Studies in Computational Intelligence 264

Olivier Sigaud
Jan Peters (Eds.)

From Motor Learning to Interaction Learning in Robots

Springer

Studies in Computational Intelligence, Volume 264

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 244. Ngoc Thanh Nguyen, Rados law Piotr Katarzyniak, and Adam Janiak (Eds.)
New Challenges in Computational Collective Intelligence, 2009
ISBN 978-3-642-03957-7

Vol. 245. Oleg Okun and Giorgio Valentini (Eds.)
Applications of Supervised and Unsupervised Ensemble Methods, 2009
ISBN 978-3-642-03998-0

Vol. 246. Thanasis Daradoumis, Santi Caballé, Joan Manuel Marquès, and Fatos Xhafa (Eds.)
Intelligent Collaborative e-Learning Systems and Applications, 2009
ISBN 978-3-642-04000-9

Vol. 247. Monica Bianchini, Marco Maggini, Franco Scarselli, and Lakhmi C. Jain (Eds.)
Innovations in Neural Information Paradigms and Applications, 2009
ISBN 978-3-642-04002-3

Vol. 248. Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri (Eds.)
Innovations in Swarm Intelligence, 2009
ISBN 978-3-642-04224-9

Vol. 249. Wesam Ashour Barbakh, Ying Wu, and Colin Fyfe
Non-Standard Parameter Adaptation for Exploratory Data Analysis, 2009
ISBN 978-3-642-04004-7

Vol. 250. Raymond Chiong and Sandeep Dhakal (Eds.)
Natural Intelligence for Scheduling, Planning and Packing Problems, 2009
ISBN 978-3-642-04038-2

Vol. 251. Zbigniew W. Ras and William Ribarsky (Eds.)
Advances in Information and Intelligent Systems, 2009
ISBN 978-3-642-04140-2

Vol. 252. Ngoc Thanh Nguyen and Edward Szczerbicki (Eds.)
Intelligent Systems for Knowledge Management, 2009
ISBN 978-3-642-04169-3

Vol. 253. Roger Lee and Naohiro Ishii (Eds.)
Software Engineering Research, Management and Applications 2009, 2009
ISBN 978-3-642-05440-2

Vol. 254. Kyandoghere Kyamakya, Wolfgang A. Halang, Herwig Unger, Jean Chamberlain Chedjou, Nikolai F. Rulkov, and Zhong Li (Eds.)
Recent Advances in Nonlinear Dynamics and Synchronization, 2009
ISBN 978-3-642-04226-3

Vol. 255. Catarina Silva and Bernardete Ribeiro
Inductive Inference for Large Scale Text Classification, 2009
ISBN 978-3-642-04532-5

Vol. 256. Patricia Melin, Janusz Kacprzyk, and Witold Pedrycz (Eds.)
Bio-inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition, 2009
ISBN 978-3-642-04515-8

Vol. 257. Oscar Castillo, Witold Pedrycz, and Janusz Kacprzyk (Eds.)
Evolutionary Design of Intelligent Systems in Modeling, Simulation and Control, 2009
ISBN 978-3-642-04513-4

Vol. 258. Leonardo Franco, David A. Elizondo, and José M. Jerez (Eds.)
Constructive Neural Networks, 2009
ISBN 978-3-642-04511-0

Vol. 259. Kasthurirangan Gopalakrishnan, Halil Ceylan, and Nii O. Attoh-Okine (Eds.)
Intelligent and Soft Computing in Infrastructure Systems Engineering, 2009
ISBN 978-3-642-04585-1

Vol. 260. Edward Szczerbicki and Ngoc Thanh Nguyen (Eds.)
Smart Information and Knowledge Management, 2009
ISBN 978-3-642-04583-7

Vol. 261. Nadia Nedjah, Leandro dos Santos Coelho, and Luiza de Macedo de Mourelle (Eds.)
Multi-Objective Swarm Intelligent Systems, 2009
ISBN 978-3-642-05164-7

Vol. 262. Jacek Koronacki, Zbigniew W. Ras, Sławomir T. Wierzchon, and Janusz Kacprzyk (Eds.)
Advances in Machine Learning I, 2010
ISBN 978-3-642-05176-0

Vol. 263. Jacek Koronacki, Zbigniew W. Raś, Sławomir T. Wierzchoń, and Janusz Kacprzyk (Eds.)
Advances in Machine Learning II, 2010
ISBN 978-3-642-05178-4

Vol. 264. Olivier Sigaud and Jan Peters (Eds.)
From Motor Learning to Interaction Learning in Robots, 2010
ISBN 978-3-642-05180-7

Olivier Sigaud and Jan Peters (Eds.)

From Motor Learning to Interaction Learning in Robots

Springer

Prof. Olivier Sigaud
Institut des Systèmes Intelligents et de Robotique (CNRS UMR 7222)
Université Pierre et Marie Curie Pyramide
Tour 55 Boîte courrier 173
4 Place Jussieu
75252 PARIS cedex 05, France
E-mail: Olivier.Sigaud@upmc.fr

Jan Peters
Max-Planck Institute for Biological Cybernetics
Dept. Schölkopf
Spemannstraße 38, Rm 223
72076 Tübingen, Germany

and

University of Southern California
Schaal/CLMC Lab
3641 Watt Way
Los Angeles, CA 90089, USA
E-mail: jan.peters@tuebingen.mpg.de

ISBN 978-3-642-05180-7 e-ISBN 978-3-642-05181-4

DOI 10.1007/978-3-642-05181-4

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2009939994

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To all our hard-working robotics students...

Preface

From an engineering standpoint, the increasing complexity of robotic systems and the increasing demand for more autonomous robots result in a surge of interest about learning methods.

This book is largely based on the successful workshop "From motor to interaction learning in robots" held at the IEEE/RSJ International Conference on Intelligent Robot Systems. The major aim of the book is to give students a chance to get started faster and researchers a helpful compandium for the study of learning in robotics.

Paris, August 2009 — Olivier Sigaud
Tübingen, August 2009 — Jan Peters

Contents

Part II: Learning Policies for Motor Control

Part III: Imitation and Interaction Learning

From Motor Learning to Interaction Learning in Robots

Olivier Sigaud and Jan Peters

Abstract. The number of advanced robot systems has been increasing in recent years yielding a large variety of versatile designs with many degrees of freedom. These robots have the potential of being applicable in uncertain tasks outside well-structured industrial settings. However, the complexity of both systems and tasks is often beyond the reach of classical robot programming methods. As a result, a more autonomous solution for robot task acquisition is needed where robots adaptively adjust their behaviour to the encountered situations and required tasks.

Learning approaches pose one of the most appealing ways to achieve this goal. However, while learning approaches are of high importance for robotics, we cannot simply use off-the-shelf methods from the machine learning community as these usually do not scale into the domains of robotics due to excessive computational cost as well as a lack of scalability. Instead, domain appropriate approaches are needed. In this book, we focus on several core domains of robot learning. For accurate task execution, we need motor learning capabilities. For fast learning of the motor tasks, imitation learning offers the most promising approach. Self improvement requires reinforcement learning approaches that scale into the domain of complex robots. Finally, for efficient interaction of humans with robot systems, we will need a form of interaction learning. This chapter provides a general introduction to these issues and briefly presents the contributions of the subsequent chapters to the corresponding research topics.

Keywords: motor learning, interaction learning, imitation learning, reinforcement learning, robotics.

Olivier Sigaud
Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222
Université Pierre et Marie Curie
Pyramide Tour 55 - Boîte Courrier 173, 4 Place Jussieu, 75252 Paris CEDEX 5, France
e-mail: Olivier.Sigaud@upmc.fr

Jan Peters
Max Planck Institute for Biological Cybernetics, Dept. Schölkopf
Spemannstrasse 38, 72076 Tübingen, Germany
e-mail: mail@jan-peters.net

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 1–12.
springerlink.com

1 Introduction

Robot learning has reached an unprecedented amount of interest in recent years. However, as the robotics domain is of particular complexity for learning approaches, it has become quite demanding for students and young researchers to get started in this area. Furthermore, due to the current high speed of development it is often hard for scientists from other areas to follow the developments. For this reason, the need for an easy entrance into this field has become strong while it is not yet time for a robot learning textbook. The idea of this book arose from these considerations.

This chapter serves two purposes: firstly, it allows us to quickly familiarize the reader with the background. In Section 1.1, we give an overview on the importance of robot learning approaches at this moment. In Section 1.2, we discuss essential background on motor, imitation and interaction learning and recommend the reader to briefly survey this part before diving into the respective chapters. Secondly, we give in Section 2 a brief overview on the chapters included in this book, before concluding in Section 3 on the necessity of a more integrated effort.

1.1 The Need for Robot Learning Approaches

At the beginning of the 21st century, robotics research is experiencing large changes in its aims and objectives. In most of the previous century, the majority of all operational robot were performing the same manufacturing task again and again in extremely structured environments such as automobile factories. Often it was easier and cheaper to build a new factory with new robots to accommodate a new car model than to reprogram an existing one. By contrast, robots are now "leaving" factory floors and start becoming part of the everyday life of average citizens. Vacuum cleaning robots have become the most sold robots to date with 4-5 million units shipped up to 2008, and programmable entertainment robots such as the RoboSapiens have become many children's favorite toy. This evolution raises the major challenge of "personalizing" the programming of our robots and making them compatible with human-inhabited environments. As a result, a variety of new issues arise that we will discuss below.

First, robots will often be in physical contact with people that are not specially trained to interact with them, thus they must be less dangerous. This concern first implies some mechanical requirements: robots must become lighter and their actuators must have inherent compliance properties as human muscles have. But in turn, these changes result in the necessity to think differently about their control loops. Either we stay with the same kind of actuator technology and we must use extremely low gains while yielding sufficient accuracy, or we come to completely new actuators like artificial muscles where the classical control knowledge is missing and learning techniques will play an important role. Finally, in any case they must never become unsafe in unforeseen situations. All these considerations result in the necessity to move from the previous standard way of thinking about robot control to new approaches that rely more on on-line, adaptive model identification and autonomous action selection properties.

Second, future robots need to be more versatile and more flexible when encountering some of the infinitely many potential situations that are part of our daily life. Despite the impressive results of human manual plan design and robot programming, these hand-crafted solutions are not likely to transfer to that large variety of different tasks and environmental states. Hence, it is becoming increasingly clear that a new approach is essential. To interact better with their environment, robots will need more and more sensing capabilities but also algorithms that can make use of this richer sensory information. Due to this increase of complexity both on the perception and action side, robots will need to *learn* the appropriate behavior in many situations. This challenge is becoming recognized in the general robotics community. It has resulted in supportive statements by well-known roboticists such as "*I have always said there would come a time for robot learning — and that time is now*" by O. Khatib (at Stanford, October 2006) and "*robot learning has become the most important challenge for robotics*" by J. Hollerbach (at the NIPS Workshop on Robotics Challenges for Machine Learning, December 2007).

Third, apart from the security aspects of human robot interaction that we have already outlined, the fact that robots will be in interaction with people must also be taken into account in their control and learning architecture. Thinking of a robot as interacting with people has a lot of consequences in the design of their control architecture, as will be investigated in subsequent parts of this book. As we will see, robots can be instructed from their human user how to perform a new task by imitation, or by physical or language-based interaction during task execution, etc.

All these changes on the way to consider robots and their control comes along with another major evolution about the hardware platforms available. In the last twenty years, a huge technological effort has resulted in the design of more complex, more efficient and more flexible platforms with the challenges above in mind. In particular, these last years have seen the emergence of humanoid robots of diverse size, capabilities and price as well as a variety of bimanual mobile robotics platforms. In these platforms, all the challenges listed above are essential problems that always need to be addressed.

This large shift in robotics objectives has resulted in an increasing visibility of the corresponding lines of research. In the last few years, we have seen an increasing amount of robot learning publications both at top machine learning (such as NIPS, ICML and ECML) conferences and mainstream robotics conferences (particularly at R:SS, ICRA and IROS). The number of learning tracks has been increasing the IEEE multi-track conferences ICRA and IROS and there have been at least 12 workshops on robot learning in 2007–2009. This development has resulted in numerous special issues in excellent robotics international journals such as the *International Journal of Robotics Research*, *Autonomous Robots*, the *International Journal of Humanoid Robots* and the *IEEE Robotics & Automation Magazine*. Recently, it even gave rise to the creation of an IEEE Technical Committee on Robot Learning.

All these considerations led us to consider that this is the good time to publish a book about Motor learning, Imitation and Interaction Learning in Robots. In the next section, we will highlight the relationships between the corresponding different subfields before giving an overview of the contributions to the book.

1.2 Motor Learning, Imitation Learning and Interaction Learning

Making humanoid robots perform movements of the same agility as human movements is an aim difficult to achieve even for the simplest tasks. Although a lot of impressive results have been obtained based on pure hand-coding of the behaviour, this approach seems too costly and, probably, even too difficult if we ever want humanoid robots and mobile manipulators to leave research labs or factories and enter human homes.

The most immediate alternative to this manual programming is *imitation learning*, also known as *learning from demonstration* or *programming by showing*. This approach is relatively well-developed and has resulted in a variety of excellent results in previous work. It includes several different teaching approaches. Some researchers record human motion in the context of a task with motion capture tools and transfer the motion on the robot, which implies solving the *correspondence problem* resulting from the differences between the mechanics of a human being and a robot system (e.g., already mapping the human arm kinematics on a non-anthropomorphic robot arm is a difficult problem). Hence, it is often easier to employ teleoperating interfaces for teaching, or even use the robot by itself as a haptic device. Furthermore, different approaches are employed in order to recover a policy; while the approaches discussed in this book directly mimic the observed behaviors, there is an alternative stream of research that employs an *inverse reinforcement learning* approach which rather attempts to recover the intent of the teacher by modeling his cost function and, subsequently, derives the policy that is optimal with respect to the cost-to-go (Abbeel and Ng, 2004; Coates et al, 2008; Ratliff et al, 2009). While several chapters in this book are dedicated to imitation learning and should yield a good start in this area, we want to point out to the reader that several important groups in imitation learning are not covered and we urge the reader to study (Atkeson and Schaal, 1997; Schaal, 1999; Ijspeert et al, 2003; Abbeel and Ng, 2004; Calinon et al, 2007; Coates et al, 2008; Ratliff et al, 2009).

Nevertheless, learning from demonstration does not suffice as the behavior of the robot will be restricted to the behaviors that have been demonstrated, even if some generalization mechanisms can slightly remediate that situation, allowing for instance adaptation to slightly changing contexts. To go beyond an initial imitation, we need the robots to adapt online so that they can react to new situations. There exist a few situations where such an adaptation can be achieved purely by supervised learning, e.g., when the functional relationship can be directly observed as in inverse dynamics model learning and a relearning after a change of the dynamics due to an external load or a failure is straightforward. However, the majority of all problems require some kind of self-improvement, e.g., we need to adapt elementary movements to an unforeseen situation, improve a policy learned from a demonstration for better performance, learn new combinations of movements or even simply learn an inverse model of a redundant system. Addressing these problems is often formalized in the *reinforcement learning* framework, which mimics the way animals and humans improve their behaviour by trial-and-error in unforeseen situations. A

key issue in reinforcement learning is exploration: since we do not know in advance which behaviour will give rise to high outcome and which will not, we have to try various different actions in order to come up with an efficient strategy.

Taken as a whole, learning of general motor capabilities, at the control level or at the behavioral strategy level, is a very hard problem. This involves the exploration of a huge space of possibilities where a lot of standard algorithmic steps boil down to hard optimization problems in a continuous domain. Given the difficulty of the exploration problem in that domain, the combination of reinforcement learning with imitation learning has been shown to be fruitful. Here, imitation provides an efficient way to initialize policies so that the explorative policy can focus on the behaviors or strategies that have a high probability of being efficient.

Finally, the third robot learning topic that we cover in this book is interaction learning. Interaction learning allows the robot to discover strategies for actively using the contact with human in its proximity. It often shares tools and methods with imitation learning, since both approaches have to take the presence of the human around the robot into account. In fact, imitation learning is a particular case of interaction learning in the sense that imitation is a particular type of interaction. However, interaction learning is not restricted to reproduce the observed behavior of a human. Instead, interacting means getting jointly involved in a common activity both taking the behavior of the respective other into account. Thus, interaction can be physical, when the human user actually exerts some force onto the robot or, conversely, when the robot does so to the human user. It can also be communicative, either through diverse modalities of language or through communicative gestures. It can finally be purely implicit, when the robot and the user try to adapt their behaviour to the other without any direct communication, just through observing. Interaction learning provides a challenging context for motor learning in general. Human motor behavior is often difficult to predict and, thus, interaction may require learning non-stationary models of the dynamics of the coupling between humans and robots.

Last, but not least, the study of human motor behavior requires a deep understanding of the connections between motor, imitation and interaction learning. For example, neurophysiological studies of human subjects suggest that motor learning processes and more cognitive learning and developmental processes have much in common, particularly when it comes to interaction with other beings. After the much celebrated discovery of the so called ”mirror neurons” relating motor learning to imitation and language acquisition, several neurophysiological studies have revealed that brain areas generally considered as motor, such as the cerebellum, or dedicated to action selection, like the basal ganglia, are in fact employed in more general cognitive functions such as learning tool use, imitation, language and so forth (Doya, 1999). Taken together, these facts advocate for a hierarchical understanding of the brain architecture where motor learning and interaction learning are tightly coupled processes at the root of cognition (Wolpert et al, 2003; Demiris and Khadhouri, 2006). These topics are highly relevant for robot learning as the human motor system is still the best prototype for us to study in order to obtain new and better algorithms.

2 Overview of the Book

From the previous section, it has become apparent that *Motor*, *Imitation* and *Interaction Learning* are highly dependent on each other and complementary. The purpose of the book is to provide a state-of-the-art view of these different subfields within the same volume so as to cast the basis for an improved dialog between them. We have divided the book into three parts, but there is a strong overlap between the topics covered by these parts.

2.1 Biologically Inspired Models for Learning in Robots

A common view in most learning approaches to robotics is that humans exhibit all the properties we want from a robot system in terms of adaptivity, learning capabilities, compliance, versatility, imitation and interaction capabilities etc. Hence, it might be a good idea to be inspired by their functionality and, as a result, a lot of robot learning approaches are bio-inspired in some sense. More precisely, in this book we can distinguish three different sources of inspiration in this line of thinking.

The first one has to do with trying to implement robot controllers on a representation that is as similar as possible to the neural substrate that one can find in the human motor system. The complexity of the computational models resulting from this line of thinking raises the problem of their validation. Here, robotics plays a prominent role as a tool to evaluate the capability of these wide scope models to account for the phenomena they address. In this book, two chapters, (Duff et al, 2010) and (Lagarde et al, 2010), are following this line of thinking. The first one proposes a biologically based cognitive architecture called Distributed Adaptive Control to explain the neuronal organization of adaptive goal oriented behavior. The second one, based on neural field models, is interested in low level, basic imitation mechanisms present early in the newborn babies, showing how the different proprioceptive signals used in the examples can be seen as bootstrap mechanisms for more complex interactions.

A second line of inspiration consists in trying to reproduce the learning properties of the human motor system as observed from outside, building models that rely on computational principles that may explain these observed properties. The work of Mitrovic et al (2010) illustrates this approach. It proposes an efficient implementation of a model of motor adaptation based on well accepted computational principles of human motor learning, using optimal feedback control methods that require a model of the dynamics of the plant in a context where this model is learned. The work of Herbort et al (2010) shares similar goals, but the authors address slightly different motor learning phenomena, with a particular focus on motor preparation. The authors propose an implementation of their system based on artificial neural networks on a simulated complex robot, perfectly illustrating the highly cross-disciplinary nature of this domain.

Finally, a third line of inspiration takes its sources in developmental psychology, giving rise to the so called developmental robotics or epigenetic robotics (Lungarella et al, 2004). In some sense, the work already discussed by Lagarde et al

(2010) can also fall into this category. Moreover, the work in (Oudeyer et al, 2010) is a prominent example of this line of thinking, investigating how a model of activity selection based on curiosity can give rise to the capability to tackle more and more difficult tasks within a life-long learning paradigm.

2.2 Learning Models and Policies for Motor Control

The differences between papers about biologically inspired models for learning in robots and the one that fall into this technical part is often small. A lot of work about learning models and policies for motor control is also inspired by biological considerations but does not attempt to provide an explanation for biological behavior. As there are important differences between the mechanics of the human musculo-skeletal system and the mechanical design of robots, severe limitations are imposed on the degree of similarity between natural and artificial controllers. Indeed, for instance, the human musculo-skeletal system has the amazing ability to control both stiffness and position of each joint independently from each other due to co-contraction. By contrast, nearly all humanoid robots to date are having a single motor per joint and, thus, offer either position access (e.g., cheap RoboSapiens designs), setting desired velocities (e.g., the Fujitsu Hoap, iCub and many others) or are torque controlled (e.g., the SARCOS humanoids). This makes robots controllers unable to make profit of the nice properties of the muscles that human people use in practice and this drives robotics control towards control principles that may differ a lot from those observed in human movement. In such a context when the standard engineering knowledge is not well developed, the contribution from (Fumagalli et al, 2010) compares two learning techniques, namely Least Squares Support Vector Machines and Neural Networks, on their capability to estimate the forces and torques measured by a single six-axis force/torque sensor placed along the kinematic chain of a humanoid robot arm.

Beyond these considerations, the chapters regrouped in this part fall in two categories. The first category is about learning models of the plant, either direct or inverse, at the kinematics, velocity kinematics and dynamics level. This kind of work, giving rise to motor adaptation capabilities, is one of the main mechanisms to obtain compliance and versatility in robots. The contribution from Salaün et al (2010) provides an overview of how learned kinematics and velocity kinematics models can be used within a feedback control loop in the Operational Space Control framework. Learning these models is a difficult self-supervised learning problem in large continuous state and action spaces, thus having an efficient learning method is crucial. A few learning techniques have emerged in the last years as particularly competitive to address this task. In particular, the most recent Locally Weighted Regression methods give rise to very fast implementations that scale well and are able to tackle large problems but suffer from the necessity to tune a lot of parameters, whereas methods based on Gaussian Processes, are computationally more intensive as the size of the problems grows but require less tuning. The chapter by Nguyen-Tuong et al (2010) proposes a local method based on Gaussian Processes that combines the good

properties of both families of approaches. They are able to show that the model works well in the context of learning inverse dynamics.

The second category of contributions in this part is about finding the good computational framework to derive efficient controllers from learning principles. In that domain, an important approach is about the automatic tuning of motor primitives, that already provided convincing results (e.g., as Ijspeert et al (2003)). But whereas in these previous approaches primitives were based on open-loop control, the chapter by Kober et al (2010) provides an extension to the case where primitives incorporate perceptual coupling to external variables, giving rise to closed-loop policies.

Taking a very different view, the chapter by Toussaint and Goerick (2010) presents a bayesian formulation of classical control techniques based on task space to joint space mapping, that results in the possibility to consider motor execution and motor planning as a unified bayesian inference mechanism. The chapter highlights the interesting robustness properties of the resulting framework and highlights deep relationships with the optimal control framework that provides convincing computational principles for motor control (Todorov and Jordan, 2002; Todorov, 2004; Todorov and Li, 2005). Still in the same category but based on different principles, (Howard et al, 2010) is focused on learning from trajectories a controller able to realize a set of tasks subject to a set of unknown constraints. Finally, the contribution from Roberts et al (2010) describes a nice application of a model-free reinforcement learning-based control methodology, based on an optimized policy gradient algorithm, to the control of an experimental system dedicated to the study of flapping wing flight.

2.3 Imitation and Interaction Learning

This part starts with a chapter by Lopes et al (2010) which provides an overview of imitation learning methods in robots. It provides a presentation of some recent developments about imitation in biological systems, as well as a focus on robotics work that consider self-modelling and self-exploration as a fundamental part of the cognitive processes required for higher-level imitation.

The contribution (Chalodhorn and Rao, 2010) describes an approach based purely on human motion capture to achieve stable gait acquisition in a humanoid robot despite its complex mechanical structure. The chapter gives a good example of the theoretical difficulties and technical intricacies that must be faced in such kind of imitation learning approaches given the "correspondence problem" that must be solved between the human musculo-skeletal system with its many redundant degrees of freedom and robot systems with their different & well-defined kinematic structures. The chapter insists on dimensionality reduction techniques that can be used to simplify the resolution of the correspondence problem.

After a chapter focused on learning one particular motor primitive from imitation, the chapter by Kulić and Nakamura (2010) proposes a broader approach for autonomous and incremental organization of a set of such primitives learned by observation of human motion, within a life-long learning framework. The hierarchical

organization makes it easier to recognize known primitives and to determine when adding a new primitive in the repertoire is necessary. The different motor primitives are represented by Hidden Markov Models or by Factorial Hidden Markov Models.

The contribution of Grollman and Jenkins (2010) is focused on the case where some task is decomposed into a set of subtasks. With a more critical standpoint than previous chapters, it examines the limits of a regression-based approach for learning a Finite State Machine controller from demonstration of a basic robot soccer goal-scoring task, based on an Aibo robot.

We already discussed in the first section of this chapter that there is a lot of potential in the combination of imitation learning (or learning from demonstration) and automatic improvement of control policies. The chapter from Argall (2010) falls into this category. It presents an approach for the refinement of mobile robot control policies, that incorporates human teacher feedback.

The chapter (Detry et al, 2010) describes a method that combines imitation learning with actual interaction with object to learn grasp affordances. More precisely, the work is about learning to grasp objects described by learned visual models from different sources of data. The focus is on the organization of the whole knowledge that an agent has about the grasping of an object, in order to facilitate reasoning on grasping solutions and their likelihood of success.

The last two chapters are more focused on interaction learning. First, the chapter from Hörnstein et al (2010) is about language acquisition in humanoid robots, based on interaction with a caregiver and using as few built in a priori knowledge or primitives as possible. The importance of motor learning in the language acquisition process is underlined. Second, the contribution from Lallee et al (2010) presents an outstanding integration effort towards language based interaction between a robot and a non-expert user in the context of a cooperation between them. The focus is put on the use of the Spoken Language Programming approach to facilitate the interaction.

3 Conclusion and Perspectives

Robot learning is a young, fruitful and exciting field. It addresses problems that will become increasingly important for robotics as the platforms get more and more complex and the environment get less and less prepared or structured. The reader will find in this book research works stemming from different areas – statistical learning of models, reinforcement learning, imitation and interaction learning – that all contribute to the global endeavour of having more adaptive robots able to deal with more challenging settings, in particular those where interaction with humans is involved. In Section 1.2, we highlighted some ways in which diverse research efforts could be combined given the complementary subproblems they address. However, when closing this book, the reader will probably have the feeling that the different contributors are working within isolated frameworks and that a global coordination effort is still missing. Our view as editors is that finding frameworks giving rise to

the possibility of such coordination is the next step in the field, and we hope this book will play its role towards this next step.

References

Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the International Conference on Machine Learning (2004)

Argall, B.D.: Mobile robot motion control from demonstration and corrective feedback. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 431–450. Springer, Heidelberg (2010)

Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Proceedings of the International Conferenec on Machine Learning, pp. 12–20 (1997)

Calinon, S., Guenter, F., Billard, A.: On Learning, Representing and Generalizing a Task in a Humanoid Robot. IEEE Transactions on Systems, Man and Cybernetics, Part B 37(2), 286–298 (2007)

Chalodhorn, R., Rao, R.P.N.: Learning to imitate human actions through eigenposes. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 357–381. Springer, Heidelberg (2010)

Coates, A., Abbeel, P., Ng, A.Y.: Learning for control from multiple demonstrations. In: Proceedings of the International Conference on Machine Learning (2008)

Demiris, Y., Khadhouri, B.: Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems 54, 361–369 (2006)

Detry, R., Baseski, E., Popovi, M., Touati, Y., Krüger, N., Kroemer, O., Peters, J., Piater, J.: Learning continuous grasp affordances from sensorimotor interaction. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 451–465. Springer, Heidelberg (2010)

Doya, K.: What are the computations of the cerebellum, the basal ganglia, and the cerebral cortex? Neural Networks 12, 961–974 (1999)

Duff, A., César, R., Costa, R., Marcos, E., Luvizotto, A.L., Giovannucci, A., Sanchez Fibla, M., Bernardet, U., Verschure, P.F.M.J.: Distributed adaptive control: A proposal on the neuronal organization of adaptive goal oriented behavior. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 15–41. Springer, Heidelberg (2010)

Fumagalli, M., Gijsberts, A., Ivaldi, S., Jamone, L., Metta, G., Natale, L., Nori, F., Sandini, G.: Learning how to exploit proximal force sensing: a comparison approach. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 149–167. Springer, Heidelberg (2010)

Grollman, D.H., Jenkins, O.C.: Can we learn finite state machine robot controllers from interactive demonstration? In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 407–430. Springer, Heidelberg (2010)

Herbort, O., Butz, M.V., Pedersen, G.: The sure reach model for motor learning and control of a redundant arm: from modeling human behavior to applications in robots. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 85–106. Springer, Heidelberg (2010)

Hörnstein, J., Gustavsson, L., Santos-Victor, J., Lacerda, F.: Multimodal language acquisition based on motor learning and interaction. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 467–489. Springer, Heidelberg (2010)

Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Methods for learning control policies from variable-constraint demonstrations. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 253–291. Springer, Heidelberg (2010)

Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Advances in Neural Information Processing Systems (NIPS), pp. 1523–1530 (2003)

Kober, J., Mohler, B., Peters, J.: Imitation and reinforcement learning for motor primitives with perceptual coupling. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 209–225. Springer, Heidelberg (2010)

Kulić, D., Nakamura, Y.: Incremental learning of full body motion primitives. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 383–406. Springer, Heidelberg (2010)

Lagarde, M., Andry, P., Gaussier, P., Boucenna, S., Hafemeister, L.: Proprioception and imitation: on the road to agent individuation. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 43–63. Springer, Heidelberg (2010)

Lallee, S., Yoshida, E., Mallet, A., Nori, F., Natale, L., Metta, G., Warneken, F., Dominey, P.F.: Human-robot cooperation based on interaction learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 491–536. Springer, Heidelberg (2010)

Lopes, M., Melo, F., Montesano, L., Santos-Victor, J.: Abstraction levels for robotic imitation: Overview and computational approaches. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 313–355. Springer, Heidelberg (2010)

Lungarella, M., Metta, G., Pfeifer, R., Sandini, G.: Developmental robotics: a survey. Connection Science 0, 1–40 (2004)

Mitrovic, D., Klanke, S., Vijayakumar, S.: Adaptive optimal feedback control with learned internal dynamics models. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 65–84. Springer, Heidelberg (2010)

Nguyen-Tuong, D., Seeger, M., Peters, J.: Real-time local gp model learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 193–207. Springer, Heidelberg (2010)

Oudeyer, P.Y., Baranes, A., Kaplan, F.: Intrinsically motivated exploration and active sensorimotor learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 107–146. Springer, Heidelberg (2010)

Ratliff, N.D., Silver, D., Bagnell, J.A.: Learning to search: Functional gradient techniques for imitation learning. Autonomous Robots 27(1), 25–53 (2009)

Roberts, J.W., Moret, L., Zhang, J., Tedrake, R.: Motor Learning at Intermediate Reynolds Number: Experiments with Policy Gradient on the Flapping Flight of a RigidWing. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 293–309. Springer, Heidelberg (2010)

Salaün, C., Padois, V., Sigaud, O.: Learning forward models for the operational space control of redundant robots. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 169–192. Springer, Heidelberg (2010)

Schaal, S.: Is imitation learning the route to humanoid robots? Trends in Cognitive Sciences 6, 233–242 (1999)

Todorov, E.: Optimality principles in sensorimotor control. Nature Neurosciences 7(9), 907–915 (2004)

Todorov, E., Jordan, M.I.: Optimal feedback control as a theory of motor coordination. Nature Neurosciences 5(11), 1226–1235 (2002)

Todorov, E., Li, W.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proceedings of the American Control Conference, pp. 300–306 (2005)

Toussaint, M., Goerick, C.: A bayesian view on motor control and planning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 227–252. Springer, Heidelberg (2010)

Wolpert, D.M., Doya, K., Kawato, M.: A unifying computational framework for motor control and social interaction. Philosophical Transactions of the Royal Society 358, 593–602 (2003)

Distributed Adaptive Control: A Proposal on the Neuronal Organization of Adaptive Goal Oriented Behavior

Armin Duff, César Rennó-Costa, Encarni Marcos, Andre L. Luvizotto, Andrea Giovannucci, Marti Sanchez-Fibla, Ulysses Bernardet, and Paul F.M.J. Verschure

Abstract. In behavioral motor coordination and interaction it is a fundamental challenge how an agent can learn to perceive and act in unknown and dynamic environments. At present, it is not clear how an agent can – without any explicitly predefined knowledge – acquire internal representations of the world while interacting with the environment. To meet this challenge, we propose a biologically based cognitive architecture called Distributed Adaptive Control (DAC). DAC is organized in three different, tightly coupled, layers of control: reactive, adaptive and contextual. DAC based systems are self-contained and fully grounded, meaning that they autonomously generate representations of their primary sensory inputs, hence bootstrapping their behavior form simple to advance interactions. Following this approach, we have previously identified a novel environmentally mediated feedback loop in the organization of perception and behavior, i.e. behavioral feedback. Additionally, we could demonstrated that the dynamics of the memory structure of DAC, acquired during a foraging task, are equivalent to a Bayesian description of foraging. In this chapter we present DAC in a concise form and show how it is allowing us to extend the different subsystems to more biophysical detailed models. These further developments of the DAC architecture, not only allow to better understand the biological systems, but moreover advance DACs behavioral capabilities and generality.

Armin Duff
INI Institute of Neuroinformatics, UNI-ETH Zürich, Winterthurerstrasse 190,
CH-8057 Zürich, Switzerland
e-mail: duffar@ini.phys.ethz.ch

Paul F.M.J. Verschure
ICREA Institució Catalana de Recerca i Estudis Avançats, Passeig Lluís Companys 23,
E-08010 Barcelona, Spain
e-mail: paul.verschure@upf.edu

All
SPECS, IUA, Technology Department, Universitat Pompeu Fabra,
Carrer de Roc Boronat 138, E-08018 Barcelona, Spain

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 15–41.
springerlink.com

1 Introduction

One of the main challenges we face in the development of novel real-world cognitive systems and robotics technologies is to construct systems with perceptual, cognitive and behavioral capabilities that allow autonomous coordination of complex sensory and effector systems under varying task conditions. Successful solutions to this challenge must account for the rapidly changing demands from the behaving system itself and its environment. Furthermore, all this is to be achieved under severe constraints placed upon available computing resources and time, and upon the basis of incomplete and only partly reliable information. Humans, and animals in general, excel in this challenge making optimal use of their acquired knowledge [20, 69]. For instance, many animals, from bees to mammals, show optimal performance in foraging by adapting their behavioral strategies to the particular reward contingencies and risks that the real world offers [20, 14, 38, 55]. For instance, rats placed in a radial arm maze, where different arms contain a varying amount of food pellets, develop an optimal foraging strategy in terms of travel time [55]. Foraging can be described as a goal-oriented exploration for resources normally motivated by food deprivation. Foraging is an advanced goal oriented behavior where prior knowledge of an environment and acquired behavioral strategies must be matched to the novelty and hazards presented by an unpredictable world. These constraints are defined at varying spatial and temporal scales where a successful forager must simultaneously satisfy local and global constraints such as performing obstacle avoidance while staying on course to reach a known feeding site whilst also allocating resources consistent with its allostatic needs. The continuous decision making by the animal on global tasks can be further decomposed into sub-tasks each serving specific goals and sub-goals, e.g. during spatial navigation a target site is found by moving from landmark to landmark in a specific order. Given natural selection and the dynamics of internal and external constraints, the information processing and behavioral strategies of a forager must be performed in a near-optimal fashion. Hence, foraging provides a suitable and challenging test case for artificial controls systems.

Behavior control is thought to depend on three distinct components: sense, think, and act. Although this view has been challenged [51, 70] and the three components are tightly interlinked [49, 6], this distinction segregates the basic components involved in foraging: sensory processing, action selection and cognition. Sensory processing is a key component of successful foraging. In a real world environment a forager is exposed to a wealth of sensory inputs. Only a subset of these inputs will provide behaviorally relevant information. In order to cope with this overload of information, the forager has to extract the behaviorally relevant information and ignore irrelevant or noisy sensory signals. In a changing environment, and under changing task conditions, these internal representations have to be adapted constantly. This process of forming and adapting internal representations of the sensory inputs is termed perceptual learning [23]. In order to reach its goals a forager has to select a set of actions depending on its internal states. In classical conditioning, for example, one distinguishes reactive actions (unconditioned response UR), which are innate and elicited independently of prior learning given a certain sensory input

(unconditioned stimulus *US*) and conditioned actions (conditioned response *CR*) elicited by sensory stimulus (conditioned stimulus *CS*) that is associated through learning to a specific action [39, 50]. In operant conditioning such a direct association is not always possible since a series of actions is needed to reach the goal [66]. However, reaching a goal is often associated with a certain value (reward or avoidance of punishment) and as a result, action selection requires as the association of value to different actions. The assessment and assignment of values to internal representations and possible actions is a critical step for an optimal foraging agent. The utility of a reward is influenced by different factors: probability of the reward, time delay and internal state of the agent [59]. This process of action evaluation and selection is termed behavioral learning. Together with the formation of internal sensory representation and the direct assignment of values and actions to these representations, foraging does also require planning and rule learning. Different forms of memory allow the forager to extend direct perception-action associations to sensory states that are not detectable at the moment of decision or that are ambiguous without the actual context.

Several models have been proposed to describe the process of sensory processing, action selection and cognition. In the field of Artificial Intelligence (AI) one distinguishes traditional AI and new AI. Traditional AI explains intelligent behavior at the knowledge level [44]. The knowledge level describes intelligent behavior in terms of knowledge, goals and actions and intelligent behavior emerges from the principle of rationality.

> . . . if the system wants to attain goal G and knows that to do act A will lead to attaining G, then it will do A. This law is a simple form of rationality that an agent will operate in its own best interest according to what it knows [44, p. 49].

The implicit assumption of the knowledge level description of intelligent behavior is that, general intelligent behavior is based on the manipulation of symbols. This view leads to a series of fundamental problems such as the frame problem [41], grounding problem [24] and the frame of reference problem [13] all deriving from the "a priori" definition of the internal representations and rules [73]. The New AI counteracts this problem by putting forward the importance of situatedness, embodiment and grounding of the artificial intelligent systems through the use of real-world systems [12]. New AI aims at minimizing the "a priori" definitions of internal representations and rules and relies on iterative methods to generate internal representations while interacting with the environment [52, 12, 11]. Where traditional AI is mainly concerned with the cognitive part new AI has aimed to solve problems embedded in the real world [52]. These different perspectives in AI raise the question if these two views on intelligence are incompatible or can be combined [76]. One motivation for trying to unify these views is that they all provide a complementary solution to the foraging problem. Where traditional AI found effective descriptions of higher-level cognitive processes such as problem solving and planning, the new AI provides solutions to real world problems. However, where traditional AI fails to ground its solutions in the real world, the new AI faces the challenge to scale up to non-trivial processes. Combining these views requires on the one hand the generation of

symbol like representation in a fully embedded and grounded process and on the other hand an approximation of a logic formalism to manipulate the generated symbols.

In biological systems the generation of internal representation of the environment is ascribed to perceptual learning. Perceptual learning has been studied in a variety of tasks (for a review see [16, 45]) showing that it is optimally adapted to statistical structure of the available sensory input [60] but also that it regards the behavioral relevance of the sensory input [7, 57, 81]. This suggests that behavioral and perceptual learning are tightly interlinked. In theoretical studies, perceptual and behavioral learning are mostly studied in separation [48, 47, 65]. The majority of perceptual learning models are based on statistical methods [29, 56, 5, 53]. In these models perception is defined by the need of reducing the dimensionality of the input signal while preserving as much relevant information as possible. Commonly, an artificial neural network is trained to compress the input while optimizing a certain statistical property. A variety of possible optimization criteria have been proposed including explained variance [46], independence [32], sparseness [48, 30, 33], temporal stability [31, 26, 86, 19, 15] and efficiency [37, 61]. This approach allows the formation of optimal representations similar to these found in biological system such as the simple and complex cells of the visual cortex [48, 31, 35] up to place cell representations as found in hippocampus [86, 19]. All these statistical learning algorithms are solely driven by the statistics of the input and the inter-dependence of perceptual learning and behavioral learning is not taken into account.

Likewise, in most of the models of behavioral learning, perceptual learning is practically ignored. Behavioral learning is reduced to associating actions to predefined states or sequences of states that describe the task domain. The acquisition of these states and the adaptation of the action association to a changing state space are generally ignored. A prominent theory for behavioral learning is reinforcement learning [65]. Going back to the law of effect proposed by Thorndike [66], reinforcement learning algorithms are based on the notion of trial and error learning. A numerical reward signal encodes the success of an executed action and learning consists in assigning different reward values to states and actions. These acquired values will guide future action selection. The values are however always assigned to predefined state spaces and suffer from the same grounding problem as traditional AI. Although the isolated investigation of the different components leads to valuable insights, the apparent interaction between the two is not regarded. In an embedded system that generates the internal representations while acting in the environment, perceptual and behavioral learning can however not be treated separately.

To breach the apparent gap between the traditional and new AI and integrate the to views that unifies perceptual and behavioral learning we have proposed a cognitive architecture called Distributed Adaptive Control (DAC) [76, 73, 79, 74, 78]. DAC proposes that the core of perceptual, cognitive and behavioral systems of the brain, including their adaptive properties, is probed through the paradigm of classical and operant conditioning. Classical conditioning is a form of associative learning where an initially neutral stimulus (conditioned stimulus *CS*) is over time associated

to a conditioned response (CR) through its contiguous representation with motivational stimulus (unconditioned stimulus US) [50]. The presentation of a US alone leads to an innate automatic response (unconditioned response UR). In a typical classical conditioning experiment in animals (such as rodents), a tone (CS) is presented together with a foot shock (US) leading to a freezing response (UR). Initially the CS is neutral and does not induce any response. After several presentations however the CS alone will induce freezing (CR). As opposed to stimulus response associations of classical conditioning, in operant conditioning, the animal is actively learning and has to perform a series of actions to reach a goal state, i.e. reward [66]. Following the law of effect [66] the animal associates particular actions to different states of the environment. In this case the US resulting from an action (CR) given a certain state (CS) act as a reinforcement. In a Skinner-Box experiment for example an animal learns to push a lever (CR) in order to receive a food reward (US).

The two paradigms of classical and operant conditioning that have been defined more than a century ago have stimulated an enormous amount of studies at both the behavioral aspects of learning and its neuronal substrate. In essence the former type of learning allows the animal to approximate the causal fabric of its environment where CSs cause USs while the latter form of learning allows the agent to infer the causal relationship between its own actions and its environment. Prediction is central to these two forms of learning. This is well captured in the, so-called, Rescorla and Wagner laws of conditioning or stimulus competition that prescribe that animals only learn when events violate their expectations [54]. Also the foraging paradigm can be described in terms of these two elementary forms of learning. Automated reflexes ($US \rightarrow UR$) generate a basic exploratory behavior approaching rewarded sides and avoiding negatively rewarded sides. Based on this exploratory behavior the agent associates approach and avoidance actions (CR) to previously neutral stimuli (CS). This actions form again the basis to learn sequences of sensory action responses as in operant conditioning.

DAC is based on the fundamental assumption that foraging can be explained on the basis of the interaction of three layers of control; reactive, adaptive and contextual (see Fig. 1). The reactive control layer provides a behaving system with a pre-wired repertoire of reflexes, which enables the behaving system to interact with its environment and to accomplish simple automatic behaviors ($US \rightarrow UR$). The activation of any reflex, however, also provides cues for learning that are used by the adaptive layer. The adaptive layer provides the mechanisms for the adaptive classification of sensory events (CS) and the reshaping of responses (CR) supporting the acquisition of simple tasks as in classical conditioning. The sensory and motor representations formed at the level of the adaptive layer also provide the inputs to the contextual layer, which acquires, retains, and expresses sequential representations using systems for short-term (or working) and long-term memory, providing a model of operant conditioning. Thus DAC proposes that the adaptive layer is both acquiring adaptive responses and a representational substrate for the planning system of the contextual layer.

DAC has been investigated using formal approaches [75, 74] and robots [76, 78, 74, 79]. The prototypical robot test case for DAC is an open arena foraging

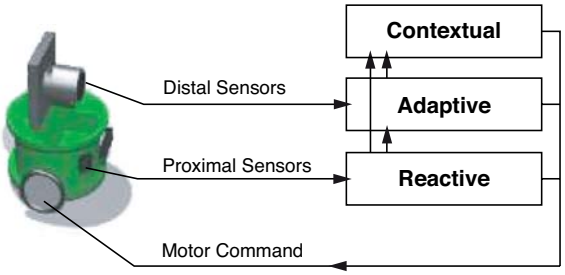


Fig. 1 The DAC architecture: The reactive layer receives inputs from the robot's proximal sensors (e.g. distance and light sensors). The adaptive layer receives inputs from the robot's distal sensors (e.g. camera). The contextual layer receives sensory motor information from the adaptive layer. All the three layers contribute to the action controlling the robot.

task. In this task, the robot, equipped with proximal and distal sensors, explores the arena in search for light sources while avoiding collisions with the surrounding wall. Colored patches scattered on the floor serve as landmarks for the navigation. In the framework of classical conditioning the proximal (e.g. distance and light) sensors serve as aversive and appetitive US. Close to the light or at a collision an UR is triggered approaching the light or turning away form the wall. The colored patches serve as CS.

In this chapter we describe how the DAC architecture provides a unification of perceptual and behavioral learning in a single behaving system that can account for a successful and structured behavior in a foraging task. We show how perception and behavior can interact synergetically via the environment, i.e. behavioral feedback, and how the knowledge level description of the foraging task can be mapped to the memory structures of the contextual layer. Further, we describe how the proposed integration serves as starting point for the investigation of more biologically detailed models of key structures of perceptual and behavioral learning.

2 Formal Description of DAC

2.1 Reactive and Adaptive Layer

The adaptive and the reactive layer comprise four neuronal groups: the unconditioned stimulus US, the conditioned stimulus CS, the internal state IS and the motor map cell group MM (see Fig. 2). The neuronal groups are modeled as mean firing rate neurons. Both the US and the CS are linked with a synaptic weight matrix to the IS. The connections V of the US to the IS are pre-wired and static. They define the reactive layer. The connections W from the CS to the IS represent the adaptive substrate of the conditioning process and are subject to learning. The IS cell group is connected to a motor map MM in a predefined way over the weight matrix U. If the activity in the IS cell group is higher than a defined threshold θ^A the motor map

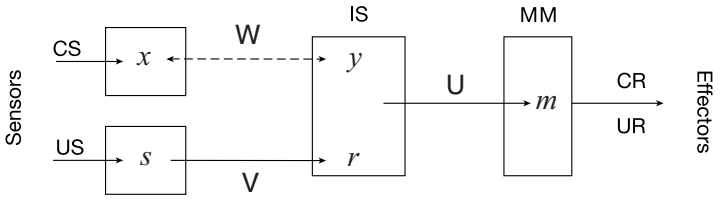


Fig. 2 Adaptive and the reactive layer: Squared boxes stand for neuronal groups. Arrows stand for static (solid line) and adaptive (dashed line) synaptic transitions between the groups. See text for further explanations.

(*MM*) is activated. A winner take-all mechanism in the *MM* cell selects the action to be executed. If the adaptive layer activates no action the reactive layer performs a default action, i.e. translate forward.

We define the following abbreviations for the activities in the different cell groups:

$$
\begin{aligned}
s &= \text{Activity of the } US \text{ cell group} \in \mathbb{R}^M \\
x &= \text{Activity of the } CS \text{ cell group} \in \mathbb{R}^N \\
z &= \text{Activity of the } IS \text{ cell group} \in \mathbb{R}^K \\
V &= \text{weight matrix from } US \text{ to } IS \text{ cell group} \in \mathbb{R}^{M\times K} \\
W &= \text{weight matrix from } CS \text{ to } IS \text{ cell group} \in \mathbb{R}^{N\times K} \\
r &= \text{Contribution of the } US \text{ to } IS \in \mathbb{R}^K \\
y &= \text{Contribution of the } CS \text{ to } IS \in \mathbb{R}^K \\
m &= \text{Activity in the } MM \text{ cell group} \in \mathbb{R}^L \\
U &= \text{weight matrix from the } IS \text{ to the } MM \text{ cell group} \in \mathbb{R}^{K\times L}
\end{aligned}
$$

Usually the dimensionality N of the CS is higher than the dimensionality K of the IS. The dimensionality M of the US is in general but not necessary similar or equal to the dimensionality K of the IS cell group. In the general case the activity of the US and the CS cell can be a nonlinear function of the sensor readings. Usually the function is however the unity function. With these definitions the forward dynamics of the adaptive and reactive layer can be written as:

$$
\begin{aligned}
r &= V^{\mathsf{T}} s \\
y &= W^{\mathsf{T}} x \\
z &= y + r \\
m &= U^{\mathsf{T}} z H(z - \theta^A)
\end{aligned}
\tag{1}
$$

The US cell group can comprise neurons for different values of USs such as appetitive and aversive stimuli. To simplify the notation they are all represented in the

vector s. The predefined weight matrix V determines what actions are triggered by the different states of US. It connects the elements of US to specific elements of IS and thus via the motor map MM sets specific actions. W describes the association of CS to IS and is subject to learning. The activity of the IS is the sum of the two contributions from the US and the CS. The activity m of the motor population MM, is determined by the predefined weight matrix U and the thresholded activity of the IS cell group where θ^A is the threshold and $H(.)$ is the Heaviside function. $H(x)$ is 1 if $x \geq 0$ and 0 if $x < 0$.

The weight matrix W is subject to learning and changes following a learning rule called predictive Hebbian learning where the changes in associations depend on the difference between actual CS x and the estimated CS e, defined as $e = Wz$ where $z = y + r$ [77]. e is the recurrent estimate calculated by the backwards projection of z to the CS cell group. e will be referred to as CS prototype and can be seen as a generalized estimation of the CS activity given the IS activity. With these definitions W changes as:

$$\Delta W = \eta(x - \gamma e)z^{\mathsf{T}} \tag{2}$$

The predictive Hebbian learning rule of DAC directly captures the central role of prediction in the acquisition of stimulus-stimulus and stimulus-response associations that was identified by Rescorla and Wagner [54]. Learning is driven by the correlative term xz^{T} that contains both the auto-correlation of the CS i.e. xx^{T} and the correlation between the CS and the US, i.e. xr^{T}. The auto-correlation term xx^{T} relates to perceptual learning as it maximizes the explained variance [46]. The correlation term xr^{T} relates to behavioral learning and drives the associative learning between the CS and the US. In this way predictive Hebbian learning unifies perceptual and behavioral learning in a single neuronal network. ΔW is small when x and e are similar, i.e. when the estimate e of the CS approximates the activity x in the CS. The parameter η is the learning rate. The parameter γ is a gain factor determining the influence of the prediction term on learning. γ allows to control the norm of the weight matrix and ultimately the amplitude of the activity y in the IS cell group. In this way, the adaptive layer fulfills its twofold task of learning the sensory motor associations and forming internal representations, i.e. the prototypes e for the planning system of the contextual layer.

2.2 Contextual Layer

The contextual layer provides mechanisms for memorizing and recall of behavioral sequences. It comprises two memory structures: a short-term memory (STM), and a long-term memory (LTM) for the permanent storage of information (see Fig. 3). They allow the system to acquire, retain and express sequences of the sensory-motor contingencies the adaptive layer generates. The interaction of the two memory structures is a double process of storing and recall based on the following assumptions [74]:

- Memorize
  - Salient sensory-motor events generated by the adaptive layer are stored in STM
  - The content of the STM is stored in LTM when a goal state is reached
- Recall
  - The content of the LTM is matched against ongoing estimations of sensory events based on their similarity and biased by chaining rules in the memory
  - Matching LTM elements bias action selection in the motor population *MM*

The STM is implemented as a ring buffer and has a fixed size N_S. An element in the memory is called a segment and contains representations of sensory-motor contingencies. A series of consecutive segments is called a sequence. At each moment the generated *CS* prototype e and the action m executed by the agent is stored in the STM. When the agent reaches a goal state, e.g. collision or target, the full content of the STM is retained in the LTM as a sequence and the STM is reset. The LTM contains a maximal number of sequences N_L. The different sequences are value labeled by the different goal states.

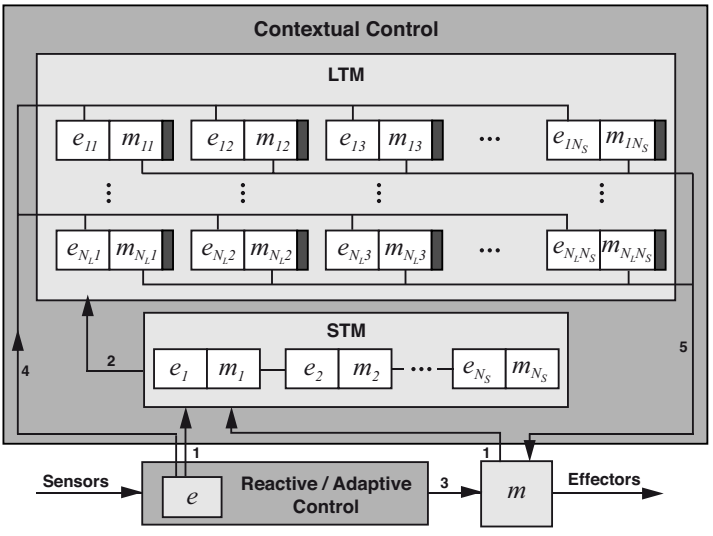


Fig. 3 Contextual layer of DAC: (1) The CS prototype e and the *MM* activity m are acquired in the STM buffer and stored as a segment if the discrepancy D falls bellow a defined threshold. (2) If a goal state occurs, the content of STM is retained in LTM as a sequence. (3) The *MM* population receives input from the *IS* populations according to the rules of the adaptive control structure. (4) If the input of the *IS* population to the *MM* is sub-threshold, the values of the current *CS* prototypes are matched against those stored in LTM. (5) The *MM* population receives the motor response calculated as a weighted sum over the memory segments as input.

The contextual layer relies on the representations formed at the level of the adaptive layer and is activated only when on average the *CS* prototype e approximates the *CS* x. This transition is controlled by an internally generated discrepancy measure D that is a running average of the actual difference between the *CS* x and the prototype e defined as:

$$d(x,e) = \frac{1}{N}\sum_{j=1}^{N}\left|\frac{x_j}{\max(x)} - \frac{e_j}{\max(e)}\right| \tag{3}$$

Initially, DAC operates only with the reactive and adaptive layer. The contextual layer is enabled when the discrepancy measure D falls bellow a certain confidence interval.

During recall the generated prototypes are matched against those stored in LTM. For every segment the so-called collector value c is calculated. c determines the contribution of the different segments to the action of the contextual layer. The collector value is determined by the distance $d(.)$ of the current prototype to the prototype stored in the segment and by a trigger value t. The trigger enables chaining through a sequence and depends on the past activation of neighboring segments earlier in the sequence. For segment l of sequence q the collector value is defined by:

$$c_{lq} = (1 - d(e, e_{lq})t_{lq}) \tag{4}$$

The default value for the trigger is 1, and does not bias the collector value. If the previous segment $l-1$ of sequence q, is activated, the value of the trigger t_{lq} is set to a value lower than 1 and relaxes asymptotically to 1 with a defined time constant. This biasing of the distance measure of the collector value allows to chain through the sequence.

The actual action of the contextual layer is a weighted sum over all the segments whose collector value c_{lq} surpasses a certain threshold θ^C and is calculated as:

$$m = \sum_{l,q \in LTM} \pm \frac{c_{lq} H(c_{lq} - \theta^C)}{\delta_{lq}} m_{lq} \tag{5}$$

where δ_{lq} is the distance measured in segments, between segment l to the end of its sequence, i.e. the distance to the goal state. By dividing the output of segment with the distance to the goal state, the segments closer to the goal state have a higher impact on the contextual response. The sign is plus if the segment belongs to a target sequence and a minus when it belongs to a collision sequence. After updating their input, the motor units compete in a WTA fashion. The winning unit will induce its corresponding motor action. The trigger value of the segments following the segments contributing to this action will be set to a low value enabling chaining. In case that the motor population does not receive any input the robot falls back into exploratory behavior.

3 Results

We investigate the DAC architecture following two complementary approaches. On the one hand we analyze the behavioral performance as well as the analytical predictions of the proposed system (see Sects. 3.1 and 3.2). On the other hand we investigate the neuronal substrates of different sub-components of DAC in biophysically detailed models. In particular we investigate the neuronal substrates underlying the reactive and adaptive layer (see Sects. 3.3, 3.4 and 3.5).

3.1 Behavioral Feedback

In the theoretical analysis of behavioral control, perception and behavior are usually seen as two separated processes. The perceptual learning process constructs compact representations of sensory events while the behavioral learning process is making the association between perceptual representations and actions and organizing them through reinforcement. In this view, the interaction between these two processes is assumed to take place internal to the agent through its neural substrate [63]. However, as already described above, perceptual and behavioral learning are tightly interlinked [84]. In order to associate the correct actions to different percepts, an agent has to perceive and analyze the current situation it is in. The way it perceives a situation does however depend on its past experience [78]. The precise mechanism of this interaction is not clear, mainly because it requires a detailed analysis of both the behavior and the corresponding neuronal substrate. The use of mobile robots allowed us to bypass this restriction as all the internal states are accessible. Here we show how in a robot foraging task, perceptual and behavioral learning interact synergetically via the environment [78].

We simulated a robot foraging task in an environment comprising lights as targets, walls as obstacles and colored patches on the floor. The task of the robot is to maximize the number of targets reached while minimizing the number of collisions. To investigate the influence of the contextual layer we compared two experimental conditions: one where the contextual layer is activated when the discrepancy D falls bellow the defined threshold (see Sect. 2.2) and one where the contextual layer is disabled. We distinguish two different phases in the experiment, a stimulation phase where the lights (US) are turned on (2000 time steps) and a recall phase (5000 time steps) where the lights are turned off and the robot has to rely on the conditioned stimulus (CS) alone in order to reach the targets (Fig. 4 A). We found that the adaptive layer improved the performance of the robot through a learning-dependent avoidance of collisions, observable in the increase of the target/collision ratio (Fig. 4 B). We also observed that at the onset of the second stimulation period, the performance of the two conditions diverges: in the enabled condition, performance is strongly enhanced compared with the disabled condition. This difference is due to the activation of the contextual control layer in the enabled condition, as can be deduced from the evolution of the discrepancy measure D (Fig. 4 C) shortly after the onset of the second stimulation period, D falls below the transition threshold. During the second stimulation and recall periods, the D value of the enabled

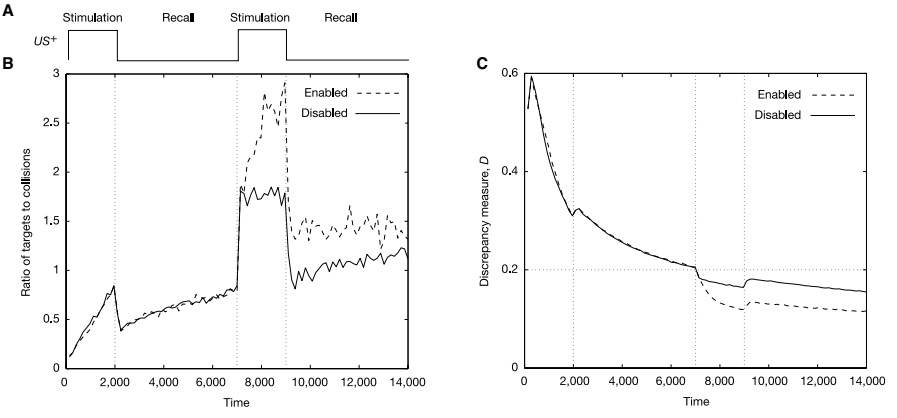


Fig. 4 Performance and input variability for the simulated robot experiment. **A** The experiment consisted in two cycles with one stimulation period (light on) and one recall period (light off) each. **B** Targets/collisions ratio in time windows of 100 time steps for the conditions where the contextual layer was disabled and enabled. **C** Discrepancy value for disabled and enabled condition. The transition threshold is 0.2 (from [78]).

condition is markedly below that of the disabled condition. This reduction is accompanied by a significantly lower value of the average absolute change in synaptic efficacies of the connections W between the CS and IS population. Hence, the transition to contextual control leads to a reduction of the discrepancy between predicted and actual CS events and to a stabilization of the synaptic weights of the adaptive control layer. However, our model has no internal feedback from the contextual to the adaptive control layer. Therefore, this difference must be due to the difference in the overt behavior generated in the two conditions and the systematic bias in the sampling of CS events that this difference causes, that is, behavioral feedback. The behavior is less variable when the contextual layer is enabled, thereby reducing the variability of the sampled sensory inputs [78]. We tested this hypothesis by comparing the entropies of behavior and sampled stimuli and found that both the behavioral entropy (positions visited by the robot) and the perceptual entropy (sampled CS events) are significantly lower for the condition where the contextual layer is activated. This shows that the structuring of the behavior due to behavioral control leads to a smaller set of input states. These results were also reproduced with a robot in a real world environment, demonstrating the effect of behavioral feedback on perceptual learning [78].

To evaluate how behavioral feedback affects performance, we run a control experiment where we compared an enabled condition with a "static" condition. In the static condition the synaptic efficacies W of the adaptive layer was switched off after the activation of the contextual layer. The performance of the robot was lower for the static condition showing that behavioral feedback directly enhances performance [78].

Theses results show that learning-dependent changes in behavior can establish a macroscopic feedback loop. The activation of the contextual layer leads to a more structured and planned behavior and the resulting restricted trajectories, consequently, reduce perceptual variability, stabilizing the behavioral patterns. Hence, this non-neuronal, environmentally mediated feedback organizes the behavior and perception through a synergistic interaction. This suggests that changes in the afferent input to sensory areas, due to behavioral feedback, can systematically change perceptual learning. This is supported by the observation that, during development, the organization and response properties of primary sensory areas can be strongly influenced by their afferent inputs [64].

3.2 DAC as an Approximation of an Optimal Bayesian Decision Making System

To unify the different views of traditional and new AI we build on the assumption that a knowledge level description of intelligence, including the principle of rationality, can be captured in the perspective of Bayesian decision making [4]. Here we summarize the argument that DAC is equivalent to an optimal decision making system in a Bayesian sense. Most importantly we show that our solution is self-contained in the sense that DAC acquires and updates its own set of prior hypotheses. This is relevant since, as traditional AI, also a Bayesian framework does not automatically solve the symbol grounding problem: It also assumes that the knowledge of a decision making system is defined "a priori". In the Bayesian case knowledge is defined by a set of prior hypotheses h and the theorem of inverse probability defines the probability that hypothesis h is true given observation o:

$$p(h|o) = \frac{p(o|h)p(h)}{p(o)}$$

where $p(o)$ is the probability of making observation o, $p(h)$ the prior probability of h being true, and $p(o|h)$ the prior probability that making observation o given h is true. The optimal action, m, can be calculated using a score function $G_g(h_n, m)$ that defines the expected gain, $\langle g \rangle_m$, of performing action m given hypothesis h_n. Bayes principle states that optimal decision making requires that the action m_* is selected, which maximizes the expectancy $\langle g \rangle$:

$$\langle g \rangle_{m_*} = \sum_{h_n \in H} p(h_n|o) G_g(h_n, m_*) = \max_{m_k \in M} [\sum_{h_n \in H} p(h_n|o) G_g(h_n, m_k)]$$

Now we phrase the foraging tasks performed with the DAC architecture in these Bayesian terms. By doing so we prove that the DAC architecture will execute exactly those actions that are optimal in a Bayesian sense (for details on this prove see [74]).

We start by identifying analogs of the inverse probability formula to the corresponding values in the DAC contextual layer. $p(h)$ is the probability of the hypothesis of the agent (its prior knowledge) about a target being reachable (or a collision suffered) when executing an action at a certain time step. This probability in DAC

will be non-zero if we actually have experienced that and have stored it in LTM. $p(o|h)$ adds the probability of observing o at a certain time step. This probability is non-zero if there is a sequence in LTM containing the action that h specifies. Each LTM segment also contains an observation, so this probability will be equivalent to a measure of similarity of prototypes of observation and the corresponding observations stored in the segment. The activity collector unit can play this role. $p(o)$ is a normalization factor so we can discard it. With these two analogs of the probability distributions to DAC memory structures, we have reformulated the expression $\langle g \rangle_{m_*}$, referring to the maximum expectation of an action in Bayesian terms (see [74] for details). We are only missing the gain/score function $G(h_n, m_k)$, which indicates the profit of the hypothesis h_n being true and action m_k executed. This profit can be defined inversely proportional to the time steps necessary to reach the goal state that h_n specifies. This reformulation in Bayesian terms proves that DAC will select the same action as optimal Bayesian decision making. Thus the action selection follows Bayesian rationality based on representations and priors formed while interacting in the environment.

3.3 The Reactive Layer and the Construction of a Synthetic Insect

The current reactive layer implements approach and avoidance responses. This limited set of behaviors might be sufficient and the right choice for an open arena foraging task. In the general case the set of behaviors provided by the reactive layer has to be matched to a specific task (see also [42, 25]). This is especially important as the basic behaviors provided by the reactive layer generate the sensory and motor inputs for the adaptive and contextual layer. To investigate how a set of more advanced stereotyped behaviors can lead to purposeful behavior we have investigated insect navigation, in particular exploration and homing. Leaving the dwelling, moving about in the environment, and finding home is a basic requirement for any agent living in the real world, a task that needs to be solved by man and bug. To be able to find the way back, two strategies are used: The animal can memorize its path based on external cues such as visual, auditory, or olfactory landmarks or by integrate over the distance and direction traveled. Commonly the former is referred to as "landmark navigation" (LMN), and the latter is called "path integration" (PI) or "Dead reckoning".

We believe that behavioral models have to be set in a behavioral context, preferably in the real world. In keeping with this paradigm, we embedded our path integration model in a behavioral task, thus expanding the conceptual model of the reactive layer to a synthetic insect system. The task of the synthetic insect system is to explore the environment, while searching for the target stimulus, and to return to the point of departure. The switching from exploratory to return behavior happens if either the duration of the exploration exceeds a given threshold or the target stimulus was found. Hence, as a behavioral model, the synthetic insect system spawns the complete nexus of information processing from the input stage, to the perceptive and integrative components, to the generation of behaviors, to the output stage.

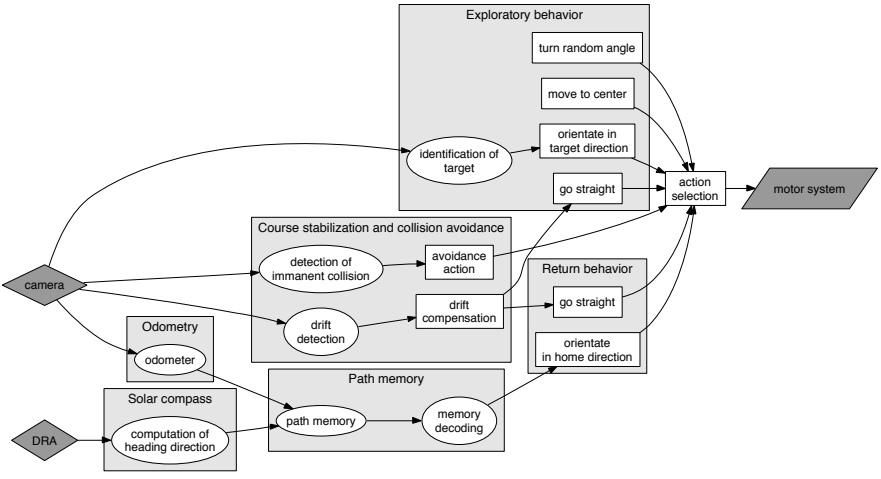


Fig. 5 Grand overview over the components and the flow of information in the synthetic insect system. Light gray boxes demarcate subsystems, whereas diamond shapes represent the input stage, ellipses indicate perceptive and integrative components, and boxes stand for actions. DRA = dorsal rim area.

Although the task outlined might seem simple, in moving from a conceptual path integration model to a synthetic insect system that is behaving in the real world, the number of components that need to be incorporated increases markedly. The system is organized in a number of subsystems (Fig. 5).

The core of the synthetic insect system is the **path integration** sub-system. To integrate the path of a journey, information about the heading direction, and distance traveled must be stored in a path memory. Evidently, also a mechanism for the readout of the path memory must be available. Path memory and readout are both implemented based on the concept of population code regarding the neurophysiological constraints [22, 9]. The heading direction assessment is based on a model of the solar compass of insects. In insects this solar compass receives input from a specialized area of the eye, referred to as the dorsal rim area (DRA) [28], capable of detecting the polarization pattern of the sky.

The goal in developing the **Exploratory behavior** subsystem was to devise a biologically plausible target search behavior. We realized this by combining a random walk behavior with a visual target detection circuit, and a Braitenberg vehicle inspired target orientation mechanism [10]. The **Return behavior** subsystem is the main "consumer" of the path memory information, and responsible for generating the behavior that effectively makes the agent return to the point of departure. The system is complemented by a subsystem for the detection and avoidance of collisions, and stabilization of the course (see [8] for a more detailed description).

For the system to fulfill its behavioral task, the behaviors and information originating from the subsystems must be orchestrated. When looking at the integration,

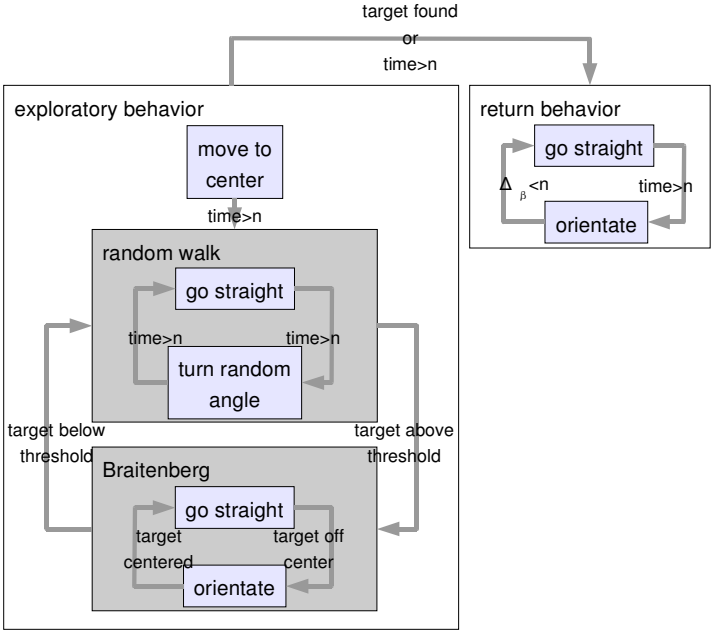


Fig. 6 Behavior regulation in the path integration model. (see text for further explanation).

it is important to realize that a subsystem can be on a purely perceptive and integrative level, or can be implemented as a complete nexus from perception to action, i.e. generate a motor output signal. The former is true for the **Solar compass** and the **Path memory** subsystems, the latter for the **exploratory behavior** and the **return behavior** subsystems. If a subsystem implements the complete nexus, it may stand for the behavior itself. We devised a hierarchical architecture comprising on the one hand a tier of lateral activity exchange, on the other hand of tiers of behavioral reflexes, and motivational "volition" serving to prioritize the actions of the agent. Our action selection circuits employed several generic mechanisms such as gating, cascaded inhibition and convergence, which we predict are commonly found in neuronal systems [62].

The behavioral elements of the synthetic insect system are organized in a nested hierarchy: The **exploratory behavior** comprises the aggregated behaviors **random walk**, and **Braitenberg**, which in turn consist of the atomic behaviors **go straight**, **orientate**, and **turn random angle**. The **return behavior** comprises the atomic actions **go straight**, and **orientate** (Fig. 6). An evasive action is part of the **course stabilization and collision** subsystem [2].

The task of the action selection on the one hand covers the lateral management of the atomic behaviors, i.e. a prioritization and resolution of conflicts between actions into a coherent overall behavior. On the other hand, action selection controls the top-down switching between the complex behaviors. The lateral interaction between behaviors corresponds to reflexes, e.g. ensuring that the collision avoidance reaction

has precedence, whereas the top-down control can be understood as the level of "volition".

Outside the action selection process, **early lateral exchange** of information between the subsystems is required. One of these interactions is the so-called "saccadic" suppression: the input to the course stabilization system is suppressed when the atomic orientate action is causing a "voluntary" rotational movement. If this suppression were absent, the course stabilization system reacts to the visual input caused by the rotation and counteracts to the movement. The **top-down action selection** realizes the "volitional" aspect of switching between exploratory and return behavior. It is "volitional" in that, as opposed to the autonomous lateral control, the switching is flexible and managed explicitly. Top-down action selection manages the external criterion of whether the target stimulus is within the desired distance, and the internal criterion of time spent exploring the environment. The first criterion is represented by the group **Target Found**, and stems from the exploratory behavior subsystem, whereas the second criterion is represented by the group **Exploration timeout**, and is implemented within the action selection process itself.

The synthetic insect system presented here is probably one of the most comprehensive models of an insect built to date. Yet the model is not complete as it lacks components such as circadian rhythms and navigation by the aid of landmarks. In the context of the reactive layer we have used the synthetic insect system as an example of how a set of basic behaviors can be orchestrated to yield advanced behaviors such as exploring and homing. Key to achieving a coherent overall behavior are lateral management of basic behaviors, i.e. a prioritization and resolution of conflicts, and top-down, "volitional", switching between the more complex behaviors. The integration of a such a rich behavioral set of behaviors in the DAC reactive layer will allow not only to deal with more advanced reactive behaviors but will also improve learning at the level of the adaptive and contextual layer as the reactive layer generates the inputs for learning in these layers. In the context of insect navigation the adaptive and contextual layer can serve for landmark navigation and as such complete the synthetic insect model.

3.4 The 'Two–Stage' Theory of Classical Conditioning

Conditioning as described in the adaptive layer of DAC involves both a perceptual and a behavioral learning component. The adaptive layer forms internal representations of the *CS* biased by the behavioral relevance represented by the US. This can be seen as an abstract model of two-phase conditioning [34]. According to the two–stage theory of classical conditioning, the association between the *CS* and the aversive stimulus (US) is formed within the first few conditioning trials, representing the initial first stage of conditioning, and results in the acquisition of emotional *CR*s. This means that before a *CR* of the skeletal-motor system can be observed, rapidly–developing *CR*s concerning the heart rate, respiration, blood pressure, pupillary size or skin conductance have already been acquired. As these *CR*s all develop regardless of the locus or type of US, they have been called *non–specific*. After the context

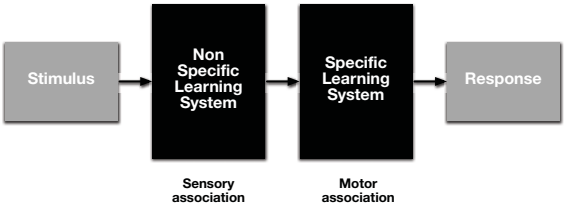


Fig. 7 Illustration of the two-phase theory of classical conditioning.

between the *CS* and *US* has been formed through stimulus–stimulus associations, the second stage, namely stimulus–response associations (S-R), is responsible for forming the specific somatic motor response that are directed to a specific *US*. Within the DAC adaptive layer the two phases are interwoven in a single system and the interaction between the two is regulated through the predictive Hebbian learning rule (Eq. 2). To understand more explicitly the interaction between the two stages we have developed a biophysically detailed model of both the first and the second stage of classical conditioning i.e. auditory cortex and cerebellum. After reviewing the two separate models we suggest how this to connect those two stages together providing the first complete account of the two phase theory of conditioning (see Fig. 7).

We have successfully modeled the roles of the amygdala, basal forebrain and auditory cortex (AC) as an example of the non-specific learning system and the cerebellum (CE) as a model of the specific learning system [58, 72, 27]. Here we provide a complete account of Konorski's proposal by integrating these two systems and thus provide the first complete biologically-grounded computational model of the two-phase theory of conditioning.

In our previous work [58] we chose the AC as a model of the non-specific learning system. In particular, it has been shown that classical fear conditioning specifically retunes the receptive fields (RFs) in the primary auditory cortex to favor the processing of the frequency which was used as the *CS*. Specifically, [3] observed a shift of the characteristic frequency of the auditory neurons towards the frequency of the *CS*. These receptive field changes rapidly appear [17] and the AC neurons continue to increase their responses to the frequency of the *CS* even in the absence of further training [21]. In summary, the RFs in the primary auditory cortex of normal, adult animals are not fixed but are modified by learning. This plasticity is sufficient to change frequency tuning to favor the processing of the conditioned stimulus. The acquired behavioral importance of the *CS*–frequency is represented by an increase in tuning and can be seen through a clear expansion of the region of the cortex that represents frequencies similar to the conditioned stimulus [82].

AC learning has been simulated using a biophysically realistic learning rule, where neurons plasticity is controlled locally by a spike-time dependent learning rule (STDP). The occurrence of an unconditioned stimulus (*US*), signaled by a burst of activity in the model amygdala/nucleus basalis, switches a bigger fraction of synapses in the A1 into a plastic mode. As a result *CS* representations are enlarged, thus resulting in a higher global response to a *CS* stimulus [58]. The simulation

protocol consists in the repeated presentation of five tones to train the learning AC model. Once the training is over, different portions of the AC represent the different frequencies of the presented tones. Afterwards, the tones are repeated again, but the fourth tone is followed by an aversive stimulus. Indeed, after few trials, the portion of AC representing the fourth tone considerably increases.

For the specific leaning we investigated an eye-blink conditioning paradigm. Our cerebellar model [72, 27] is able to associate a *CS* to a *CR* that reduces the impact of the aversive stimulus. Moreover, the *CR*s that are produced are well-timed, correctly anticipating the aversive *US*. The neural mechanism responsible for the association between *CS* and *US* is the plasticity at the parallel fibers (carrying the *CS* signal) to purkinje cells. Such plasticity is controlled by the learning modulatory effect of the climbing fiber action (carrying the *US* signal). The modulation of the synaptic strength at the parallel fibers prukinje cell synapse allows to control whether a given *CS* will produce a *CR* and the time separating the *CS* to the *CR*. The adaptive timing of the response, i.e. the fact that the *CR* correctly anticipates the *US*, is controlled by a negative feedback mechanism that inhibits the climbing fiber signal whenever a well-timed conditioned response is generated. For a detailed explanation see [72].

The next step is to combine the two models in a single behaving system. The non-specific learning stage is able to filter out non relevant stimuli by enhancing the stimuli that are related to the *CS*. A gating mechanism should be sufficient to separate *CS*–related from unrelated neural activity. Once this activity has been relayed to the cerebellum, other activity not related to the *CS* should be silenced as much as possible in order to facilitate stable learning in the cerebellum. Such a system provides on the one hand a complete computational account of Konorsky's two-phase theory of classical conditioning, and on the other hand, demonstrates that the interplay between enhanced perception and adaptation is fundamental. The integration of these detailed biophysical models in the DAC architecture will allow us to test how they perform in a more complex foraging task and how the representations formed in the first phase can be utilized in the contextual layer for sequence learning and planning.

3.5 General Principals for Perceptual Learning

The perceptual learning component of the adaptive layer provides behaviorally biased feature extraction of the states of the distal sensors. However, the linear nature of the learning rule (Eq. 2) will only allow the extraction of features based on second order statistics. As a means to support complex behavior in a real-world scenario it is necessary to consider the complexity and the richness of natural stimuli in the sensory representation involving higher order statistics. As an example, visual stimuli can range from blinking monochromatic lights to three-dimensional objects and human faces while sounds can range from a sinusoidal tone to a human voice. Each of these inputs requires specialized processing with different degrees of complexity. The human brain has evolved to be highly adapted to these high order properties. For this reason we explored brain based higher order statistical optimization to understand and advance the feature extraction capabilities of the adaptive layer.

Objective functions optimization is a general approach for higher order statistical learning. By means of unsupervised learning methods, a statistically defined objective is used to optimize a neural network response to express a desired firing profile. Using this approach, it has been shown that optimally sparse representations resembling so-called simple cells [48, 37] and optimally stable representations giving rise to complex cells [80, 18, 83] can be learned from visual stimulus consistent with physiology of the primary visual cortex V1. With this procedure it has been possible to generate artificial neural structures with similar receptive fields of visual V1 cells [48] and auditory nerve cells [15]. The theory of general computational principles is not only functionally coherent with the brain but it is also in agreement with cortical organization. Given the relative uniform structure of the neuronal substrate underlying perception, cognition and behavior one can assume that also the number of computational principles should be fairly restricted. For instance, this would suggest that all proprieties of the visual system of the cerebral cortex from the restricted local tunings in V1 to the invariant representation of space found in the place cells of the hippocampus can emerge from very few principles. Indeed, many physiological features of the cerebral cortex have been modeled following this approach.

We have investigated how an objective function approach can help for feature extraction and the formation of internal representations in a foraging robot. A mobile robot performing a Braitenberg like behavior with a CCD camera mounted on its top was embedded in a rectangular arena (Fig. 8 A). The output of the camera was its unique input connected to a neural model (Fig. 8 B). The processing hierarchy was composed of five-layers of leaky integrator units, providing them with a local transient memory, with both inhibitory intra-area and excitatory feed-forward inter-area connections. The convergence of the feed-forward connectivity and the temporal integration constant increase while moving up in the hierarchy. The objective function is evaluated autonomously at each layer. The weights of the feed-forward connections are updated by an online unsupervised learning algorithm (gradient descent) guided by the objective function.

In this model the three computational principles used are: temporal stability, decorrelation and activity regulation. Temporal stability refers to the variation of the activity in time. The main concept behind this principle is that high-level representations are invariant to fast-varying features. Therefore the profile of the activity must follow a behavioral time scale, this is usually slower than a neural time scale, i.e., hundreds of milliseconds compared to milliseconds. Decorrelation refers to the relation of the activity of different units in one network. Different cells are expected to respond to different stimuli to minimize redundancy and to avoid the convergence of all units to a single salient feature of the input. The last principle seeks the regulation of activity to have the most energy saving representation.

After a training period the network converged to express stable values for the objective functions. The higher layers converged only after earlier layers had converged showing the bootstrapping nature of the model. This indicates that to obtain high-level representations a stable pre-processing of the input is needed. In addition, the model showed that the first layer had strong spatial frequency tuning, as observed in V1. Moreover, the output response of the highest layer when

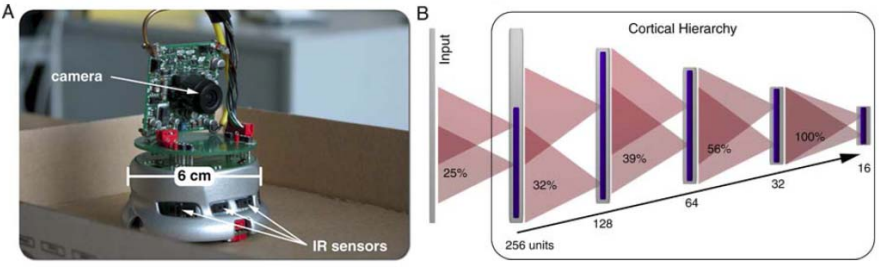


Fig. 8 The Micro-Robot Khepera and the Neural Network Structure Used for Sensory Processing. **A** camera mounted on top of the cylindrical body provides the visual input. The infra-red (IR) sensors are used for obstacle avoidance during exploration of a real-world office environment. **B** Diagram showing the hierarchical network comprising five levels of identical computational units. Units are arranged uniformly within a two-dimensional square lattice, and their number per level decreases with a constant factor of 0.5 moving up the hierarchy. Each efferent unit receives input from a topographically aligned square region within the afferent level and connects laterally to all the units in the same level with which it shares feed-forward input. The average relative size of a units feed-forward arbor within the afferent level (as given in percentages), and consequently also the lateral degree of connectivity, increases with the hierarchical level and reaches 100% for the units at the highest level. The input to the network has a resolution of 16x16 pixels. [86]

plotted over the robot position (obtained by a tracking system) revealed place fields as observed in the hippocampal place cells. This indicates that as its biological counterpart this layer was representing the position of the robot. The results also showed that this position representation is invariant on agent orientation. These experiments show that very few computational principles can drive perceptual learning to emerge high-level representations from complex stimuli.

The objective function approach shows how perceptual structures can be acquired. However, another problem faced by a real-world perceptual system is speed. Humans, for example are able to detect an animal in a previously unseen image within as little as 150 ms [67]. To address this issue we investigated a temporal coding scheme in the form of the so-called Temporal Population Code (TPC) [85]. TPC relies on a recurrent connected network to generate temporal codes for the invariant identification of perceptual inputs [85]. The network consists of a two dimensional array of conductance-based leaky integrate-and-fire neurons (Fig. 9). Each neuron is connected to a circular neighborhood with synapses of equal strength, modeled as instantaneous excitatory connections, whereas transmission delays are related to the Euclidean distance between the positions of the pre-and postsynaptic neurons. The stimuli are presented continuously to the network first passing through an edge-detection stage. The resulting contours are projected topographically onto the array of neurons [87]. The spatially integrated activity of all cells are projected, as a sum of their action potentials, onto the readout circuit resulting on the so called Temporal Population Code or TPC. This representation is position- and rotation-invariant and robust to stimulus variability with an encoding speed consistent to

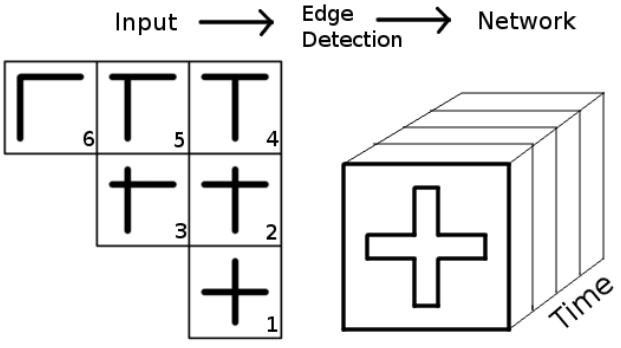


Fig. 9 The stimuli are composed by two solid bars of same length intersecting at different points for each of the individual classes. After passing over an edge-detection stage, the resulting contour is projected topographically onto a map of cortical neurons. Because of the lateral intra-cortical interactions, the stimulus becomes encoded in the networks activity trace.

physiological data [68]. We tested TPC on handwritten digits of a standard benchmark set in the domain of character recognition. The model was capable of classifying 94.8 % of the stimuli correctly, showing the potential of the temporal population coding in clustering different classes of objects with a realistic degree of variability.

Objective functions and TPC are complementary approaches to perception. Integrated within the DAC architecture, they allow the extraction of high order statistical features and the invariant identification of features. The generality of these methods and the independence from the input modality allows a wide range of applications.

4 Discussion

In this chapter we have shown how perceptual and behavioral learning is integrated in the DAC architecture, a self-contained cognitive system. This integration not only generates structured and successful behavior in different foraging tasks but does also allow to investigate the complex nature of the interaction between perceptual and behavioral learning. In addition we have shown how key parts of DAC have been mapped to the neuronal substrates underlying the reactive and adaptive layer. The integration of theses detailed models in the DAC architecture do not only enrich the behavioral capabilities of DAC but also allow to study the interaction between the different systems.

DAC is a proposal on the neuronal organization of adaptive goal oriented behavior. Different cognitive architectures have been proposed (see [71] for a review). DAC is unique by providing a self-contained learning system that demonstrates how perception, problem solving and behavioral control can be understood in strictly bottom-up terms. A reactive control layer which uses only prewired reflexes is equipped with a minimal behavioral competence to deal with its environment. The adaptive control layer associates this reactive behavior to distal sensory inputs and forms internal representations of the environment. At the level of contextual control these

representations are used to form more complex representations in order to express relationships of sensory and motor events over time. It has been shown that the contextual control structure uses these representations in an optimal Bayesian way to achieve its goals in simulated and real foraging tasks. The learning model is self-contained in the sense that the prior and conditional probabilities are acquired through interaction with the world and are continuously updated in relation to the experience of the agent, by changes in the classification of sensory events at the adaptive layer or the formation of new sequences by the contextual layer. A key difference between DAC and traditional rational systems is that the former becomes rational due to its continuous interaction with the world while the latter are rational as a result of prior specification. The symbols DAC is integrating in its optimal decision making are acquired and not predefined. An important consequence of this is that where the rationality of traditional systems is bounded by the logical closure of their predefined world models that of the DAC architecture is bounded by the complexity of the real world in which it operates and the direct interfaces it possesses.

How to acquire these representations is a key problem in the unification of traditional and new AI. Note, that the internal representations formed at the level of the adaptive layer could also serve as a basis for more traditional AI reasoning systems [1, 36] or for localization and mapping methods in autonomous robots [43, 40]. The adaptive layer combines perceptual and behavioral learning in a single system. This facilitates the formation of internal representations that do not only capture the statistical properties of the perceptual input but are also behaviorally relevant. The learning rule presented in equation 2 only captures linear relationships. To overcome this limitation we have investigated objective function optimization methods. We showed that such an approach yields invariant and stable representations of the environment. Complementary to the firing rate coding of the objective functions we have proposed a temporal population code (TPC). With the TPC we were able to identify entities independent on position and rotation. However these two methods are only driven by the statistics of the inputs. The behavioral relevance is not taken into account. In the two-phase model of conditioning we investigate how behavioral learning influences perceptual learning. As a result we find that the two-phase model of conditioning can be exploited as a behavioral relevance filter enhancing the behavioral relevant and suppress irrelevant features. In combination this two methods allow the adaptive layer to form stable and rich internal representations that can be manipulated in the contextual layer. In such a system, combining perceptual, cognitive and behavioral learning behavioral feedback is inevitable. The use of robots allowed us to show how behavioral feedback synergetically structures perception and behavior. This accentuates the necessity to study artificial intelligent systems in an embodied and grounded context.

Acknowledgement

This work was supported by the EU projects Synthetic Forager-SF (EC-FP7-217148) and PRESENCCIA (IST-2006-27731). Andrea Giovannucci is funded by a Juan De La Cierva contract (JCI-2008-03006).

References

1. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. Psychol. Rev. 111(4), 1036–1060 (2004)
2. Bermudez i Badia, S., Pyk, P., Verschure, P.F.M.J.: A fly-locust based neuronal control system applied to an unmanned aerial vehicle: the invertebrate neuronal principles for course stabilization, altitude control and collision avoidance. Int. J. Robot Res. 26, 759–772 (2007)
3. Bakin, J.S., Weinberger, N.M.: Classical conditioning induces CS-specific receptive field plasticity in the auditory cortex of the guinea pig. Brain Res. 536(1-2), 271–286 (1990)
4. Bayes, M., Price, M.: An essay towards solving a problem in the doctrine of chances. Philos. Trans. R Soc. London 53, 370–418 (1763)
5. Becker, S., Plumbley, M.: Unsupervised neural network learning procedures for feature extraction and classification. Appl. Intell. 6(3), 185–203 (1996)
6. Bell, A.J.: Levels and loops: the future of artificial intelligence and neuroscience. Philos. Trans. R Soc. Lond B Biol. Sci. 354(1392), 2013–2020 (1999)
7. Berlau, K.M., Weinberger, N.M.: Learning strategy determines auditory cortical plasticity. Neurobiol. Learn. Mem. 89(2), 153–166 (2008)
8. Bernardet, U.: The neurobiological basis of perception and behavior: the iqr large-scale neuronal system simulator and its application. Ph.D. thesis, University of Zurich (2007)
9. Bernardet, U., Bermúdez i Badia, S., Verschure, P.F.M.J.: A model for the neuronal substrate of dead reckoning and memory in arthropods: a comparative computational and behavioral study. Theory Biosci. 127(2) (2008)
10. Braitenberg, V.: Vehicles, experiments in synthetic psychology. MIT Press, Cambridge (1984)
11. Brooks, R.: Intelligence without representation. Artif. Intell. 47(991), 139–159 (1991)
12. Brooks, R.: New approaches to robotics. Science 253(5025), 1227–1232 (1991)
13. Clancey, W.: Situated Cognition: On human knowledge and computer representations. Cambridge University Press, Cambridge (1996)
14. Davis, H.: Underestimating the rat's intelligence. Brain Res. Cogn. Brain Res. 3(3-4), 291–298 (1996)
15. Duff, A., Wyss, R., Verschure, P.F.M.J.: Learning temporally stable representations from natural sounds: Temporal stability as a general objective underlying sensory processing. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4669, pp. 129–138. Springer, Heidelberg (2007)
16. Edeline, J.M.: Learning-induced physiological plasticity in the thalamo-cortical sensory systems: a critical evaluation of receptive field plasticity, map changes and their potential mechanisms. Prog. Neurobiol. 57(2), 165–224 (1999)
17. Edeline, J.M., Pham, P., Weinberger, N.M.: Rapid development of learning-induced receptive field plasticity in the auditory cortex. Behav. Neurosci. 107(4), 539–551 (1993)
18. Foldiak, P.: Learning invariance from transformation sequences. Neural Comput. 3(2), 194–200 (1991)
19. Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness lead to place, head-direction, and spatial-view cells. PLoS Comput. Biol. 3(8), e166 (2007)
20. Gallistel, C.R.: The Organization of Learning. MIT Press, Cambridge (1990)
21. Galvan, V.V., Weinberger, N.M.: Long-term consolidation and retention of learning-induced tuning plasticity in the auditory cortex of the guinea pig. Neurobiol. Learn. Mem. 77(1), 78–108 (2002)
22. Georgopoulos, A.: New concepts in generation of movement. Neuron 13, 257–268 (1994)

23. Gibson, J.J.: The Ecological Approach to Visual Perception. Lawrence Erlbaum, New Jersey (1979)
24. Harnad, S.: The symbol grounding problem. Physica D 42(1-3), 335–346 (1990)
25. Herbort, O., Butz, M.V., Pedersen, G.: The sure reach model for motor learning and control of a redundant arm: from modeling human behavior to applications in robots. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 85–106. Springer, Heidelberg (2010)
26. Hipp, J., Einhäuser, W., Conradt, J., König, P.: Learning of somatosensory representations for texture discrimination using a temporal coherence principle. Network 16(2-3), 223–238 (2005)
27. Hofstötter, C., Mintz, M., Verschure, P.F.M.J.: The cerebellum in action: a simulation and robotics study. Eur. J. Neurosci. 16(7), 1361–1376 (2002)
28. Homberg, U.: In search of the sky compass in the insect brain. Naturwissenschaften 91, 199–208 (2004)
29. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. U S A 79(8), 2554–2558 (1982)
30. Hoyer, P.O., Hyvärinen, A.: A multi-layer sparse coding network learns contour coding from natural images. Vision Res. 42(12), 1593–1605 (2002)
31. Hurri, J., Hyvärinen, A.: Simple-cell-like receptive fields maximize temporal coherence in natural video. Neural Comput. 15(3), 663–691 (2003)
32. Hyvarinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. J. Wiley, New York (2001)
33. Klein, D.J., König, P., Körding, K.P.: Sparse spectrotemporal coding of sounds. Eurasip Jasp 3, 659–667 (2003)
34. Konorski, J.: Integrative activity of the brain: An interdisciplinary approach. University of Chicago Press, Chicago (1967)
35. Körding, K.P., Kayser, C., Einhäuser, W., König, P.: How are complex cell properties adapted to the statistics of natural stimuli? J. Neurophysiol. 91(1), 206–212 (2004)
36. Laird, J.: Using a computer game to develop advanced AI. Computer 34(7), 70–75 (2001)
37. Lewicki, M.S.: Efficient coding of natural sounds. Nat. Neurosci. 5(4), 356–363 (2002)
38. MacDonall, J.S., Goodell, J., Juliano, A.: Momentary maximizing and optimal foraging theories of performance on concurrent VR schedules. Behav. Processes 72(3), 283–299 (2006)
39. Mackintosh, N.J.: Conditioning and associative learning. Oxford psychology series. Clarendon Press, Oxford (1990) (Reprint)
40. Martinez-Cantin, R., de Freitas, N., Brochu, E., Castellanos, J., Doucet, A.: A Bayesian Exploration-Exploitation Approach for Optimal Online Sensing and Planning with a Visually Guided Mobile Robot. Auton Robots (in press, 2009)
41. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Mach. Intell. 4, 463–502 (1969)
42. Mitrovic, D., Klanke, S., Vijayakumar, S.: Adaptive optimal feedback control with learned internal dynamics models. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 65–83. Springer, Heidelberg (2010)
43. Montemerlo, M., Thrun, S.: FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics (Springer Tracts in Advanced Robotics). Springer, New York (2007)
44. Newell, A.: Unified theories of cognition. Harvard University Press, Cambridge (1990)
45. Ohl, F.W., Scheich, H.: Learning-induced plasticity in animal and human auditory cortex. Curr. Opin. Neurobiol. 15(4), 470–477 (2005)

46. Oja, E.: A simplified neuron model as a principal component analyzer. J. Math. Biol. 15(3), 267–273 (1982)
47. Oja, E., Ogawa, H., Wangviwattana, J.: Principal component analysis by homogeneous neural networks, Part I: The weighted subspace criterion. IEICE Trans. Inf. Syst. 75, 366–375 (1992)
48. Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381(6583), 607–609 (1996)
49. O'Regan, J.K., Noe, A.: A sensorimotor account of vision and visual consciousness. Behav. Brain Sci. 24(5), 939–973 (2001)
50. Pavlov, I.P.: Conditioned reflexes: an investigation of the physiological activity of the cerebral cortex. Oxford University Press, Oxford (1927)
51. Pfeifer, R., Scheier, C.: From perception to action: the right direction? In: Gaussier, P., Nicoud, J. (eds.) From Perception to Action Conference, 1994, Proceedings, Los Alamitos, California, pp. 1–11 (1994)
52. Pfeifer, R., Scheier, C.: Understanding Intelligence. MIT Press, Cambridge (1999)
53. Rao, R., Olshausen, B., Lewicki, M.: Probabilistic Models of the Brain: Perception and Neural Function. MIT Press, Cambridge (2002)
54. Rescorla, R., Wagner, A.: A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In: Black, A., Prokasy, W.F. (eds.) Classical Conditioning II: Current Research and Theory, pp. 64–99. Appleton Century Crofts, New York (1972)
55. Roberts, W.: Foraging by rats on a radial maze:learning, memory, and decision rules. In: Gormezano, I., Wasserman, E. (eds.) Learning and memory: The behavioral and biological substrates, pp. 7–24. Lawrence Erlbaum, New Jersey (1992)
56. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychol. Rev. 65(6), 386–408 (1958)
57. Rutkowski, R.G., Weinberger, N.M.: Encoding of learned importance of sound by magnitude of representational area in primary auditory cortex. Proc. Natl. Acad. Sci. U S A 102(38), 13664–13669 (2005)
58. Sanchez-Montanes, M.A., König, P., Verschure, P.F.M.J.: Learning sensory maps with real-world stimuli in real time using a biophysically realistic learning rule. IEEE Trans. Neural Netw. 13(3), 619–632 (2002)
59. Schultz, W.: Behavioral theories and the neurophysiology of reward. Annu. Rev. Psychol. 57, 87–115 (2006)
60. Simoncelli, E.P., Olshausen, B.A.: Natural image statistics and neural representation. Annu. Rev. Neurosci. 24, 1193–1216 (2001)
61. Smith, E.C., Lewicki, M.S.: Efficient auditory coding. Nature 439(7079), 978–982 (2006)
62. Sporns, O., Kötter, R.: Motifs in brain networks. PLoS Biol. 2(11), e369 (2004)
63. Squire, L.R., Kandel, E.R.: Memory: From mind to molecules. Scientific American Library, New York (1999)
64. Sur, M., Leamey, C.A.: Development and plasticity of cortical areas and networks. Nat. Rev. Neurosci. 2(4), 251–262 (2001)
65. Sutton, R., Barto, A.G.: Reinforcement learning: An Introduction. MIT Press, Cambridge (1998)
66. Thorndike, E.: Animal Intelligence. Macmillan, New York (1911)
67. Thorpe, S., Fize, D., Marlot, C.: Speed of processing in the human visual system. Nature 381(6582), 520–522 (1996)
68. Tovee, M.J., Rolls, E.T., Treves, A., Bellis, R.P.: Information encoding and the responses of single neurons in the primate temporal visual cortex. J. Neurophysiol. 70(2), 640–654 (1993)

69. Tversky, A., Slovic, B., Kahneman, B.: Judgment under uncertainty: heuristics and biases. Cambridge University Press, Cambridge (2001)
70. Varela, F., Thompson, E., Rosch, E.: The Embodied Mind: Cognitive Science and Human Experience. MIT Press, Cambridge (1991)
71. Vernon, D., Metta, G., Sandini, G.: A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. IEEE Trans. Evol. Comput. 11(2), 151–180 (2007)
72. Verschure, P., Mintz, M.: A real-time model of the cerebellar circuitry underlying classical conditioning: A combined simulation and robotics study. Neurocomputing 38(40), 1019–1024 (2001)
73. Verschure, P.F.M.J.: Synthetic epistemology: The acquisition, retention, and expression of knowledge in natural and synthetic systems. In: IEEE World Conference on Computational Intelligence, Proceedings, pp. 147–152. Anchorage, Alaska (1998)
74. Verschure, P.F.M.J., Althaus, P.: A real-world rational agent: Unifying old and new AI. Cogn. Sci. 27, 561–590 (2003)
75. Verschure, P.F.M.J., Coolen, A.C.C.: Adaptive fields: distributed representations of classically conditioned associations. Network 2(2), 189–206 (1991)
76. Verschure, P.F.M.J., Krose, B., Pfeifer, R.: Distributed adaptive control: The self-organization of structured behavior. Rob. Auton. Syst. 9, 181–196 (1993)
77. Verschure, P.F.M.J., Pfeifer, R.: Categorization, representations, and the dynamics of system-environment interaction: a case study in autonomous systems. In: Meyer, J.A., Roitblat, H., Wilson, S. (eds.) From Animals to Animats: Proceedings of the Second International Conference on Simulation of Adaptive behavior, Honolulu, Hawaii, pp. 210–217. MIT Press, Cambridge (1992)
78. Verschure, P.F.M.J., Voegtlin, T., Douglas, R.J.: Environmentally mediated synergy between perception and behaviour in mobile robots. Nature 425(6958), 620–624 (2003)
79. Voegtlin, T., Verschure, P.F.M.J.: What can robots tell us about brains? A synthetic approach towards the study of learning and problem solving. Rev. Neurosci. 10(3-4), 291–310 (1999)
80. Wallis, G.: Using spatio-temporal correlations to learn invariant object recognition. Neural Netw. 9(9), 1513–1519 (1996)
81. Weinberger, N.M.: Learning-induced changes of auditory receptive fields. Curr. Opin. Neurobiol. 3(4), 570–577 (1993)
82. Weinberger, N.M.: Physiological memory in primary auditory cortex: characteristics and mechanisms. Neurobiol. Learn. Mem. 70(1-2), 226–251 (1998)
83. Wiskott, L., Sejnowski, T.J.: Slow feature analysis: unsupervised learning of invariances. Neural Comput. 14(4), 715–770 (2002)
84. Wyss, R.: Sensory and motor coding in the organization of behavior. Ph.D. thesis, ETHZ (2003)
85. Wyss, R., König, P., Verschure, P.F.M.J.: Invariant representations of visual patterns in a temporal population code. Proc. Natl. Acad. Sci. U S A 100(1), 324–329 (2003)
86. Wyss, R., König, P., Verschure, P.F.M.J.: A model of the ventral visual system based on temporal stability and local memory. PLoS Biol. 4(5), e120 (2006)
87. Wyss, R., Verschure, P.F.M.J., Konig, P.: Properties of a temporal population code. Rev. Neurosci. 14(1-2), 21–33 (2003)

Proprioception and Imitation: On the Road to Agent Individuation

M. Lagarde, P. Andry, P. Gaussier, S. Boucenna, and L. Hafemeister

Abstract. In this paper, we will show that different kinds of interactive behaviors can emerge according to the kind of proprioceptive function available in a given sensori-motor system. We will study three different examples. In the first one, an internal proprioceptive signal is available for the learning of the visuo-motor coordination between an arm and a camera. An imitation behavior can emerge when the robot's eye focuses on the hand of the experimenter instead of its own hand. The imitative behavior results from the error minimization between the visual signal and the proprioceptive signal. In the second example, we will show that similar modifications of the robot's initial dynamics allows to learn some of the space-time properties of more complex behaviors under the form of a sequence of sensori-motor associations. In the third example, a robot head has to recognize the facial expression of the human caregiver. Yet, the robot has no visual feedback of its own facial expression. The human expressive resonance will allow the robot to select the visual features relevant for a particular facial expression. As a result, after few minutes of interactions, the robot can imitates the facial expression of the human partner. We will show that the different proprioceptive signals used in the examples can be seen as bootstrap mechanisms for more complex interactions. Applied as a crude model of the human, we will propose that these mechanisms play an important role in the process of individuation.

1 Introduction

Our motivation is to understand the key processes of cognitive systems. To do so, we adopt a bottom-up approach : we design perception-action control

P. Gaussier
IUF

M. Lagarde, P. Andry, P. Gaussier, S. Boucenna, and L. Hafemeister
ETIS, ENSEA, Univ Cergy-pontoise, CNRS UMR 8051
F-95000 Cergy-Pontoise
e-mail: {andry,gaussier,lagarde,boucenna,hafemeister}@ensea.fr

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 43–63.
springerlink.com

architectures with on-line learning and categorization abilities, and we study how the combination of low-level mechanisms allows the emergence of higher level abilities. In other words, we are studying how simple sensori-motor loops allow situated and embedded robots to adapt to their physical and social environment.

If in the last few decades studies have shown how simple and elegant architectures allow the emergence of behaviors adapted to the physical environment (Braitenberg's Vehicles [1], the role of the embodiment in adapted dynamics [2], the use of multi-agent dynamics in problem solving [3], or the role of vision in navigation), it seem less clear how emergent behaviors can serve the adaptation of robots to the social environment, especially in the frame of one to one Human-Robot Interaction (HRI). With the recent development of robotics, an important effort has been made to build robots able to interact easily with humans. This effort mainly result in (1) researches trying to enhance the robot's control architecture in order to detect and take into account the other (from speech recognition, to the addition of special primitives processing face/body detection and framing and/or emotion recognition [4]) and (2) technics used to enhance the expressiveness and the aspect of the robot [5, 6]. (1) is dedicated to enhance the robot's ability to interact and (2) is dedicated to enhance the human acceptation of the robot, and both approaches intend to facilitate the exchanges between humans and robots. Putted together, (1) and (2) are used to frame the interaction, by giving appropriate but also a priori information explicitly defining for the robot what another "agent" is (processing speech, seeking for a given model of face, detecting a given skin tone or a pre-defined emotional pattern), and how the human can accept the robot as an interactive agent. Nevertheless, as underlined by decades of studies in developmental psychology, understanding the roots of communication is still a major issue [7, 8, 9], and we still not know what are the right "properties" that a system must embed in order to detect, exchange, take turn easily as the young infants do, even at a pre-verbal stage [10, 11, 12]. For our research, the question turns to be : how can adaptive systems build autonomously the notion of what an "agent" and what "self" are? Is it possible, starting from minimal mechanisms but without notion of self or of the other, to obtain systems where these notion can emerge with a progressive individuation from the social and physical environment? Of course, it is not here question of consciousness nor any complex notion of agency, but rather to be able to design a minimalist system allowing to explain how collaborative functions can emerge from simple non-verbal interactions. On the other hand this issue is also close to the following question: If we build a system that have no notion of self or of the other, how far can we go in the learning of complex tasks, or complex interactions?

In this study, our starting point is always a simple perception-action system initially designed as a homeostatic machine, without embedding pre-defined "interactive functions". Such systems have a very poor dynamics, which consist to satisfy the homeostatic principle : to maintain an equilibrium between

the different modalities (vision, proprioception, etc...). They try to regulate (using their actions and their learning properties) the sensori-motor information. Interestingly, if the human is introduced in the robot's environment (and therefore modifies by his/her action the robot's perceptions) it will simply induce in some cases, the emergence of new behaviors adapted to interactive situations.

To explain our approach, we will give three examples, where the architectures are initially not designed to interact with others. They are only designed to learn new sensory-motor associations when detecting changes in self dynamics (imbalances). As an introduction, we will briefly explain the two first examples : (1) how a low-level imitative behavior can emerge when a human is acting in front of a simple eye-arm system controlled by an homeostatic controller, thanks to the perception ambiguity. Imitation is here a side effect of the human moves, actively modifying the behavioral dynamics of the robot. Then, (2) how the modifications of the initial dynamics of a robot can be used to learn sensori-motor associations. Here, the role of the human imitative behavior allow the learning and recognition of human expressions by the robot without any a priori or pre-defined model of the other.

From theses two first examples, we wish to underline the fact that just by modifying the interaction dynamics of the hemostat, we can get for free low level imitation or expression recognition: what would have been the prerequisite of classical HRI studies appear here to be the emergent result of an unconstrained interaction with an adaptive system.

In the third part of this chapter (3), we will describe more precisely a third interactive situation, in order to tackle the issue of task learning with an autonomous robot. Here, we wish to investigate the role of the richness of the initial dynamics of the robot to see if sequences of sensori-motor associations can be anchored and synchronized with self dynamics, and learned as one new behavior. This study is applied in the frame of a mobile robot, associating place recognition with movements, and then these place-movement "units" in a more complex path of moves (see also [13] for a similar issue with top down analysis). The learning is triggered when the robot detects change in self dynamics provoked by the human pulling on a leash attached to the robot.

2 From Visuo-motor Learning to Low Level Imitation

Our model is a perception-action architecture designed to equilibrate 2 kinds of information : Visual information (V), about the external environment and Proprioception information (Prop), i.e information about the self movements that are being done (angle of joints ϕ_1,ϕ_2,...,ϕ_n, speed of the joints $\frac{d}{d\phi}$, etc.. of the devices). In order to compute basic behaviors, like tracking, pointing, reaching objects, the architecture is designed like an homeostatic machine (Figure 1 a.). It tries to minimize the differences between the inputs V and

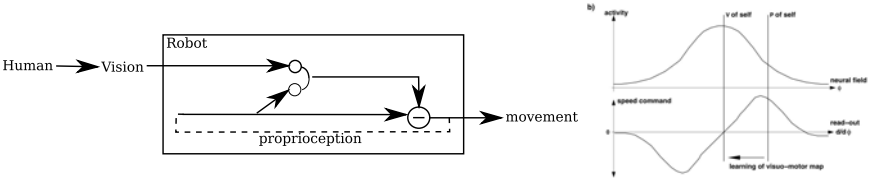


Fig. 1 a. The homeostatic controller trying to equilibrate visual and proprioceptive information. The motor command is computed by the neural field and the readout groups whose activities are illustrated in b. (example in 1 dimension). In this case Visual and proprioceptive information are different. This unbalance will trigger the learning of a new association on the visuo-motor map.

Prop, and the imbalances generate the movements of the system in order to reduce the corresponding error. To do so, V excites a Neural Field (NF) that compute a dynamical attractor centered on the input stimuli [14]. If Prop can be expressed in the same referential as the NF, then it is easy to extract the command to give to the joints of the arm in order to move this latter (and therefore the value of Prop) toward the maximum of the attractor (Figure 1 b.). This read-out [15] mechanism is the value of the derivative of the attractor at the position of Prop. Hence, behaviors of the robot are NF attractors triggered by V information (representing sensory-motor "target"). Obviously, the read-out mechanism is only possible if V and Prop can be computed in the same space. In previous work, we have shown that all information can be computed in a simple 2D space (corresponding to the vision produced by a sole CDD camera). To do so, we used a visuo-motor map of neurons, that associate multi-dimensional and redundant information about the joints into a simple 2D "visual" response. The associations are formed during a learning phase: the system produces initial random arm movements in order to learn to equilibrate input V and Prop information projected on the Visuo-motor map. The result is a system that is able to transform motor information (ϕ_1,ϕ_2,...,ϕ_n) into a simpler 2D space corresponding to the visual one. Then, the movements to reach visual goals are computed in the visual space thanks to the read-out mechanism . The association of an adaptive visuo-motor map with two 1D neural fields can be seen as a simple and global dynamical representation of the working space controlling an arbitrary number of degrees of freedom according to 2D information coming from the visual system [16]. Moreover, having a system that is not able to discriminate visual information about self movements with the movements of others can be, for example, useful to trigger an imitative behavior [17]. Based on movement detection, the system can't differentiate its extremity from another moving target. As a consequence, moving in front of the robot induces visual changes (movement detection) that the robot interprets as an unforeseen self movement. The robot acts as an *homeostat*, it tends to correct by producing the opposite movements, inducing the follow-up of the demonstrator gesture (numerous psychological works show comparable human behaviors when visual

Fig. 2 The low-level imitation principle applied to a robotic setup.

perception is ambiguous [18, 19, 20]). Applied to an eye-arm robotic system, the generated error will induce movements of the robotic arm reproducing the moving path of the human hand: an imitative behavior emerges . Using this setup, we showed that our robot can imitate several kind of movements (square or circle trajectories, up and down movements [21]. Hence, the imitative behavior emerges as a side effect of the perception ambiguity (and limitations).

Thanks to this ambiguity (i.e. confusing its grip and the hand of the other), we are able to modify the sensori-motor dynamic of the robot.

3 Robot Response to an Expressive Human

This second work is motivated by the question of how a robotic system (Figure 3, right), able to exhibit a set of emotional expressions can learn autonomously to a associate theses expressions with those of others. Here, “autonomously" refers to the ability to learn without the use of any external supervision. A robot with this property could therefore be able to associate its expressions with those of others, linking intuitively its behaviors with the responses of the others. This question is close to the understanding of how babies learn to recognize the facial expressions of their caregivers without having any explicit teaching signal allowing to associate for instance an “happy face” with their own internal emotional state of happiness. Using the cognitive system algebra [22], we showed that a simple sensori-motor architecture (see Figure 3, left) using a classical conditioning paradigm could solve the task if and only if we suppose that the baby produces first facial expressions according to his/her internal emotional state and that next the parents imitate the facial expression of their baby allowing in return the baby to associate these expressions with his/her internal state [23]. If the adult facial expressions are not synchronized with the baby facial expression, the task cannot be learned. Moreover, recent psychological experiments [24] have shown that humans ”reproduce” involuntary a facial expression when observing and trying to recognize it. Interestingly, this facial response has also been observed in presence of our robotic head. This low level resonance to the facial expression of the other can be considered as a natural bootstrap for the baby learning (“empathy" from the parents). Because the agent representing the baby must not be explicitly supervised, a simple solution is to suppose the

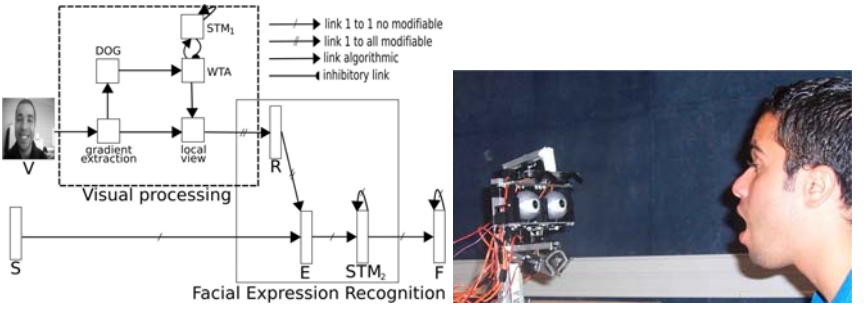


Fig. 3 The architecture used to associate a collection of feature points extracted from the visual flow with the emotion expressed by the robot. The R layer is categorizing the output of the Visual processing system (competition between the visual feature points). The S layer is the internal stimulus provoking the expression (happy, sad, surprise, etc...). The expression is executed (rotation of the head's motors) according to the activity of F, equivalent to S (one to one links). The E layer is learning to associate a category of R with an internal expression of S. If a human come in front of the robot and starts to imitate the expression, (s)he will close the loop between vision and expression and allow the system to learn to recognize emotions of others.

agent representing the parent is nothing more than a mirror. We obtain an architecture allowing the robot to learn the "internal state"-"facial expression" associations.

We showed that from our departure control architecture, learning is only possible if the parent agent (supposed to be the teacher) imitates the baby agent. The roles are switched according to the classical point of view of AI and learning theory. This advocates that taking account of the interaction dynamics between two agents can change our way of thinking learning and more generally cognition problems. To go further, we will show in the next section that modification of the dynamics of the robot can also be used to learn sensori-motor associations that can be used, in turn, to build a novel and more complex behavior. We illustrate it in the context of learning by correction/demonstration where a mobile robot learns path in concatenating place-movement associations.

4 Learning a Path as a Sequence of Sensori-Motor Associations

In the same manner as an eye-arm robot can learn the associations between vision of its end point and motor information about self arm configuration, we have developed a controller for mobile robots which is able to associate visual information (a panorama of the environment) with self orientation (using a compass representing the direction of the actual movement). This controller

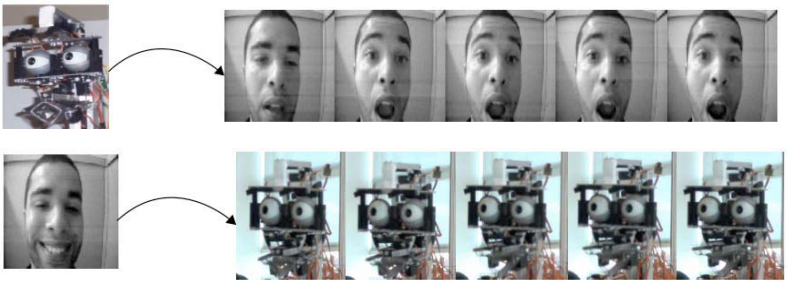


Fig. 4 Up : the robot produces a facial expression (reflex behavior) and the human imitates the robot. By doing this, the human is closing the loop of the interaction and returns to the robot an equivalent of its own facial expression. A categorization of the visual scene is then possible with an unconstrained vision system that just learn the properties of the features points of *all* the scene (no notion of what is a face or an agent). Bottom : once the salient features of the scene are extracted and associated with the corresponding internal emotion, the robot is able to respond to the human with the same emotional expression. The role are reversed and he robot is now able to recognize and imitate the facial expression of the human. The robot is able to display 4 facial expressions plus neutral face.

is designed as a sensori-motor loop based on a neurobiological model testing some of the spatial properties of the hippocampus [25]. Numerous studies [26, 27] show and discuss the spatial selectivity of rat hippocampal neurons. These specific neurons are the so-called "place cells", because of their activity firing for specific parts of the environment [28] . Our model allows the robot to learn and recognize some regions of the environment (places). It also allows to associate the response of the place cells with a movement (a direction) and therefore to obtain a mobile robot able to recognize and reach learned places (attractor of the sensori-motor space).

Fewer researches highlight the temporal properties of the hippocampal loop and the fact that populations of cells can also learn the timing of the transitions between input events [29, 30]. From theses studies we have come to design control architectures allowing a robot to learn sequences of sensori-motor transitions [31, 32]. Previous studies have suggested that successive sensori-motor "units" could be chained and glued one after the other under the form of a sequence of predictions of the sensori-motor transitions [33]. We are now interested in the properties that can allow a robot to learn complex sequences (i.e sequences containing many occurrences of the same "unit") and how a whole sequence can be learned and synchronized [34].

Indeed, it is important to distinguish how a "behavior" can be learned : on one side you can choose to anchor the different steps in the environment. You will obtain a coding linked to the environment, where each step, each association is dependent of the environment (and the recognition of this environment). For example our associative robot is able to recognize different

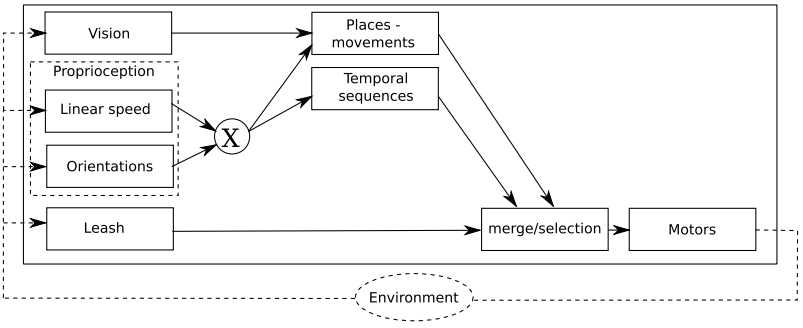


Fig. 5 Model of 2 sensori-motor loops (i) place-movement associations learning and (ii) temporal complex sequence learning.

places that can be associated to a particular move (recognizing place 1 can be associated with the move of going in a particular direction leading the robot to arrive to place 2, an so on..). In this case no synchronization is required as the changes of the external environment (due to the robot actions) naturally elect the new association. On the other side, you can choose to learn this behavior independently of the external environment, “blindly", for example by anchoring the proprioceptive changes according to an internal timeline . Of course, it is interesting to notice that both solution seem to complete each other, and the goal of this section is to present a model (figure 5) where both dynamics (temporal and spatial) are learned by the same model in two sensori-motor loops.

The subsection 1.3.1 will present the “place-movement" mechanism, emphasizing how orientation changes can be robustly associated with place recognition. The subsection 1.3.2 will present the “temporal sequence" mechanism, emphasizing how orientation changes can be associated to an internal dynamics composed of batteries of continuous time recurrent networks. We will present the result of a simple experiment showing how both mechanism can replace and complete each other allowing to group the sensori-motor associations in one, more complex path. To drive the robot , the human teacher use a leash to correct the robot’s behavior. When the robot senses the leash, it follows the direction desired by the teacher (direct pathway).

4.1 Place-Movement Associations

A place is defined by a constellation of visual features (couple landmark-azimuth) extracted from a panorama (figure 8) compressed in a place code. The place code results of the merging of “what” information provided by the visual system that extracts local view centered on point of interest (landmark recognition), and a “where” information provided by a magnetic compass acting as proprioceptive information (spatial localization in the visual field).

Fig. 6 Example of setup allowing to a human to learn paths to a mobile robot. The control architecture is designed to detect and learn changes in the robot self motor dynamics (strong differences in the proprioceptive flow of the wheels). Thanks to a leash, the human can pull on a sensing device provoking orientation changes of the robot (direct pathway). The robotic arms are not used in this study.

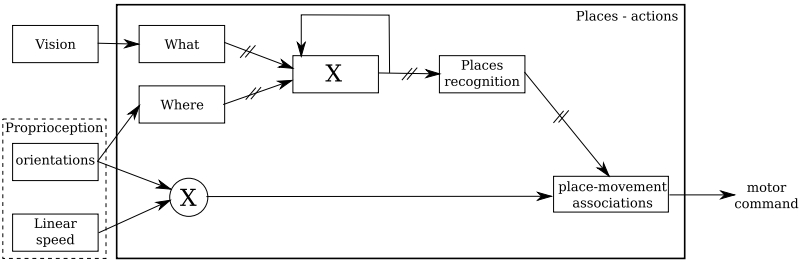


Fig. 7 Model of place-movement associations learning. "what" and "where" information are extracted from vision and proprioception. They are merged and compressed in place code allowing place recognition. These place codes are associated to proprioceptive information.

The merging of the "what" and "where" information is performed in a product space (*i.e.* a second-order tensor compressed into a vector of product neurons $m_k(t)$ called *merging neurons*, but see [35] for more classical sigma-pi units) defining a tensorial place code $M(t)$. The multiplicative merging realizes an analogical "AND" operation. The recruited merging neurons characterize a point (or a region) in the *landmark-azimuth* space. A new merging neuron is learned as follow :

$$\Delta\omega_{ak}^{LM} = \Gamma_1(l_l(t)) \cdot R_k^M(t) \tag{1}$$

$$\Delta\omega_{lk}^{AM} = \Gamma_1(\theta_a(t)) \cdot R_k^M(t) \tag{2}$$

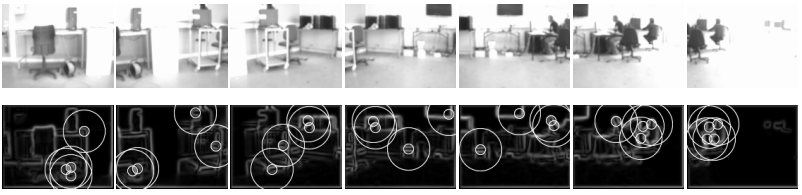


Fig. 8 Example of visual features extraction on half visual panorama. The system computes a gradient on each view. 4 visual features are extracted from each gradient.

with $\Gamma_1(x) = 1 \; if \; x \geq 1 \; otherwise \; 0$ (Heaviside function). l_l and θ_a respectively the l^{th} landmark and the a^{th} azimuth neuron. R_k^M is the recruitment signal of the k^{th} neuron : $R_k^M = 1$ if the neuron k is the new recruited neuron, *otherwise* 0.

Once the learning is done, a $m_k(t)$ activity is the place code of one panorama. Each component of the places codes responds proportionally to the learned couple landmark-azimuth :

$$\mathcal{L}_k(t) = \sum_{l=1}^{n_L} \omega_{lk}^{LM}(t).l_l(t)$$

$$\mathcal{A}_k(t) = \sum_{a=1}^{n_\Theta} \omega_{ak}^{AM}(t).\theta_a(t)$$

with $\omega_{lk}^{LM}(t)$ and $\omega_{ak}^{AM}(t)$ respectively the weights of the synapses of the kth merging neuron coming from the l^{th} landmark and a^{th} azimuth neuron (ω_{lk}^{LM} and ω_{ak}^{AM} are initially null). n_L and n_Θ are the number of recruited landmarks and azimuth neurons.

At last, the response of each place-code m_k depends on the short term memory (STM) and of the new inputs from $\mathcal{L}_k$ and $\mathcal{A}_k$. The STM property is used to maintain the m_k activity during all the process of the panorama.

$$m_k(t) = \max\left(\mathcal{L}_k(t) \cdot \mathcal{A}_k(t), \left[\lambda^M(t).m_k(t - d_t) - r_k(t)\right]^+\right) \tag{3}$$

with $r_k(t)$ a signal reseting the activity of the neuron k at the beginning of the visual exploration of a panoramic image. A forgetting term $\lambda^M(t)$ applied on the activity of the k^{th} merging neuron is also used to act as STM.

Once a visual panorama is encoded, a new place cell is learned. A place cell p follows the one-shot learning rule :

$$\Delta\omega_{kp}^P = \Gamma_1\left(m_k(t)\right) \cdot R_p^P(t) \tag{4}$$

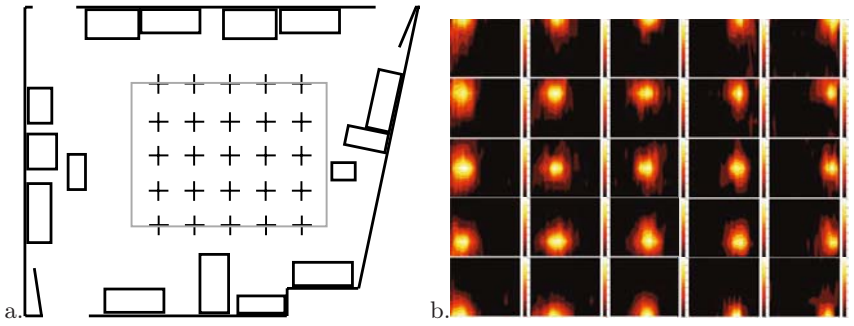


Fig. 9 a. Room with regular markers where the robot learns each place. b. 25 places are regularly learned and tested in the room. A competition between all place cells leads to paving the environment.

with ω_{kp}^{P} (initially null and taking binary values $0, 1$) the connections between the k^{th} merging neuron and the recruited neuron p coding the place. The recruitment algorithm is the same as the one used in eq 1 and 2.

The activity of a place cell results from the computation of the distance between the learned place code and the current place code (at the end of each visual exploration). Thus, the activity $P_p(t)$ of the p^{th} place cell can be expressed as follows:

$$P_p(t) = \frac{1}{W_p}\left(\sum_{k=1}^{n_M} \omega_{kp}^{P}(t) m_k(t)\right) \tag{5}$$

where $\omega_{kp}^{P}(t)$ expresses the fact that the couple k (*i.e.* the k^{th} merging neuron which activity is $m_k(t)$) has been used to encode the place-cell p. The number of couples used by the p^{th} place-cell is given by $W_p = \sum_{k=1}^{n_M} \omega_{kp}^{P}$, and with n_M the number of recruited neurons in the *what* and *where* tensor.

In order to illustrate the response of the place cells, we have placed a mobile robot on 25 markers positioned as we can see in figure 9.a. It is important to understand that the robot does not see the markers. After the learning of the 25 places, we let the robot go straight on each line of markers. The figure 9.b shows the activity of each place cell at different region of the environment. A place cell learned in the place A responds maximally in A and creates a large decreasing place field around A.

Such a system is able to learn several regions of the environment. Consequently, it can be visually located thanks to place cells which react at particular regions of the environment. When the teacher draws the leash, it modifies the robot's dynamics. Leash information can be seen as an input that automatically override the motor command of the wheels (figure 5). The resulting change in the proprioception (orientation change) triggers the learning of an associations between this new orientation of self and the winning place cell.

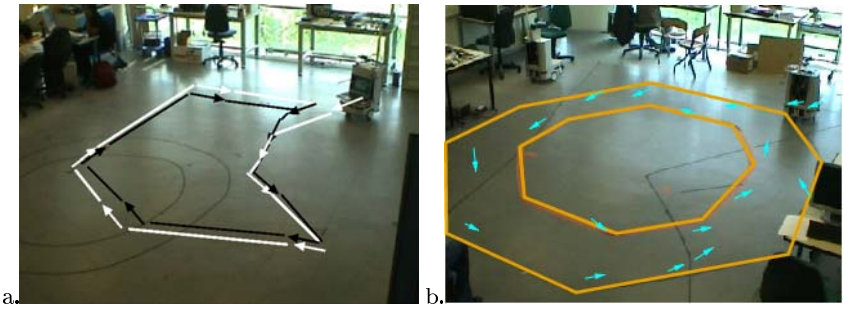


Fig. 10 Learning of a path by correction of the dynamic of the robot. The robot has its own initial dynamics (it goes straight) and the teacher corrects it by modifying the orientation of the robot. For this, the teacher uses either a leash or a joystick. Each arrow represents a correction applied by the teacher. Hence, the robot learn the association between the place and the movement. a. The system learns a round precisely and one time after 1 round of correction by the teacher (light arrows) and reproduces the trajectory autonomously (dark arrows). b. After 3 rounds of learning (arrows show where the robot has associated the corresponding movement), the teacher does not have to correct the robot anymore.

From this simple interaction, the robot learns the new movement given by the teacher and associates it at the place where the robot is. Step by step, by associating the movements with the different places, the system shapes progressively a sensori-motor attractor corresponding to the trajectory of the path (figure 10). Such a system does not learn the path like such, but only different parts of the path without connections between them: putted all together, the iterative constraints (pulls of the leash) have shaped the sensori-motor attractor representing the path, but there is no unified representation of this path in the neural memory of the architecture. Of course, such a coding also have the following limitations : the robot can not learn complex behaviors. For instance, only one movement can be associated to a place. Moreover, the coding being based on the visual sensations, the system is dependent of visual quality of the environment: in the dark, the robot can not recognizes the places and therefore not execute the appropriate movement in order to continue the sequence.

4.2 The Role of Internal Dynamic for More Complex Behavior Learning

To allow the robot to learn a path as a sequence of movements (proactive navigation) and not only as a movement anchored in a particular place (reactive navigation), we test a second sensori-motor loop (Figure 11) based on a neurobiological model [30] inspired from the temporal properties of the cerebellum

and the hippocampus. In the scope of learning a sequence, neural networks based solutions are interesting. Among the models, chaotic neural networks are based on recurrent network (RN). In [36], a fully connected RN learns a sequence thanks to a single layer of neurons. The dynamics generated by the network helps to learn a short sequence but after a few iterations, the learned sequence is progressively lost. In [37], a random RN (RRN) learns a sequence thanks to a combination of two layers of neurons. The first layer generates an internal dynamic by means of a RRN. The second layer generates a resonance phenomenon. The network learns short sequences of 7 or 8 states. But this model is highly sensitive to noises on the stimuli and does not learn long sequences. A similar model is the Echo States Network (ESN) based on RRN for their short term memory property [38] (STM). Under certain conditions (detailed in [39]), the activation of each neuron in the hidden layer is a function of the input history presented to the network; this is the echo function. Once again, the idea is to use a "reservoir" of dynamics from which the desired output is learned in conjunction with the effect of the input activity.

In the context of robotics, many models concern gesture learning (see for example [40] and [41] in this book for learning by imitation). By means of nonlinear dynamical systems, [42] develops control policies to approximate the recorded movements and to learn them with a fitting of mixture model using a recursive least square regression technique. In [43], the trajectories of gestures are acquired by the construction of motor skills with a probabilistic representation of the movement. Trajectories can be learnt through via points [44] with parallel vector-integration-to-endpoint models [45]. In our work, we wish to be able to re-use and detect sub-sequences and possibly, combine them. Thus, we need to learn some of the important components of the sequence and not only to approximate the trajectory of the robot.

In a first approach [46], we proposed a neural network for learning on-line the timing between the events of one simple sequences (with non ambiguous states like "A B C"). Now we propose an extension of this model to the learning of complex sensori-motor sequences (with ambiguous states like A and B in "A B A C B"). In order to remove the ambiguous states, we use batteries of oscillators as a reservoir of diversity allowing to separate the inputs appearing repeatedly in the sequence. This model uses an associative learning rules linking the past inputs (STM) with the actual inputs. This internal states are coded and based on different and un-ambiguous patterns of activities. As a result, our architecture manages to learn/predict and reproduce complex temporal sequences.

The dynamics are generated by a set of oscillators based on CTRNN [47] (figure 12). An oscillator is 2 coupled neurons computed following :

$$\tau_e . \frac{dx}{dt} = -x + S((w_{ii} * x) - (w_{ji} * y) + w_{e_{const}}) \tag{6}$$

$$\tau_i . \frac{dy}{dt} = -y + S((w_{jj} * y) + (w_{ij} * x) + w_{i_{const}}) \tag{7}$$

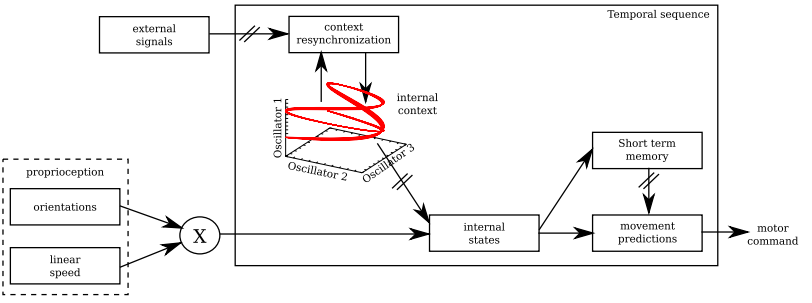


Fig. 11 Model of temporal sequence learning. Inputs are associated at the internal dynamic generated by 3 CTRNN oscillators to create an internal state. The system learns transitions of movements in associating the past movements in TS and the present movement.

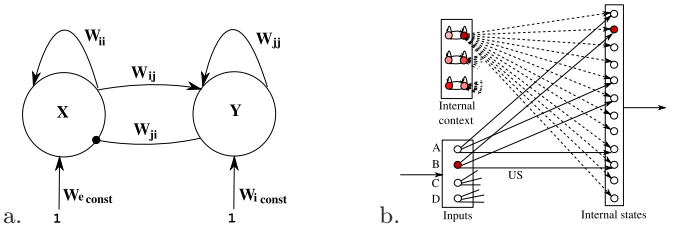


Fig. 12 The internal dynamics are generated by a set of oscillators based of CTRNN. a. Model of an oscillator composed of coupled neurons (1 excitatory, 1 inhibitory) fully connected. b. Model of the neural network used to associate an input state with internal dynamics. Only few links are represented for the legibility. Dashed links are modifiable connections. Solid links are fixed connections.

with τ_e a time constant for the excitatory neuron and τ_i for the inhibitory neuron. x and y are the activities of the excitatory and the inhibitory neurons respectively. w_{ii} is the weight of the recurrent link of the excitatory neuron, w_{jj} the weight of the recurrent link of the inhibitory neuron. w_{ij} is the weight of the link from the excitatory neuron to inhibitory neuron. w_{ji} is the weight of the link from the inhibitory neuron to excitatory neuron. $w_{e_{const}}$ and $w_{i_{const}}$ are the weights of the links from the constant inputs. And S is the identity function of each neuron. In our model, we use three oscillators with $\tau_e = \tau_i$.

In order to use repeatedly the same input in given sequence, each input is associated with the internal dynamics generated by the oscillators. The learning process of an association is :

$$US = w_i * x_i \tag{8}$$

with w_i the weight of the link from input state i, and x_i activity of the input state i. If $US > threshold$, we compute the potential and the activity of the neuron as follow :

$$Pot_j = \sum_{j=0}^{M_{osci}} |(w_j - e_j)| \qquad Act_j = \frac{1}{1 + Pot_j} \tag{9}$$

with M_{osci} the number of oscillators, w_j the weight of the link from oscillator j, and e_j the activity of the oscillator j. The neuron that has the minimum activity is recruited : $Win = Argmin_j(Act_j)$. Initial weights of connections have high values. The internal dynamics is learnt according to the error of distance $\Delta w_j = \varepsilon(e_j - w_j)$ with ε a learning rate, w_j weight of link from oscillator j, and e_j activity of oscillator j.

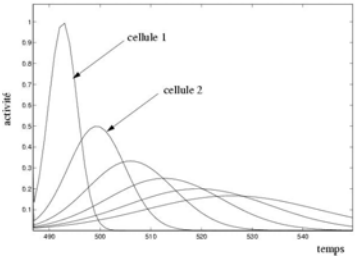


Fig. 13 Activity time spectrum memorizing past inputs.

As said, the model uses an associative learning rules between past inputs memorized as TS (figure 13) and present inputs in order to learn the timing of sequences. The TS is computed following :

$$Act_{j,l}^{TS}(t) = \frac{1}{m_j} \cdot \exp - \frac{(t - m_j)^2}{2 \cdot \sigma_j} \tag{10}$$

with $Act_{j,l}^{TS}$ the activity of the cell at index l on the line j, t the time, m_j a time constant and σ_j the standard deviation. Neurons on one line share their activities in the time and represent a temporal trace of the internal states. Learning of an association is on the weights of links between movement prediction group and the TS. The normalization of the activity coming from neurons of TS is performed due to the normalization of the weights.

$$W_{prediction(i,j)}^{TS(j,l)} = \begin{cases} \frac{Act_{j,l}^{TS}}{\sum_{j,l}\left(Act_{j,l}^{TS}\right)^2} \; if \; Act_j^{TS} \neq 0 \\ unchanged \quad otherwise \end{cases} cx \tag{11}$$

Moreover, this model has the property to work when the same input comes several times consecutively. Thanks to a derivative input group, only the first occurrence of an input is detected, and two successive occurrences of the same state are not considered to be ambiguous for the detector ("A A" = "A").

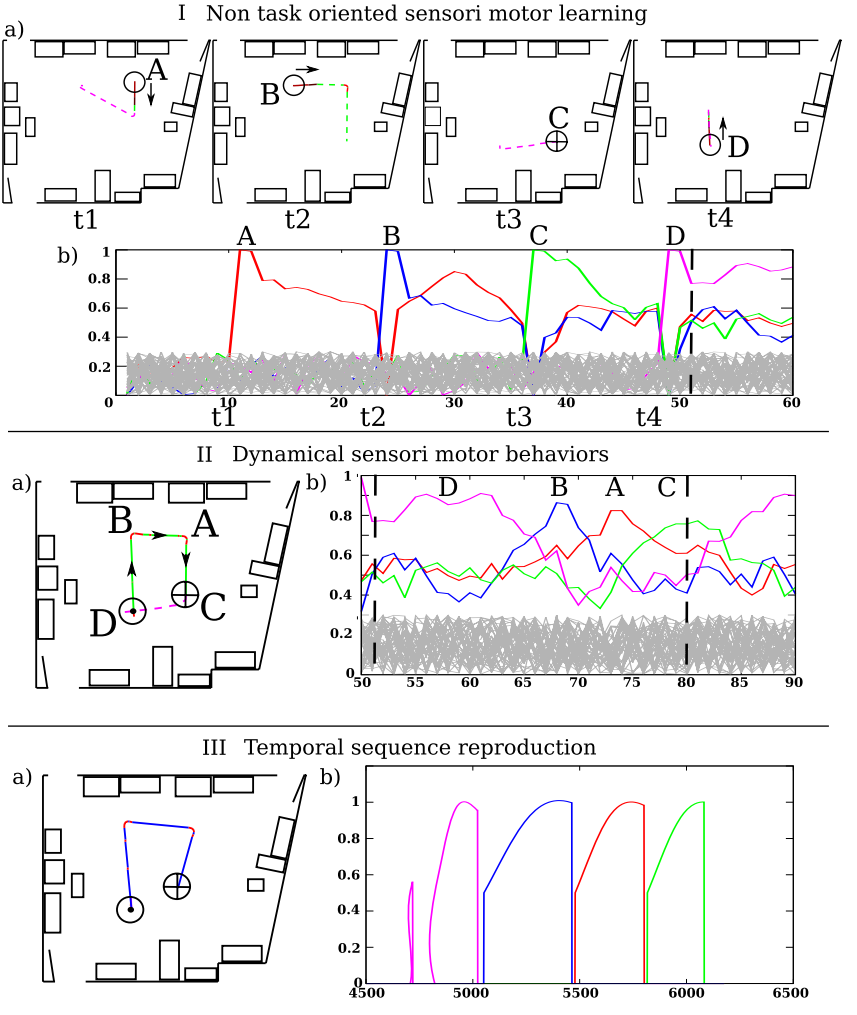


Fig. 14 I.a - The operator guides the robot to different places of the room. The robot learns each visual place and it associates its current movement - an orientation/linear speed couple (*go down* at place "A", *go right* at place "B", *stop down* at place "C" and *go up* at place "D"). I.b - The activities of the recognition of learnt visual places ("A", "B", "C" and "D".)

II.a - The robot is taken back to place "D" in the state *stop up* by the operator. Thanks to the place recognition, the robot predicts the associated movement *go up*. During the reconstruction of the trajectory, the system learns the temporal sequence of its orientations/linear speed couple. II.b - The activities of visual places recognized.

III.a - The robot is taken back at the beginning of the temporal sequence in the state *stop up* by the operator. The operator hides the vision of the robot, consequently the system can not recognize the visual places. Thanks to temporal sequence, the robot predicts the next movement and it reproduces the trajectory. III.b - Activities of the temporal predictions.

Recent experiments shows that this architecture can be transposed to learning by demonstration [34], and also to the learning of complex sequences of displacements with a navigating robot [48]. In this case, the coupling between the teacher and the robot can also be performed via the visual environment (see also [49] for a similar dynamical coupling) but it is important to mention that the architecture always learns the changes of its own dynamics, whatever the media of the perturbation is (pulling a leash, using a joystick, moving in front of the visual field, etc...). With such a model, we also have the opportunity to test the complementarity of these two sensori-motor loops known to be involved in the learning of tasks (place-movement learning vs temporal learning). In an on going experiment, we are testing the benefits and limitation of using these two learning at the same time. Figure 14 illustrate how, one learning procedure (via interaction with the teacher) can be the support of a first encoding (place-movement) which demonstration supports in turn an internal re-encoding of the sequence (learning of the timing of the transitions). The experiment is conducted as follow : (I) The robot learns different places in interaction with the teacher giving the movement to execute at each places (figure 14.I). At this moment, the robot does not learn a path, but several independent movements associated with the different places. (II) Next, the robot is kidnapped and then navigate autonomously, but reactively (figure 14.II): it rapidly converge toward the spatial attractor of the path built previously. During the reproduction of the path, the system learns the timing of ruptures of its own movements (orientations and/or linear speed changes). Hence, the robot learns solely the path as a succession of temporal movements in addition to the spatial attractor. (III) Finally, we kidnapped the robot to the same place that previously. At this moment, the robot is able to navigate autonomously reactively (spatial attractor) and proactively (temporal prediction). Finally if we switch off the vision of the robot (figure 14.III), it is not able to recognize the places, but still able to reproduce the path thanks to the proactive navigation and the temporal sequence learned by itself in (II).

5 Discussion

These examples show the importance of the interaction with a partner for the robot development and learning. Imitation appears as an important mechanism both for learning and communication. Interestingly, in our developmental procedure there is no need at first to suppose that the robot is able to recognize the human partner as a model. At the opposite, the robot development is based on the perception ambiguity. The robot considers the human partner as a part of itself. The interaction and the associated correlations extend the proprioception of the robot to the human partner. They become a single dynamical system able to learn things that the robot could not learn by itself. Hence imitative behaviors would be the result at first of the agent

inability to perceive something as being different from itself. We can imagine the capability to predict the feedback signal can be used to verify if there is some novelty in the sensory feedback related to the current robot action. Bounded novelty may result from the interaction with a partner hence allowing to recognize him/her as such. Hence we can propose a scheme in which the individuation comes after a phase of fusion/merging with the environment. The proprioceptive signal is necessary to close the loop and to build a global system composed of the agent and his/her caregivers allowing simple learning mechanisms to be sufficient for the learning of more and more complex behaviors (autopoietic loop). First correlations detections allow to build new predictive signals allowing then to differentiate the agent “inner part" from the outside and latter the other agents. We have shown for instance that the capability to discriminate faces from non faces could be the result of the emotional interaction and not one of its prerequisites as usually supposed in classical image processing works. In the same vein, joint attention to a given object could be the result of the emotional association between the facial expression of the experimenter and an object (social referencing) and not the opposite as it is classically supposed. Social interactions can be seen in the first periods of the development as a way to increase the potential of the proprioceptive function allowing to maintain more complex and stable dynamics that can be used for the learning of more complex tasks. Hence, in an epigenetic approach, the sensori-motor control appears as an essential element to avoid the symbol grounding problem and to build autonomous robots able to develop more and more cognitive capabilities thanks to the social interactions.

Acknowledgments

This work was supported by the French Region Ile de France, the Institut Universitaire de France (IUF) and the FEELIX GROWING european project. (FP6 IST-045169)

References

1. Braitenberg, V.: Vehicles : Experiments in Synthetic Psychology. MIT Press, Bradford Books, Cambridge (1984)
2. Pfeifer, R., Scheier, C.: Understanding intelligence. MIT Press, Cambridge (1999)
3. Michel, F., Ferber, J., Drogoul, A.: Multi-Agent Systems and Simulation: a Survey From the Agents Community’s Perspective in Multi-Agent Systems: Simulation and Applications. CRC Press - Taylor and Francis (2009)
4. Breazeal, C.: Regulation and entrainment for human-robot interaction. International Journal of Experimental Robotics 10-11(21), 883–902 (2003)
5. Kozima, H., Nakagawa, C., Yano, H.: Can a robot empathize with people? Artificial Life and Robotics 8(1), 83–88 (2004)

6. Masahiro, M.: On the uncanny valley. In: Proceedings of the Humanoids 2005 workshop: Views of the Uncanny Valley, Tsukuba, Japan (December 2005)
7. Meltzoff, N., Moore, M.K.: Imitation of facial and manual gestures by humans neonates. Science 198, 75–82 (1977)
8. Nadel, J., Croué, S., Mattlinger, M., Canet, P., Hudelot, C., Lecuyer, C., Martini, M.: Do autistic children have expectancies about the social behaviour of unfamiliar people?: A pilot study with the still face paradigm. Autism 2, 133–145 (2000)
9. Meltzoff, A., Decety, J.: What imitation tells us about social cognition: a rapprochementbetween developmental psychology and cognitive neuroscience. Phil. Trans. R. Soc. Lond. B 358, 491–500 (2003)
10. Kozima, H., Nakagawa, C., Yano, H.: Attention coupling as a prerequisite for social interaction. In: 2th IEEE International Workshop on Robot and Human Interactive Communication, IEEE, October 2003, pp. 109–114. IEEE Press, Los Alamitos (2003)
11. Kozima, H., Nakagawa, C., Yano, H.: Can a robot empathize with people? Artif. Life Robotics 8, 83–88 (2004)
12. Revel, A., Andry, P.: Emergence of structured interactions: From a theoretical model to pragmatic robotics. Neural Network 22, 116–125 (2009)
13. Ratliff, N., Silver, D., Bagnell, J.: Learning to search: Functional gradient techniques for imitation learning. Autonomous Robots 27(1), 25–53 (2009)
14. Amari, S.: Dynamic of pattern formation in lateral-inhibition type by neural fields. Biological Cybernetics 27, 77–87 (1977)
15. Schöner, G., Dose, M., Engels, C.: Dynamics of behavior: theory and applications for autonomous robot architectures. Robotic and Autonomous System 16(2-4), 213–245 (1995)
16. Andry, P., Gaussier, P., Hirsbrunner, B., Nadel, J.: Learning invariant sensorymotor behaviors: A developmental approach of imitation mechanisms. Adaptive behavior 12(2) (2004)
17. Gaussier, P., Moga, S., Quoy, M., Banquet, J.: From perception-action loops to imitation processes: a bottom-up approach of learning by imitation. Applied Artificial Intelligence 12(7-8), 701–727 (1998)
18. Nielsen, T.: Volition: A new experimental approach. Scandinavian Journal of Psychology 4, 225–230 (1963)
19. Fourneret, P., Jeannerod, M.: Limited conscious monitoring of motor performance in normal subjects. Neuropsychologia 36, 1133–1140 (1998)
20. Jeannerod, M.: To act or not to act. perspectives on the representation of actions. Quaterly Journal of Experimental Psychology 52A, 1–29 (1999)
21. Andry, P., Gaussier, P., Nadel, J.: From sensorimotor coordination to low level imitation. In: Second international workshop on epigenetic robotics, pp. 7–15 (2002)
22. Gaussier, P.: Toward a cognitive system algebra: A perception/action perspective. In: European Workshop on Learning Robots (EWRL), pp. 88–100 (2001)
23. Gaussier, P., Prepin, K., Nadel, J.: Toward a cognitive system algebra: Application to facial expression learning and imitation. In: Iida, F., Pfeifer, R., Steels, L., Kuniyoshi, Y. (eds.) Embodied Artificial Intelligence. LNCS (LNAI), vol. 3139, pp. 243–258. Springer, Heidelberg (2004)
24. Nadel, J., Simon, M., Canet, P., Soussignan, R., Blancard, P., Canamero, L., Gaussier, P.: Human responses to an expressive robot. In: Epirob 2006 (2006)

25. Banquet, J., Gaussier, P., Dreher, J.C., Joulain, C., Revel, A., Günther, W.: Space-time, order, and hierarchy in fronto-hippocampal system: A neural basis of personality. In: Matthews, G. (ed.) Cognitive Science Perspectives on Personality and Emotion, vol. 124, pp. 123–189. North Holland, Amsterdam (1997)
26. Zipser, D.: A computational model of hippocampal place fields. Behavioral Neuroscience 99(5), 1006–1018 (1985)
27. Wiener, S., Berthoz, A., Zugaro, M.: Multisensory processing in the elaboration of place and head direction responses by limbic system neurons. Cognitive Brain Research 14, 75–90 (2002)
28. O'Keefe, J.: The hippocampal cognitive map and navigational strategies. Oxford University Press, Oxford (1991)
29. Grossberg, S., Merrill, J.: A neural network model of adaptively timed reinforcement learning and hippocampal dynamics. Cognitive Brain Research 1, 3–38 (1992)
30. Gaussier, P., Moga, S., Banquet, J.P., Quoy, M.: From perception-action loops to imitation processes. Applied Artificial Intelligence (AAI) 1(7), 701–727 (1998)
31. Moga, S., Gaussier, P.: A neuronal structure for learning by imitation. In: Floreano, D., Mondada, F. (eds.) ECAL 1999. LNCS, vol. 1674, pp. 314–318. Springer, Heidelberg (1999)
32. Andry, P., Gaussier, P., Moga, S., Banquet, J., Nadel, J.: Learning and communication in imitation: An autonomous robot perspective. IEEE transactions on Systems, Man and Cybernetics, Part A 31(5), 431–444 (2001)
33. Andry, P., Gaussier, P., Nadel, J.: Autonomous learning and reproduction of complex sequence: a multimodal architecture for bootstrapping imitation games. In: Lund University Cognitive Studies (ed.) Proceedings of the Fith Internationnal Workshop on Epigenetic Robotics, Modeling Cognitive development in robotic Systems, EPIROB 2005, July 2005, pp. 97–100 (2005)
34. Lagarde, M., Andry, P., Gaussier, P.: The role of internal oscillators for the one-shot learning of complex temporal sequences. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 934–943. Springer, Heidelberg (2007)
35. Rumelhart, D.E., Mcclelland, J.L.: Parallel distributed processing: explorations in the microstructure of cognition. psychological and biological models, vol. 2 (1986)
36. Molter, C., Salihoglu, U., Bersini, H.: Learning cycles brings chaos in continuous hopfield networks. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN) conference (2005)
37. Daucé, E., Quoy, M., Doyon, B.: Resonant spatio-temporal learning in large random neural networks. Biological Cybernetics 87, 185–198 (2002)
38. Jaeger, H.: Short term memory in echo state networks. Technical Report GMD Report 152, German National Research Center for Information Technology (2001)
39. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology (2001)

40. Lopes, M., Melo, F., Montesano, L., Santos-Victor, J.: Abstraction Levels for Robotic Imitation: Overview and Computational Approaches. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 313–355. Springer, Heidelberg (2010)
41. Chalodhorn, R., Rao, R.: Learning to imitate human actions through eigenposes. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 357–381. Springer, Heidelberg (2010)
42. Ijspeert, A.J., Nakanishi, J., Shibata, T., Schaal, S.: Nonlinear dynamical systems for imitation with humanoid robots. In: Proceedings of the IEEE/RAS International Conference on Humanoids Robots (Humanoids 2001), pp. 219–226 (2001)
43. Calinon, S., Billard, A.: Learning of Gestures by Imitation in a Humanoid Robot. In: Dautenhahn, K., Nehaniv, C.L. (eds.). Cambridge University Press, Cambridge (2006)
44. Hersch, M., Billard, A.: A biologically-inspired model of reaching movements. In: Proceedings of the 2006 IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, Pisa (2006)
45. Bullock, D., Grossberg, S.: Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. Psychological Review 95, 49–90 (1988)
46. Ans, B., Coiton, Y., Gilhodes, J.C., Velay, J.L.: A neural network model for temporal sequence learning and motor programming. Neural Networks 7(9), 1461–1476 (1994)
47. Beer, R.: On the dynamics of small continuous–time recurrent networks. Technical Report CES–94–18, Cleveland, OH (1994)
48. Lagarde, M., Andry, P., Gaussier, P., Giovannangeli, C.: Learning new behaviors: Toward a control architecture merging spatial and temporal modalities. In: Workshop on Interactive Robot Learning - International Conference on Robotics: Science and Systems (RSS 2008) (June 2008) (to appear)
49. Cheng, G., Kuniyoshi, Y.: Complex continuous meaningful humanoid interaction: A multi sensory-cue based approach. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2000), April 2000, pp. 2235–2242 (2000)

Adaptive Optimal Feedback Control with Learned Internal Dynamics Models

Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar

Abstract. Optimal Feedback Control (OFC) has been proposed as an attractive movement generation strategy in goal reaching tasks for anthropomorphic manipulator systems. Recent developments, such as the Iterative Linear Quadratic Gaussian (ILQG) algorithm, have focused on the case of non-linear, but still analytically available, dynamics. For realistic control systems, however, the dynamics may often be unknown, difficult to estimate, or subject to frequent systematic changes. In this chapter, we combine the ILQG framework with learning the forward dynamics for simulated arms, which exhibit large redundancies, both, in kinematics and in the actuation. We demonstrate how our approach can compensate for complex dynamic perturbations in an online fashion. The specific adaptive framework introduced lends itself to a computationally more efficient implementation of the ILQG optimisation without sacrificing control accuracy – allowing the method to scale to large DoF systems.

1 Introduction

The human motion apparatus is by nature a highly redundant system and modern humanoid robots, designed to mimic human behaviour and performance, typically exhibit large degrees of freedom (DoF) in the kinematics domain (joints) and in the dynamics domain (actuation). Many recent humanoid system designs are extending the classic joint torque operated designs (i.e., one motor per joint) by redundantly actuated systems based on antagonistic or pseudo-antagonistic architectures (e.g., [11, 32]). Therefore producing even the simplest movement, such as reaching towards a particular position, involves an enormous amount of information processing and a controller has to make a choice from a very large space of possible movements

Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar
School of Informatics, University of Edinburgh,
10 Crichton Street, Edinburgh EH8 9AB, United Kingdom
e-mail: {d.mitrovic,s.klanke,sethu.vijayakumar}@ed.ac.uk

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 65–84.
springerlink.com

to achieve a task. An important question to answer therefore is how to resolve this redundancy?

Optimal control theory [23] answers this question by establishing a certain cost function, and selecting the solution with minimal cost (e.g., minimum jerk [10], minimum torque change [29]). Quite often these control schemes are only concerned with trajectory *planning* and an open loop optimisation of the control commands, while the correction of errors during *execution* is left to simple PID controllers. As an alternative, closed loop optimisation models are aimed at providing a control law which is explicitly based on feedback from the system. In the ideal case, the system state is directly mapped to control signals during execution, and the form of this mapping is again governed by a cost function [25].

Another characteristic property of anthropomorphic systems, besides the high redundancies, is a lightweight and flexible-joint construction which is a key ingredient for achieving compliant human-like motion. However such a morphology complicates analytic dynamics calculations, which usually are based on unrealistic rigid body assumptions. Moreover, even if the different links of a manipulator could be modelled as a rigid body, the required parameters such as mass and inertia may be unknown or hard to estimate. Finally, unforeseen changes in the plant dynamics are hard to model based purely on analytic dynamics. In order to overcome these shortcomings we can employ online supervised learning methods to extract dynamics models driven by data from the movement system itself. This enables the controller to adapt *on the fly* to changes in dynamics conditions due to wear and tear or external perturbations. Applying such methods has previously been studied in robot control [8, 6, 18, 31] but has not found much attention in the perspective of the optimal control framework. Indeed the ability to adapt to perturbations is a key feature of biological motion systems and enabling optimal control to be adaptive is a valuable theoretical test-bed for human adaptation experiments.

By combining optimal control with dynamics learning we can create a powerful framework for the realisation of efficient control for high dimensional systems. This will provide a viable and principled control strategy for the biomorphic based highly redundant actuation systems that are currently being developed. Furthermore, we would like to exploit this framework for understanding optimal control and its link to biological motor control.

2 Optimal Feedback Control

In the past, although many control problems have been described within the framework of optimality, most optimal motor control models have focused on open loop (feed-forward) optimisation [10, 29]. Assuming deterministic dynamics (i.e., no perturbations or noise), open-loop control will produce a sequence of optimal motor signals or limb states. However if the system leaves the optimal path due to inevitable modelling imperfections, it must be corrected for example with a hand-tuned PID controller. This will often lead to suboptimal behaviour, because the error feedback has not been incorporated into the optimisation process.

Stable optimal performance can only be achieved by constructing an *optimal feedback law* that produces a mapping from states to actions by making use of all available sensory data. In such a scheme, which is referred to as *optimal feedback control* (OFC) [26], there is no separation anymore between trajectory planning and trajectory execution for the completion of a given task. Rather, one directly seeks to obtain the gains of a feedback controller which produce an optimal mapping from state to control signals (control law). A key property of OFC is that errors are only corrected by the controller if they adversely affect the task performance, otherwise they are neglected (minimum intervention principle [27]). This is an important property especially in systems that suffer from control dependent noise, since task-irrelevant correction could destabilise the system beside expending additional control effort.

In this work, we focus on the investigation of OFC in limb reaching movements for highly nonlinear and redundant systems. Let $\mathbf{x}(t)$ denote the state of a plant and $\mathbf{u}(t)$ the applied control signal at time t. The state consists of the joint angles $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ of a robot, and the actuator control signals $\mathbf{u}$. If the system would be deterministic, we could express its dynamics as $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$, whereas in the presence of noise we write the dynamics as a stochastic differential equation

$$d\mathbf{x} = \mathbf{f}(\mathbf{x},\mathbf{u})dt + \mathbf{F}(\mathbf{x},\mathbf{u})d\boldsymbol{\omega}. \tag{1}$$

Here, $d\boldsymbol{\omega}$ is assumed to be Brownian motion noise, which is transformed by a possibly state- and control-dependent matrix $\mathbf{F}(\mathbf{x},\mathbf{u})$. We formally specify the problem of carrying out a (reaching) movement as follows: Given an initial state $\mathbf{x}_0$ at time $t = 0$, we seek a control sequence $\mathbf{u}(t)$ such that the system's state is $\mathbf{x}^*$ at time $t = T$. Stochastic optimal control theory approaches the problem by first specifying a cost function which is composed of (i) some evaluation $h(\mathbf{x}(T))$ of the final state, usually penalising deviations from the desired state $\mathbf{x}^*$, and (ii) the accumulated cost $c(t,\mathbf{x},\mathbf{u})$ of sending a control signal $\mathbf{u}$ at time t in state $\mathbf{x}$, typically penalising large motor commands. Introducing a policy $\boldsymbol{\pi}(t,\mathbf{x})$ for selecting $\mathbf{u}(t)$, we can write the expected cost of following that policy from time t as [28]

$$v^{\boldsymbol{\pi}}(t,\mathbf{x}(t)) = \left\langle h(\mathbf{x}(T)) + \int_t^T c(s,\mathbf{x}(s),\boldsymbol{\pi}(s,\mathbf{x}(s)))ds \right\rangle. \tag{2}$$

In OFC one then aims to find the policy $\boldsymbol{\pi}$ that minimises the total expected cost $v^{\boldsymbol{\pi}}(0,\mathbf{x}_0)$. Thus, in contrast to classical control, calculation of the trajectory (planning) and the control signal (execution) is handled in one go. Notably, optimal control provides a principled approach to resolve redundancy: Whereas redundant degrees of freedom are often a nuisance for kinematic path planning, in OFC redundancy can actually be exploited in order to decrease the cost.

If the dynamics $\mathbf{f}$ is linear in $\mathbf{x}$ and $\mathbf{u}$, the cost is quadratic, and the noise is Gaussian, the resulting so-called LQG or LQR[1] problem is convex and can be solved

[1] LQR stands for linear quadratic regulator and describes the optimal controller for linear systems and quadratic costs. LQG also includes an optimal state estimator (under the assumption of Gaussian noise), but because for linear systems estimation and control are independent of each other, LQR and LQG essentially compute the same control law.

analytically [23]. Finding an optimal control policy for nonlinear systems, in contrast, is a much harder challenge. Global solutions could be found in theory by applying dynamic programming methods [5] that are based on the Hamilton-Jacobi-Bellman equations. However, in their basic form these methods rely on a discretisation of the state and action space, an approach that is not viable for large DoF systems. Some research has been carried out on random sampling in a continuous state and action space [24], and it has been suggested that sampling can avoid the curse of dimensionality if the underlying problem is simple enough [2], as is the case if the dynamics and cost functions are very smooth.

A promising alternative to global OFC methods are approaches that compromise between open loop and closed loop optimisation and iteratively compute an optimal trajectory together with a locally valid feedback law. These trajectory-based methods are not directly subject to the curse of dimensionality but still yield locally optimal controllers. Differential dynamic programming (DDP) [9, 12] is a well-known successive approximation technique for solving nonlinear dynamic optimisation problems. This method uses second order approximations of the system dynamics and cost function to perform dynamic programming in the neighbourhood of a nominal trajectory. A more recent algorithm is the Iterative Linear Quadratic Regulator (ILQR) [16]. This algorithm uses iterative linearisation of the nonlinear dynamics around the nominal trajectory, and solves a locally valid LQR problem to iteratively improve the trajectory. However, ILQR is still deterministic and cannot deal with control constraints. A recent extension to ILQR, the Iterative Linear Quadratic Gaussian (ILQG) framework [28], allows to model nondeterministic dynamics by incorporating a Gaussian noise model. Furthermore it supports control constraints like non-negative muscle activations or upper control boundaries and therefore is well suited for the investigation of biologically inspired systems. The ILQG framework has been shown to be computationally significantly more efficient than DDP [16] and also has been previously tested on biologically inspired movement systems and therefore is the favourite approach for us to investigate further.

The ILQG algorithm starts with a time-discretised initial guess of an optimal control sequence and then iteratively improves it w.r.t. the performance criteria in v (eq. 2). From the initial control sequence $\bar{\mathbf{u}}^i$ at the i-iteration, the corresponding state sequence $\bar{\mathbf{x}}^i$ is retrieved using the deterministic forward dynamics $\mathbf{f}$ with a standard Euler integration $\bar{\mathbf{x}}^i_{k+1} = \bar{\mathbf{x}}^i_k + \Delta t\, \mathbf{f}(\bar{\mathbf{x}}^i_k, \bar{\mathbf{u}}^i_k)$. In a next step the discretised dynamics (eq. 1) are linearly approximated around $\bar{\mathbf{x}}^i_k$ and $\bar{\mathbf{u}}^i_k$.:

$$\delta\mathbf{x}_{k+1} = \left(\mathbf{I} + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\bar{\mathbf{x}}_k}\right)\delta\mathbf{x}_k + \Delta t \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\Big|_{\bar{\mathbf{u}}_k}\delta\mathbf{u}_k + \sqrt{\Delta t}\left(\mathbf{F}(\mathbf{u}_k) + \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\Big|_{\bar{\mathbf{u}}_k}\delta\mathbf{u}_k\right)\boldsymbol{\xi}_k. \qquad (3)$$

Similarly to the linearised dynamics in (3) one can derive an approximate cost function which is quadratic in $\delta\mathbf{u}$ and $\delta\mathbf{x}$ (for details please see [28]). Both approximations are formulated as deviations of the current optimal trajectory $\delta\mathbf{x}^i_k = \mathbf{x}^i_k - \bar{\mathbf{x}}^i_k$ and $\delta\mathbf{u}^i_k = \mathbf{u}^i_k - \bar{\mathbf{u}}^i_k$ and therefore form a *local* LQG problem. This linear quadratic problem can be solved efficiently via a modified Ricatti-like set of equations. The

optimisation supports constraints for the control variable $\mathbf{u}$, such as lower and upper bounds. After the optimal control signal correction $\delta\bar{\mathbf{u}}^i$ has been obtained, it can be used to improve the current optimal control sequence for the next iteration using $\bar{\mathbf{u}}_k^{i+1} = \bar{\mathbf{u}}_k^i + \delta\bar{\mathbf{u}}^i$. At last $\bar{\mathbf{u}}_k^{i+1}$ is applied to the system dynamics (eq. 1) and the new total cost along the trajectory is computed. The algorithm stops once the cost v cannot be significantly decreased anymore. After convergence, ILQG returns an optimal control sequence $\bar{\mathbf{u}}$ and a corresponding optimal state sequence $\bar{\mathbf{x}}$ (i.e., trajectory). Along with the optimal open loop parameters $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$, ILQG produces a feedback matrix $\mathbf{L}$ which may serve as optimal feedback gains for correcting local deviations from the optimal trajectory on the plant.

Since the focus of this work is on utilising dynamics learning within ILQG, and its implications to adaptivity, we do not utilise an explicit noise model $\mathbf{F}$ for the sake of clarity of results. In fact it has been shown that a matching feedback control law is only marginally superior to one that is optimised for a deterministic system [28]. We also do not include any model for estimating the state, that is, we assume that noise-free measurements of the system are available (full observability). However an ILQG implementation for systems with partial observability has been been developed recently [17].

3 Adaptive Optimal Feedback Control

As mentioned earlier a major shortcoming of ILQG (and other OFC methods) is the dependence on an analytic form of the system dynamics, which often may be unknown or subject to change. We overcome this limitation by learning an adaptive internal model of the system dynamics using an online, supervised learning method. We consequently use the learned model to derive an ILQG formulation that is computationally efficient, reacts optimally to transient perturbations, and most notably adapts to systematic changes in plant dynamics. We name this algorithm *ILQG with learned dynamics (ILQG–LD)*.

The idea of learning the system dynamics in combination with iterative optimisations of trajectory or policy has been explored previously in the literature, e.g., for learning to swing up a pendulum [4] using some prior knowledge about the form of the dynamics. Similarly, Abeel et al. [1] proposed a hybrid reinforcement learning algorithm, where a policy and an internal model get subsequently updated from "real life" trials. In contrast to their method, however, we employ a second-order optimisation method, and we refine the control law solely from the internal model. To our knowledge, learning dynamics in conjunction with control optimisation has not been studied in the light of adaptability to changing plant dynamics.

From a biological point of view, enabling OFC to be adaptive would allow us to investigate the role of optimal control in human adaptation scenarios. Indeed, adaptation in humans, for example towards external perturbations, is a key property of human motion and is a very active area of research since nearly two decades [21, 22].

3.1 ILQG with Learned Dynamics (ILQG–LD)

In order to eliminate the need for an analytic dynamics model and to make ILQG adaptive, we wish to learn an approximation $\tilde{\mathbf{f}}$ of the real plant forward dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. Assuming our model $\tilde{\mathbf{f}}$ has been coarsely pre-trained, for example by motor babbling, we can refine that model in an online fashion as shown in Fig. 1. For optimising and carrying out a movement, we have to define a cost function (where also the desired final state is encoded), the start state, and the number of discrete time steps because the ILQG algorithm in its current form requires a specified final time. Given an initial torque sequence $\bar{\mathbf{u}}_k^0$, the ILQG iterations can be carried out as described in the Section 2, but utilising the learned model $\tilde{\mathbf{f}}$. This yields a locally optimal control sequence $\bar{\mathbf{u}}_k$, a corresponding desired state sequence $\bar{\mathbf{x}}_k$, and feedback correction gain matrices $\mathbf{L}_k$. Denoting the plant's true state by $\mathbf{x}$, at each time step k, the feedback controller calculates the required correction to the control signal as $\delta\mathbf{u}_k = \mathbf{L}_k(\mathbf{x}_k - \bar{\mathbf{x}}_k)$. We then use the final control signal $\mathbf{u}_k = \bar{\mathbf{u}}_k + \delta\mathbf{u}_k$, the plant's state $\mathbf{x}_k$ and its change $d\mathbf{x}_k$ to update our internal forward model $\tilde{\mathbf{f}}$. As we show in Section 4, we can thus account for (systematic) perturbations and also bootstrap a dynamics model from scratch.

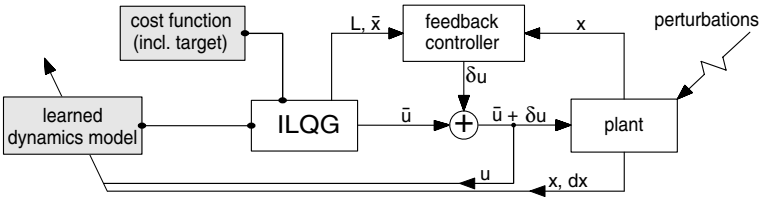


Fig. 1. Illustration of our ILQG–LD learning and control scheme.

3.2 Learning the Dynamics

Various machine learning algorithms could be applied to the robot control learning problem just mentioned. Global learning methods like sigmoid neural networks often suffer from the problem of negative interference, i.e., interference between learning in different parts of the input space when input data distributions are not uniform [20]. Local learning methods, in contrast, represent a function by using small simplistic patches - e.g. first order polynomials. The range of these local patches is determined by weighting kernels, and the number and parameters of the local kernels are adapted during learning to represent the non-linear function. Because any given training sample activates only a few patches, local learning algorithms are robust against global negative interference. This ensures the flexibility of the learned model towards changes in the dynamics properties of the arm (e.g. load, material wear, and different motion). Furthermore the domain of real-time robot control demands certain properties of a learning algorithm, namely fast learning rates and high computational efficiency for predictions and updates if the model is trained

incrementally. Locally Weighted Projection Regression (LWPR) has been shown to exhibit these properties, and to be very efficient for incremental learning of non-linear models in high dimensions [30].

During LWPR training, the parameters of the local models (locality and fit) are updated using incremental Partial Least Squares, and local models can be pruned or added on an as-need basis, for example, when training data is generated in previously unexplored regions. Usually the areas of validity (also termed its receptive field) of each local model are modelled by Gaussian kernels, so their activation or response to a query vector $\mathbf{z} = (\mathbf{x}^T, \mathbf{u}^T)^T$ (combining the *state* and *control* inputs of the forward dynamics $\mathbf{f}$) is given by

$$w_k(\mathbf{z}) = \exp\left(-\frac{1}{2}(\mathbf{z}-\mathbf{c}_k)^T \mathbf{D}_k (\mathbf{z}-\mathbf{c}_k)\right), \tag{4}$$

where $\mathbf{c}_k$ is the centre of the k^{th} linear model and $\mathbf{D}_k$ is its distance metric. Treating each output dimension[2] separately for notational convenience, and ignoring the details about the underlying PLS computations [14], the regression function can be written as

$$\tilde{f}(\mathbf{z}) = \frac{1}{W}\sum_{k=1}^{K} w_k(\mathbf{z})\psi_k(\mathbf{z}), \quad W = \sum_{k=1}^{K} w_k(\mathbf{z}), \tag{5}$$

$$\psi_k(\mathbf{z}) = b_k^0 + \mathbf{b}_k^T(\mathbf{z}-\mathbf{c}_k), \tag{6}$$

where b_k^0 and $\mathbf{b}_k$ denote the offset and slope of the k-th model, respectively.

LWPR learning has the desirable property that it can be carried out online, and moreover, the learned model can be adapted to changes in the dynamics in real-time. A forgetting factor λ [30], which balances the trade-off between preserving what has been learned and quickly adapting to the non-stationarity, can be tuned to the expected rate of external changes. In order to provide some insight, LWPR internally uses update rules within each receptive field of the form $E_{new} = \lambda \cdot E_{old} + w \cdot e_{cur}$. In this example, E is the sufficient statistics for the squared prediction error, and e_{cur} is the error from the current training sample alone, but the same principle applies for other quantities such as the correlation between input and output data. In this way, after N updates to a receptive field, the original value of the sufficient statistics has been down-weighted (or forgotten) by a factor of λ^N. As we will see later, the factor λ can be used to model biologically realistic adaptive behaviour to external force-fields.

3.3 Reducing the Computational Cost

So far, we have shown how the problem of unknown or changing system dynamics can be addressed within ILQG–LD. Another important issue to discuss is the

[2] In the case of learning forward dynamics, the target values are the joint accelerations. We effectively learn a separate model for each joint.

computational complexity. The ILQG framework has been shown to be the most effective locally optimal control method in terms of convergence speed and accuracy [15]. Nevertheless the computational cost of ILQG remains daunting even for simple movement systems, preventing their application to real-time optimal motion planning for large DoF systems. A large part of the computational cost arises from the linearisation of the system dynamics, which involves repetitive calculation of the system dynamics' derivatives $\partial\mathbf{f}/\partial\mathbf{x}$ and $\partial\mathbf{f}/\partial\mathbf{u}$. When the analytical form of these derivatives is not available, they must be approximated using finite differences. The computational cost of such an approximation scales linearly with the sum of the dimensionalities of $\mathbf{x} = (\mathbf{q};\dot{\mathbf{q}})$ and $\mathbf{u} = \boldsymbol{\tau}$ (i.e., $3N$ for an N DoF joint torque controlled robot). In simulations, our analysis show that for the 2 DoF manipulator, 60% of the total ILQG computations can be attributed to finite differences calculations. For a 6 DoF arm, this rises to 80%.

Within our ILQG–LD scheme, we can avoid finite difference calculations and rather use the analytic derivatives of the learned model, as has similarly been proposed in [3]. Differentiating the LWPR predictions (5) with respect to $\mathbf{z} = (\mathbf{x};\mathbf{u})$ yields terms

$$\frac{\partial \tilde{f}(\mathbf{z})}{\partial \mathbf{z}} = \frac{1}{W}\sum_k \left(\frac{\partial w_k}{\partial \mathbf{z}}\psi_k(\mathbf{z}) + w_k\frac{\partial \psi_k}{\partial \mathbf{z}}\right) - \frac{1}{W^2}\sum_k w_k(\mathbf{z})\psi_k(\mathbf{z})\sum_l \frac{\partial w_l}{\partial \mathbf{z}} \tag{7}$$

$$= \frac{1}{W}\sum_k \left(-\psi_k w_k \mathbf{D}_k(\mathbf{z}-\mathbf{c}_k) + w_k\mathbf{b}_k\right) + \frac{\tilde{f}(\mathbf{z})}{W}\sum_k w_k\mathbf{D}_k(\mathbf{z}-\mathbf{c}_k) \tag{8}$$

for the different rows of the Jacobian matrix $\begin{pmatrix}\partial\tilde{\mathbf{f}}/\partial\mathbf{x}\\ \partial\tilde{\mathbf{f}}/\partial\mathbf{u}\end{pmatrix} = \frac{\partial}{\partial\mathbf{z}}(\tilde{f}_1,\tilde{f}_2,\ldots\tilde{f}_N)^T$.

Table 1 illustrates the computational gain (mean CPU time per ILQG iteration) across 3 test manipulators – highlighting added benefits for more complex systems. On a notebook running at 1.6 GHz, the average CPU times for a *complete* ILQG trajectory using the analytic method are 0.8 sec (2 DoF), 1.9 sec (6 DoF), and 9.8 sec (12 DoF), respectively. Note that LWPR is a highly parallelisable algorithm: Since the local models learn independently of each other, the respective computations can be distributed across multiple processors or processor cores, which can yield a further significant performance gain [14].

Table 1. CPU time for one ILQG–LD iteration (sec).

	finite differences	**analytic Jacobian**	**improvement factor**
2 DoF	0.438	0.193	2.269
6 DoF	4.511	0.469	9.618
12 DoF	29.726	1.569	18.946

4 Evaluation

In this section we evaluate ILQG–LD in several setups with increasing complexity. We start with joint torque controlled manipulators setups first, which will be analysed under stationary and non-stationary conditions. We then present ILQG–LD results from an antagonistic humanoid arm model which embodies the challenge of large redundancies in the dynamics domain.

All simulations are performed with the Matlab Robotics Toolbox [7]. This simulation model computes the non-linear plant dynamics using standard equations of motion. For an N-DoF manipulator the joint torques $\boldsymbol{\tau}$ are given by

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{b}(\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}), \tag{9}$$

where $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the joint angles and joint velocities respectively; $\mathbf{M}(\mathbf{q})$ is the N-dimensional symmetric joint space inertia matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ accounts for Coriolis and centripetal effects, $\mathbf{b}(\dot{\mathbf{q}})$ describes the viscous and Coulomb friction in the joints, and $\mathbf{g}(\mathbf{q})$ defines the gravity loading depending on the joint angles $\mathbf{q}$ of the manipulator.

We study movements for a fixed motion duration of one second, which we discretise into $K = 100$ steps ($\Delta t = 0.01$s). The manipulator starts at an initial position $\mathbf{q}_0$ and reaches towards a target $\mathbf{q}_{tar}$. During movement we wish to minimise the energy consumption of the system. We therefore use the cost function

$$v = w_p \,|\, \mathbf{q}_K - \mathbf{q}_{tar} \,|^2 + w_v \,|\, \dot{\mathbf{q}}_K \,|^2 + w_e \sum_{k=0}^{K} |\, \mathbf{u}_k \,|^2 \Delta t, \tag{10}$$

where the factors for the target position accuracy (w_p), for the zero end-point velocity (w_v), and for the energy term (w_e) weight the importance of each component. We compare the control results of ILQG–LD and ILQG with respect to the number of iterations, the end point accuracy and the generated costs. In this paper we will refer to *cost* as total cost defined in (10) and to *running cost* to the energy consumption only, i.e., the summation term in (10).

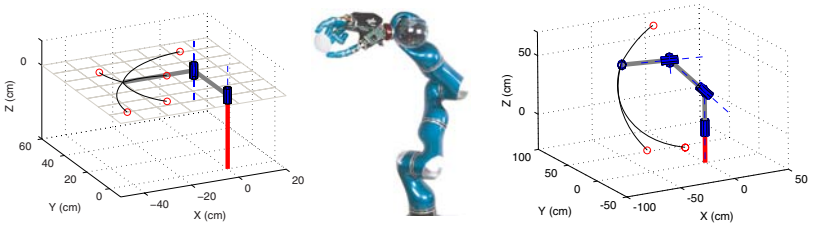


Fig. 2. Two different joint-torque controlled manipulator models with selected targets (circles) and ILQG generated trajectories as benchmark data. All models are simulated using the Matlab Robotics Toolbox. Left: 2 DoF planar manipulator model; Middle: picture of the Kuka Light-Weight Robot arm (LWR); Right: Simulated 6 DoF LWR model (without hand).

4.1 Planar Arm with 2 Torque-Controlled Joints

The first setup (Fig. 2 left) is a horizontally planar 2 DoF manipulator similar to the one used in [28]. The arm is controlled by directly commanding joint torques. This low DoF system is ideal for performing extensive (quantitative) comparison studies and to test the manipulator under controlled perturbations and force fields during planar motion.

4.1.1 Stationary Dynamics

First, we compared the characteristics of ILQG–LD and ILQG (both operated in open loop mode) in the case of stationary dynamics without any noise in the 2 DoF plant. Fig. 3 shows three trajectories generated by learned models of different predictive quality, which is reflected by the different normalised means square errors (nMSE) on test data. The nMSE is defined as $nmse(y,\tilde{y}) = \frac{1}{n\sigma_y^2}\sum_{i=1}^{n}(y_i-\tilde{y}_i)^2$ where y is the desired output data set of size n and $\tilde{y}$ represents the LWPR predictions. The nMSE takes into account the output distribution of the data (variance σ_y^2 in the data) and therefore produces a "dimensionless" error measure. As one would expect, the quality of the model plays an important role for the final cost, the number of ILQG–LD iterations, and the final target distances (cf. the table within Fig. 3). For the final learned model, we observe a striking resemblance with the analytic ILQG performance.

Next, we carried out a reaching task to 5 reference targets covering a wide operating area of the planar arm. To simulate control dependent noise, we contaminated the commands $\mathbf{u}$ just before feeding them into the plant, using Gaussian noise with 50% of the variance of the signal $\mathbf{u}$. We then generated motor commands to move the system towards the targets, both with and without the feedback controller. As expected, closed loop control (utilising gain matrices $\mathbf{L}_k$) is superior to open loop operation regarding reaching accuracy. Fig. 4 depicts the performance of ILQG–LD and ILQG under both control schemes. Averaged over all trials, both methods show similar endpoint variances and behaviour which is statistically indistinguishable.

ILQG–LD	**(L)**	**(M)**	**(H)**	**ILQG**
No. of training points	111	146	276	–
Prediction error (nMSE)	0.80	0.50	0.001	–
Iterations	19	17	5	4
Cost	2777.36	1810.20	191.91	192.07
Eucl. target distance (cm)	19.50	7.20	0.40	0.01

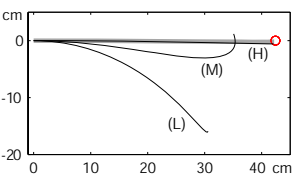
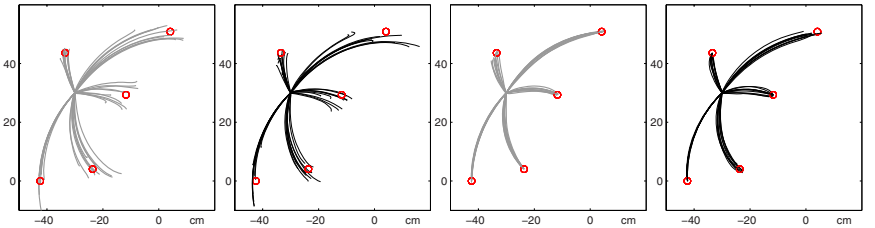


Fig. 3. Behaviour of ILQG–LD for learned models of different quality: (L)-Low, (M)-Medium, (H)-High. Right: Trajectories in task space produced by ILQG–LD (black lines) and ILQG (grey line).



open loop ILQG
$d = 2.71 \pm 1.84$ cm

open loop ILQG–LD
$d = 3.20 \pm 1.60$ cm

closed loop ILQG
$d = 0.19 \pm 0.001$ cm

closed loop ILQG–LD
$d = 0.18 \pm 0.001$ cm

Fig. 4. Illustration of the target reaching performances for the planar 2 DoF in the presence of strong control dependent noise, where d represents the average Euclidean distance to the five reference targets.

4.1.2 Non-stationary Dynamics

A major advantage of ILQG–LD is that it does not rely on an accurate analytic dynamics model; consequently, it can adapt *on the fly* to external perturbations and to changes in the plant dynamics that may result from altered morphology or wear and tear. We carried out adaptive reaching experiments in our simulation similar to the human manipulandum experiments in [21]. First, we generated a constant unidirectional force field (FF) acting perpendicular to the reaching movement (see Fig. 5). Using the ILQG–LD models from the previous experiments, the manipulator gets strongly deflected when reaching for the target because the learned dynamics model cannot account for the *spurious* forces. However, using the resultant deflected trajectory (100 data points) as training data, updating the dynamics model online brings the manipulator nearer to the target with each new trial. We repeated this procedure until the ILQG–LD performance converged successfully. At that point, the internal model successfully accounts for the change in dynamics caused by the FF. Then, removing the FF results in the manipulator overshooting to the other side, compensating for a non-existing FF. Just as before, we re-adapted the dynamics online over repeated trials.

Fig. 5 summarises the results of the sequential adaptation process just described. The closed loop control scheme clearly converges faster than the open loop scheme, which is mainly due to the OFC's desirable property of always correcting the system towards the target. Therefore, it produces more relevant dynamics training data. Furthermore, we can accelerate the adaptation process significantly by tuning the forgetting factor λ, allowing the learner to weight the importance of new data more strongly [30]. A value of $\lambda = 0.95$ produces significantly faster adaptation results than the default of $\lambda = 0.999$. As a follow-up experiment, we made the force field dependent on the velocity $\mathbf{v}$ of the end-effector, i.e. we applied a force

$$\mathbf{F} = \mathbf{B}\mathbf{v}, \quad \text{with} \quad \mathbf{B} = \begin{pmatrix} 0 & 50 \\ -50 & 0 \end{pmatrix} Nm^{-1}s \tag{11}$$

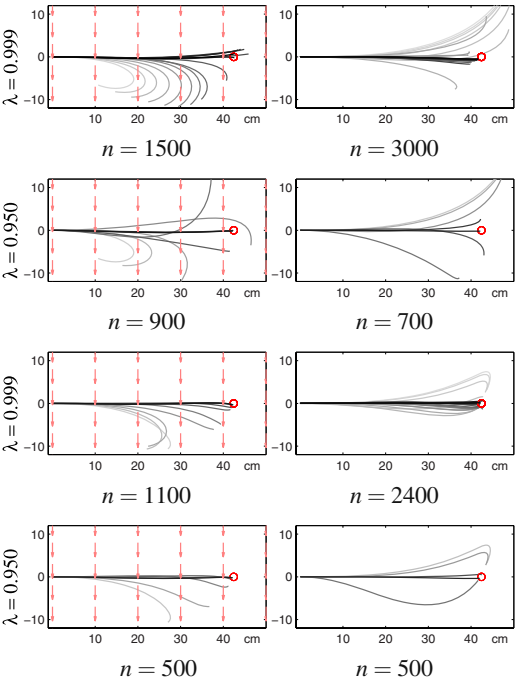


Fig. 5. Illustration of adaptation experiments for open loop (rows 1,2) and closed loop (rows 3,4) ILQG–LD. Arrows depict the presence of a (constant) force field; n represents the number of training points required to successfully update the internal LWPR dynamics model. Darker lines indicate better trained models, corresponding to later trials in the adaption process.

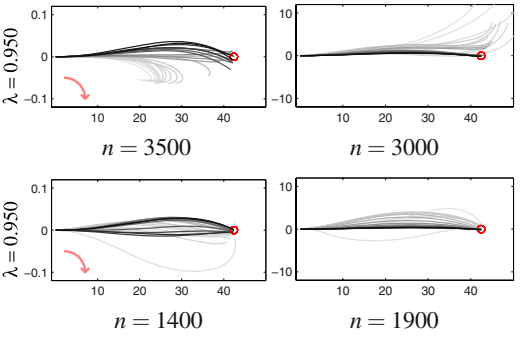


Fig. 6. Adaptation to a velocity-dependent force field (as indicated by the bent arrow) and re-adaptation after the force field is switched off (right column). Top: open loop. Bottom: closed loop.

to the end-effector. The results are illustrated in Fig. 6: For the more complex FF, more iterations are needed in order to adapt the model, but otherwise ILQG–LD shows a similar behaviour as for the constant FF. Interestingly, the overshoot behaviour depicted in Fig. 5 and 6 has been observed similarly in human adaptation experiments where it was referred to as "after effects" [21]. We believe this to be an interesting insight for future investigation of ILQG–LD and its role in modeling sensorimotor adaptation data in the (now extensive) human reach experimental paradigm [22].

4.2 Anthropomorphic 6 DoF Robot Arm

Our next experimental setup is a 6 DoF manipulator (Fig. 2, right), the physical parameters (i.e., link inertia, mass, etc.) of which are a faithful model of the first 6 links of the *Kuka Light-Weight Robot* (LWR).

Using this arm, we studied reaching targets specified in *Cartesian* coordinates $\mathbf{r} \in \mathbb{R}^3$ in order to highlight the redundancy resolution capability and trial-to-trial variability in large DoF systems. We set up the cost function (cf. eq. 10) as

$$v = w_p \,|\, \mathbf{r}(\mathbf{q}_K) - \mathbf{r}_{tar} \,|^2 + w_v \,|\, \dot{\mathbf{q}}_K \,|^2 + w_e \sum_{k=0}^{K} |\, \mathbf{u}_k \,|^2 \Delta t, \tag{12}$$

where $\mathbf{r}(\mathbf{q})$ denotes the end-effector position as calculated from forward kinematics. It should be noted that for the specific kinematic structure of this arm, this 3D position depends only on the first 4 joint angles. Joints 5 and 6 only change the orientation of the end-effector[3], which does not play a role in our reaching task and correspondingly in the cost function. In summary, our arm has *one redundant* and further *two irrelevant* degrees of freedom for this task.

Table 2. Comparison of the performance of ILQG–LD and ILQG for controlling a 6 DoF robot arm. We report the number of iterations required to compute the control law, the average running cost, and the average Euclidean distance **d** to the three reference targets.

	ILQG			ILQG–LD		
Targets	**Iter.**	**Run. cost**	**d (cm)**	**Iter.**	**Run. cost**	**d (cm)**
(a)	51	**18.50** $\pm$ 0.13	**2.63** $\pm$ 1.63	51	**18.32** $\pm$ 0.55	**1.92** $\pm$ 1.03
(b)	61	**18.77** $\pm$ 0.25	**1.32** $\pm$ 0.69	99	**18.65** $\pm$ 1.61	**0.53** $\pm$ 0.20
(c)	132	**12.92** $\pm$ 0.04	**1.75** $\pm$ 1.30	153	**12.18** $\pm$ 0.03	**2.00** $\pm$ 1.02

Similar to the 2 DoF experiments, we bootstrapped a forward dynamics model through extensive data collection (i.e., motor babbling). Next, we used ILQG–LD (closed loop, with noise) to train our dynamics model online until it converged to stable reaching behaviour. Fig. 7 depicts reaching trials, 20 for each reference target,

[3] The same holds true for the 7th joint of the original LWR arm.

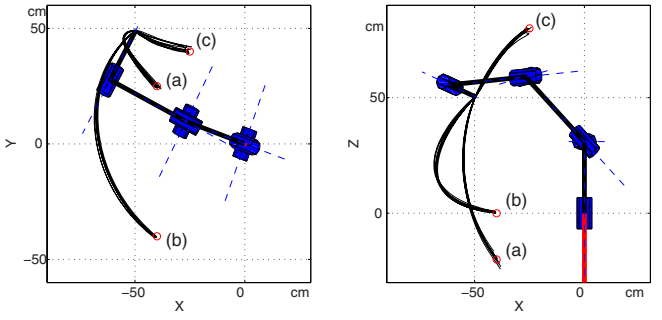


Fig. 7. Illustration of the trial-to-trial variability of the 6-DoF arm when reaching towards target (a,b,c). Left: top-view, right: side-view.

using ILQG–LD with the final learned model. Table 2 quantifies the performance. The targets are reached reliably and no statistically significant differences can be spotted between ILQG–LD and ILQG. An investigation of the trials in *joint angle* space also shows similarities. Fig. 8 depicts the 6 joint angle trajectories for the 20 reaching trials towards target (c). Please note the high variance of the joint angles especially for the irrelevant joints 5 and 6, which nicely show that task irrelevant errors are not corrected unless they adversely affect the task (minimum intervention principle of OFC). Moreover, the joint angle variances (trial-to-trial variability) between the ILQG–LD and ILQG trials are in a similar range, indicating an equivalent corrective behaviour – the shift of the absolute variances can be explained by the slight mismatch between the learned and analytical dynamics. We can conclude from our results that ILQG–LD scales up very well to 6 DoF, not suffering from any losses in terms of accuracy, cost or convergence behaviour. Furthermore, its computational cost is significantly lower than the one of ILQG.

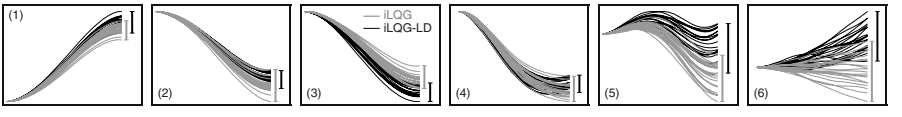


Fig. 8. Illustration of the trial-to-trial variability in the joint angles (1–6) over time when reaching towards target (c). Grey lines indicate ILQG, black lines stem from ILQG–LD.

4.3 Antagonistic Planar Arm

In order to analyse ILQG–LD in a dynamically redundant scenario, we studied a two DoF planar human arm model, which is actuated by four single-joint and two double-joint antagonistic muscles (Fig. 9 left). The arm model described in this section is based on [13]. Although kinematically simple, the system is over-actuated and therefore an interesting testbed for our control scheme, because large

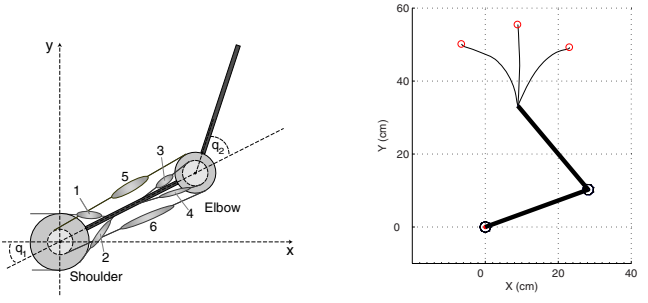


Fig. 9. Left: Human arm model with 6 muscles (adapted from [13]). Right: Same arm model with selected targets (circles) and ILQG generated trajectories as benchmark data. The physics of the model is simulated using the Matlab Robotics Toolbox [7].

redundancies in the dynamics have to be resolved. The dimensionality of the control signals makes adaptation processes (e.g., to external force fields) quite demanding. Indeed this arm poses a harder learning problem than the 6-DoF manipulator of the previous section, because the muscle-based actuation makes the dynamics less linear.

As before the dynamics of the arm is in part based on standard equations of motion, given by

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}. \tag{13}$$

Given the antagonistic muscle-based actuation, we cannot command joint torques directly, but rather we have to calculate effective torques from the muscle activations $\mathbf{u}$. For the present model the corresponding transfer function is given by

$$\boldsymbol{\tau}(\mathbf{q},\dot{\mathbf{q}},\mathbf{u}) = -\mathbf{A}(\mathbf{q})^{\mathbf{T}}\mathbf{T}(\mathbf{l},\dot{\mathbf{l}},\mathbf{u}), \tag{14}$$

where $\mathbf{A}$ represents the moment arm. For simplicity, we assume $\mathbf{A}$ to be constant and independent of the joint angles $\mathbf{q}$:

$$\mathbf{A}(\mathbf{q}) = \mathbf{A} = \begin{pmatrix} a_1 & a_2 & 0 & 0 & a_5 & a_6 \\ 0 & 0 & a_3 & a_4 & a_7 & a_8 \end{pmatrix}^T. \tag{15}$$

The muscle lengths $\mathbf{l}$ depend on the joint angles $\mathbf{q}$ through the affine relationship $\mathbf{l} = \mathbf{l}_m - \mathbf{A}\mathbf{q}$, which also implies $\dot{\mathbf{l}} = -\mathbf{A}\dot{\mathbf{q}}$. The term $\mathbf{T}(\mathbf{l},\dot{\mathbf{l}},\mathbf{u})$ in (14) denotes the muscle tension, for which we follow the Kelvin-Voight model [19] and define:

$$\mathbf{T}(\mathbf{l},\dot{\mathbf{l}},\mathbf{u}) = \mathbf{K}(\mathbf{u})\big(\mathbf{l}_r(\mathbf{u}) - \mathbf{l}\big) - \mathbf{B}(\mathbf{u})\dot{\mathbf{l}}. \tag{16}$$

Here, $\mathbf{K}(\mathbf{u})$, $\mathbf{B}(\mathbf{u})$, and $\mathbf{l}_r(\mathbf{u})$ denote the muscle stiffness, the muscle viscosity and the muscle rest length, respectively. Each of these terms depends linearly on the motor commands $\mathbf{u}$, as given by

$$\mathbf{K}(\mathbf{u}) = diag(\mathbf{k}_0 + k\mathbf{u}), \qquad \mathbf{B}(\mathbf{u}) = diag(\mathbf{b}_0 + b\mathbf{u}), \qquad \mathbf{l}_r(\mathbf{u}) = \mathbf{l}_0 + r\mathbf{u}. \tag{17}$$

The elasticity coefficient k, the viscosity coefficient b, and the constant r are given from the muscle model. The same holds true for $\mathbf{k}_0$, $\mathbf{b}_0$, and $\mathbf{l}_0$, which are the intrinsic elasticity, viscosity and rest length for $\mathbf{u} = \mathbf{0}$, respectively. For the exact values of these coefficients please refer to [13]. ILQG has been applied previously to similar antagonistic arm models, that are slightly more complex. Most notably, non-constant moment arms $\mathbf{A}(\mathbf{q})$, stochastic control signals, and a muscle activation dynamics which increase the dimensionality of the state space have been used [15].

Please note that in contrast to standard torque-controlled robots, in our arm model the dynamics (13) is *not* linear in the control signals, since $\mathbf{u}$ enters (16) quadratically. We follow the same cost function as before (eq. 10) and the same fixed motion duration of one second. Here we discretise the time into $K = 50$ steps ($\Delta t = 0.02$s).

4.3.1 Stationary Dynamics

In order to make ILQG–LD converge for our three reference targets we coarsely pretrained our LWPR model with a focus on a wide coverage of the workspace. The training data are given as tuples consisting of $(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u})$ as inputs (10 dimensions in total), and the observed joint accelerations $\ddot{\mathbf{q}}$ as the desired two-dimensional output. We stopped training once the normalised mean squared error (nMSE) in the predictions reached ≤ 0.005. At this point LWPR had seen $1.2 \cdot 10^6$ training data points and had acquired 852 receptive fields, which is in accordance with the previously discussed high non-linearity of the plant dynamics.

We carried out a reaching task to the 3 reference targets (Fig. 9, right) using the feedback controller (feedback gain matrix $\mathbf{L}$) that falls out of ILQG(-LD). To compare the stability of the control solution, we simulated control dependent noise by contaminating the muscle commands $\mathbf{u}$ just before feeding them into the plant. We applied Gaussian noise with 50% of the variance of the signal $\mathbf{u}$.

Fig. 10 depicts the generated control signals and the resulting performance of ILQG–LD and ILQG over 20 reaching trials per target. Both methods show similar

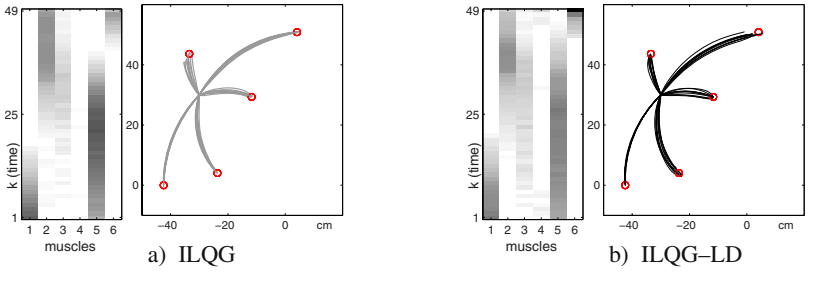


Fig. 10. Illustration of an optimised control sequence (left) and resulting trajectories (right) when using a) the known analytic dynamics model and b) the LWPR model learned from data. The control sequences (left target only) for each muscle (1–6) are drawn from bottom to top, with darker grey levels indicating stronger muscle activation.

Table 3. Comparison of the performance of ILQG–LD and ILQG with respect to the number of iterations required to compute the control law, the average running cost, and the average Euclidean distance to the three reference targets (left, center, right).

	ILQG			ILQG–LD		
Targets	**Iter.**	**Run. cost**	**d (cm)**	**Iter.**	**Run. cost**	**d (cm)**
Center	19	**0.0345** $\pm$ 0.0060	**0.11** $\pm$ 0.07	14	**0.0427** $\pm$ 0.0069	**0.38** $\pm$ 0.22
Left	40	**0.1873** $\pm$ 0.0204	**0.10** $\pm$ 0.06	36	**0.1670** $\pm$ 0.0136	**0.21** $\pm$ 0.16
Right	41	**0.1858** $\pm$ 0.0202	**0.57** $\pm$ 0.49	36	**0.1534** $\pm$ 0.0273	**0.19** $\pm$ 0.12

endpoint variances and trajectories which are in close match. As can be seen from the visualisation of the control sequences, antagonistic muscles (i.e., muscle pairs 1/2, 3/4, and 5/6 in Fig. 9, left) are never activated at the same time. This is a direct consequence of the cost function, which penalises co-contraction as a waste of energy. Table 3 quantifies the control results of ILQG–LD and ILQG for each target with respect to the number of iterations, the generated running costs and the end point accuracy.

4.3.2 Adaptation Results

As before we carried out adaptive reaching experiments (towards the center target) and we generated a constant unidirectional force field (FF) acting perpendicular to the reaching movement (see Fig. 11). Using the ILQG–LD model from the previous experiment, the manipulator gets strongly deflected when reaching for the target because the learned dynamics model cannot yet account for the "spurious" forces. However, using the resultant deflected trajectory as training data, updating the dynamics model online brings the manipulator nearer to the target with each new trial. In order to produce enough training data, as is required for a successful adaptation, we generated 20 slightly jittered versions of the optimised control sequences, ran these on the plant, and trained the LWPR model with the resulting 50 samples each. We repeated this procedure until the ILQG–LD performance converged successfully, which was the case after 27000 training samples. At that point, the internal model successfully accounted for the change in dynamics caused by the FF. Then, we switched off the FF while continuing to use the adapted LWPR model. This resulted in an overshooting of the manipulator to the other side, trying to compensate for non-existing forces. Just as before, we re-adapted the dynamics online over repeated trials. The arm reached the target again after 7000 training points. One should note that compared to the initial *global* motor babbling, where we required $1.2 \cdot 10^6$ training data points, for the *local* (re-)adaptation we need only a fraction of the data points.

Fig. 11 summarises the results of the sequential adaptation process just described. Please note how the optimised *adapted* control sequence contains considerably stronger activations of the extensor muscles responsible for pulling the arm to the right (denoted by "2" and "6" in Fig. 9), while still exhibiting practically no co-contraction.

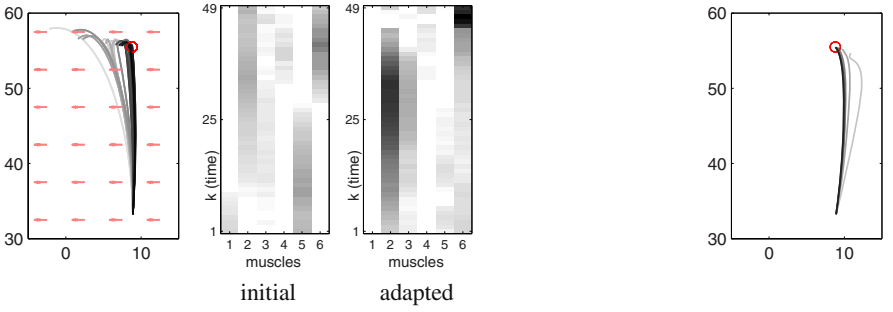


Fig. 11. Left: Adaptation to a unidirectional constant force field (indicated by the arrows). Darker lines indicate better trained models. In particular, the left-most trajectory corresponds to the "initial" control sequence, which was calculated using the LWPR model (from motor babbling) *before* the adaptation process. The fully "adapted" control sequence results in a nearly straight line reaching movement. Right: Resulting trajectories during re-adaptation after the force field has been switched off (i.e., after effects).

5 Discussion

In this work we introduced ILQG–LD, a method that realises adaptive optimal feedback control by incorporating a learned dynamics model into the ILQG framework. Most importantly, we carried over the favourable properties of ILQG to more realistic control problems where the analytic dynamics model is often unknown, difficult to estimate accurately or subject to changes. As with ILQG control, redundancies are implicitly resolved by the OFC framework through a cost function, eliminating the need for a separate trajectory planner and inverse kinematics/dynamics computation.

Utilising the derivatives (8) of the learned dynamics model $\tilde{\mathbf{f}}$ avoids expensive finite difference calculations during the dynamics linearisation step of ILQG. This significantly reduces the computational complexity, allowing the framework to scale to larger DoF systems. We empirically showed that ILQG–LD performs reliably in the presence of noise and that it is adaptive with respect to systematic changes in the dynamics; hence, the framework has the potential to provide a unifying tool for modelling (and informing) non-linear sensorimotor adaptation experiments even under complex dynamic perturbations. As with ILQG control, redundancies are implicitly resolved by the OFC framework through a cost function, eliminating the need for a separate trajectory planner and inverse kinematics/dynamics computation.

Our future work will concentrate on implementing the ILQG–LD framework on an anthropomorphic hardware – this will not only explore an alternative control paradigm, but will also provide the only viable and principled control strategy for the biomorphic *variable stiffness* based highly redundant actuation system that we are currently developing. Indeed, exploiting this framework for understanding OFC and its link to biological motor control is another very important strand.

References

1. Abbeel, P., Quigley, M., Ng, A.Y.: Using inaccurate models in reinforcement learning. In: Proc. Int. Conf. on Machine Learning (ICML), pp. 1–8 (2006)
2. Atkeson, C.G.: Randomly sampling actions in dynamic programming. In: Proc. Int. Symp. on Approximate Dynamic Programming and Reinforcement Learning, pp. 185–192 (2007)
3. Atkeson, C.G., Moore, A., Schaal, S.: Locally weighted learning for control. AI Review 11, 75–113 (1997)
4. Atkeson, C.G., Schaal, S.: Learning tasks from a single demonstration. In: Proc. Int. Conf. on Robotics and Automation (ICRA), Albuquerque, New Mexico, pp. 1706–1712 (1997)
5. Bertsekas, D.P.: Dynamic programming and optimal control. Athena Scientific, Belmont (1995)
6. Conradt, J., Tevatia, G., Vijayakumar, S., Schaal, S.: On-line learning for humanoid robot systems. In: Proc. Int. Conf. on Machine Learning (ICML), pp. 191–198 (2000)
7. Corke, P.I.: A robotics toolbox for MATLAB. IEEE Robotics and Automation Magazine 3(1), 24–32 (1996)
8. D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: Proc. Int. Conf. on Intelligence in Robotics and Autonomous Systems (IROS), Hawaii, pp. 298–303 (2001)
9. Dyer, P., McReynolds, S.: The Computational Theory of Optimal Control. Academic Press, New York (1970)
10. Flash, T., Hogan, N.: The coordination of arm movements: an experimentally confirmed mathematical model. Journal of Neuroscience 5, 1688–1703 (1985)
11. Grebenstein, M., van der Smagt, P.: Antagonism for a highly anthropomorphic hand-arm system. Advanced Robotics 22(1), 39–55 (2008)
12. Jacobson, D.H., Mayne, D.Q.: Differential Dynamic Programming. Elsevier, New York (1970)
13. Katayama, M., Kawato, M.: Virtual trajectory and stiffness ellipse during multijoint arm movement predicted by neural inverse model. Biological Cybernetics 69, 353–362 (1993)
14. Klanke, S., Vijayakumar, S., Schaal, S.: A library for locally weighted projection regression. Journal of Machine Learning Research 9, 623–626 (2008)
15. Li, W.: Optimal Control for Biological Movement Systems. PhD dissertation, University of California, San Diego (2006)
16. Li, W., Todorov, E.: Iterative linear-quadratic regulator design for nonlinear biological movement systems. In: Proc. 1st Int. Conf. Informatics in Control, Automation and Robotics (2004)
17. Li, W., Todorov, E.: Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system. International Journal of Control 80(9), 14391–14453 (2007)
18. Nguyen-Tuong, D., Peters, J., Seeger, M., Schoelkopf, B.: Computed torque control with nonparametric regressions techniques. In: American Control Conference (2008)
19. Özkaya, N., Nordin, M.: Fundamentals of biomechanics: equilibrium, motion, and deformation. Van Nostrand Reinhold, New York (1991)
20. Schaal, S.: Learning Robot Control. In: The handbook of brain theory and neural networks, pp. 983–987. MIT Press, Cambridge (2002)
21. Shadmehr, R., Mussa-Ivaldi, F.A.: Adaptive representation of dynamics during learning of a motor task. The Journal of Neurosciene 14(5), 3208–3224 (1994)

22. Shadmehr, R., Wise, S.P.: The Computational Neurobiology of Reaching and Ponting. MIT Press, Cambridge (2005)
23. Stengel, R.F.: Optimal control and estimation. Dover Publications, New York (1994)
24. Thrun, S.: Monte carlo POMDPs. In: Advances in Neural Information Processing Systems (NIPS), pp. 1064–1070 (2000)
25. Todorov, E.: Optimality principles in sensorimotor control. Nature Neuroscience 7(9), 907–915 (2004)
26. Todorov, E., Jordan, M.: Optimal feedback control as a theory of motor coordination. Nature Neuroscience 5, 1226–1235 (2002)
27. Todorov, E., Jordan, M.: A minimal intervention principle for coordinated movement. In: Advances in Neural Information Processing Systems (NIPS), pp. 27–34. MIT Press, Cambridge (2003)
28. Todorov, E., Li, W.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proc. of the American Control Conference (2005)
29. Uno, Y., Kawato, M., Suzuki, R.: Formation and control of optimal trajectories in human multijoint arm movements: minimum torque-change model. Biological Cybernetics 61, 89–101 (1989)
30. Vijayakumar, S., D'Souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Computation 17, 2602–2634 (2005)
31. Vijayakumar, S., D'Souza, A., Shibata, T., Conradt, J., Schaal, S.: Statistical learning for humanoid robots. Autonomous Robots 12(1), 55–69 (2002)
32. Wolf, S., Hirzinger, G.: A new variable stiffness design: Matching requirements of the next robot generation. In: Proc. Int. Conf. on Robotics and Automation (ICRA), pp. 1741–1746 (2008)

The SURE_REACH Model for Motor Learning and Control of a Redundant Arm: From Modeling Human Behavior to Applications in Robotics

Oliver Herbort, Martin V. Butz, and Gerulf Pedersen

Abstract. The recently introduced neural network SURE_REACH (sensorimotor unsupervised redundancy resolving control architecture) models motor cortical learning and control of human reaching movements. The model learns redundant, internal body models that are highly suitable to flexibly invoke effective motor commands. The encoded redundancy is used to adapt behavior flexible to situational constraints without the need for further learning. These adaptations to specific tasks or situations are realized by a neurally generated movement plan that adheres to various end-state or trajectory-related constraints. The movement plan can be implemented by proprioceptive or visual closed-loop control. This chapter briefly reviews the literature on computational models of motor learning and control and gives a description of SURE_REACH and its neural network implementation. Furthermore, we relate the model to human motor learning and performance and discuss its neural foundations. Finally, we apply the model to the control of a dynamic robot platform. In sum, SURE_REACH grounds highly flexible task-dependent behavior on a neural network framework for unsupervised learning. It accounts for the neural processes that underlie fundamental aspects of human behavior and is well applicable to the control of robots.

Oliver Herbort
Universität Würzburg, Department of Psychology, Röntgenring 11,
97070 Würzburg, Germany
e-mail: oliver.herbort@psychologie.uni-wuerzburg.de

Martin V. Butz
Universität Würzburg, Department of Psychology, Röntgenring 11,
97070 Würzburg, Germany
e-mail: butz@psychologie.uni-wuerzburg.de

Gerulf Pedersen
Universität Würzburg, Department of Psychology, Röntgenring 11,
97070 Würzburg, Germany
e-mail: gerulf@psychologie.uni-wuerzburg.de

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 85–106.
springerlink.com

1 Introduction

Virtually all of the brain's capabilities, from the simplest automatic mechanisms to the most complex cognitive operations, are mediated by the human motor system. Only movements of our bodies can cause consistent manipulation of the environment. Thus, understanding the human motor system is of paramount importance to understand human behavioral control and the involved cognitive processes. However, many open questions remain in our understanding of how the brain translates our will into actual body movements.

Despite this lack of explicit knowledge, our brains transform goals into movements exceedingly well and with astonishing ease. For example, movements are generally executed in a fast, accurate, and energy conserving way [23, 71]. Multiple sources of information are integrated when forming goals or during the control of ongoing movements [18, 47]. If obstacles block the way or our own mobility is restricted, the motor system adapts to these task-dependent constraints from one moment to the other and it even aligns the way we carry out current movements to facilitate future actions [17, 22, 61]. On top of these facts, it needs to be remembered that all these capabilities are acquired by unsupervised motor learning. In the CNS, cortical motor areas have been associated with the unsupervised acquisition of motor behavior and new motor skills [19, 25, 32].

The computational principles underlying motor learning and control are not yet very well understood. A review of existing computational models reveals that most models fall in either of two groups. Some models scrutinize how the fully developed motor system might work offering an account for the flexibility of human behavior, but these models do not account for motor learning (e.g. [14, 63, 64]). Other models focus mainly on the acquisition of motor control structures but they cannot explain the flexibility of human behavioral control (e.g. [3, 4, 7, 9, 40, 43, 49]).

The SURE_REACH neural network model of the cortical control of human reaching aims at integrating these aspects [10, 28, 30, 31]: It offers an account for unsupervised motor learning; It explains how humans can flexibly adapt to changing task constraints; And finally, it is based on plausible mechanisms and structures on the neural as well as functional level.

This is achieved by providing neural learning and control mechanisms that extracts as much information as possible about the relationship between motor commands and changes in sensory input. The abundance of information enables to flexibly generate novel movement pattern and thus adjust quickly to changing situations (c.f. [73] in this volume). This approach offers unprecedented flexibility in movement control and offers new perspectives on understanding motor learning, in both humans and robots (c.f. [60]; [66] in this volume).

In the remainder of this article, we first review related models of motor learning and control. Next, SURE_REACH is described, including the mathematical formulations of spatial representations, learning mechanisms, and the motor control networks. After that, several examples show that the model is well able to account for motor learning and flexible behavior. Additionally, the biological and theoretical foundations of the model are discussed. Finally, we review an application of the underlying framework to the flexible control of a robot arm.

2 Theories of Motor Learning, Movement Preparation, and Control

To enable goal-directed action, the brain has to convert the representation of a goal into a sequence of efferent motor commands, which result in muscle activations and finally in overt body movements. A neural structure that encodes such a mapping from a desired goal location into motor commands is termed *inverse model* [42].[1] Inverse models are at the heart of any theory of goal-directed movement and closely relate to the "memory trace" in Adams' closed-loop theory [1] or the "recall schema" in Schmidt's schema theory [67] (for older accounts see: [6, 27, 52]). Each body reacts differently to different motor commands and thus, an inverse model for controlling movements needs to be acquired by body-dependent motor learning. Even more so, the learning mechanisms need to operate unsupervised from the beginning and without the help of an internal or external teacher. Finally, each possible goal might be reached by an abundance of different motor command patterns (due to motor redundancy). Thus, there is no easy way to directly acquire an inverse model as each goal may be reached with various, alternative movement patterns and the involved learning needs to be unsupervised.

Several theories address how such inverse models might be acquired (e.g. [3, 4, 9, 40, 41, 43, 49, 58]). These theories differ broadly regarding representations, controlled parameters, and learning mechanisms. However, they all have one aspect in common: All these theories assume that an *optimal* inverse model is acquired, which stores for each goal a specific movement that optimizes additional criteria. For example, it might encode those motor commands that would reach a goal location with the smoothest possible trajectory [74]. Some of these theories rely on learning mechanisms that require a one-to-one mapping between goals and motor commands (e.g. [3, 49, 58]). They fail to account for the acquisition of inverse models for more complex, redundant bodies the control of which requires the resolution of a one-to-many mapping. Other theories that are mostly addressing the acquisition of cerebellar control structures (e.g. [43, 41] but see [40]) require at least a coarse inverse model that serves as the teacher. Thus, these models fail to account for unsupervised learning. However, under certain conditions, it is possible to acquire an inverse model unsupervised [7, 9].

The very notion that inverse models directly map goals onto optimal (but fixed) sequences of motor commands is problematic. Such a model only encodes one possible way to reach a goal and neglects alternatives—particularly those that yielded worse performance at the time of motor learning but may actually prove advantageous later on. Since the environment, the controlled body, and tasks change all the time, goals need to be reachable in different ways, depending on the current circumstances—sometimes we only need to reach a certain point in space, sometimes we also need to fulfill proprioceptive constraints, for example, when aligning hand and forearm to a pointing gesture. Likewise, we carry out one action in a way

[1] Please note, the term "model" may refer to a scientific theory or the above-mentioned mapping in this text.

that facilitates the next one [22, 78] or we bypass obstacles with ease [17]. If only one way was represented to attain a certain goal, it is impossible to model the flexibility with which we adjust our movements to novel situations so rapidly. Thus, inverse models cannot implement a direct mapping from goals to (optimal) actions, but have to provide a set of useful action alternatives. A selection amongst those alternatives is then necessary to generate actual behavior.

Furthermore, from a computational point of view, a direct mapping from goals to motor commands implies that the process of mapping goals onto actions is rather simple and somewhat akin to the read-out of a neural look-up table. However, movement preparation in the brain seems to be a rather involved process. This is evident as the time to prepare a movement depends on many features of the goal, the response, and the context of the movement [45, 48, 56, 62]. Hence, theories that predicate a direct mapping from goals to motor commands fail to assign a meaningful role to the obvious complexity of movement preparation.

On the other hand, theory of task-dependent movement preparation exist. Most notably, the posture-based motion planning theory [63, 64, 65] details how different movement alternatives may be evaluated, selected, and refined according to the constraints of the situation. The theory accounts for a wide range of behaviors, including reaching, grasping, and the avoidance of obstacles. It presumes that neural mechanisms link sensory (goal) representations and motor commands but it is mute regarding their neural structure and their acquisition. Besides such abstract approaches, also neural network models for flexible motor behavior have not yet offered an explanation for motor learning [14, 15].

SURE_REACH integrates theories of motor learning and theories of task-dependent movement preparation into a biologically plausible neural network framework. It thereby extents related approaches by accounting for the unsupervised acquisition of internal sensorimotor models and flexible movement planning (e.g. [55]). The next section outlines the general approach of the model and details its structure.

3 Description of the Model

Before the model is formulated in detail, an overview over the principles underlying the model is given. The discussion of the literature revealed that many conceptual problems of recent neural network models arise because the proposed learning mechanisms strive to only encode optimal motor commands. This causes some network approaches to fail at controlling a redundant body, others require supervised teaching mechanisms, and again others are left with a highly restrained behavioral repertoire. In contrast, SURE_REACH encodes *all* motor commands relevant for behavioral control. This can be quite easily achieved by ideomotor learning mechanisms. Such mechanisms encode contingent sensorimotor relationships, that is, they store which sensory effects result from which motor commands, depending on the initial state [21, 33, 34, 37]. Together, these sensorimotor contingencies represent how the body may be controlled and thus constitute a task-independent internal body model. However, they may not be directly used to generate movements. On

the one side, for certain goals no sensorimotor contingencies may be stored that provide a single motor command to move from the initial state to the goal. This problem is solved by the combination of multiple sensorimotor contingencies. On the other side, many sensorimotor contingencies may provide useful motor commands in the current context, but, of course, only one motor command can be executed at a time. Thus, specific motor commands have to be selected. According to the model, these processes of combining and task-dependently selecting the encoded sensorimotor information is what happens during movement preparation: A task-independent, general body model is used to tailor a task-specific inverse model to the unique demands and constraints of the current situation. Thus, the model links behavioral flexibility, movement preparation processes, and the way humans learn to control their bodies in a meaningful, interdependent manner.

3.1 Neural Network Implementation

In the following, a more detailed account for the specific computational stages and processes is given. First, the simulated controlled body is briefly described. The neural networks have to control a planar kinematic arm with three joints. Each joint is attached to two "muscles" which rotate the joint proportional to the excitation level of the innervating motor neuron. Each muscle is activated by motor commands ranging between $0.0 \leq mc_i \leq 1.0$. To compute the final movement of a joint ϕ_i, activations of antagonistic motor commands are subtracted and the result is multiplied by a gain factor g which scales movement velocity (see appendix for parameter values):

$$\phi_i(t+1) = \phi_i(t) + g(mc_{2i-1} - mc_{2i}), i = 1,2,3$$

This body is surely rather simple but it incorporates two important features. First, in most cases a sequence of motor commands is necessary to reach a goal. Second, the arm is redundant on the kinematic (multiple postures may realize a hand position) and sensorimotor (multiple trajectories may realize transitions between postures) level.

Figure 1 shows the staged structure of the model, which reflects the common notion that goals, such as specific hand locations, are transformed stepwise into a sequence of motor commands [12, 26, 35, 39]. By including multiple layers of representations and different nested transformation processes, it is possible to account for the flexible incorporation of constraints of different modalities during movement preparation.

In the current implementation, SURE_REACH transforms a desired hand location into a sequence of motor commands, given certain constraints.[2] The individual transformations are realized by interacting, adaptive neural network modules. First, the *posture memory* module encodes a mapping from visual hand space to proprioceptive posture space. Second, the *sensorimotor module* encodes sensorimotor

[2] The origin of these goals or constraints is not part of the model.

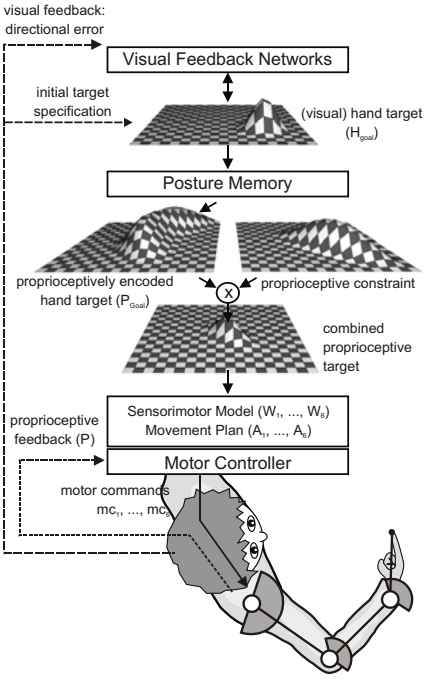


Fig. 1 The cartoon of the model reflects multiple stages of processing from the goal to motor commands. Intermediate representation and transformation processes may be adjusted to incorporate task-demands into the movement preparation process.

contingencies in posture space. Based on the sensorimotor model, movements are prepared and controlled in posture space.

3.1.1 Body Spaces Representation, Internal models, and Motor Learning

Before movements can be executed, the neural networks have to be learned. The posture memory has to encode the relationship between visually encoded hand positions and proprioceptively encoded arm postures. It stores the kinematic relationship between the two sensory modalities. The sensorimotor model has to encode the relationship between issued motor commands and state transitions in proprioceptive posture space. It stores the relationship between action and perception. To simulate early infant movements, random motor commands are executed and related to the consequent sensory input.[3] Throughout the model, goals, sensory input, and motor output are represented by populations of neurons (population codes), in which each

[3] This does not imply that infant movements are merely random, as is clearly not the case [76]. However, the neural networks are learned sufficiently well based on simple random movements. Learning speed and accuracy might be further improved if more structured exploratory movements were employed.

neuron represents a certain stimulus, such as a hand position or joint configuration [8, 13, 24, 69]. Hand coordinates are encoded by a population of neurons H. Each neuron h_i of H fires, if the hand coordinates (x,y) are close enough to the neuron's preferred hand location (h_i^x,h_i^y)

$$h_i = \max(1.0 - \frac{|x - h_i^x|}{d_{hand}};0) \cdot \max(1.0 - \frac{|y - h_i^y|}{d_{hand}};0),$$

where d_{hand} is the distance between the preferred values of neurons with adjacent receptive fields. The preferred hand locations are arranged in a grid and cover the arm's workspace. Arm postures are coded in a population of neurons P, where each neuron p_i is activated according to the equation

$$p_i = \prod_{j=1}^{3} \max(1.0 - \frac{|\phi_j - p_i^{\phi_j}|}{d_{posture,j}};0),$$

where $p_i^{\phi_j}$ are the preferred joint angles of each neuron p_i and $d_{posture,j}$ is the distance in the j-th dimension of joint space between the preferred values of neurons with adjacent receptive fields. During the initial learning movements, hand positions and postures are represented by different populations of neurons that constitute neural hand space and posture space. An associative learning rule strengthens the connections between simultaneously activated neurons in hand and posture space, thus creating a mapping between both representations. This mapping is termed posture memory. In each time step of the simulation the current hand (H) and arm state (P) are associated by Hebbian learning:

$$W_{PM}(t) = W_{PM}(t-1) + \varepsilon PH^T,$$

where ε is the learning rate and W_{PM} is the weight matrix that constitutes the posture memory.

The sensorimotor model consists of several neural representations of the posture space, each of which is associated to a specific motor neuron. If the motor neuron associated to an individual neural network is active, connections between the neuron representing the current posture and the neurons representing just visited postures are formed in the respective network. Together, these connections encode sensorimotor contingencies: They represent which transitions in posture space may occur, if the associated motor neuron is activated.

As the simulated arm is controlled by six different motor commands (two for each joint), there are six recurrent neural networks A_i each of which consists of a layer of mutually interconnected neurons. The neural layers A_i have the same size as the posture representation P and their interconnections are encoded by weight matrices W_i. During learning, the neural layers A_i have the following dynamics.

$$A_i(t) = \rho A_i(t-1) + mc_i(t-1)P(t-1),$$

where P is a representation of the current arm posture, ρ is a decay coefficient that enables the learning of temporally extended posture transitions by maintaining a trace of past posture representations, and mc_i is the activation of the i-th motor command during learning. Neural network weights are updated according to the following associative learning rule:

$$w_i^{jk}(t) = w_i^{jk}(t-1) + \delta a_i^j(t) p^k(t)(\theta - w_i^{jk}(t-1)),$$

where w_i^{jk}, a_i^j, and p^k are single values of the weight matrices, neuron layers, and the representation of the current posture, respectively, δ is the learning rate, and $\theta = 0.1$ is a ceiling value that prevent weights from increasing indefinitely.

3.1.2 Movement Preparation

The previous paragraphs revealed the model's space representations, internal models, and learning mechanisms. The following paragraphs answer how the acquired internal models may be used to plan movements. SURE_REACH accounts for movements to desired hand locations. The input to the model is thus a population encoded hand position. The desired hand position (H_{goal}) is transformed by the *posture memory* into a likewise encoded representation of *all* those arm postures that realize the respective hand location (P_{goal}; *proprioceptively encoded hand target* in Figure 1). It thus transforms the goal from a visual, hand-based into a proprioceptive, posture-based frame of reference. This is modeled by the equation

$$P_{goal} = W_{PM} \times H_{goal}.$$

The redundant representations of postures may be further constrained, for example, by inhibiting neurons which represent undesired final joint angles (see *proprioceptive constraint* in Fig 1). Movement planning is based on this redundant representation. The neural representation of acceptable end-postures is fed into the different neural networks of the motor controller whose connections constitute the sensorimotor model. The activity is propagated through these connections, the dynamics of which are modeled by the following equation:

$$A_i^* \leftarrow \max\{\beta(\gamma \frac{\sum_j^{j \neq i} A_j}{n-1} + (1-\gamma)A_i), P_{goal}\}$$
$$A_i \leftarrow A_i^* + W_i \times A_i^*,$$

where n is the number of neural networks, *max* returns the entry-wise maximum of two vectors, β reduces neural activity, γ specifies the intensity of crosstalk between networks, and P_{goal} is the representation of suitable goal postures normalized so that single values add up to 1.0.

Due to the learning scheme, activity is propagated to neurons that represent postures from which the goal can be reached by executing the motor command associated with the neuron's network. In each network, activity is propagated somewhat differently due to differing synaptic connectivity after learning. Thus, different

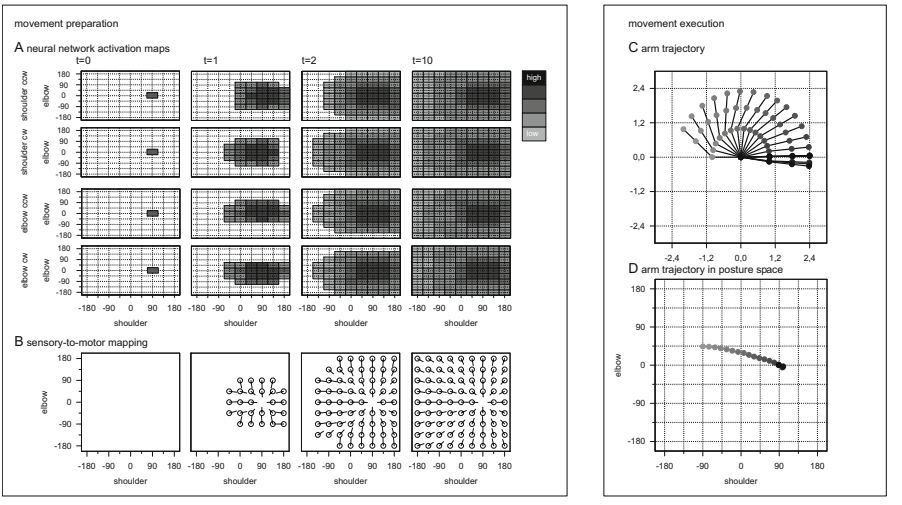


Fig. 2 A) The maps display the activation pattern of neurons of the sensorimotor model networks, which are associated to different motor neurons (rows), at various time points (columns, cw = clockwise, ccw = counterclockwise). White areas are not activated at all, dark areas are highly activated. The activation pattern constitute the neural basis of the movement plan or sensory-to-motor mapping. Please note, for illustration purposes, only those parts of the networks that are necessary to control shoulder and elbow are shown. The networks for control of a three joint arm are more complex. B) Motor commands may be derived from weighting the activations of neurons representing the same arm posture but in different networks (associated to different motor neurons). The arrows show the effects that the motor commands generated by the motor controller would have, depending on the actual arm posture. C) The chart shows the resulting arm movement, which starts from the left (light Grey) and terminates at the rightmost posture (black). D) The trajectory in posture space leads quickly from the start posture to the target.

patterns of activity emerge in the different networks. When the activities of neurons representing identical postures between the individual networks are compared, neurons of those networks are activated strongest whose associated motor neuron is most suitable to reach the goal. Thus, the relationship between the activities of neurons in the different networks defines the movement plan.

Figure 2A illustrates the movement preparation process. The network in the first row is associated to the motor neuron that causes counterclockwise shoulder rotations, the network in the second row is associated to the antagonistic motor neuron. In the first row, activity, which originates from the goal posture, is propagated stronger to the right, that is, to neurons that represent postures from which the goal can be reached by a counterclockwise shoulder rotation. In the second row, activity is propagated mostly to the left and hence to neurons that represent postures from which the goal can be reached by a clockwise shoulder rotation. Thus, after movement preparation, the neural activity pattern of the networks constitute the sensorimotor model for the neural representation of the movement plan (or sensory-to-motor mapping)

because they encode which motor neurons should be activated to reach the goal given any possible posture. Note that both movement plan and sensorimotor model are represented in the same networks: the current movement plan is encoded in the activity of the neurons whereas the sensorimotor model is encoded in the neural connections.

The movement plan can be considered as an inverse model that is generated online for the forthcoming target. Furthermore, the movement preparation process can be adapted to different situational constraints. For example, neurons representing postures that would collide with an obstacle can be inhibited, resulting in different activation pattern and consequently a movement that bypasses the obstacle. Likewise, the contribution of connections that are associated with motor commands that cause undesirable movements can be limited and thus, for example, decrease or even prevent movements of certain joints.

3.1.3 Movement Execution

To execute the movement, the movement plan is read out in a closed-loop fashion. This is realized by generating motor neuron activations dependent on the relative activations of those neurons in the different sensorimotor model networks (that is, the movement plan) that represent the current arm posture. The read-out of the movement plan is modeled by the equation

$$mc_i^* = P^T A_i$$
$$mc_i = \frac{\max(mc_i^* - mc_{anta(i)}^*;0)}{\sum_{i=1\ldots6}\max(mc_i^* - mc_{anta(i)}^*;0)},$$

where P is the current posture, and $mc_{anta(i)}$ is the antagonistic motor command to mc_i. The equation computes a normalized (1-norm) set of motor commands that moves the arm towards the goal. Figure 2B shows the movements in joint space that would result from reading out the movement plan at different postures and at different time points during movement preparation. Figure 2C and D show an exemplar movement which implements the prepared movement plan. The exemplar movement starts in an area of posture space (light Grey dots in Figure 2D), where the activation of the network that is associated with the clockwise shoulder rotation is higher than the activations in other networks. Thus, a movement that is mostly based on a clockwise shoulder rotation is generated (Figure 2D). However, also the somewhat higher activation of networks associated to a counterclockwise elbow rotation contribute to the movement.

4 Simulation of Reaching Movements

The previous section describes the basic components and connectivity of the model. In this section, we review how the model accounts for highly flexible behavior. However, before any goal directed movements can be performed, the neural networks need to learn the relationships between hand positions and arm postures and

the relationship between arm movements and motor commands. Figure 3A–G shows how the neural network controller performs when trying to execute goal directed movements after an increasing number of unsupervised learning iterations. Initially, the controller is not yet able to reach the target (Fig 3A–B). However, as more and more experience with the simulated arm is gathered, movements get increasingly accurate and efficient (Fig 3C–I). Finally, the arm can be moved accurately to all goals within the arm's reach (Fig 3H,J).

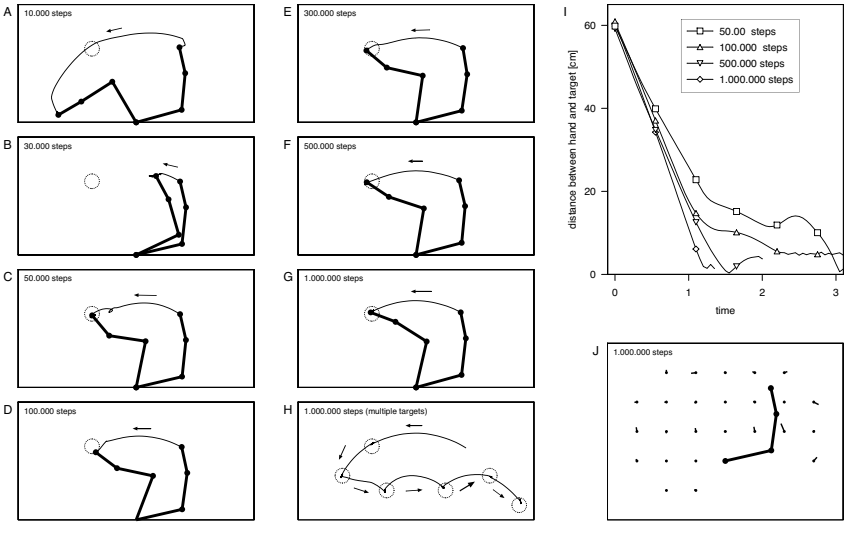


Fig. 3 A–G) The charts show the hand trajectory, initial and final arm posture of movements to an exemplar target (dotted circle) after different durations of motor learning. Movement accuracy and efficiency increases during learning. H) The chart shows the hand trajectories of movements to a number of targets after motor learning. I) The chart shows the distance between hand and target over the duration of the movements that are charted in C, D, F, G. With increasing durations of motor learning, movements get more efficient (faster) and more accurate. J) After learning, movements to reachable targets are quite accurate. Dots display targets, lines point to the actually reached hand position. If lines are not visible, the final hand position is within the respective dot. For the shown target, the average distance between hand and target is 1.80 cm (SD 1.31cm).

The evaluation of the model's learning performance shows not only that movement accuracy increases but also that movement preparation time (time needed until the movement plan is sufficiently prepared to generate movements) decreases. This reproduces a salient aspect of the interaction between movement preparation and motor learning [51, 53]. A detailed analysis of the general performance of the model during learning can be found elsewhere [10]. To summarize, the proposed neural networks and learning mechanisms are well able to learn to control a redundant arm.

4.1 Posture and Trajectory Redundancy

Next, we review how SURE_REACH can account for highly adaptive, flexible behavior. First the representation of redundant goal postures allows SURE_REACH to terminate a movement to a single hand target with different arm postures. Figure 4A–B show that that the final posture partially depends on the starting posture. A systematic analysis of such movements reveals that SURE_REACH exploits arm posture redundancy, as do humans [71], to reduce the overall trajectory length. Second, by simply inhibiting certain areas of the posture-based goal representations, movements to hand goals may be constrained to finally acquire a specific angle in a certain joint, for example, when aligning wrist and forearm to make a pointing movement (Fig 4C). Thus, it is possible to integrate visual (hand target) and proprioceptive constraints in the goal representation (Figure 1 exemplifies this process). Third, representing postural redundancy not only enables to adhere to constraints posed by the past (as the starting posture of a movement) or by the present (as constraints on a final joint configuration as in pointing) but also to constraints posed by future actions. This is especially important as movements are usually embedded in a larger sequence of actions and most movements can be carried out in ways that facilitate subsequent movements. Indeed, several experimental studies reveal how future goals of humans influence their present movements [46, 22, 38, 57, 70, 78]. In SURE_REACH, the representation of the redundant postures can be overlaid with a movement plan to a future goal to select those postures among the possible ones for an immediate movement that are also good starting postures for the subsequently planned movement, minimizing subsequent movement paths (Fig 5, see [28] for details). The resulting movements are then aligned to the overarching goals of a movement sequence. Recently, supporting evidence for this mode of movement planning has been provided [29]. Thus, representing redundant postures enables the anticipatory adjustment of movements to the demands of future goals (see [22] for a comparable account).

To summarize, the representation of posture redundancies enables the model to account for behavior in which the final state of the arm is not only defined by a desired hand location and a fixed optimality criterion, but also by other implicitly or explicitly defined constraints.

Behavior may get more adaptive by not only making use of the end-posture redundancy but also by adjusting the movement trajectory. As mentioned above, during movement preparation, small pieces of sensorimotor information are put together. Until now, no motor or trajectory constraints were imposed and the shortest movement from start to goal was prepared. This is not always desirable and thus it is possible to adjust the movement preparation process in different ways. To bypass an obstacle, for example, it is sufficient to simply inhibit neurons in the movement plan that represent postures that collide with an obstacle. Another example is reduced joint mobility. In some situations we might want to reduce the motion of an arthralgic joint or a joint that is immobilized by a cast. Movement preparation may be adapted to such inconveniences by trying to prepare a movement plan that relies

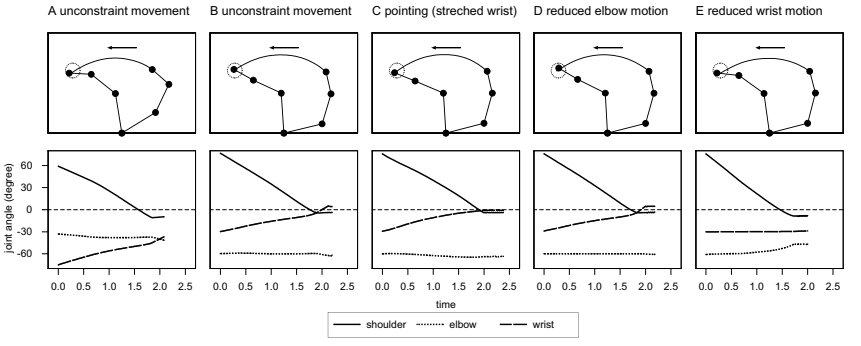


Fig. 4 The top charts show hand trajectories, initial and final arm postures. The dotted circles show the target of the hand. The bottom charts show the trajectory of the angle of shoulder, elbow, and wrist during those movements. A–B) Movements to identical hand targets may end in different arm postures, depending on the starting position. C) Targets can be reached while adhering to postural constraints, e.g. aligning forearm and wrist to make a pointing gesture. D–E) Movement planning may be adjusted to avoid moving specific joints, e.g. if in a cast or arthralgic. The bottom charts show how such constraints influence the trajectories of individual joints.

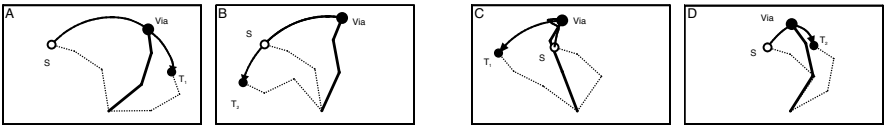


Fig. 5 The hand had to move sequentially from a start position (S) to a first target (Via) and then to one of two possible second targets (T_1 or T_2). The movements in A and B start from identical start postures and proceed to identical first targets (Via) but the second target differs (T_1 in A and T_2 in B). SURE_REACH can anticipate requirements of upcoming goals and thus choose different postures at the first target (compare black posture in A and B), dependent on the subsequent targets. The assumed posture at the first target facilitated the movement to the second target. Charts C and D show another example.

mainly on executable motor commands and thus minimizes the motion of impaired joints (Figure 4D–E).

4.2 Multimodal Feedback Control

The described neural networks are able to use a visual goal representation to plan a movement, but so far visual feedback of the relationship between a goal and and the hand is not used during movement execution. However, human movement accuracy depends considerably on visual feedback [20, 44, 54, 79]. Due to the hierarchical structure of SURE_REACH, visual feedback can be integrated into the model without compromising its ability to account for flexible behavior [30]. In this case, *visual feedback networks* (Fig 1) decouple the hand-based goal representation from direct

visual input and adjust an internal hand target according to a visual error signal (Fig 6A). For example, if the hand is currently slightly left of the target, the internal hand target may be shifted to the right. The hand-based internal target is then transformed into a posture-based representation that can be combined with kinematic constraints as described above. This enables a higher final goal reaching accuracy while keeping behavior flexible. Moreover, on the behavioral side this model of visual feedback generates human-like corrective movements. Figure 6 shows that the model reproduces movements with a fast initial approach component and slow final corrective movements, mimicking human reaching movements even closer [20, 79]. On the neurophysiological side, the model reflects the notion that movements are controlled by a cascade of nested control loops [12, 35].

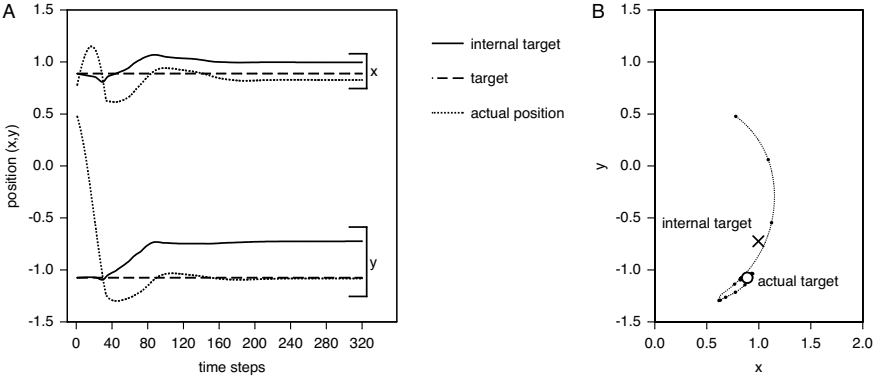


Fig. 6 A) The left chart shows the x- (top three lines) and y-coordinates (lower three lines) of the actual target (dashed), the internal target (black), and the hand location (dotted) of an exemplar movement. B) The right chart displays the hand trajectory, the actual target (circle), and the final position of the internal target (cross). The internal target shifts considerably to compensate for the initial overshoot. The charted location of the internal target is the mean of the preferred values of the dynamic target representation weighted by their activations.

5 Theoretical, Biological and Psychological Implications

The various simulation experiments have demonstrated three important claims. First, SURE_REACH is able to learn and control a redundant body with an unsupervised learning scheme. Its functionality thus exceeds other neural network models of unsupervised motor learning. Second, the encoded motor redundancy enables to account for behavioral flexibility to a high degree. Third, the simulation of specific features of human movement preparation and execution show that the implemented system has strong correlations with actual psychological processes.

More specifically, SURE_REACH contributes to the debate of the computational bases of movement control in several ways (for an overview see [11]). Neural network theories of motor learning typically assume that an inverse model that encodes

one single "best" way to reach a goal is acquired during motor learning and invariably used later on. In contrast, SURE_REACH learns a task-independent body model, which encodes very general properties of the relationships between motor commands and body movements. These general properties may be easily acquired unsupervised and may encode redundant motor control patterns. However, to make use of the body model, movement preparation processes are needed that extract a task-specific inverse model from the task-independent body model. Thus, while movement preparation is necessary to be able to use representations that can be learned unsupervised, the same movement preparation also enables the adjustment of motor plans to the demands and constraints of the current situation. Considering abstract theories of movement preparation, SURE_REACH offers a way to link such models to neural representations and sensorimotor learning mechanisms. In fact, as discussed elsewhere [10], many aspects of the posture-based motion planning theory [64] are realized with the proposed neural network architecture.

The model does not only offer an interesting computational account but it is also supported by neurophysiological and psychological findings. In SURE_REACH, sensory information or goals are represented by populations of many neurons in population codes. This property lays the foundation for unsupervised learning and the representation of motor redundancy but also reflects properties of cortical representations. Single-cell recording studies revealed that sensory, motor, and sensorimotor representations are shaped likewise in the motor cortex and parietal cortex of monkeys [13, 24, 69] and encoded in different coordinate systems, including posture based ones [2, 68].

SURE_REACH also fits in macroscopic theories of the brain, which consider the cerebral motor cortex — which is the site the model relates to — an area in which learning is unsupervised (as opposed to, e.g. the basal ganglia or the cerebellum, see e.g. [19, 36]). One of the key aspects of the model is its ability to account for the representation of redundant goal representations (that is, more than a single possible goal state). While corresponding representations have been recorded from motor areas during movement preparation [5, 13], behavioral studies and theoretical arguments suggest redundant goal representations during movement control [16, 50, 72]. Finally, as in humans or monkeys, movement preparation and movement control is more or less decoupled [5, 75], which enables the adjustment of ongoing movements or the preplanning but withholding of an upcoming movement. In conclusion, the model offers a comprehensive account of how humans learn to control their bodies and how their motor control mechanisms adapt flexibly to ever-changing situational requirements and constraints.

6 Application in Robotics

While the previous sections have focused on SURE_REACH as a model for human behavior, this section addresses how the derived principles may be used to make robots more flexible. To this aim, the general idea of SURE_REACH needs to be extended. To control a dynamic robot arm dynamic control components need

to be added and need to efficiently interact with the SURE_REACH-based control components.

A simulation of the KUKA KR16 arm was controlled by SURE_REACH enhanced with adaptive PD-control mechanisms [59]. Several modifications were necessary to make the application to the actual physical robot arm possible in the future. First, the learning mechanisms of SURE_REACH were replaced by hard-coded connections. This is possible due to the exact knowledge of the kinematics of the KUKA KR16 arm. Thus, while SURE_REACH shows that unsupervised learning of the neural network structures is possible, the KUKA KR16 arm application shows that it is not necessary given perfect prior knowledge of the targeted arm. Second, the muscle-based sensorimotor models were collapsed into one joint-motor-based sensorimotor model. This allows a more compact representation and thus a speed-up of the activity propagation in posture space—an important step to make real-time control possible. Third, an adaptive PD controller was added to translate the

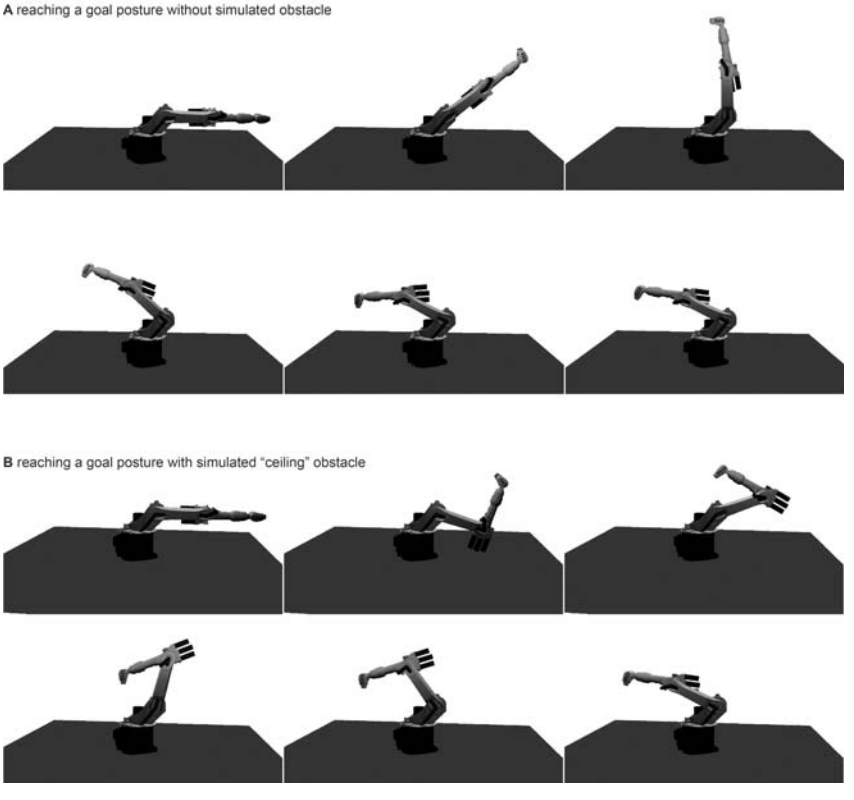


Fig. 7 The figures show the simulated KUKA KR 16 arm that is controlled by a dynamic SURE_REACH system with adaptive PD controller. A shows the straight transition from a start posture to a goal posture in an unconstrained environment, B shows that a more complex movement is exerted if the presence of a "ceiling" obstacle is suggested to the controller.

movement commands from SURE_REACH into actual motor control commands. Adding adaptivity to the controller helped to counteract increasing momentum in the KUKA KR16 arm. Fourth, the activity propagation signal was used to estimate the distance to the goal in order to provide the PD controller with movement direction and distance estimates. In effect, the resulting system was able to control the KUKA KR16 arm in simulation, which was realized with the commercially available simulation platform Webots [77].

Figure 7 shows two typical control sequences of the KUKA KR16 arm. In the upper panel, the arm has to move from a fully right-extended posture to a fully left-extended posture. Since the easiest way to achieve this is to only rotate the base joint, the main movements are observable in this joint. In the lower panel, the presence of a "ceiling" obstacle was suggested to the system. Thus, in order to reach the goal posture form the same starting posture, the arm flexes its upper two joints much more to avoid the obstacle and finally re-extends those joints to reach the goal posture.

To conclude, while the adapted SURE_REACH takes care of the kinematics enabling flexible movement planning and behavioral adjustments, the adaptive PD controller invokes suitable motor commands to maintain dynamic system stability while executing the suggested movement plan. Further evaluations of the system confirm the general robustness of the approach within other dynamic arm and with other added constraints and start-goal location or posture combinations [59].

7 Conclusion

To conclude, the described model offers an interesting perspective on the interplay between movement planning, unsupervised motor learning, and flexible task-dependent motor control in humans. Furthermore, the model's underlying principles are well-suited to control robots in a dynamic environment. In turn, the application of the model (and computational models in general) to real-world robots may hint at critical aspects that have yet to be considered in the models and contribute to a deeper understanding of neural processes. Thus, the application of biological theories to robotics may result in more flexible, robust, and adaptive machines. On the other hand, these applications can be expected to also enhance our knowledge about the intricate neural mechanisms that enable animals and humans to move their bodies seemingly effortlessly and with unmatched sophistication.

Appendix

The parameter settings for the simulations depicted in Figures 2 can be found in [10]. The parameter settings for the simulations depicted in Figures 5 can be found in [28]. The simulation results depicted in Figures 3, 4, and 6 are as follows. The lengths of the upper arm, forearm, and hand are $l_1 = 32cm$, $l_2 = 25cm$ and $l_3 = 18cm$, respectively; The shoulder, elbow and wrist joints can assume any angle within $-60^\circ \leq \phi_1 \leq 115^\circ$, $-160^\circ \leq \phi_2 \leq 0^\circ$, and $-75^\circ \leq \phi_3 \leq 50^\circ$, respectively; The

gain factor was set to $g = 0.9°$; The preferred hand locations are distributed in a fixed 31×25 grid with $d_{hand} = 5cm$ distance. The grid covers a $150cm \times 120cm$ rectangle, which covers the the arm's work space. Posture neurons are arranged in a $8 \times 7 \times 6$ grid covering the entire posture space. The distance between two adjacent neurons is approximately $d_{posture,1} = d_{posture,3} = 25°$ and $d_{posture,2} = 26.7°$. The neural networks are trained by moving the arm randomly for 1.000.000 time steps. During learning, new sets of motor command are randomly generated and executed for a random duration between 1 to 8 time steps. Sets of motor commands are generated by setting each individual motor neuron to 1.0 with a probability of $p = 0.3$ and to zero otherwise. The learning rate of the posture memory is set to $\varepsilon = 0.001$. The learning rate for the motor controller decays exponentially from $\delta_0 = 0.1$ to $\delta_{1,000,000} = 0.01$ during learning. The upper weight threshold is set to $\theta = 0.1$. The parameters of the equations modeling movement preparation are set to $\beta = 0.17$ and $\gamma = 0.43$. In the charts, the duration unit included 50 time steps. That is, each joint can move 45° in each duration unit.

References

1. Adams, J.A.: A closed-loop theory of motor learning. J. Mot. Behav. 3(2), 111–149 (1971)
2. Aflalo, T.N., Graziano, M.S.A.: Partial tuning of motor cortex neurons to final posture in a free-moving paradigm. Proc. Natl. Acad. of Sci. 8, 2909–2914 (2006)
3. Baraduc, P., Guigon, E., Burnod, Y.: Where does the population vector of motor cortical cells point during reaching movements? In: Kearns, M., Solla, S., Cohn, D. (eds.) Advances in Neural Information Processing Systems, vol. 11, pp. 83–89. MIT Press, Cambridge (1999)
4. Barto, A.G., Fagg, A.H., Sitkoff, N., Houk, J.C.: A cerebellar model of timing and prediction in the control of reaching. Neural Comp. 11, 565–594 (1999)
5. Bastian, A., Schöner, G., Riehle, A.: Preshaping and continuous evolution of motor cortical representations during movement preparation. Eur. J. Neurosci. 18, 2047–2058 (2003)
6. Bernstein, N.A.: The co-ordination and regulation of movements. Pergamon Press, Oxford (1967)
7. Berthier, N.E., Rosenstein, M.T., Barto, A.G.: Approximate optimal control as a model for motor learning. Psychol. Rev. 112(2), 329–346 (2005)
8. Bowers, J.S.: On the biological plausibility of grandmother cells: Implications for neural network theories in psychology and neuroscience. Psychol. Rev. 116(1), 220–251 (2009)
9. Bullock, D., Grossberg, S., Guenther, F.H.: A self-organizing neural model of motor equivalent reaching and tool use by a multijoint arm. J. Cogn. Neurosci. 5(4), 408–435 (1993)
10. Butz, M.V., Herbort, O., Hoffmann, J.: Exploiting redundancy for flexible behavior: Unsupervised learning in a modular sensorimotor control architecture. Psychol. Rev. 114(4), 1015–1046 (2007)
11. Butz, M.V., Herbort, O., Pezzulo, G.: Anticipatory, goal-directed behavior. In: Pezzulo, G., Butz, M.V., Castelfranchi, C., Falcone, R. (eds.) The Challenge of Anticipation. LNCS (LNAI), vol. 5225, pp. 85–113. Springer, Heidelberg (2008)
12. Cisek, P.: Integrated neural processes for defining potential actions and deciding between them: A computational model. J. Neurosci. 26(38), 9761–9770 (2006)

13. Cisek, P., Kalaska, J.F.: Neural correlates of reaching decisions in dorsal premotor cortex: Specification of multiple direction choices and final selection of action. Neuron 45(5), 801–814 (2005)
14. Cruse, H., Steinkühler, U.: Solution of the direct and inverse kinematic problems by a common algorithm based on the mean of multiple computations. Biol. Cybern. 69, 341–351 (1993)
15. Cruse, H., Steinkühler, U., Burkamp, C.: MMC - a recurrent neural network which can be used as manipulable body model. In: Pfeifer, R., Blumberg, B., Meyer, J.A., Wilson, S. (eds.) From Animals to Animats 5: The Fifth International Conference on the Simulation of Adaptive Behavior, pp. 381–389. MIT Press, Cambridge (1998)
16. de Freitas, S.M.S.F., Scholz, J.P., Stehman, A.J.: Effect of motor planning on use of motor abundance. Neurosci. Lett. 417(1), 66–71 (2007)
17. Dean, J., Brüwer, M.: Control of human arm movements in two dimensions: Paths and joint control in avoiding simple linear obstacles. Exp. Brain Res. 97, 497–514 (1994)
18. Desmurget, M., Grafton, S.: Forward modeling allows feedback control for fast reaching movements. Trends Cogn. Sci. 4(11), 423–431 (2000)
19. Doya, K.: Complementary roles of basal ganglia and cerebellum in learning and motor control. Curr. Opin. Neurobiol. 10(6), 732–739 (2000)
20. Elliott, D., Helsen, W.F., Chua, R.: A century later: Woodworth's two-component model of goal-directed aiming. Psychol. Bull. 127(3), 342–357 (1899)
21. Elsner, B., Hommel, B.: Effect anticipations and action control. J. Exp. Psychol. 27(1), 229–240 (2001)
22. Fischer, M.H., Rosenbaum, D.A., Vaughan, J.: Speed and sequential effects in reaching. J. Exp. Psychol.: Hum. Percept. Perform. 23(2), 404–428 (1997)
23. Flash, T., Hogan, N.: The coordination of arm movements: An experimentally confirmed mathematical model. J. Neurosci. 5(7), 1688–1703 (1985)
24. Georgopoulos, A.P.: Current issues in directional motor control. Trends Neurosci. 18(11), 506–510 (1995)
25. Hallett, M.A., Pascual-Leone, A., Topka, H.: The acquisition of motor behavior in vertebrates. In: Bloedel, J.R., Ebner, T.J., Wise, S.P. (eds.) Adaptation and skill learning: Evidence for different neural substrates, pp. 289–301. MIT Press, Cambridge (1996)
26. Haruno, M., Wolpert, D.M., Kawato, M.: Hierarchical mosaic for movement generation. In: Ono, T., Matsumoto, G., Llinas, R., Berthoz, A., Norgren, R., Nishijo, H., Tamura, R. (eds.) Excepta Medica International Congress Series, vol. 1250 (2003)
27. Herbart, J.F.: Psychologie als Wissenschaft neu gegründet auf Erfahrung, Metaphysik und Mathematik. In: Zweiter analytischer Teil [Psychology as a Science newly founded on Experience, Metaphysics and Mathematics: Second, Analytical Part]. August Wilhelm Unzer., Königsberg (1825)
28. Herbort, O., Butz, M.V.: Encoding complete body models enables task dependent optimal control. Proc. Int. Jt. Conf. Neural Netw. 20, 1639–1644 (2007)
29. Herbort, O., Butz, M.V.: Anticipatory planning of sequential hand and finger movements. J. Mot. Behav. (in press)
30. Herbort, O., Butz, M.V., Hoffmann, J.: Multimodal goal representations and feedback in hierarchical motor control. In: Proc. Int. Conf. Cogn. Syst. (2008)
31. Herbort, O., Ognibene, D., Butz, M.V., Baldassarre, G.: Learning to select targets within targets in reaching tasks. In: Proc. 6th Int. IEEE Conf. Dev. Learn., vol. 6, pp. 7–12 (2007)
32. Hikosaka, O., Nakamura, K., Sakai, K., Nakahara, H.: Central mechanisms of motor skill learning. Curr. Opin. Neurobiol. 12, 217–222 (2002)

33. Hoffmann, J.: Vorhersage und Erkenntnis: Die Funktion von Antizipationen in der menschlichen Verhaltenssteuerung und Wahrnehmung [Anticipation and cognition: The function of anticipations in human behavioral control and perception]. Hogrefe, Göttingen (1993)
34. Hoffmann, J.: Anticipatory behavior control. In: Butz, M.V., Sigaud, O., Gérard, P. (eds.) Anticipatory Behavior in Adaptive Learning Systems. LNCS (LNAI), vol. 2684, pp. 44–65. Springer, Heidelberg (2003)
35. Hoffmann, J., Butz, M.V., Herbort, O., Kiesel, A., Lenhard, A.: Spekulationen zur Struktur ideo-motorischer Beziehungen [Speculations about the structure of ideomotor relations]. Z. Sportpsychol. 14(3), 95–103 (2007)
36. Jackson, A., Mavoori, J., Fetz, E.E.: Long-term motor cortex plasticity induced by an electronic neural implant. Nat. 444, 56–60 (2006)
37. James, W.: The principles of psychology, vol. 1. Holt, New York (1890)
38. Johnson-Frey, S.H., McCarty, M.E., Keen, R.: Reaching beyond spatial perception: Effects of intended future actions on visually guided prehension. Vis. Cogn. 11(2-3), 371–399 (2004)
39. Jordan, M.I., Wolpert, D.M.: Computational motor control. In: Gazzaniga (ed.) The Cognitive Neuroscience, pp. 601–620. MIT Press, Cambridge (1999)
40. Karniel, A., Inbar, G.F.: A model for learning human reaching movements. Biol. Cybern. 77, 173–183 (1997)
41. Kawato, M.: Feedback-error-learning neural network for supervised learning. In: Eckmiller, R. (ed.) Advanced neural computers, pp. 365–372. North-Holland, Amsterdam (1990)
42. Kawato, M.: Internal models for motor control and trajectory planning. Curr. Opin. Neurobiol. 9, 718–727 (1999)
43. Kawato, M., Furukawa, K., Suzuki, R.: A hierarchical neural-network model for control and learning of voluntary movement. Biol. Cybern. 57, 169–185 (1987)
44. Khan, M.A., Franks, I.M., Goodman, D.: The effect of practice on the control of rapid aiming movements: Evidence for an interdependency between programming and feedback processing. Q. J. Exp. Psychol. Section A 51(2), 425–443 (1998)
45. Klapp, S.T., Erwin, C.I.: Relation between programming time and duration of the response being programmed. J. Exp. Psychol.: Hum. Percept. Perform. 2(4), 591–598 (1976)
46. Klein Breteler, M.D., Hondzinski, J.M., Flanders, M.: Drawing sequences of segments in 3d: Kinetic influences on arm configuration. J. Neurophysiol. 89, 3253–3263 (2003)
47. Körding, K.P., Wolpert, D.M.: Bayesian integration in sensorimotor learning. Nat. 427, 244–247 (2004)
48. Kunde, W., Koch, I., Hoffmann, J.: Anticipated action effects affect the selection, initiation, and execution of actions. Q. J. Exp. Psychol. Section A: Human Exp. Psychol. 57, 87–106 (2004)
49. Kuperstein, M.: Neural model of adaptive hand-eye coordination for single postures. Sci. 239, 1308–1311 (1988)
50. Latash, M.L., Scholz, J.P., Schöner, G.: Motor control strategies revealed in the structure of motor variability. Exerc. & Sport Sci. Rev. 30(1), 26–31 (2002)
51. Lavrysen, A., Helsen, W.F., Tremblay, L., Elliott, D., Adam, J.J., Feys, P., Buekers, M.J.: The control of sequential aiming movements: The influence of practice and manual asymmetries on the one-target advantage. Cortex 39, 307–325 (2003)
52. Lotze, H.R.: Medicinische Psychologie oder Physiologie der Seele [Medical Psychology or Physiology of the Soul]. Weidmannsche Buchhandlung, Leipzig (1852)

53. Ludwig, D.A.: Emg changes during the acquisition of a motor skill. Am. J. Phys. Medicine 61(5), 229–243 (1982)
54. Ma-Wyatt, A., McKee, S.P.: Visual information throughout a reach determines endpoint precision. Exp. Brain Res. 179(1), 55–64 (2007)
55. Morasso, P., Sanguineti, V., Spada, G.: A computational theory of targeting movements based on force fields and topology representing networks. Neurocomputing 15(3-4), 411–434 (1997)
56. Munro, H., Plumb, M.S., Wilson, A.D., Williams, J.H.G., Mon-Williams, M.: The effect of distance on reaction time in aiming movements. Exp. Brain Res. 183(2), 249–257 (2007)
57. Mutsaarts, M., Steenbergen, B., Bekkering, H.: Anticipatory planning deficits and task context effects in hemiparetic cerebral palsy. Exp. Brain Res. 172(2), 151–162 (2006)
58. Ognibene, D., Mannella, F., Pezzulo, G., Baldassarre, G.: Integrating reinforcement-learning, accumulator models, and motor-primitives to study action selection and reaching in monkeys. In: Fum, D., Del Missier, F., Stocco, A. (eds.) Proc. Seventh International Conference on Cognitive Modeling (ICCM 2006), pp. 214–219. Edizioni Goliardiche, Trieste (2006)
59. Pedersen, G., Butz, M.V., Herbort, O.: Integrating dynamics into a human behavior model for highly flexible autonomous manipulator control. IEEE Systems, Man & Cybernetics B (submitted)
60. Peters, J., Schaal, S.: Learning to control in operational space. Int. J. Robot. Res. 27, 197–212 (2008)
61. Robertson, E.M., Miall, R.C.: Multi-joint limbs permit a flexible response to unpredictable events. Exp. Brain Res. 117, 148–152 (1997)
62. Rosenbaum, D.A.: Human movement initiation: Specification of arm, direction and extent. J. Exp. Psychol.: Gen. 109, 444–474 (1980)
63. Rosenbaum, D.A., Engelbrecht, S.E., Bushe, M.M., Loukopoulos, L.D.: A model for reaching control. Acta Psychol. 82(1-3), 237–250 (1993)
64. Rosenbaum, D.A., Loukopoulos, L.D., Meulenbroek, R.G.J., Vaughan, J., Engelbrecht, S.E.: Planning reaches by evaluating stored postures. Psychol. Rev. 102(1), 28–67 (1995)
65. Rosenbaum, D.A., Meulenbroek, R.G.J., Vaughan, J., Jansen, C.: Posture-based motion planning: Applications to grasping. Psychol. Rev. 108(4), 709–734 (2001)
66. Salaun, C., Padois, V., Sigaud, O.: Learning forward models for the operational space control of redundant robots. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 169–192. Springer, Heidelberg (2010)
67. Schmidt, R.A.: A schema theory of discrete motor skill-learning. Psychol. Rev. 82(4), 229–261 (1975)
68. Schwartz, A.B., Moran, D.W., Reina, G.A.: Differential representation of perception and action in the frontal cortex. Sci. 303, 380–383 (2004)
69. Shadmehr, R., Wise, S.P.: The Computational Neurobiology of Reaching and Pointing: A foundation for motor learning. MIT Press, Cambridge (2005)
70. Short, M.W., Cauraugh, J.H.: Precision hypothesis and the end-state comfort effect. Acta Psychol. 100(3), 243–252 (1999)
71. Soechting, J.F., Buneo, C.A., Herrmann, U., Flanders, M.: Moving effortlessly in three dimensions: Does Donders' law apply to arm movement? J. Neurosci. 15, 6271–6280 (1995)
72. Todorov, E., Jordan, M.I.: Optimal feedback control as a theory of motor coordination. Nat. Neurosci. 5(11), 1226–1235 (2002)
73. Toussaint, M., Goerick, C.: A bayesian view on motor control and planning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 227–252. Springer, Heidelberg (2010)

74. Uno, Y., Kawato, M., Suzuki, R.: Formation and control of optimal trajectory in human multijoint arm movement: Minimum torque-change model. Biol. Cybern. 61(2), 89–101 (1989)
75. van Sonderen, J.F., Dernier van der Gon, J.J.: Reaction-time-dependent differences in the initial movement direction of fast goal-directed arm movements. Hum. Mov. Sci. 10(6), 713–726 (1991)
76. von Hofsten, C.: An action perspective on motor development. Trends Cogn. Sci. 8(6), 266–272 (2004)
77. Webots, C.L.: Commercial mobile robot simulation software, http://www.cyberbotics.com
78. Weigelt, M., Kunde, W., Prinz, W.: End-state comfort in bimanual object manipulation. Exp. Psychol. 53(2), 143–148 (2006)
79. Woodworth, R.S.: The Accuracy of Voluntary Movement. New Era, Lancaster (1899)

Intrinsically Motivated Exploration for Developmental and Active Sensorimotor Learning*

Pierre-Yves Oudeyer, Adrien Baranes, and Frédéric Kaplan

Abstract. Intrinsic motivation is a central mechanism that guides spontaneous exploration and learning in humans. It fosters incremental and progressive sensorimotor and cognitive development by pushing exploration of activities of intermediate complexity given the current state of capabilities. This chapter presents and studies two computational intrinsic motivation systems that share similarities with human intrinsic motivation systems, IAC and R-IAC, that aim at self-organizing and efficiently guiding exploration for sensorimotor learning in robots. IAC was initially introduced to model the qualitative formation of developmental motor stages of increasing complexity, as shown in the Playground Experiment which we will outline. In this chapter, we argue that IAC and other intrinsically motivated learning heuristics could also be viewed as active learning algorithms that are particularly suited for learning forward models in unprepared sensorimotor spaces with large unlearnable subspaces. Then, we introduce a novel formulation of IAC, called R-IAC, and show that its performances as an intrinsically motivated active learning algorithm are far superior to IAC in a complex sensorimotor space where only a small subspace is "interesting", i.e. neither unlearnable nor trivial. We also show results in which the learnt forward model is reused in a control scheme. Finally, an open-source accompanying software containing these algorithms as well as tools to reproduce all the experiments in simulation presented in this paper is made publicly available.

Index Terms: active learning, intrinsically motivated learning, exploration, developmental robotics, artificial curiosity, sensorimotor learning.

1 Intrinsically Motivated Exploration and Learning

Developmental robotics approaches are studying mechanisms that may allow a robot to continuously discover and learn new skills in unknown environments and in

Pierre-Yves Oudeyer and Adrien Baranes
INRIA, France
e-mail: (http://flowers.inria.fr)

Frédéric Kaplan
CRAFT-EPFL, Switzerland

* Material presented in this chapter is based on several previous publications of the authors (in particular [27, 61]).

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 107–146.
springerlink.com

a life-long time scale [1], [2]. A main aspect is the fact that the set of these skills and their functions are at least partially unknown to the engineer who conceive the robot initially, and are also task-independent. Indeed, a desirable feature is that robots should be capable of exploring and developing various kinds of skills that they may re-use later on for tasks that they did not foresee. This is what happens in human children, and this is also why developmental robotics shall import concepts and mechanisms from human developmental psychology.

1.1 The Problem of Exploration in Open-Ended Learning

Like children, the "freedom" that is given to developmental robots to learn an open set of skills also poses a very important problem: as soon as the set of motors and sensors is rich enough, the set of potential skills become extremely large and complicated. This means that on the one hand, it is impossible to try to learn all skills that may potentially be learnt because there is not enough time to physically practice all of them. Furthermore, there are many skills or goals that the child/robot could imagine but never be actually learnable, because they are either too difficult or just not possible (for example, trying to learn to control the weather by producing gestures is hopeless). This kind of problem is not at all typical of the existing work in machine learning, where usually the "space" and the associated "skills" to be learnt and explored are well-prepared by a human engineer. For example, when learning hand-eye coordination in robots, the right input and output spaces (e.g. arm joint parameters and visual position of the hand) are typically provided as well as the fact that hand-eye coordination is an interesting skill to learn. But a developmental robot is not supposed to be provided with the right subspaces of its rich sensorimotor space and with their association with appropriate skills: it would for example have to discover that arm joint parameters and visual position of the hand are related in the context of a certain skill (which we call hand-eye coordination but which it has to conceptualize by itself) and in the middle of a complex flow of values in a richer set of sensations and actions.

1.1.2 Intrinsic Motivations

Developmental robots, like humans, have a sharp need for mechanisms that may drive and self-organize the exploration of new skills, as well as identify and organize useful sub-spaces in its complex sensorimotor experiences. Psychologists have identified two broad families of guidance mechanisms which drive exploration in children:

1) **Social learning**, which exists in different forms such as stimulus enhancement, emulation, imitation or demonstration, and which many groups try to implement in robots [e.g. 3,4,5,6,7,8,9,10,11,12,13,14];
2) **Internal guiding mechanisms**, also studied by many robotics research groups (e.g. see [15,16,17,18,19,20]) and in particular intrinsic motivation, responsible of spontaneous exploration and curiosity in humans, which is the mechanisms underlying the algorithms presented in this paper.

Intrinsic motivations are mechanisms that guide curiosity-driven exploration, that were initially studied in psychology [21]-[23] and are now also being approached in neuroscience [24]-[26]. Machine learning and robotics researchers have proposed

that such mechanism might be crucial for self-organizing developmental trajectories as well as for guiding the learning of general and reusable skills in machines and robots [27,28]. A large diversity or approaches for operationalizing intrinsic motivation have been presented in the literature [e.g. 29,30,31,32,33,34,28,27,35], and see [27] for a general overview. Several experiments have been conducted in real-world robotic setups, such as in [27,36,34] where an intrinsic motivation system was shown to allow for the progressive discovery of skills of increasing complexity, such as in the Playground Experiment that we will present in section 4. In these experiments, the focus was on the study of how developmental stages could self-organize into a developmental trajectory of increasing complexity without a direct prespecification of these stages and their number. As we will explain in section 4, this can lead to stimulating models of the self-organization of structured developmental trajectories with both universal tendencies and diversity as observed in humans [60]. Furthermore, in this chapter, we argue that such intrinsic motivation systems can be used as efficient active learning algorithms. With this view, we present a novel system, called **R-IAC**, which improves **IAC** over a number of features. Through several experiments, we will show that it can be used as an efficient active learning algorithm to learn forward and inverse models in complex unprepared sensorimotor spaces with unlearnable subspaces.

2 IAC and R-IAC for Intrinsically Motivated Active Learning

2.1 Developmental Active Learning

In **IAC**, intrinsic motivation is implemented as a heuristics which pushes a robot to explore sensorimotor activities for which learning progress is maximal, i.e. subregions of the sensorimotor space where the predictions of the learnt forward model improve fastest [27]. Thus, this mechanism regulates actively the growth of complexity in sensorimotor exploration, and can be conceptualized as a **developmental active learning** algorithm. This heuristics shares properties with statistical techniques in optimal experiment design (e.g. [37]) where exploration is driven by expected information gain, as well as with attention and motivation mechanisms proposed in the developmental psychology literature (e.g. [22], [38], or see [23] for a review) where it has been proposed that exploration is preferentially focused on activities of intermediate difficulty or novelty [39,40], but differs significantly from many active learning heuristics in machine learning in which exploration is directed towards regions where the learnt model is maximally uncertain or where predictions are maximally wrong (e.g. [41, 42], see [27] for a review). As argued in [27], developmental robots are typically faced with large sensorimotor spaces which cannot be entirely learnt (because of time limits among other reasons) and/or in which subregions are not learnable (potentially because it is too complicated for the learner, or because there are no correlations between the input and output variables, see examples in the experiment section and in [27]). In these sensorimotor spaces, exploring zones of maximal uncertainty or unpredictability is bound to be an inefficient strategy since it would direct exploration towards subspaces in which there are no learnable correlations, while a heuristics based on learning progress allows to avoid unlearnable parts as well as to focus exploration on zones of gradually increasing complexity.

In [27, 34], experiments such as the Playground Experiment described in section 4 showed how **IAC** can allow an AIBO robot, equipped with a set of parameterized motor primitives (in a 5 DOF motor space), as well as a set of perceptual primitives (in a 3 DOF perceptual space), to self-organize a developmental trajectory in which a variety of affordances uses of the motor primitives where learnt in spite of not having been specified initially. In [36], a slightly modified version of **IAC** allowed an AIBO robot, equipped with parameterized central pattern generators (CPG's) in a 24 DOF motor space and 3 DOF perceptual space, to learn a variety of locomotion skills. Yet, these previous results focused on qualitative properties of the self-organized developmental trajectories, and **IAC** was not optimized for efficient active learning per se.

Here, we present a novel formulation of **IAC**, called **Robust-IAC (R-IAC)**, and show that it can efficiently allow a robot to learn actively, fast and correctly forward and inverse kinematic models in an unprepared sensorimotor space. As we will explain, **R-IAC** introduces four main advances compared to **IAC**:

- **Probabilistic action selection:** instead of choosing actions to explore the zone of maximal learning progress at a given moment in time (except in the random action selection mode), R-IAC explores actions on sensorimotor subregions probabilistically chosen based on their individual learning progress;
- **Multi-resolution monitoring of learning progress:** in R-IAC, when sensorimotor regions are split into subregions, parent regions are kept and one continues to monitor learning progress in them, and they continue to be eligible regions for action selection. As a consequence, learning progress is monitored simultaneously at various regions scales, as opposed to IAC where it was monitored only in child regions and thus at increasing small scales;
- **A new region splitting mechanism** that is based on the direct optimization of learning progress dissimilarity among regions;
- **The introduction of a third exploration mode** hybridizing learning progress heuristics with more classic heuristics based on the exploration of zones of maximal unpredictability;

2.2 Prediction Machine and Analysis of Error Rate

We consider a robot as a system with motor/actions channels **M** and sensory/state channels **S. M** and **S** can be low-level such as torque motor values or touch sensor values, or higher level such as a "go forward one meter" motor command or "face detected" visual sensor". Furthermore, **S** can correspond to internal sensors measuring the internal state of the robot or encoding past values of the sensors. Real valued action/motor parameters are represented as a vector $\mathbf{M}(\mathbf{t})$, and sensors, as $\mathbf{S}(\mathbf{t})$, at a time t. $\mathbf{SM}(\mathbf{t})$ represents a sensorimotor context, i.e. the concatenation of both motors and sensors vectors.

We also consider a Prediction Machine **PM** (Fig. 1), as a system based on a learning algorithm (neural networks, KNN, etc.), which is able to create a forward model of a sensorimotor space based on learning examples collected through self-determined sensorimotor experiments. Experiments are defined as series of

actions, and consideration of sensations detected after actions are performed. An experiment is represented by the set $(\mathbf{SM(t)}, \mathbf{S(t+1)})$, and denotes the sensory/state consequence **S(t+1)** that is observed when actions encoded in **M(t)** are performed in the sensory/state context **S(t).** This set is called a "**learning exemplar**". After each trial, the prediction machine **PM** gets this data and incrementally updates the forward model that it is encoding, i.e. the robot incrementally increases its knowledge of the sensorimotor space. In this update process, **PM** is able to compare, for a given context $\mathbf{SM}(\boldsymbol{t})$, differences between predicted sensations $\tilde{\mathbf{S}}(\boldsymbol{t+1})$ (estimated using the created model), and real consequences $\mathbf{S}(\boldsymbol{t+1})$. It is then able to produce a measure of error $\boldsymbol{e(t+1)}$, which represents the quality of the model for sensorimotor context $\mathbf{SM}(\boldsymbol{t})$. This is summarized in figure 1.

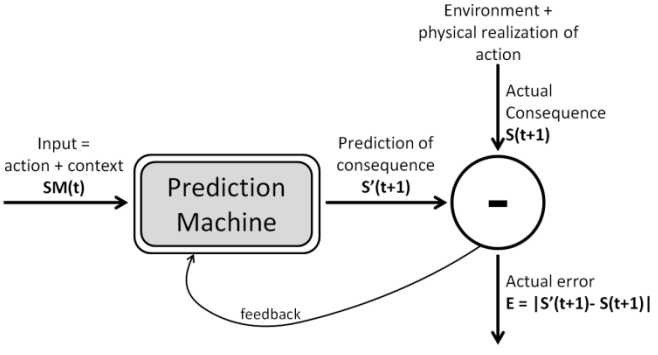


Fig. 1 The prediction learning machine (e.g. a neural network, an SVM, or Gaussian process regression based algorithm).

Then, we consider a module able to analyze learning evolutions over time, called Prediction Analysis Machine **PAM**, Fig. 2. In a given subregion $\boldsymbol{R_n}$ of the sensorimotor space (which we will define below), this system monitors the evolution of errors in predictions made my **PM** by computing its derivative, i.e. the learning progress, $\boldsymbol{LP_n} = e_N - e_F$ in this particular region over a sliding time window (see Fig. 2). $\boldsymbol{LP_n}$ is then used as a measure of interestingness used in the action selection scheme outlined below. The more a region is characterized by learning progress, the more it is interesting, and the more the system will perform experiments and collect learning exemplars that fall into this region. Of course, as exploration goes on, the learnt forward model becomes better in this region and learning progress might decrease, leading to a decrease in the interestingness of this region.

To precisely represent the learning behavior inside the whole sensorimotor space and differentiate its various evolutions in various subspaces/subregions, different **PAM** modules, each associated to a different subregion R_i of the sensorimotor space, need to be built. Therefore, the learning progress $\boldsymbol{LP_i}$ provided as the output values of each **PAM** becomes representative of the interestingness of the associated region $\boldsymbol{R_i}$. Initially, the whole space is considered as one single gion $\boldsymbol{R_0}$, associated to one **PAM**, which will be progressively split into subregions with their own **PAM** as we will now describe.

2.3 The Split Machine

The Split Machine **SpM** (Fig. 3) possesses the capacity to memorize all the experimented learning exemplars $(\mathbf{SM(t)}, \mathbf{S(t+1)})$, and the corresponding errors values $\boldsymbol{e(t+1)}$. It is both responsible for identifying the region and **PAM**

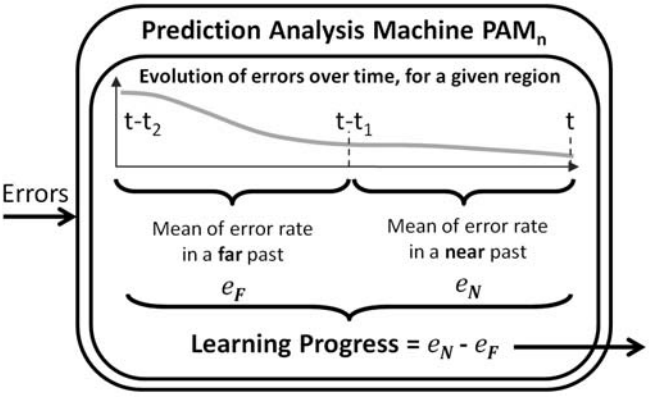


Fig. 2 Internal mechanism of the Prediction Analysis Machine $\mathbf{PAM_n}$ associated to a given subregion $\boldsymbol{R_n}$ of the sensorimotor space. This module considers errors detected in prediction by the Prediction Machine **PM**, and returns a value representative of the learning progress in the region. Learning progress is the derivative of errors analyzed between a far and a near past in a fixed length sliding window.

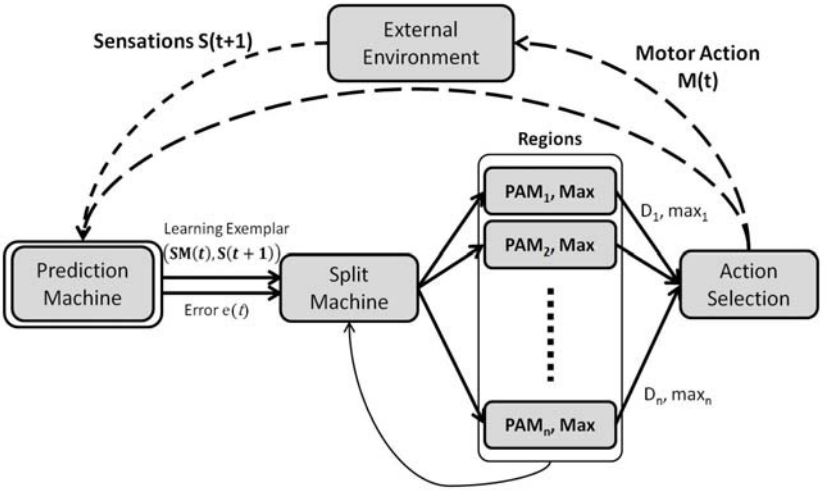


Fig. 3 General architecture of IAC and R-IAC. The prediction Machine is used to create a forward model of the world, and measures the quality of its predictions (errors values). Then, a split machine cuts the sensorimotor space into different regions, whose quality of learning over time is examined by Prediction Analysis Machines. Then, an Action Selection system, is used to choose experiments to perform.

corresponding to a given **SM(t)**, but also responsible of splitting (or creating in R-IAC where parent regions are kept in use) sub-regions from existing regions.

1) Region Implementation

We use a tree representation to store the list of regions as shown in Fig. 4. The main node represents the whole space, and leafs are subspaces. $\mathbf{S(t)}$ and $\mathbf{M(t)}$ are here normalized into $[0;1]^n$. The main region (first node), called R_0, represents the whole sensorimotor space. Each region stores all collected exemplars that it covers. When a region contains more than a fixed number $\mathbf{T_{split}}$ of exemplars, we split it into two ones in **IAC**, or create two new regions in **R-IAC**. Splitting is done with hyperplanes perpendicular to one dimension. An example of split execution is shown in Fig. 4, using a two dimensions input space.

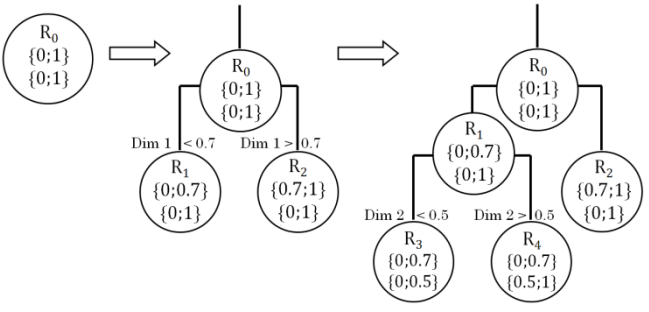


Fig. 4 The sensorimotor space is iteratively and recursively split into sub-spaces, called "regions". Each region R_n is responsible for monitoring the evolution of the error rate in the anticipation of consequences of the robot's actions, if the associated contexts are covered by this region.

2) IAC Split Algorithm

In the **IAC** algorithm, the idea was to find a split such that the two sets of exemplars into the two subregions would minimize the sum of the variances of $\mathbf{S}(\boldsymbol{t}+\mathbf{1})$ components of exemplars of each set, weighted by the number of exemplars of each set. Hence, the split takes place in the middle of zones of maximal change in the function $\boldsymbol{SM}(\boldsymbol{t}) \rightarrow \boldsymbol{S}(\boldsymbol{t}+\mathbf{1})$. Mathematically, we consider $\varphi_n = \left\{ \left(\mathbf{SM}(\boldsymbol{t}), \mathbf{S}(\boldsymbol{t}+\mathbf{1})\right)_i \right\}$ as the set of exemplars possessed by region R_n. Let us denote j a cutting dimension and v_j, an associated cutting value. Then, the split of $\varphi_n$into φ_{n+1} and φ_{n+2} is done by choosing j and v_j such that:

(1) All the exemplars $(\mathbf{SM}(\boldsymbol{t}), \mathbf{S}(\boldsymbol{t}+\mathbf{1}))_{\boldsymbol{i}}$ of φ_{n+1} have a j^{th}component of their $\mathbf{SM}(\boldsymbol{t})$ smaller than v_j

(2) All the exemplars $(\mathbf{SM}(\boldsymbol{t}), \mathbf{S}(\boldsymbol{t}+\mathbf{1}))_{\boldsymbol{i}}$ of φ_{n+2} have a j^{th}component of their $\mathbf{SM}(\boldsymbol{t})$ greater than v_j

(3) The quantity :

$$Qual(j, v_j) = \\ |\varphi_{n+1}|.\sigma(\{\mathbf{S}(\boldsymbol{t}+\mathbf{1})|(\mathbf{SM}(\boldsymbol{t}), \mathbf{S}(\boldsymbol{t}+\mathbf{1})) \in \varphi_{n+1}\}) \\ + |\varphi_{n+2}|.\sigma(\{\mathbf{S}(\boldsymbol{t}+\mathbf{1})|(\mathbf{SM}(\boldsymbol{t}), \mathbf{S}(\boldsymbol{t}+\mathbf{1})) \in \varphi_{n+2}\})$$

is **minimal**, where

$$\sigma(\mathrm{S}) = \frac{\sum_{v \in \mathrm{S}} \left\| s - \frac{\sum_{v \in S} v}{|\mathrm{S}|} \right\|^2}{|\mathrm{S}|}$$

where S is a set of vectors, and $|\mathrm{S}|$, its cardinal. Finding the exact optimal split would be computationally too expensive. For this reason, we use the following heuristics for optimization: for each dimension j, we evaluate N_{sp} cutting values v_j equally spaced between the extrema values of φ_n, thus we evaluate $N_{sp}.|\{j\}|$ splits in total, and the one with minimal $Qual(j, v_j)$ is finally chosen. This computationally cheap heuristics has produced acceptable results in all the experiments we ran so far. It could potentially be improved by allowing region splits cutting multiple dimensions at the same time in conjunction with a Monte-Carlo based sampling of the space of possible splits.

3) R-IAC Split Algorithm

In **R-IAC**, the splitting mechanism is based on comparisons between the learning progress in the two potential child regions. The principal idea is to perform the **separation which maximizes the dissimilarity of learning progress** comparing the two created regions. This leads to the direct detection of areas where the learning progress is maximal, and to separate them from others (see Fig. 5). This contrasts with **IAC** where regions where built independently of the notion of learning progress.

Reusing the notations of the previous section, in **R-IAC** the split of φ_n into φ_{n+1} and φ_{n+2} is done by choosing j and v_j such that:

$$Qual(j, v_j) = \\ (LP_{n+1}(\{\mathbf{e}(\boldsymbol{t}+\mathbf{1})|(\mathbf{SM}(\boldsymbol{t}), \mathbf{S}(\boldsymbol{t}+\mathbf{1})) \in \varphi_{n+1}\}) \\ - LP_{n+2}(\{\mathbf{e}(\boldsymbol{t}+\mathbf{1})|(\mathbf{SM}(\boldsymbol{t}), \mathbf{S}(\boldsymbol{t}+\mathbf{1})) \in \varphi_{n+2}\}))^2$$

is **maximal**, where

$$LP_k(E) = \frac{\sum_{i=1}^{\frac{|E|}{2}} e(i) - \sum_{i=\frac{|E|}{2}}^{|E|} e(i)}{|E|}$$

Where E is a set of errors values $\{e(i)\}$ with errors indexed by their relative order i of encounter (e.g. error *e(9)* corresponds to a prediction made by the robot before

another prediction which resulted in *e(10)*: this implies that the order of exemplars collected and associated prediction errors are stored in the system), and $LP_k(E)$is the learning progress of region R_k. The heuristics used to find an approximate maximal split is the same as the one described above for **IAC**.

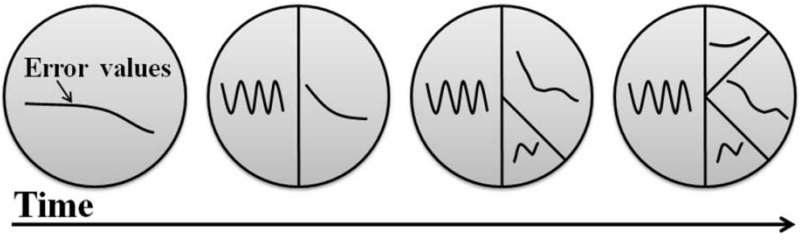


Fig. 5 Evolution of the sensorimotor regions over time. The whole space is progressively subdivided in such a way that the dissimilarity of each sub-region in terms of learning progress is maximal.

2.4 Action Selection Machine

We present here an implementation of Action Selection Machine **ASM**. The **ASM** decides of actions $\mathbf{M}(\boldsymbol{t})$ to perform, given a sensory context $\mathbf{S}(\boldsymbol{t})$. (See Fig. 3.). The ASM heuristics is based on a mixture of several **modes**, which differ between **IAC** and **R-IAC**. Both **IAC** and **R-IAC** algorithms share the same global loop in which modes are chosen probabilistically:

Outline of the global loop of IAC and R-IAC algorithms:

- **Action Selection Machine ASM**: given **S(t)**, execute an action $\mathbf{M}(\boldsymbol{t})$ using the **mode** $(\boldsymbol{n})$ with probability $\boldsymbol{p_n}$and based on data stored in the region tree, with $\boldsymbol{n} \in \{\mathbf{1}, \mathbf{2}\}$ for IAC and $\boldsymbol{n} \in \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$ for R-IAC;
- **Prediction Machine PM**: Estimate the predicted consequence $\tilde{\boldsymbol{S}}_{\boldsymbol{t+1}}$ using the prediction machine **PM ;**
- **External Environment**: Measure the real consequence $\boldsymbol{S}_{\boldsymbol{t+1}}$
- **Prediction Machine PM**: Compute the error $\boldsymbol{e(t+1)} = \boldsymbol{abs}(\tilde{\boldsymbol{S}}_{\boldsymbol{t+1}} - \boldsymbol{S}_{\boldsymbol{t+1}})$**;**
- Update the **prediction machine PM** with $\big(\mathbf{SM(t)}, \mathbf{S(t+1)}\big)$
- **Split Machine SpM:** update the region tree with $\big(\mathbf{SM(t)}, \mathbf{S(t+1)}\big)$ and $\boldsymbol{e(t+1)}$**;**
- **Prediction Analysis Machine PAM:** update evaluation of learning progress in the regions that cover $\big(\mathbf{SM(t)}, \mathbf{S(t+1)}\big)$

We now present the different exploration modes used by the Action Selection Machine, in **IAC** and **R-IAC** algorithm:

1) Mode 1: Random Babbling Exploration

The **random babbling** mode corresponds to a totally random exploration (random choice of $\mathbf{M(t)}$ with a uniform distribution), which does not consider previous

actions and context. This mode appears in both **IAC** and **R-IAC** algorithm, with a probability $\boldsymbol{p_1}$ typically equal to 30%**.**

*2) **Mode 2: Learning Progress Maximization Exploration***

This mode, chosen with a probability $\boldsymbol{p_1}$ typically equal to 70%**,** aims to maximize learning progress, but with two different heuristics in **IAC** and **R-IAC**:

IAC: In the **IAC** algorithm, mode *2* action selection is straightforward: among the leaf regions that cover the current state $\mathbf{S(t)}$ (i.e. for which there exists a $\mathbf{M(t)}$ such that $\mathbf{SM(t)}$ is in the region - there are typically many), the leaf region which learning progress is maximal is found, and then a random action within this region is chosen;

R-IAC: In the **R-IAC** algorithm, we take into account the fact that many regions may have close learning progress values, and thus should be selected roughly equally often, by taking a probabilistic approach to region selection. This avoids the problems of a winner take-all strategy when the region splits do not reflect well the underlying learnability structure of the sensorimotor space. Furthermore, instead of focusing on the leaf regions like in **IAC**, **R-IAC** continues to monitor learning progress in node regions and select them if they have more learning progress: thus learning progress is monitored simultaneously at several scales in the sensorimotor space. Let us give more details:

i) Probabilistic approach to region selection

A region R_n is chosen among all eligible regions $R = \{R_i\}$ (i.e. for which there exists a $\mathbf{M(t)}$ such that $\mathbf{SM(t)}$ is in the region) with a probability $\boldsymbol{P_n}$ proportional to its learning progress LP_n, stored in the associated $\boldsymbol{PAM_n}$:

$$\boldsymbol{P_n} = \frac{|LP_n - min(LP_i)|}{\sum_{i=1}^{|R|} |LP_i - min(LP_i)|}$$

j) Multi-resolution monitoring of learning progress

In the **IAC** algorithm, the estimation of learning progress only happens in leaf regions, which are the only eligible regions for action selection. In **R-IAC**, learning progress is monitored in all regions created during the system's life time, which allows us to track learning progress at multiple resolution in the sensorimotor space. This implies that when a new exemplar is available, **R-IAC** updates the evaluation of learning progress in all regions that cover this exemplar (but only if the exemplar was chosen randomly, i.e. not with mode *3* as described below). Because regions are created in a top-down manner and stored in a tree structure which was already used for fast access in IAC, this new heuristics does not bring computational overload and can be implemented efficiently.

In **R-IAC mode *2***, when a region has been chosen with the probabilistic approach and the multi-resolution scheme, a random action is chosen within this region with a probability $\boldsymbol{p_2}$ typically equal to 60%, (which means this is the dominant mode.

*3) **Mode 3: Error Maximization Exploration***

Mode 3 combines a traditional active learning heuristics with the concept of learning progress: in mode 3, a region is first chosen with the same scheme as in **R-IAC** mode 2. But once this region has been chosen, an action in this region is selected such that the expected error in prediction will be maximal. This is currently implemented through a k-nearest neighbor regression of the function $SM(t) \rightarrow e(t+1)$ which allows finding the point of maximal error, to which is added small random noise (to avoid to query several times exactly the same point). Mode 3 is typically chosen with a probability $\boldsymbol{p_3} = 10\%$ in R-IAC (and does not appear in **IAC**).

2.5 Pseudo-code of R-IAC

RIAC($\boldsymbol{PM, p_1, p_2, p_3, T_{split}, l, \eta, \Gamma, \kappa, \zeta}$)

Init

- Let $\boldsymbol{R_0}$ be the whole space of mathematically possible values of the sensorimotor context **SM(t)** (typically a hypercube in $\mathbb{R}^{\boldsymbol{d}}$);
- Let $\boldsymbol{LP_0 = 0}$ be the learning progress associated to $\boldsymbol{R_0}$**;**
- Let $\boldsymbol{Lex_{R_0}} = \{\emptyset\}$ (later on in the algorithm, $\boldsymbol{Lex_{R_k}}$ will be the set $\{((\mathbf{SM_i(t)}, \mathbf{S_i(t+1)}), \mathbf{e_i(t+1)}, \boldsymbol{\omega_i})\}$ where the set of $(\mathbf{SM_i(t)}, \mathbf{S_i(t+1)})$ components is the set of learning examplars collected in R_k, the set of $\mathbf{e_i(t+1)}$ components is the set of associated prediction errors, and $\boldsymbol{\omega_i}$ is an indice whose value indicates the relative order in which each particular learning examplar was collected within $\boldsymbol{R_k}$);
- Init the prediction/learning machine **PM** with an empty set of learning exemplars;

Loop

Let **S(t)** be the current state;
Let $\boldsymbol{R} = \{\boldsymbol{R_0}, \boldsymbol{R_1}, \ldots, \boldsymbol{R_n}\}$ be the set of subregions $\boldsymbol{R_l}$ of the sensorimotor space such that there exists a **M(t)** such that **SM(t)** $\in \boldsymbol{R_l}$;
For all $\boldsymbol{n}$, let $\boldsymbol{LP_n}$ be the learning progress associated to $\boldsymbol{R_n}$**;**

Action Selection

- Select action selection mode *mode* among **mode 1**, **mode 2** and **mode 3** with probabilities $\boldsymbol{p_1}$, $\boldsymbol{p_2}$, $\boldsymbol{p_3}$;
- If *mode* = **mode 1**
  - Let **M(t)** be a random vector (uniform distribution)
- If *mode* = **mode 2**
  - For $l = 0 \ldots n$,
    let $\boldsymbol{P}_l = \frac{\left|LP_l - min_{LP_i \in \boldsymbol{R}}(LP_i)\right|}{\sum_{i=1}^{|R|}\left|LP_i - min_{LP_i \in \boldsymbol{R}}(LP_i)\right|}$
  - Let $\boldsymbol{R_k}$ be a subregion in $\boldsymbol{R}$ chosen with probability $\boldsymbol{P_k}$, $k \in \{0, \ldots, n\}$ in a roulette wheel manner ;
  - Let **M(t)** be a random vector such that $\boldsymbol{SM(t)} \in \boldsymbol{R_k}$ (uniform distribution);
- If *mode* = **mode 3**
  - For $l = 0 \ldots n$,
    let $\boldsymbol{P}_l = \frac{\left|LP_l - min_{LP_i \in \boldsymbol{R}}(LP_i)\right|}{\sum_{i=1}^{|R|}\left|LP_i - min_{LP_i \in \boldsymbol{R}}(LP_i)\right|}$
  - Let $\boldsymbol{R_k}$ be a subregion in $\boldsymbol{R}$ chosen with probability $\boldsymbol{P_k}$, $k \in \{0, \ldots, n\}$in a roulette-wheel manner ;
  - Let $\boldsymbol{Err_{R_k}}$ be a model of the errors made in prediction in $\boldsymbol{R_k}$ in the past, built with a l -nearest neighbor algorithm on the last η learning examplars collected in $\boldsymbol{R_k}$, belonging to $\boldsymbol{Lex_{R_k}}$ **;**
  - Let **Mmax(t) =** $\boldsymbol{argmax_{M(t)} Err_{R_k}}\left(\boldsymbol{SM(t)}\right)$ obtained by sampling uniformly randomly **Γ** candidates **M(t) ;**
  - **Let M(t) = Mmax(t) +** $\boldsymbol{\varepsilon}$ with $\boldsymbol{\varepsilon}$ a small random number between **0** and **σ** along a uniform distribution.
- **Execute M(t) ;**

Prediction and measurement of the consequences of action

- Estimate the predicted consequence $\tilde{\mathbf{S}}(\boldsymbol{t+1})$ of executing $\mathbf{M}(\mathbf{t})$ in the environment with state **S(t)** using the prediction machine **PM ;**
- Measure the real consequence $\mathbf{S}(\mathbf{t+1})$ after execution of $\mathbf{M}(\mathbf{t})$ in the environment with state **S(t);**
- Compute the error $\boldsymbol{e(t+1)} = \boldsymbol{abs}\left(\tilde{\mathbf{S}}(\boldsymbol{t+1}) - \mathbf{S}(\mathbf{t+1})\right)$**;**
- Update the prediction machine **PM** with the new learning plar $\left(\mathbf{SM}(\mathbf{t}), \mathbf{S}(\mathbf{t+1})\right)$**;**

Update of region models

- Let Ex $= \left(\mathbf{SM(t)}, \mathbf{S(t+1)}, \mathbf{e(t+1)}\right)$
- Let γ be the total number of regions created by the system so far;
- For all regions $\boldsymbol{R_k}$ such that $\mathbf{SM(t)} \in \boldsymbol{R_k}$
  - **Let** $\boldsymbol{\omega}$ be the maximum $\boldsymbol{\omega_i}$ index in $\boldsymbol{Lex_{R_k}}$**;**
  - $\boldsymbol{Lex_{R_k}} = \boldsymbol{Lex_{R_k}} + \left\{\left(\left(\mathbf{SM(t)}, \mathbf{S(t+1)}\right), \mathbf{e(t+1)}, \boldsymbol{\omega}+\mathbf{1}\right)\right\}$ where $\boldsymbol{\omega}+\mathbf{1}$ is an indice used to keep track of the order in which this learning exemplar was stored in relation to others (see below);
  - If card($\boldsymbol{Lex_{R_k}}$) $= \boldsymbol{T_{split}}$

    Create two new regions $\boldsymbol{R}_{\gamma+1}$ and $\boldsymbol{R}_{\gamma+2}$ as subregions of $\boldsymbol{R_k}$ with j, a cutting dimension and v_j, an associated cutting value optimized through random uniform sampling of κ possible candidates and such that:

    1. $\boldsymbol{Lex}_{\boldsymbol{R}_{\gamma+1}}$ is initialized with all the elements in $\boldsymbol{Lex_{R_k}}$ that have a j^{th}component of their $\mathbf{SM}(\boldsymbol{t})$ smaller than v_j;
    2. $\boldsymbol{Lex}_{\boldsymbol{R}_{\gamma+2}}$ is initialized with all the elements in $\boldsymbol{Lex_{R_k}}$ that have a j^{th}component of their $\mathbf{SM}(\boldsymbol{t})$ greater than v_j;
    3. The difference between learning progresses $LP_{\gamma+1}$ and $LP_{\gamma+2}$ measured in both subregions is maximal, i.e.

$$\left(\boldsymbol{LP_{\gamma+1}}\left(\left\{\mathbf{e_i}(\boldsymbol{t}+\mathbf{1})_{\boldsymbol{v}} | (\mathbf{SM_i}(\boldsymbol{t}), \mathbf{S_i}(\boldsymbol{t}+\mathbf{1}), \mathbf{e_i}(\boldsymbol{t}+\mathbf{1}), \boldsymbol{\omega_i}) \in \boldsymbol{Lex}_{\boldsymbol{R}_{\gamma+1}}\right\}\right) - \boldsymbol{LP_{\gamma+2}}\left(\left\{\mathbf{e_i}(\boldsymbol{t}+\mathbf{1})_{\boldsymbol{v}} | (\mathbf{SM_i}(\boldsymbol{t}), \mathbf{S_i}(\boldsymbol{t}+\mathbf{1}), \mathbf{e_i}(\boldsymbol{t}+\mathbf{1}), \boldsymbol{\omega_i}) \in \boldsymbol{Lex}_{\boldsymbol{R}_{\gamma+2}}\right\}\right)\right)^2$$

    is **maximal**, where errors are indexed by their relative order of measurement $\boldsymbol{v}$ calculated from $\boldsymbol{\omega}$ values where

$$LP(E) = \frac{\sum_{i=card(E)-\frac{\zeta}{2}}^{card(E)-\zeta} e_i - \sum_{i=card(E)-\zeta}^{card(E)-\frac{\zeta}{2}+1} e_i}{card(E)}$$

    where ζ defines the time window used to compute learning progress achieved through the acquisition of most recent learning examplars in each region;
  - Store the learning progresses $\mathbf{LP_{\gamma+1}}$ and $\mathbf{LP_{\gamma+2}}$ of the two newly created regions;
  - $\gamma = \gamma + 1$
- For all regions $\boldsymbol{R_k}$ such that $\mathbf{SM(t)} \in \boldsymbol{R_k}$ (except $\boldsymbol{R}_{\gamma+1}$ and $\boldsymbol{R}_{\gamma+2}$ if a split was performed), recompute $\boldsymbol{LP_k}$ and store the value;

EndLoop

2.6 Software

An open-source Matlab-based software library containing the source code of the **IAC** and **R-IAC** algorithms, as well as tools and a tutorial that allow to reproduce all experiments presented in sections IV and V below is made publicly available at: http://flowers.inria.fr/riac-software.zip

2.7 Remarks

Regulation of the growth of complexity. As argumented in detail in [28], the heuristics consisting in preferentially exploring subregions of the sensorimotor space where learning progress is maximal has the practical consequence to lead the robot to explore zones of intermediate complexity/difficulty/contingency, which has been advocated by developmental psychologists (e.g. [22,23,38]) as being the key property of spontaneous exploration in humans. Indeed, subregions which are trivial to learn are quickly characterized by a low plateau in prediction errors, and thus become uninteresting. On the other end of the complexity spectrum, subregions which are unlearnable are characterized with a high plateau in prediction errors and thus are also quickly identified as uninteresting. In between, exploration first focuses on subregions where prediction errors decrease fastest, which typically correspond to lower complexity situations, and when these regions are mastered and a plateau is reached, exploration continues in more complicated subregions where large learning progress is detected.

Key advances of R-IAC over IAC and robustness to potential inaccurate and large number of region splits. Among the various differences between IAC and R-IAC, the two most crucial ones are 1) the ***probabilistic choice of regions*** in R-IAC as opposed to the winner take all strategy in IAC, and 2) the ***multiresolution monitoring of learning progress*** in R-IAC as opposed to the only lowest scale monitoring of IAC. The combination of these two innovations allows the system to cope with potentially inaccurate and supernumerary region splits. Indeed, a problem in IAC was that if for example one homogeneous region with high learning progress was split, the winner-take-all strategy typically biased the system to explore later on only one of the two subregions, which was very inefficient. Furthermore, the more regions were split, which happened continuously given the splitting mechanism, the smaller they became, and because only child regions were monitored, exploration was becoming increasingly focused on smaller and smaller subregions of the sensorimotor space, which was also often quite inefficient. While the new splitting mechanism introduced in this paper allows the system to minimize inaccurate splits, the best strategy to go around these problems was to find a global method whose efficiency depends only loosely on the particular region split mechanism. The probabilistic choice of actions makes the system robust to the potentially unnecessary split of homogeneous regions, and the multiresolution scheme allows the system to be rather insensitive to the creation of an increasing number of small regions.

3 The Prediction Machine: Incremental Regression Algorithms for Learning Forward and Inverse Models

The **IAC** and **R-IAC** system presented above are mostly agnostic regarding the kind of learning algorithm used to implement the prediction machine, i.e. used to learn forward models. The only property that is assumed is that learning must be incremental, since exploration is driven by measures of the improvement of the learnt forward models as new learning examplars are collected. But among incremental algorithm, methods based on neural networks, memory-based learning algorithms, or incremental statistical learning techniques could be used [43]. This agnosticity is an interesting feature of the system since it constitutes a single method to achieve active learning with multiple learning algorithms, i.e. with multiple kinds of learning biases that can be peculiar to each application domain, as opposed to a number of statistical active learning algorithms designed specifically for particular learning methods such as support vector machines, Gaussian mixture regression, or locally weighted regression [41]. Nevertheless, what the robot will learn eventually will obviously depend both on **IAC** or **R-IAC** and on the capabilities of the prediction machine/regression algorithm for which **IAC/R-IAC** drives the collection of learning exemplars.

In robot learning, a particular important problem is to learn the forward and inverse kinematics as well as the forward and inverse dynamics of the body [44,45,46,47]. A number of regression algorithms have been designed and experimented in this context in the robot learning literature, and because a particularly interesting use of **IAC/R-IAC** is for driving exploration for the discovery of the robot's body, as it will be illustrated in the experiments in the next section (and was already illustrated for **IAC** in [27,36]), it is useful to look at state-of-the-art statistical regression methods for this kind of space. An important family of such algorithms is locally weighted regression [45], among which Locally Weighted Projection Regression (LWPR) has recently showed a strong ability to learn incrementally and efficiently forward and inverse models in high-dimensional sensorimotor spaces [46,45]. Gaussian process regression has also proven to allow for very high generalization performances [48]. Another approach, based on Gaussian mixture regression [49,3], is based on the learning of the joint probability distribution of the sensorimotor variables, instead of learning a forward or an inverse model, and can be used online for inferring specific forward or inverse models by well-chosen projections of the joint density. Gaussian mixture regression (GMR) has recently shown a number of good properties for robot motor learning in a series of real-world robotic experiments [3]. It is interesting to note that these techniques come from advances in statistical learning theory, and seem to allow significantly higher performances than for example approaches based on neural-networks [50].

Because it is incremental and powerful, LWPR might be a good basic prediction algorithm to be used in the R-IAC framework for conducting robot experiments. Yet, LWPR is also characterized by a high number of parameters which tuning is not straightforward and thus makes its use not optimal for repeated experiments about **IAC/R-IAC** in various sensorimotor spaces. On the other hand, Gaussian processes and Gaussian mixture regression have much less parameters (only one parameter for GMR, i.e. the number of Gaussians) and are much easier

to tune. Unfortunately, they are batch methods which can be computationally very demanding as the dataset grows. Thus, they cannot be used directly as prediction machines in the **IAC/R-IAC** framework.

This is why we have developed a regression algorithm, called ILO-GMR (Incremental Local Online Gaussian Mixture Regression) which mixes the ease of use of GMR with the incremental memory-based approach of local learning approaches. The general idea is to compute online local few-components GMM/GMR models based on the datapoints in memory whose values in the input point dimensions are in the vicinity of this input point. This local approach allows directly to take into account any novel single datapoint/learning exemplar added to the database since regression is done locally and online. It can be done computationally efficiently thanks to the use of few GMM components, and crucially thanks to the use of an incremental approximate nearest neighbor algorithm derived from recent batch-mode approximate nearest neighbor algorithms [51,52,53]. ILO-GMR has only two parameters: the number of components for local models, and a parameter that defines the notion of local vicinity. Another feature of ILO-GMR is that given its incremental and online nature, with a single set of parameters it can in principle approximate and adapt efficiently to a high variety of mapping to be learnt that may differ significantly in their length scale and might require differ. The technical details and comparison of performances of the ILO-GMR algorithm will be presented in a future paper. Initial experiments to learn the forward kinematics of 6 to 10 DOF's robotic arms have shown that ILO-GMR (tuned with the optimal number of components and vicinity) allows to reach prediction performances in generalization slightly worse than GMR (tuned with the optimal number of components) but similar to LWPR (tuned with the experimentally optimal parameters), the difference between LWPR and ILO-GMR being that ILO-GMR is much easier to tune but slower in prediction due to its only computational of local joint density models. Yet, for the 10 DOF systems of our experiments, these prediction times appear to be compatible with real-time control.

Learning forward motor models is mainly useful if it can be re-used for robot control, hence for inferring inverse motor models [46,48]. This brings up difficult challenges since most robotic systems are highly redundant, which means that the mapping from motor targets in the task space to motor commands in the joint/articulatory space is not a function: one target may correspond to many motor articulatory commands. This is why learning directly inverse models with standards regression algorithm is bound to fail in redundant robots, since when asked to find an articulatory configuration that yields a given target configuration, it will typically output the mean of accurate solutions which is itself not an accurate solutions. Fortunately, there are various approaches to go around this problem [46,48], and one of them is specific to the GMM/GMR approach [50], called the single component least square estimate (SLSE): because this approach encodes joint distributions rather than functions, redundancies are encoded in the GMM and inverse models can be computed by projecting the joint distribution on the corresponding output dimensions and then doing regression based only a the single Gaussian component that gives the highest posterior probability at the given input point. This approach is readily applicable in ILO-GMR, which we have done for the hand-eye-clouds experiment described below.

4 Self-organizing Developmental Trajectories with IAC and Motor Primitives in the Playground Experiment

In this section, we will present the Playground Experiment in which it is shown how the IAC system can drive the exploration and learning of motor primitives by an AIBO robot, and focus on the self-organization of behavioural developmental trajectories of increasing complexity. An extended presentation of these results is available in [27]. Further sections will then present experiments focused on the compared efficiency of **IAC** and **R-IAC** for active learning.

The Playground Experiment setup involves a physical robot as well as a complex sensorimotor system and environment. We use a Sony AIBO robot which is put on a baby play mat with various toys that can be bitten, bashed or simply visually detected (see figure 6). The environment is very similar to the ones in which two or three month old children learn their first sensorimotor skills, although the sensorimotor apparatus of the robot is here much more limited. We have developed a web site which presents pictures and videos of this set-up: http://playground.csl.sony.fr/.

4.1 Motor Primitives

The robot is equipped initially with several parameterizable motor primitives that control its fore arms and its head. Its back legs are frozen such that it cannot walk around. There are three motor primitives: turning the head, bashing and crouch biting. Each of them is controlled by a number of real number parameters, which are the action parameters that the robot controls. The “turning head" primitive is controlled with the pan and tilt parameters of the robot's head. The “bashing" primitive is controlled with the strength and the angle of a whole leg movement (a lower-level automatic mechanism takes care of setting the individual motors controlling the leg and takes care of choosing which leg –left or right- is used depending on the angle parameter). The “crouch biting" primitive is a complex movement consisting in sequencing a crouching with the robot chest while opening the mouth, and then closing the mouth. It is controlled by the depth of crouching (and the robot crouches in the direction in which it is looking at, which is determined by the pan and tilt parameters). To summarize, choosing an action consists in setting the parameters of the 5-dimensional continuous vector **M(t)**:

$$\mathbf{M(t)\} = (p, t, bs, ba, d)}$$

where p is the pan of the head, t the tilt of the head, bs the strength of the bashing primitive, ba the angle of the bashing primitive, and d the depth of the crouching of the robot for the biting motor primitive. All values are real numbers between 0 and 1, plus the value -1 which is a convention used for not using a motor primitive: for example, **M(t)** =(0.3, 0.95, -1, -1, 0.29) corresponds to the combination of turning the head with parameters p=0.3 and t=0.95 with the biting primitive with the parameter d=0.29 but with no bashing movement.

Fig. 6. The Playground Experiment setup.

4.2 Perceptual Primitives

The robot is equipped with three high-level sensors/perceptual primitives based on lower-level sensors. The sensory vector **S(t)** is thus 3-dimensional:

$$\mathbf{S(t)} = (\boldsymbol{Ov}, \boldsymbol{Bi}, \boldsymbol{Os})$$

where:

- ***Ov*** is the binary value of an object visual detection sensor: It takes the value 1 when the robot sees an object, and 0 in the other case. In the playground, we use simple visual tags that we stick on the toys and are easy to detect from the image processing point of view;
- ***Bi*** is the binary value of a biting sensor: It takes the value 1 when the robot has something in its mouth and 0 otherwise. We use the cheek sensor of the AIBO;
- ***Os*** is the binary value of an oscillation sensor: It takes the value 1 when the robot detects that there is something oscillating in front of it, and 0 otherwise. We use the infra-red distance sensor of the AIBO to implement this high-level sensor. This sensor can detect for example when there is an object that has been bashed in the direction of the robot's gaze, but can also detect events due to human walking around the playground (we do not control the environment).

It is crucial to note that initially the robot knows nothing about sensorimotor affordances. For example, it does not know that the values of the object visual detection sensor are correlated with the values of its pan and tilt. It does not know that the values of the biting or object oscillation sensors can become 1 only when biting or bashing actions are performed towards an object. It does not know that some objects are more prone to provoke changes in the values of the ***Bi*** and ***Os*** sensors when only certain kinds of actions are performed in their direction. It does not know for example that to get a change in the value of the oscillation sensor, bashing in the correct direction is not enough, because it also needs to look in the right direction (since its oscillation sensors are on the front of its head). These remarks allow us to understand easily that a random strategy will not be efficient in this environment. If the robot would do random action selection, in a vast majority of cases nothing would happen (especially for the ***Bi*** and ***Os*** sensors).

4.3 The Sensorimotor Loop

The mapping that the robot has to learn is:

$$\mathbf{SM(t) = (}p, t, bs, ba, d\mathbf{)} \rightarrow \mathbf{S(t+1) =(}Ov', Bi', Os'\mathbf{)}$$

The robot is equipped with the **IAC** system. In this experiment, the sensorimotor loop is rather long: when the robot chooses and executes an action, it waits that all its motor primitives have finished their execution, which lasts approximately one second, before choosing the next action. This is how the internal clock for the **IAC** system is implemented in this experiment. On the one hand, this allows the robot to make all the measures necessary for determining adequate values of (***Ov, Bi, Os***). On the other hand and most importantly, this allows the environment to come back to its ``resting state". This means that environment has no memory: after an action has been executed by the robot, all the objects are back in the same state. For example, if the object that can be bashed has actually been bashed, then it has stopped oscillating before the robots tries a new action. This is a deliberate choice to have an environment with no memory: while keeping all the advantages, the constraints and the complexity of a physical embodiment, this makes that mapping from actions to perception learnable in a reasonable time. This is crucial if one wants to do many experiments (already in this case, each experiment lasts for nearly one day). Furthermore, introducing an environment with memory frames the problem of the maximization of internal reward within delayed reward reinforcement problems, for which there exists powerful and sophisticated techniques whose biases would certainly make the results more advanced but would make it more difficult to understand the specific impact and properties of the intrinsic motivation system.

4.4 Results

During an experiment we continuously measure a number of features which help us characterize the dynamics of the robot's development. First, we measure the frequency of the different kinds of actions that the robot performs in a given time window. More precisely:

- The percentage of actions which do not involve the biting and the bashing motor primitive in the last 100 actions (i.e. the robot's action boils down to ``just looking" in a given direction).
- The percentage of actions which involve the biting motor primitive in the last 100 actions.
- The percentage of actions which involve the bashing motor primitive;

Then, we track the gaze of the robot and at each action measure if it is looking towards 1) the bitable object, or 2) the bashable object, or 3) no object. This is possible since from an external point of view we know where the objects are and so it is easy to derive the information from the head position.

Third, we measure the evolution of the frequency of successful biting actions and the evolution of successful bashing actions. A successful biting action is defined as an action which provokes a ``1" value on the ***Bi*** sensor (an object has actually be bitten). A successful bashing action is defined as an action which provokes an oscillation in the ***Os*** sensor.

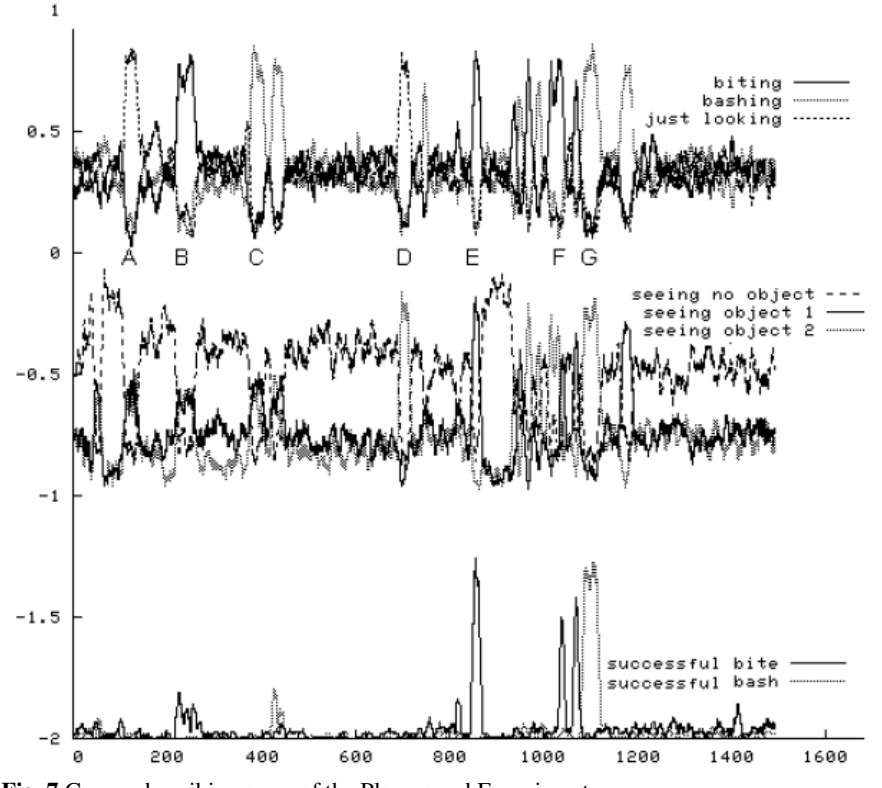


Fig. 7 Curves describing a run of the Playground Experiment.

Top 3: Frequencies for certain action types on windows 100 time steps wide.

Mid 3: Frequencies of gaze direction towards certain objects in windows 200 time steps wide: “object 1" refers to the bitable object, and “object 2" refers to the bashable object.

Bottom 3: Frequencies of successful bite ans successful bash in windows 200 time steps wide.

Figure 7 shows an example of result, showing the evolution of the three kinds of measures on three different levels. A striking feature of these curves is the formation of sequences of peaks. Each of these peaks means basically that at the moment it occurs the robot is focusing its activity and its attention on a small subset of the sensorimotor space. So it is qualitatively different from random action performance in which the curves would be stationary and rather flat. By looking in details at these peaks and at their co-occurence (or not) within the different kinds of measures, we can make a description of the evolution of the robot's behaviour. On figure 7, we have marked a number of such peaks with letters from A to G. We can see that before the first peak, there is an initial phase during which all actions are produced equally often, that most often no object is seen, and that a successful bite or bash only happens extremely rarely. This corresponds to a phase of random action selection. Indeed, initially the robot categorizes the sensorimotor space using only one big region (and so there is only one category), and so all actions in any contexts are equally interesting. Then we observe a peak (A) in the “just looking" curve: this means that for a while, the robot stops biting and bashing, and focuses on just moving its head around. This means that at this point the robot has split the space into several regions, and that a region corresponding to the sensorimotor loop of ``just looking around" is associated to the highest learning progress from the robot's point of view. Then, the next peak (B) corresponds to a focus on the biting action primitive (with various continuous parameters), but it does not co-occur with the looking towards the bitable object. This means that the robot is trying to bite basically in all directions around him : it did not discover yet the affordances of the biting actions with particular objects. The next peak (C) corresponds to a focus on the bashing action primitive (with various continuous parameters) but again the robot does not look towards a particular direction. As the only way to discover that a bashing action can make an object move is by looking in the direction of this object (because the IR sensor is on the cheek), this means that the robot does not use at this point the bashing primitive with the right affordances. The next peak (D) corresponds to a period within which the robot stops again biting and bashing and concentrates on moving the head, but this times we observe that the robot focuses these ``looking" movement in a narrow part of the visual field : it is basically looking around one of the objects, learning how it disappears/reappears in its field of view. Then, there is a peak (E) corresponding to a focus on the biting action, which is this time coupled with a peak in the curve monitoring the looking direction towards the bitable object, and a peak in the curve monitoring the success in biting. It means that during this period the robot uses the action primitive with the right affordances, and manages to bite the bitable object quite often. This peak is then repeated a little bit later (F). Then finally a co-occurrence of peaks (G) appears that corresponds to a period during which the robot concentrates on using the bashing primitve with the right affordances, managing to actually bash the bashable object quite often.

This example shows that several interesting phenomena have appeared in this run of the experiment. First of all, the presence and co-occurrence of peaks of various kinds shows a self-organization of the behavior of the robot, which focuses on particular sensorimotor loops at different periods in time. Second, when we

observe these peaks, we see that they are not random peaks, but show a progressive increase in the complexity of the behaviour to which they correspond. Indeed, one has to remind that the intrinsic dimensionality of the “just looking" behaviour (pan and tilt) is lower than the “biting" behaviour (which adds the depth of the crouching movement), which is itself lower than the “bashing" behaviour (which adds the angle and the strength dimensions). The order of appearance of the periods within which the robot focuses on one of these activities is precisely the same. If we look in more details, we also see that the biting behaviour appears first in a non-affordant version (the robot tries to bite things which cannot be bitten), and then only later in the affordant version (where it tries to bite the biteable object). The same observation holds for the bashing behaviour: first it appears without the right affordances, and then it appears with the right affordances. The formation of focused activities whose properties evolve and are refined with time can be used to describe the developmental trajectories that are generated in terms

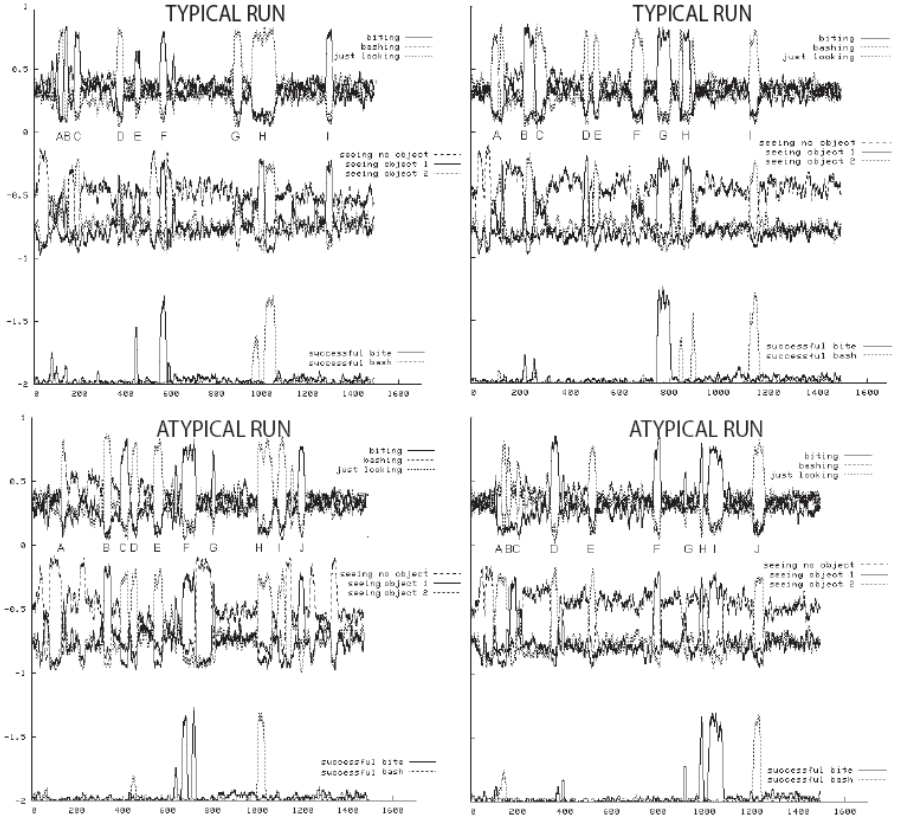


Fig. 8 Various runs of the simulated experiments. In the top squares, we observe two typical developmental trajectories corresponding to the ``complete scenario" described by measure 1. In the bottom curve, we observe rare but existing developmental trajectories.

of stages: indeed, one can define that a new stage begins when a co-occurence of peaks that never occured before happens (and so which denotes a novel kind of focused activity).

We ran several times the experiment with the real robots, and whereas each particular experiment produced curves which were different in the details, it seemed that some regularities in the patterns of peak formation, and so in terms of stage sequences, were present. We then proceeded to more experiments in order to assess precisely the statistical properties of these self-organized developmental trajectories. Because each experiment with the real robot lasts several hour, an in order to be able to run many experiments (200), we developed a model of the experimental set-up. Thanks to the fact that the physical environment was memoryless after each action of the robot, it was possible to make an accurate model of it using the following procedure: we let the robot perform several thousands actions and we recorded each time **SM(t)** and **S(t+1)**. Then, from this database of examples we trained a prediction machine based on locally weighted regression. This machine was then used as a model of the physical environment and the IAC algorithm of the robot was directly plugged into it.

Using this simulated world set-up, we ran 200 experiments, each time monitoring the evolution using the same measures as above. We then constructed higher-level measures about each of the runs, and based on the structure of the peak sequence. Peaks where here defined using a threshold on the height and width of the bumps in the curves. These measures correspond to the answer to these following questions:

- (Measure 1) **Number of peaks?:** How many peaks are there in the action curves (top curves) ?
- (Measure 2) **Complete scenario?:** Is the following developmental scenario matched: first there is a “just looking" peak, then there is a peak corresponding to “biting" with the wrong affordances which appears before a peak corresponding to “biting" with the right affordances, and there is a peak corresponding to “bashing" with the wrong affordances which appears before a peak corresponding to “bashing" with the right affordance (and the relative order between “biting"-related peaks and “bashing"-related peaks is ignored). Biting with the right affordance is here defined as the co-occurence between a peak in the “biting" curve and a peak in the “seeing the biteable object" curve, and biting with the wrong affordances is defined as all other situations. The corresponding definition applies to “bashing".
- (Measure 3) **Nearly complete scenario?:** Is the following less constrained developmental scenario matched: there is a peak corresponding to “biting" with the wrong affordances which appears before a peak corresponding to “biting" with the right affordances, and there is a peak corresponding to “bashing" with the wrong affordances which appears before a peak corresponding to “bashing" with the right affordances (and the relative order between “biting"-related peaks and “bashing"-related peaks is ignored).

- (Measure 4) **Non-affordant bite before affordant bite?:** Is there is a peak corresponding to “biting" with the wrong affordances which appears before a peak corresponding to “biting" with the right affordances?
- (Measure 5) **Non-affordant bash before affordant bash?:** there is a peak corresponding to “bashing" with the wrong affordances which appears before a peak corresponding to “bashing" with the right affordances?
- (Measure 6) **Period of systematic successful bite?:** Does the robot succeeds systematically in biting often at some point (= is there a peak in the “successful bite" curve)?
- (Measure 7) **Period of systematic successful bash?:** Does the robot succeeds systematically in bashing often at some point (= is there a peak in the “successful bash" curve?
- (Measure 8) **Bite before bash?:** Is there a focus on biting which appears before a focus on bashing (independantly of affordance) ?
- (Measure 9) **Successful bite before successful bash?:** Is there a focus on successfully biting which appear before a focus on successfully bashing ?

Table 1 Statistical measures on the 200 simulation-based experiments.

Measures	Results
(1) Number of peaks?	9.67
(2) Complete scenario?	Yes: 34 %, **No: 66 %**
(3) Near complete scenario?	**Yes: 53 %**, No: 47%
(4) Non-affordant bite before affordant bite?	**Yes: 93 %**, No: 7 %
(5) Non-affordant bash before affordant bash?	**Yes: 57 %**, No: 43 %
(6) Period of systematic successful bite?	**Yes: 100 %**, No: 0 %
(7) Period of systematic successful bash?	**Yes: 78 %**, No: 11 %
(8) Bite before bash?	**Yes: 92 %**, No: 8 %
(9) Successful bite before successful bash?	**Yes: 77 %**, No: 23 %

The numerical results of these measures are summarized in table 1. This table shows that indeed some structural and statistical regularities arise in the self-organized developmental trajectories. First of all, one has to note that the complex and structured trajectory described by Measure 2 appears in 34 percent of the cases, which is high given the number of possible co-occurences of peaks which

define a combinatorics of various trajectories. Furthermore, if we remove the test on “just looking", we see that in the majority of experiments, there is a systematic sequencing from non-affordant to affordant actions for both biting and bashing. This shows an organized and progressive increase in the complexity of the behaviour. Another measure confirms this increase of complexity from another point of view: if we compare the relative order of appearance of periods of focused bite or bash, then we find that “focused bite" appears in the large majority of the cases before the “focused bash", which corresponds to their relative intrinsic dimension (3 for biting and 4 for bashing). Finally, one can note that the robot reaches in 100 percent of the experiments a period during which it repeatedly manages to bite the biteable object, and in 78 percent of the experiments it reaches a period during which it repeatedly manages to bash the bashable object. This last point is interesting since the robot was not pre-programmed to achieve this particular task.

These experiments show how the intrinsic motivation system which is implemented (**IAC**) drives the robot into a self-organized developmental trajectory in which periods of focused sensorimotor activities of progressively increasing complexity arise. We have seen that a number of structural regularities arose in the system, such as the tendency of non-affordant behaviour to be explored before affordant behaviour, or the tendency to explore a certain kind of behaviour (bite) before another kind (bash). Yet, one has also to stress that these regularities are only statistical: two developmental trajectories are never exactly the same, and more importantly it happens that some particular trajectories observed in some experiments differ qualitatively from the mean. Figure 8 illustrate this point. The figures on the top-left and top-right corners presents runs which are very typical and corresponds to the “complete scenario" described by Measure 1. On the contrary, the runs presented on the bottom-left and bottom-right corners corresponds to atypical results. The experiment of which curves are presented in the bottom-left corner shows a case where the focused exploration of bashing was performed before the focused exploration of biting. Nevertheless, in this case the regularity “non-affordant before affordant" is preserved. On the bottom-right corner, we observe a run in which the affordant bashing activity appears very early and before any other focused activity. This balance between statistical regularities and diversity has parallels in infant sensorimotor development [60]: there are some strong structural regularities but from individual to individual there can be some substantial differences (for e.g. some infants learn how to crawl before they can sit and other do the reverse).

5 Experimenting and Comparing R-IAC and IAC with a Simple Simulated Robot

In this section, we describe the behavior of the **IAC** and **R-IAC** algorithms in a simple sensorimotor environment that allows us to show visually significant qualitative and quantitative differences, as well as compare them with random exploration.

5.1 Robotics Configuration

We designed a simulated mechanical system, using the *Matlab robotics toolbox* [54]. It consists of a robotic arm using two degrees of freedom, represented by the two rotational axes $\boldsymbol{q_1, q_2}$ as shown on figure 6. The upper part of the arm has been conceived as a bow, which creates a redundancy in the system: for each position and orientation of the tip of the arm, there are two corresponding possible articulatory/joint angle configurations.

This system's sensory system consists in a one-pixel camera, returning an intensity value $\boldsymbol{p}$, set on its extremity as shown on figure 8. The arm is put in a cubic painted environment $\boldsymbol{V}$, whose wallpapers are visible to the one-pixel camera, according to articulatory configurations.

Intensity values measured by the cameras are consequences of both environment $\boldsymbol{V}$ and rotational axes $\boldsymbol{q_1, q_2}$**. So,** we can describe the system input/output mapping with two input dimensions, and one output as:

$$\boldsymbol{p = V(q_1, q_2)}$$

Thus, in this system the mapping to be learnt is state independent since here trajectories are not considered (only end positions are measured) and the perceptual result of applying motor joint angle commands does not depend on the starting configuration.

5.2 Environment Configuration

The front wall consists of an increasing precision checker (Fig. 10), conceived with a black and white pattern. The designed ceiling contains animated wallpaper with white noise, returning a random value to the camera when this one is watching upward bound. Finally, other walls and ground are just painted in white (Fig. 9).

The set up of the system is such that we can sort three kinds of subregions in the sensorimotor space:

- The arm is positioned such that the camera is watching the front wall: for most learning algorithm, this subregion is rather difficult to learn with an increasing level of complexity from left to right (on fig. 7). This feature makes it particularly interesting to study whether **IAC** or **R-IAC** are able to spot these properties and control the complexity of explored sub-subregions accordingly.
- The arm is positioned such that the camera is watching the ceiling: the measured intensity values are random, and thus there are no correlations between motor configurations and sensory measures. Hence, once a few statistical properties of the sensory measures have potentially been learnt (such as the mean), nothing more can be learnt and thus no learning progress can happen.
- The arm is positioned such that a white wall is in front of the camera: the measured intensity value is always 0, so the input/output correlation is trivial. Thus, after it has been learnt that intensity values are constant in this area, nothing can be further learnt.

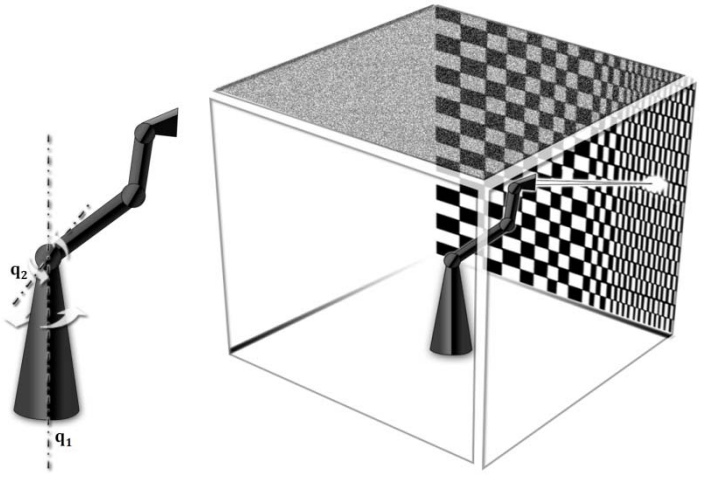


Fig. 9 Representation of a 2 axes arm, with a one pixel camera mounted on its extremity. This arm is put in the center of a cubic room, with different painted walls of different complexities.

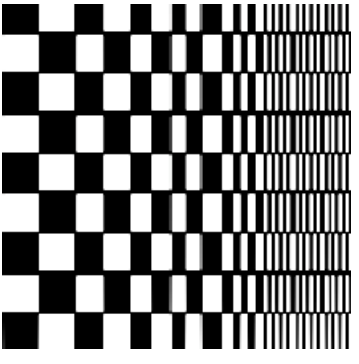


Fig. 10 Wallpaper disposed in the front wall. For many learning algorithms, the complexity increases from left to right.

Because the system has just two motor dimensions and one sensory dimension, it can be visualized using a 2D projection on a plane such as in figure 11. This projection shows a central vertical zone corresponding to the dynamic noise projected on the ceiling. Then, we can easily distinguish the front wall, represented on both sides of the noisy area, because of the redundancy of the arm. The remaining white parts correspond to other walls and the floor.

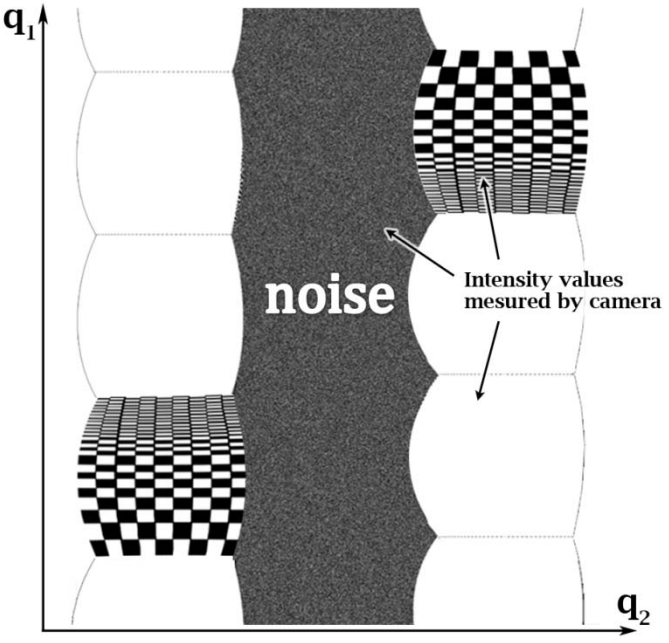


Fig. 11 2D visualization of the sensorimotor space of the robot, with two motor dimensions one sensory dimension.

5.3 Results: Exploration Areas

First, it is interesting to perform qualitative comparisons of the exploration behavior generated by random exploration, **IAC** exploration and **R-IAC** exploration methods.

For each exploration method, the system is allowed to explore its sensorimotor space through 20000 sensorimotor experiments, i.e. it is allowed to collect 20000 learning exemplars. During each run of a given method, every 2000 sensorimotor experiments made by the system one computes a 2D smoothed histogram representing the distribution of explored sensorimotor configurations in the last 500 sensorimotor experiments. This allows us to visualize the evolution of the exploration focus, over time, for each system.

Random exploration obviously leads to a flat histogram.

Fig. 12 presents typical results obtained with **R-IAC** (on the left) and **IAC** (on the right), on a grey scale histogram where darker intensities denote low exploration focus and lighter intensities denote higher exploration focus. First, we observe that **R-IAC** is focalizing on the front wall, containing the image of the checker, using its two possible redundant exploration positions. It avoids the region which contains the white noise, and also the regions just containing a white color. In contrast, we cannot observe the same accuracy to concentrate sensorimotor experiments over interesting areas with the **IAC** exploration method. Here, the algorithm

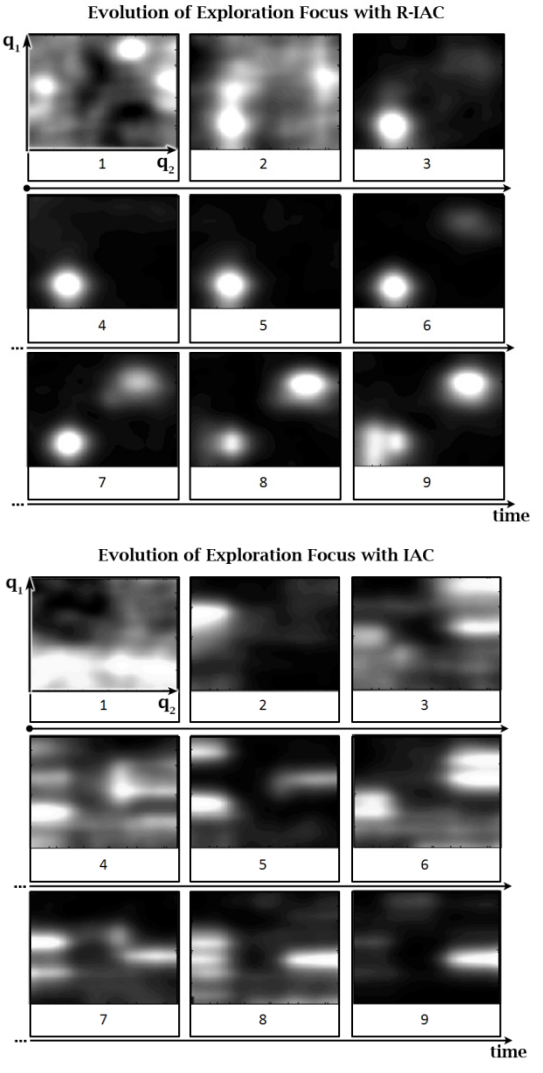


Fig. 12 Evolution of the exploration focus when using **R-IAC** as an exploration heuristics (left) or **IAC** (right). Each square represents the smoothed distribution of explored motor configurations at different times in a given run and over a sliding time window. Darker intensities denote low exploration focus and lighter intensities denote higher exploration focus. We observe that **R-IAC** leads the system to explore preferentially motor configurations such that the camera is looking at the checkerboard, while avoiding zones that are trivial to learn or unlearnable zones. On the contrary, **IAC** is unable to organize exploration properly and "interesting" zones are much less explored.

is indeed avoiding the noise, but we cannot observe precisely some interest toward the front wall, and the system seems to find some things to learn in the back wall, as we can see, watching the bottom-right part of the two last images.

The histograms in figure 12 were smoothed with a large spatial frequency filter to allow us to visualize well the global exploratory behavior. Nevertheless, it is also interesting to use a smaller spatial frequency smoother in order to zoom in and visualize the details of the exploration behavior in the front wall region. Fig. 13 shows a typical result obtained with **R-IAC**, just considering exemplars performed watching the front wall in the bottom-left side of the 2D projection. This sequence shows very explicitly that the system first focuses exploration on zones of lower complexity and progressively shifts its exploration focus towards zones of higher complexity. The system used is thus here able to evaluate accurately the different complexities of small parts of the world, and to drive the exploration based on this evaluation.

5.4 Results: Active Learning

We can now compare the performances of random exploration, **IAC** exploration and **R-IAC** exploration in terms of their efficiency for learning as fast as possible the forward model of the system. For the **R-IAC** method, we included here a version of **R-IAC** without the multi-resolution scheme to assess the specific contribution of multi-resolution learning progress monitoring in the results.

For each exploration method, 30 experiments were run in order to be able to measure means and standard deviations of the evolution of performances in generalization. In each given experiment, every 5000 sensorimotor experiment achieved by the robot, we freezed the system and tested its performances in generalization for predicting $\boldsymbol{p}$ from $(\boldsymbol{q_1}, \boldsymbol{q_2})$ on a test database generated beforehand

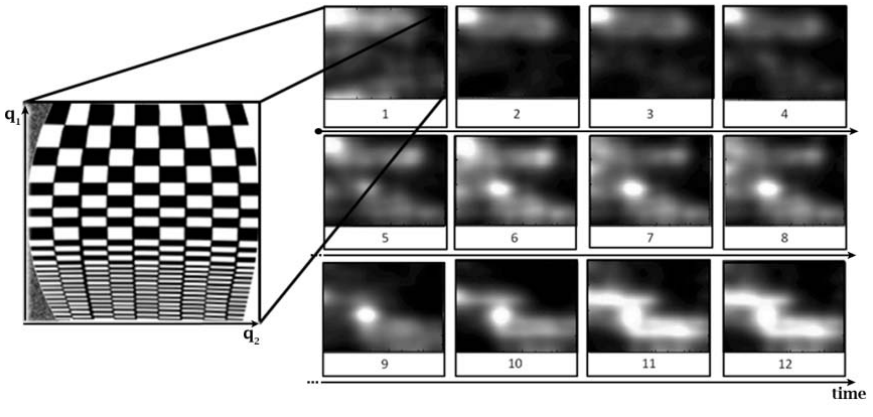


Fig. 13 A zoom into the evolution of the distribution of explored sensorimotor experiments in one of the two subregions where the camera is looking at the checkerboard when **R-IAC** is used. We observe that exploration is first focused on zones of the checkerboard that have a low complexity (for the given learning algorithm), and progressively shifts towards zones of increasing complexity.

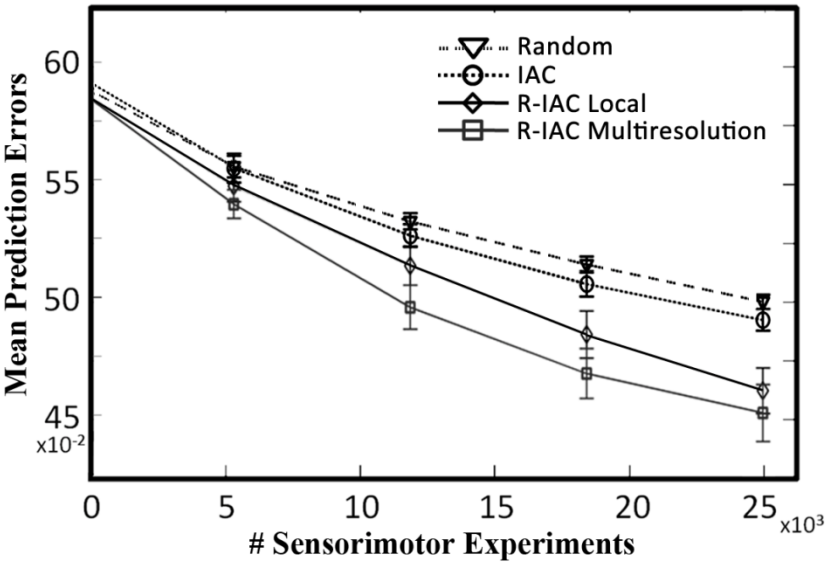


Fig. 14 Comparison of performances of the first and two new implementation of IAC, compared with the random exploration approach.

and independently consisting of random uniform queries in the sensorimotor subspace where there are learnable input/output correlations (i.e. excluding the zone with white noise). Results are provided on figure 14. As we can easily observe, and as already shown in [27], using **IAC** leads to learning performances that are statistically significantly higher than with **RANDOM** exploration. Yet, as figure 14 shows, results of **R-IAC** are statistically significantly much higher than **IAC**, and the difference between **IAC** and **R-IAC** is larger than between **IAC** and random exploration. Finally, we observe that including the multi-resolution scheme into **R-IAC** provides a clear improvement over **R-IAC** without multi-resolution, especially in the first half of the exploration trajectory where inappropriate or too early region splits can slow down the efficiency of exploration if only leaf regions are taken into account for region selection.

6 The Hand-Eye-Clouds Experiment

We will now compare the performances of **IAC** and **R-IAC** as active learning algorithms to learn a forward model in a more complex 6-dimensional robotic sensorimotor space that includes large unlearnable zones. Both algorithms will also be compared with baseline random exploration.

6.1 Robotics Configuration

In this experiment, a simulated robot has two 2-D arms, each with two links and two revolute joints whose angles are controlled by motor **inputs** q_{11}, q_{12}, q_{21}, q_{22}

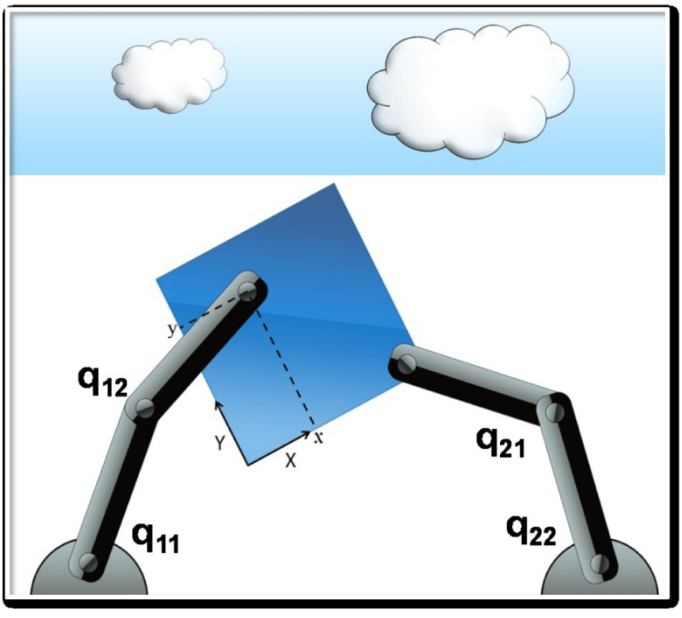


Fig. 15 Experimental setup. The 2D robot has two arms, each with two links and two revolute joints. At the tip of the right arm is rigidly attached a square camera/eye which can sense either the position of the tip of the other arm in its own referential (X, Y) if it is above it, but which can also sense the position of randomly moving clouds when the right arm motor configuration is such that the camera is looking over the top grey area (the « window »). When the camera senses something, the robot does not know initially whether this corresponds to the tip of its left arm or to a cloud. In subregions corresponding to the first alternative, the motor/sensor mapping is correlated and a lot can be learnt. In subregions corresponding to the second alternative, there are no correlations between motors and sensors and nothing can be learnt except some basic statistical properties of the random movement of clouds. There is a third alternative, which actually happens most of the time if the joint space is sampled randomly: the camera looks below the window but does not see its left arm tip. In this very large subregion, the motor to sensor mapping is trivial.

(see figure 15). On the tip of one of the two arms is attached a square camera capable to detect the sensory position $(\boldsymbol{x}, \boldsymbol{y})$ of point-blobs relative to the square. These point-blobs can be either the tip of the other arm or clouds in the sky (see figure 15). This means that when the right arm is positioned such that the camera is over the clouds, which move randomly, the relation between motor configurations and perception is quasi-random. If on the contrary the arms are such that the camera is on top of the tip of the other arm, then there is an interesting sensorimotor relationship to learn. Formally, the system has the relation:

$$(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{E}(\boldsymbol{q_{11}}, \boldsymbol{q_{12}}, \boldsymbol{q_{21}}, \boldsymbol{q_{22}})$$

where (x, y) is computed as follows:

(1) The camera is placed over the white wall: nothing has been detected: $(x, y) = (-10, -10)$;
(2) The camera is on top of the left hand: the value (x, y) of the relative position of the hand in the camera referential C is taken. According to the camera size, the **x** and **y** values are in the interval [0; 6];
(3) The camera is looking at the window: Two random values (x, y) playing the role of random clouds displacement are chosen for output. The interval of outputs corresponds to camera size.
(4) The camera is looking at the window and sees both hand and cloud: the output value (x, y) is random, like if just a cloud had been detected.

This setup can be thought to be similar to the problems encountered by infants discovering their body: they do not know initially that among the blobs moving in their field of view, some of them are part of their "self" and can be controlled, such as the hand, and some other are independent of the self and cannot be controlled (e.g. cars passing in the street or clouds in the sky).

Thus, in this sensorimotor space, the "interesting" potentially learnable subspace is next to a large unlearnable subspace, and also next to a large very simple subspace (when the camera is looking neither to the clouds not to the tip of the other arm).

6.2 Results

In these experiments, the parameters of **IAC** and **R-IAC** are $Tsplit = 250$, the learning progress window is 50, $p_1 = 0.3$, $p_2 = 0.6$, $p_3 = 0.1$. Experiments span a duration of 100000 sensorimotor experiments. The incremental learning algorithm that is used to learn the forward model is the ILO-GMR system described in section 3.

A first study of what happens consists in monitoring the distance between the center of the eye (camera), and the hand (tip of the other arm). A small distance means that the eye is looking the hand, and a high, that it is focusing on clouds (noisy part) or on the white wall. Fig. 16 shows histograms of these distances. We first observe the behavior of the Random exploration algorithm. The curve shows that the system is, in majority, describing actions with a distance of 22, corresponding to the camera looking at clouds or at the white wall. Interestingly, the curve of the **IAC** algorithm is similar but slightly displaced towards shorter distance: this shows that **IAC** pushed the system to explore the "interesting" zone a little more. We finally observe that **RIAC** shows a large difference with both **IAC** and random exploration: the system spends three times more time in a distance inferior to 8, i.e. exploring sensorimotor configurations in which the camera is looking at the other arm's tip. Thus, the difference between **R-IAC** and **IAC** is more important than the difference between **IAC** and random exploration.

Then, we evaluated the quality of the learnt forward model using the three exploration algorithms. We considered this quality in two respects: 1) the capability of the model to predict the position of the hand in the camera given

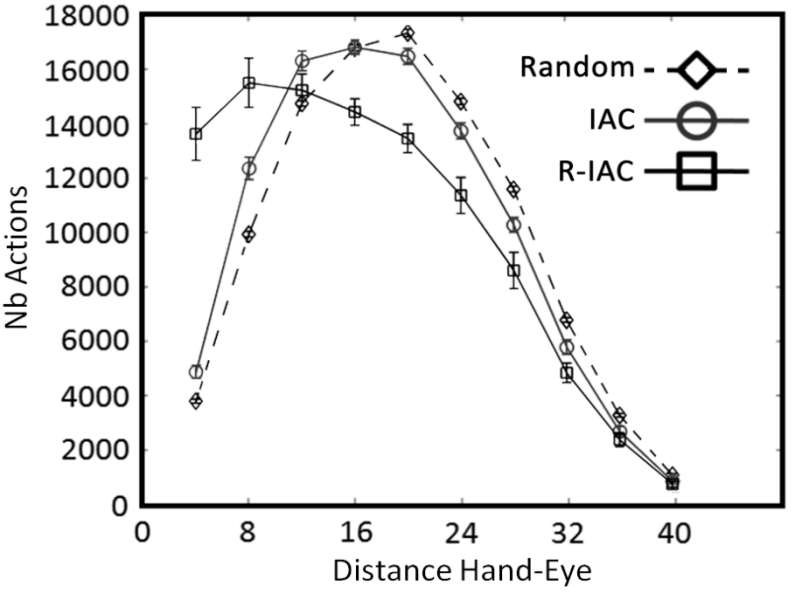


Fig. 16 Mean distributions of hand-center of eye distances when exploration is random, guided by **IAC**, or guided by **R-IAC**. We observe that while **IAC** pushes the system to explore slightly more than random exploration the zones of the sensorimotor space where the tip of the left arm is perceived by the camera or near the camera, **R-IAC** is significantly more efficient than **IAC** for driving exploration in the "interesting" area.

motor configurations for which the hand is within the field of view of the robot; 2) the capacity to use the forward model to control the arm: given a right arm configuration and a visual objective, we tested how far the forward model could be used to drive the left arm to reach this visual objective with the left hand.The first kind of evaluation was realized by first building independantly a test database of 1000 random motor configurations for which the hand is within the field of view, and then using it for testing the learnt models built by each algorithm at various stages of their lifetime (the test consisted in predicting the position of the hand in the camera given joint configurations). Thirty simulations were run, and the evolution of mean prediction errors is shown on the right of figure 17. The second evaluation consisted in generating a set of $\{(\boldsymbol{x},\boldsymbol{y})_C, \boldsymbol{q_{21}}, \boldsymbol{q_{22}} | \boldsymbol{x} > 0 \; and \; \boldsymbol{y} > 0\}$ values that are possible given the morphology of the robot, and then use the learnt forward models to try to move the left arm, i.e. find $(\boldsymbol{q_{11}}, \boldsymbol{q_{12}})$ to reach the $(\boldsymbol{x},\boldsymbol{y})_C$ objectives corresponding to particular $\boldsymbol{q_{21}}, \boldsymbol{q_{22}}$ values**.** Control was realized through inferring an inverse models using ILO-GMR as presented in section 3. The distance between the reached point and the objective point was each time measured, and results, averaged over 30 simulations, are reported in the left graph of figure 17.

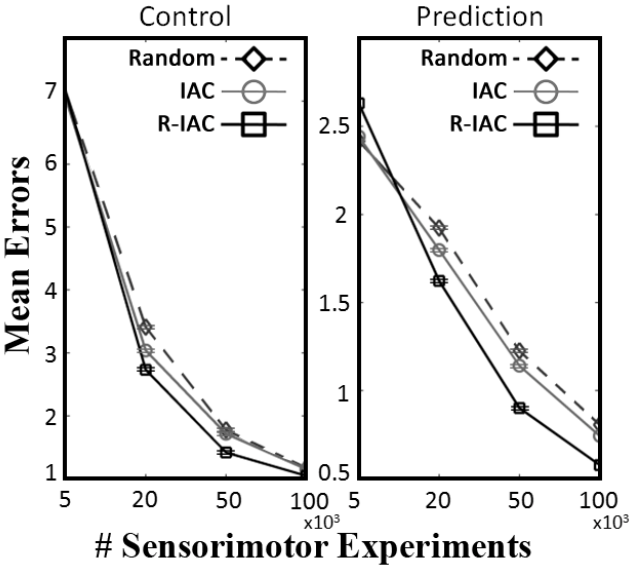


Fig. 17 Left: evolution of performances in control based on the forward model learnt through Random, IAC, and R-IAC exploration heuristics, averaged over 30 simulations. Right : evolution of the generalization capabilities of the learnt forward model with Random, IAC, and R-IAC explo., av. over 30 simulations.

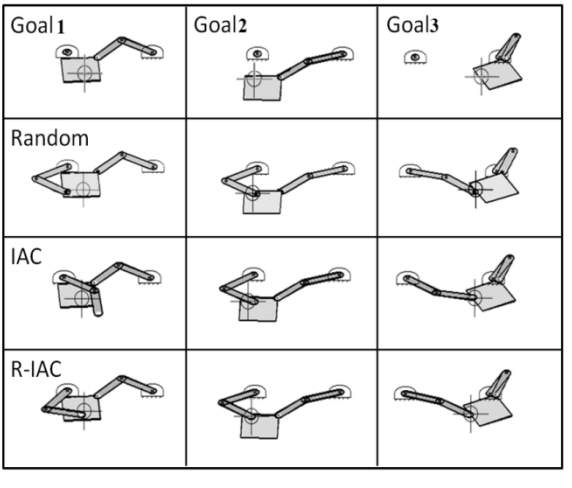


Fig. 18 Examples of performances obtained in control. The first row corresponds to goals fixed. Here, values fixed are the joints of the right hand, and the position in the referential of its eye. The challenge consists of reaching the target (position fixed in the eye) with the left arm.

Both curves on figure 17 confirm clearly the qualitative results of the previous figure: **R-IAC** outperforms significantly **IAC**, which is only slighlty better than random exploration. We have thus shown that **R-IAC** is much more efficient in such an example of complex inhomogeneous sensorimotor space. We also illustrate on figure 18 configurations obtained, considering fixed goals $\{(\boldsymbol{x}, \boldsymbol{y})_C,\ \boldsymbol{q_{21}}, \boldsymbol{q_{22}}\}$, and estimated positioning of the left hand.

7 Conclusion

In this chapter, we have presented two computational intrinsic motivation systems, **IAC** and **R-IAC**, that share substantial properties with human intrinsic motivation systems. As for humans, we have shown that they can be successfully used to self-organize complex sensorimotor developmental trajectories in learning robots, with the formation of stages of increasing complexity. This opens in return new modelling insigths to understand better developmental dynamics in humans [27]. Furthermore, thanks to their capacity to actively regulate the growth of complexity in the exploration process, we have shown that these systems can also be very efficient to drive the motor learning of forward and inverse models in spaces which contain large subregions that are either trivial or unlearnable. For this kind of sensorimotor spaces, typically encountered by developmental robots, we have explained why these intrinsic motivation systems, which we may call developmental active learning systems, will be much more efficient than more traditional active learning heuristics based on the maximization of uncertainty or unpredictability.

Furthermore, we have introduced a novel formulation of **IAC**, called **R-IAC**, and shown that its performances as an intrinsically motivated active learning algorithm were far superior to **IAC** in a complex sensorimotor space where only a small subspace was interesting. We have also shown results in which the learnt forward model was reused in a control scheme.

Further work will study extensions of the current results in several directions. First, experiments with **R-IAC** presented in this chapter were achieved in simulated robots. In spite of the fact that **IAC** was already evaluated in high-dimensional real robotic systems [27,36,34], these experiments were focusing on the self-organization of patterns in developmental trajectories. Evaluating **IAC** and **R-IAC** as active learning methods in high-dimensional real sensorimotor robotic spaces remains to be achieved. Second, both **IAC** and **R-IAC** heuristics could also be conceptualized as mechanisms for generating internal immediate rewards that could serve as a reward system in a reinforcement learning framework, such as for example in intrinsically motivated reinforcement learning [28,33,35]. Leveraging the capabilities of advanced reinforcement learning techniques for sequential action selection to optimize cumulated rewards might allow **IAC** and **R-IAC** to be successfully applied in robotic sensorimotor spaces where dynamical information is crucial, such as for example for learning the forward and inverse models of a force controlled high-dimensional robot, for which guided exploration has been identified as a key research target for the future [47,48].

Also, as argued in [55], it is possible to devise "competence-based" intrinsic motivation systems in which the measure of interestingness characterizes goals in the task space rather than motor configurations in the motor/joint space such as in knowledge-based intrinsic motivation systems like **IAC** or **R-IAC.** We believe that a competence based version of **R-IAC** would allow us to increase significantly exploration efficiency in massively redundant sensorimotor spaces. Finally, an issue of central importance to be studied in the future is how intrinsically motivated exploration and learning mechanisms can be fruitfully coupled with social learning mechanisms, which would be relevant not only for motor learning [56,57,58], but also for developmental language learning grounded in sensorimotor interactions [59].

Acknowledgements

The work presented here was partly realized and supported by the Sony CSL Laboratory, Paris, France (the Playground Experiment in particular), and partly realized and supported by INRIA.

References

[1] Weng, J., McClelland, J., Pentland, A., Sporns, O., et al.: Autonomous mental development by robots and animals. Science 291, 599–600 (2001)

[2] Lungarella, M., Metta, G., Pfeifer, R., Sandini, G.: Developmental robotics: A survey. Connection Sci. 15(4), 151–190 (2003)

[3] Calinon, S., Guenter, F., Billard, A.: On Learning, Representing and Generalizing a Task in a Humanoid Robot. IEEE Transactions on Systems, Man and Cybernetics, Part B, Special issue on robot learning by observation, demonstration and imitation 37(2), 286–298 (2007)

[4] Lopes, M., Melo, F.S., Montesano, L.: Affordance-based imitation learning in robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1015–1021 (2007)

[5] Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st International Conference on Machine Learning (ICML 2004), pp. 1–8 (2004)

[6] Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Proc. 14th International Conference on Machine Learning, pp. 12–20. Morgan Kaufmann, San Francisco (1997)

[7] Alissandrakis, A., Nehaniv, C.L., Dautenhahn, K.: Action, state and effect metrics for robot imitation. In: 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2006), pp. 232–237. Hatfield, United Kingdom (2006)

[8] Argall, B., Chernova, S., Veloso, M.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5), 469–483 (2009)

[9] Asada, M., Ogino, M., Matsuyama, S., Oga, J.: Imitation learning based on visuo-somatic mapping. In: Marcelo, O.K., Ang, H. (eds.) 9th Int. Symp. Exp. Robot., vol. 21, pp. 269–278. Springer, Berlin (2006)

[10] Andry, P., Gaussier, P., Moga, S., Banquet, J.P., Nadel, J.: Learning and communication via imitation: an autonomous robot perspective. IEEE Transactions on Systems, Man, and Cybernetics, Part A 31(5), 431–442 (2001)

[11] Demiris, Y., Meltzoff, A.: The Robot in the Crib: A developmental analysis of imitation skills in infants and robots. Infant and Child Development 17, 43–53 (2008)

[12] Pardowitz, M., Knoop, S., Zollner, R.D., Dillmann, R.: Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. IEEE Transactions on Systems, Man and Cybernetics - Part B 37(2), 322–332 (2007)

[13] Oztop, E., Kawato, M., Arbib, M.: Mirror neurons and imitation: A computationally guided review. Neural Networks 19(3), 254–271 (2006)

[14] Rao, R., Shon, A., Meltzoff, A.: A Bayesian model of imitation in infants and robots. In: Imitation and social learning in robots, humans, and animals. Cambridge University Press, Cambridge (2007)

[15] Arkin, R.C.: Moving Up the Food Chain: Motivation and Emotion in Behavior-based Robots. In: Fellous, J., Arbib, M. (eds.) Who Needs Emotions: The Brain Meets the Robot. Oxford University Press, Oxford (2005)

[16] Fellous, J.M., Arbib, M. (eds.): Who Needs Emotions: The Brain Meets the Robot. Oxford University Press, Oxford (2005)

[17] McFarland, D., Bosser, T.: Intelligent Behavior in Animals and Robots. MIT Press, Cambridge (1993)

[18] Manzotti, R., Tagliasco, V.: From behaviour-based robots to motivation-based robots. Robot. Auton. Syst. 51(2-3), 175–190 (2005)

[19] Stoytchev, A., Arkin, R.: Incorporating Motivation in a Hybrid Robot Architecture. JACIII 8(3), 269–274 (2004)

[20] Arkin, R.C., Fujita, M., Takagi, T., Hasegawa, R.: An ethological and emotional basis for human-robot interaction. Robotics and Autonomous Systems 42(3), 191–201 (2003)

[21] White, R.: Motivation reconsidered: The concept of competence. Psychological 66, 297–333 (1959)

[22] Berlyne, D.: Curiosity and Exploration. Science 153(3731), 25–33 (1966)

[23] Deci, E., Ryan, R.: Intrinsic Motivation and Self-Determination in Human Behavior. Plenum Press, New York (1985)

[24] Schultz, W.: Getting Formal with Dopamine and Reward. Neuron 36, 241–263 (2002)

[25] Dayan, P., Balleine, B.: Reward, Motivation and Reinforcement Learning. Neuron 36, 285–298 (2002)

[26] Redgrave, P., Gurney, K.: The Short-Latency Dopamine Signal: a Role in Discovering Novel Actions? Nature Reviews Neuroscience 7(12), 967–975 (2006)

[27] Oudeyer, P.-Y., Kaplan, F., Hafner, V.: Intrinsic Motivation Systems for Autonomous Mental Development. IEEE Transactions on Evolutionary Computation 11(2), 265–286 (2007)

[28] Barto, A., Singh, S., Chentanez, N.: Intrinsically motivated learning of hierarchical collections of skills. In: Proc. 3rd Int. Conf. Development Learn., San Diego, CA, pp. 112–119 (2004)

[29] Blanchard, A., Cañamero, L.: Modulation of Exploratory Behavior for Adaptation to the Context. In: Biologically Inspired Robotics (Biro-net) in AISB 2006: Adaptation in Artificial and Biological Systems, Bristol, UK (2006)

[30] Der, R., Herrmann, M., Liebscher, R.: Homeokinetic approach to autonomous learning in mobile robots. In: Dillman, R., Schraft, R.D., Wörn, H. (eds.) Robotik 2002, pp. 301–306. VDI, Dusseldorf (2002)

[31] Blank, D.S., Kumar, D., Meeden, L., Marshall, J.: Bringing up robot: Fundamental mechanisms for creating a self-motivated, self-organizing architecture. Cybernetics and Systems 36(2) (2005)

[32] Huang, X., Weng, J.: Novelty and Reinforcement Learning in the Value System of Developmental Robots. In: Proc. Second International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Edinburgh, Scotland, August 10 - 11 (2002)

[33] Schmidhuber, J.: Curious model-building control systems. In: Proc. Int. Joint Conf. Neural Netw., Singapore, vol. 2, pp. 1458–1463 (1991)

[34] Oudeyer, P.-Y., Kaplan, F.: Discovering Communication. Connection Science 18(2), 189–206 (2006)

[35] Schembri, M., Mirolli, M., Baldassarre, G.: Evolution and Learning in an Intrinsically Motivated Reinforcement Learning Robot. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 294–303. Springer, Heidelberg (2007)

[36] Kaplan, F.: Intrinsically Motivated Machines. In: Lungarella, M., Iida, F., Bongard, J.C., Pfeifer, R. (eds.) 50 Years of Aritficial Intelligence. LNCS (LNAI), vol. 4850, pp. 304–315. Springer, Heidelberg (2007)

[37] Fedorov, V.: Theory of Optimal Experiment. Academic, New York (1972)

[38] Gibson, E.J.: Principles of perceptual learning and development. Appleton-Century-Crofts, New-York (1969)

[39] Berlyne, D.: Conflict, Arousal, and Curiosity. McGraw-Hill, New York (1960)

[40] Csikszentmihalyi, M.: Creativity-Flow and the Psychology of Discovery and Invention. Harper Perennial, New York (1996)

[41] Cohn, D., Ghahramani, Z., Jordan, M.: Active learning with statistical models. J. Artif. Intell. Res. 4, 129–145 (1996)

[42] Hasenjager, M., Ritter, H.: Active Learning in Neural Networks. In: New learning paradigms in soft computing, pp. 137–169. Physica-Verlag GmbH, Berlin (2002)

[43] Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, Chichester (2006)

[44] Vijayakumar, S., Schaal, S.: LWPR: An O(n) Algorithm for Incremental Real Time Learning in High Dimensional Space. In: Proc. of Seventeenth International Conference on Machine Learning (ICML 2000) Stanford, California, pp. 1079–1086 (2000)

[45] D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: IEEE International Conference on Intelligent Robots and Systems (IROS 2001). IEEE, Piscataway (2001)

[46] Peters, J., Schaal, S.: Learning to control in operational space. International Journal of Robotics Research 27, 197–212 (2008)

[47] Salaün, C., Padois, V., Sigaud, O.: Control of redundant robots using learned models: an operational space control approach. In: IEEE International Conference on Intelligent Robots and Systems, IROS 2009 (2009)

[48] Yeung, D.Y., Zhang, Y.: Learning inverse dynamics by Gaussian process regression under the multi-task learning framework. In: Sukhatme, G.S. (ed.) The Path to Autonomous Robots, pp. 131–142. Springer, Heidelberg (2009)

[49] Ghahramani, Z.: Solving inverse problems using an EM approach to density estimation. In: Mozer, M.C., Smolensky, P., Toureztky, D.S., Elman, J.L., Weigend, A.S. (eds.) Proceedings of the 1993 Connectionist Models Summer School, pp. 316–323. Erlbaum Associates, Hillsdale (1993)

[50] Rasmussen, C.E.: Evaluation of Gaussian Process and other Methods for Non-linear Regression. PhD thesis, Department of Computer Science, University of Toronto (1996)

[51] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An Optimal Algorithm for Approximate Nearest Neighbor Searching. Journal of the ACM 45, 891–923 (1998)

[52] Maneewongvatana, S., Mount, D.M.: Analysis of Approximate Nearest Neighbor Searching with Clustered Point Sets, Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges. In: Goldwasser, M.H., Johnson, D.S., McGeoch, C.C. (eds.) Fifth and Sixth DIMACS Implementation Challenges. The DIMACS Series in Discr. Math. and Theoret. Comp. Sci, vol. 59, pp. 105–123. AMS (2002)

[53] Filliat, D.: A visual bag of words method for interactive qualitative localization and mapping. In: Proceedings of the International Conference on Robotics and Automation, ICRA (2007)

[54] Corke, P.I.: A robotics toolbox for Matlab. IEEE Robotics and Automation Magazine 1(3), 24–32 (2006)

[55] Oudeyer, P.-Y., Kaplan, F.: How can we define intrinsic motivation? In: Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems, Lund University Cognitive Studies. LUCS, Brighton, Lund (2008)

[56] Kuniyoshi, Y., Yorozu, Y., Inaba, M., Inoue, H.: From visuo-motor self learning to early imitation-a neural architecture for humanoid learning. In: IEEE Int. Conf. Robotics and Automation, vol. 3, pp. 3132–3139 (2003)

[57] Lopes, M., Mello, F., Montesano, L., Santos-Victor, J.: Abstraction Levels for Robotic Imitation: Overview and Computational Approaches. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 313–355. Springer, Heidelberg (2010)

[58] Thomaz, A.L., Breazeal, C.: Experiments in Socially Guided Exploration: Lessons learned in building robots that learn with and without human teachers. Connection Science, Special Issue on Social Learning in Embodied Agents 20(2,3), 91–110 (2008)

[59] Kaplan, F., Oudeyer, P.-Y., Bergen, B.: Computational Models" in the Debate over Language Learnability. Infant and Child Development 17(1), 55–80 (2008)

[60] Thelen, E., Smith, L.B.: A Dynamic Systems Approach to the Development of Cognition and Action. MIT Press, Cambridge (1994)

[61] Baranes, A., Oudeyer, P.-Y.: R-IAC: Robust Intrinsically Motvated Active Learning. In: Proceedings of the IEEE International Conference on Development and Learning (2009)

Learning to Exploit Proximal Force Sensing: A Comparison Approach

Matteo Fumagalli, Arjan Gijsberts, Serena Ivaldi, Lorenzo Jamone, Giorgio Metta, Lorenzo Natale, Francesco Nori, and Giulio Sandini

Abstract. We present an evaluation of different techniques for the estimation of forces and torques measured by a single six-axis force/torque sensor placed along the kinematic chain of a humanoid robot arm. In order to retrieve the external forces and detect possible contact situations, the internal forces must be estimated. The prediction performance of an analytically derived dynamic model as well as two supervised machine learning techniques, namely Least Squares Support Vector Machines and Neural Networks, are investigated on this problem. The performance are evaluated on the normalized mean square error (NMSE) and the comparison is made with respect to the dimension of the training set, the information contained in the input space and, finally, using a Euclidean subsampling strategy.

Keywords: Force sensing, machine learning, humanoid robotics.

1 Introduction

Emerging applications require robots to act safely in dynamic, unstructured environments. In order to avoid damaging the robot and the surrounding environment (physical objects and/or interacting agents), special sensors are usually applied to detect contact situations [21, 1]. Classical approaches to manipulation exploit a force/torque (FT) sensor placed on the end-effector, where most of the interactions occur. External forces acting on the other parts of the arm, however, cannot be measured with this configuration. Furthermore, it may not be feasible to put the sensor on the end-effector, due to its size or weight. An alternative solution is to place the sensor at the base of the manipulator or along the kinematic chain (e.g. after the shoulder) [19, 6].

Robotics, Brain and Cognitive Science Department
Italian Institute of Technology
Via Morego, 30 Genoa 16163
e-mail: name.surname@iit.it

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 149–167.
springerlink.com

In this case, the FT sensor measures both external and internal forces, the latter being the ones depending on gravitational, Coriolis and inertial forces. This solution allows the robot to detect interaction with the environment not only on the end-effector (e.g. voluntarily touching or grasping an object), but on the whole arm (e.g. hitting unexpected obstacles, being stopped by a human agent during motion). In order to accurately detect the external contribution of the forces, the manipulator dynamics must be compensated, i.e. the internal forces must be known, modeled or estimated.

Multiple approaches can be used for the estimation of these internal forces. Firstly, the functional estimation can be done using an analytical model describing the physics of the system, or at least its most significant properties. Model-based estimation strongly relies on the availability of a (mathematical) model of the robot [20], and is recommended only if the kinematic and dynamic parameters are known or identifiable with high accuracy. To this purpose, rigid multi-body dynamic modeling is generally used and some or all the parameters are identified [24] in order to improve the model accuracy. Within this context, the overall model accuracy is primarily limited by the (potentially nonlinear) effects which the model does not explicitly take into account (e.g. gearbox backlash). Alternatively, supervised machine learning approaches can be used to approximate the internal dynamic model from a set of training examples. This approach may be preferred when explicitly modeling all possible nonlinear effects is cumbersome [25]. The main drawback of supervised learning methods is the need for collecting a rich and significant training set. Furthermore, it may be necessary to perform the training phase offline, due to the high computational requirements of these learning methods. In contrast, the model-based approach only needs to identify a small set of significant parameters; this identification technique requires much fewer data and computational resources and therefore can typically be performed efficiently online.

In this chapter, we investigate an analytical model and two supervised machine learning methods (Least Squares Support Vector Machines and Feed-forward Neural Networks) for the estimation of internal forces in a robotic arm, which is equipped with a six-axis FT sensor inside the kinematic chain. It seems reasonable to assume, however, that the results can be generalized to similar problems in robotics (i.e. problems related to the estimation of the dynamical parameters of a kinematic chain).

Firstly, we focus our attention on the amount of training data necessary to obtain accurate predictions. The qualitative measure for the prediction is the average Normalized Mean Square Error (NMSE) for the forces and torques in three dimensions. In our framework, the minimum amount of external forces that the robot can detect is proportional to the magnitude of this estimation error; this value is critical in order to have safe interaction with the environment. We expect machine learning methods to benefit from larger data sets; on the other hand, model based techniques should be more insensitive to the size of the training set.

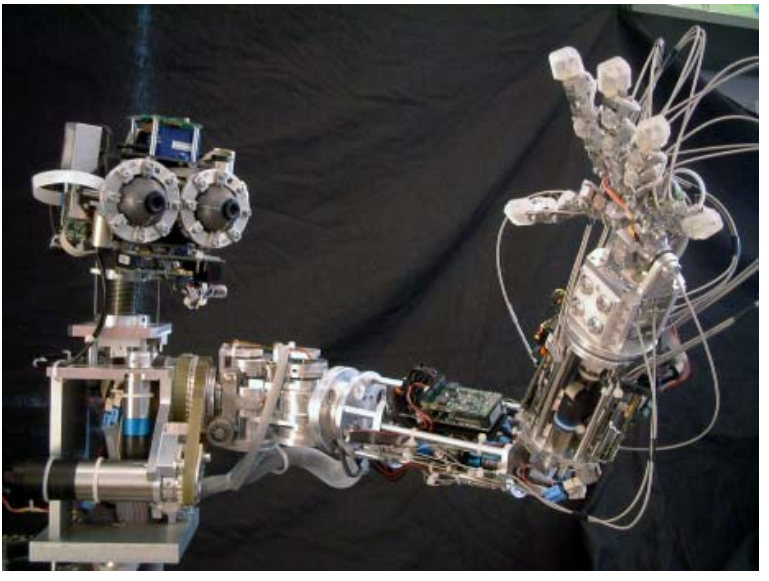


Fig. 1 The humanoid robot James.

Secondly, we pose our interest in understanding how the type of supplied data influences the estimation error. In particular, we verify empirically the usefulness of velocity and acceleration measurements when estimating (relatively complex) dynamical models. This issue is particularly important in the field of humanoid robotics, where smooth motions are usually preferred. As a consequence, velocities and accelerations need to satisfy some smoothness requirements, which prevent data from being completely random[1]. Therefore, successfully exploiting velocity and acceleration data is difficult, especially without specific sensors dedicated to their measurement.

At last, we analyze the effect of sampling distribution on the generalization performance. This analysis can give information about the way training data should be gathered or subsampled to practical dimensions from a larger training set.

The chapter is organized as follows. In Section 2 the robotic platform and the estimation problem are described. Section 3 describes three different approaches for internal forces estimation: an analytical model with identified parameters and the two machine learning methods. Experimental results are reported and discussed in Section 4. Section 5 contains the conclusions.

[1] Typical identification techniques make strong assumptions on the supplied data set. Machine learning techniques assume sufficiently distributed samples that cover the variability of the underlying function. Model based approaches assume persistently exciting conditions (see [10] for a definition).

2 Robot Setup and Problem Formulation

This work has been carried out on the humanoid robot James [8]. James is a humanoid torso, consisting of a 7 DOF head, a 7 DOF left arm and a 8 DOF left hand, with the overall size of a 10 years old boy (cf. Fig. 1). Among the 7 degrees of freedom of the arm (3 for the shoulder, 1 for the elbow and 3 for the wrist), only 4 (the shoulder and elbow) have been considered in this work[2].

At the top of the upper arm, just below the shoulder, a single 6-axis FT sensor (ATI mini45 [15]) is placed (see Fig. 2(a)). This solution has been chosen because most of the space in the upper and forearm is occupied by the motors actuating the wrist, elbow and fingers, and by the DSP boards used to control them. Furthermore, this placement enables the robot to detect both internal forces due to arm motion and external forces due to contacts between the arm/hand and the environment.

As explained before, the FT sensor measures both internal and external forces, the latter being the ones to be determined for interaction control purpose (e.g. obstacle detection and avoidance). The whole arm surface is taken into account for possible contact points (so the contact may happen in any point on the arm, not only on the end-effector). In the following we will discuss the retrieval of the external forces and the consequent need to estimate the internal ones.

Let us consider an open kinematic chain with n degrees of freedom. Let $\mathbf{q} \in \mathbb{R}^n$ be the generalized coordinates describing the pose of the kinematic chain. The FT sensor measurement will be denoted $\mathbf{x} = [\mathbf{f}^\top, \boldsymbol{\tau}^\top]^\top \in \mathbb{R}^6$. As previously said, this quantity contains both external and internal forces $\mathbf{f} \in \mathbb{R}^3$ and torques $\boldsymbol{\tau} \in \mathbb{R}^3$. Specifically we have:

$$\mathbf{x} = \mathbf{x}_I + \mathbf{x}_E \ , \tag{1}$$

where $\mathbf{x}_I$ and $\mathbf{x}_E$ refer to the internal and external forces/torques, respectively. More precisely, neglecting the effect of the elasticity of the transmissions and defining $\mathbf{f}_E, \boldsymbol{\tau}_E$ as the external forces and torques applied at the contact point, equation (1) can be expanded as follows (see [20] for details on the derivations):

$$\begin{bmatrix} \mathbf{f} \\ \boldsymbol{\tau} \end{bmatrix} = \underbrace{M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q})}_{\mathbf{x}_I} + \underbrace{T(\mathbf{q}, \mathbf{d}) \begin{bmatrix} \mathbf{f}_E \\ \boldsymbol{\tau}_E \end{bmatrix}}_{\mathbf{x}_E} \ , \tag{2}$$

where M, C and G are the inertial, Coriolis and gravity matrices of the dynamic system equations, and T is a roto-translation matrix describing the transformation of the external forces from the contact point reference frame

[2] The justification for this simplification is that state of the wrist (position, velocity and acceleration) only has a minor effect on the internal forces, due to the relatively negligible amount of mass after the wrist (i.e. the hand mass) with respect to the whole arm.

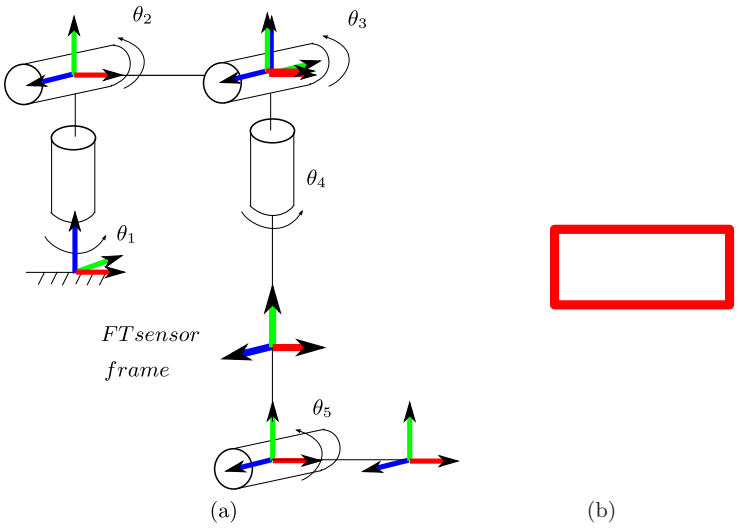


Fig. 2 (a) The reference frames for James' arm. Of the five joints, only four are independently actuated, θ_3 and θ_4 being mechanically coupled. (b) Detail of James' upper-arm, showing the FT sensor and its placement (red square).

to the sensor reference frame, with $\mathbf{d}$ being the distance vector from the contact point to the sensor.

Whenever the robot interacts with an external object, a non-null external force component $\mathbf{x}_E$ arises: in order to detect a collision or a contact, $\mathbf{x}_E$ must be identified from the sensor measurements $\mathbf{x}$. Practically, the identification can be performed by subtracting the internal forces ($\mathbf{x}_I$) from the measured ones ($\mathbf{x}$):

$$\mathbf{x}_E = \mathbf{x} - \mathbf{x}_I \ , \tag{3}$$

which yields an indirect measurement of the external forces and torques[3]. Then, the vector $\mathbf{x}_I$ must be computed from the model, or derived from experimental data. When the robot moves freely in its workspace, the sensor only perceives the internal components of forces and torques (i.e. $\mathbf{x}_I = \mathbf{x}$). These components only depend on position, velocity and acceleration of the joints. The problem of retrieving $\mathbf{x}_E$ is therefore reduced to the estimation of the internal forces and torques, i.e. the mapping from $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ to $\mathbf{f}, \boldsymbol{\tau}$:

$$\mathbf{x}_I = f(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}). \tag{4}$$

[3] Notice that $\mathbf{x}_E$ does not correspond to the real external forces $\mathbf{f}_E$ and torques $\boldsymbol{\tau}_E$, but to their projection on the sensor, i.e. $\mathbf{x}_E = T(\mathbf{q}, \mathbf{d})[\mathbf{f}_E^\top, \boldsymbol{\tau}_E^\top]^\top$. To retrieve the real external forces and torques, we must also know the distance $\mathbf{d}$ from the contact point to the sensor. In the future, we plan to mount a full body sensing skin [5], which will provide the necessary feedback to detect the contact location.

3 Proposed Approaches

Two distinct ways to identify $f(\cdot)$ in equation (4) are: (1) deriving it analytically, (2) approximating it using a set of examples (i.e. machine learning). In the latter case, the learning algorithm is agnostic to the underlying dynamics model that is used to produce the examples. One advantage of this approach is that nonlinear effects do not need to be explicitly modeled, as these are learned implicitly by the algorithm. In this section, we will detail the model-based approach and two machine learning algorithms, namely Least Squares Support Vector Machines and Neural Networks.

Remark 1. It is worth discussing some issues related to the noise effecting the measurement equation (4) which is at the base of all the proposed identification methods. Among the different contributions, the F/T sensor itself is a primary source of noise (see the specifications in [15]) but it is not the only one. As a matter of fact, also the velocity and acceleration measurements are subject to numerical inaccuracies, as these are computed using first and second order numerical differentiation of the position measurements. These derivatives are estimated based on the difference between the samples at time t and $t - W$, i.e.

$$\dot{\mathbf{q}}_t = \frac{\mathbf{q}_t - \mathbf{q}_{t-W}}{W\Delta T} \ , \ \ddot{\mathbf{q}}_t = \frac{\dot{\mathbf{q}}_t - \dot{\mathbf{q}}_{t-W}}{W\Delta T} \quad \text{for } t = W, \ldots, \infty \ . \tag{1}$$

This computation is performed on the DSP boards embedded in the robot arm at 1 kHz rate, i.e. $\Delta T = 1$ ms. The window length W has been set to 35, i.e. $W\Delta T = 35$ms[4]. Other sources of noise include communication delays effecting the synchronization of sensory measurements and reliability of the position measurements in presence of elasticity in the actuation design (elastic tendons and rubber transmission belts). The complex interaction of these various sources of noise makes it very difficult to characterize the overall system noise and therefore a complete analysis will be left outside the scope of the current paper.

3.1 Model-Based Approach

Let us consider a robotic manipulator with n degrees of freedom and links, and a force/torque sensor located in the middle of the kinematic chain, immediately after one of the joints. As already pointed out, the sensor will measure both internal ($\mathbf{x}_I$) and external ($\mathbf{x}_E$) force/torque component acting on the following links. In this section, we assume that the FT sensor measures only the internal forces, i.e. $\mathbf{x}_E \equiv \mathbf{0}$. Therefore, (2) can be written as:

[4] This window length has been chosen specifically to low-pass filter the position measurements and the computed velocities, whilst maintaining sufficient accuracy.

$$\begin{bmatrix}\mathbf{f}\\ \boldsymbol{\tau}\end{bmatrix} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) \ . \tag{2}$$

Starting from this formulation, we derive a model based approach for estimating the parameters in (2). In order to tune this model, a set of parameters that best fits the force/torque acquisition, given a certain data set of joint positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$ and accelerations $\ddot{\mathbf{q}}$, needs to be found. Equation (2) is written as the linear product of a matrix $D(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ and a vector $\boldsymbol{\eta}$ (see [10] for details). The matrix $D(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ depends solely on the joint positions, velocities and accelerations, whereas $\boldsymbol{\eta}$ contains the dynamical parameters that we would like to estimate. Practically:

$$\mathbf{x} = \begin{bmatrix}\mathbf{f}\\ \boldsymbol{\tau}\end{bmatrix} = D(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\boldsymbol{\eta} \ , \tag{3}$$

where $\boldsymbol{\eta}$ has a complex structure that can be formalized as follows:

$$\boldsymbol{\eta} = [\psi,\ \Psi]^\top \ .$$

The row vectors ψ and Ψ depend only on the system dynamical parameters and have the following structure:

$$\psi = \left[\, m_1\varphi \cdots\ m_n\varphi \,\right] \in \mathbb{R}^{n^\psi} \tag{4}$$

$$\Psi = \left[\, \mathbf{r}_1 \cdots\ \mathbf{r}_n \,\right] \in \mathbb{R}^{n^\Psi} \ , \tag{5}$$

where

$$\varphi = \left[\, \mathbf{l}_1^\top \cdots\ \mathbf{l}_n^\top,\ \mathbf{c}_1^\top \cdots\ \mathbf{c}_n^\top \,\right] \in \mathbb{R}^{n^\varphi} \tag{6}$$

$$\mathbf{r}_i = \left[\, \mathbf{s}_{i,1} \cdots\ \mathbf{s}_{i,n},\ I_i^1 \cdots\ I_i^6 \,\right] \tag{7}$$

$$\mathbf{s}_{i,j} = \left[\, m_i\varphi_j^2\ m_i\varphi_j\varphi_{j+1} \cdots\ m_i\varphi_j\varphi_{n_\varphi} \,\right] \ . \tag{8}$$

For each link $i = 1,\ldots,n$, $\mathbf{l}_i \in \mathbb{R}^3$ is the vector representing the lengths of the link in the x, y and z directions with respect to the previous joint's reference frame, $\mathbf{c}_i \in \mathbb{R}^3$ is the vector of the center of mass of each link, with respect to the same reference frame[5]. Further, $m_i \in \mathbb{R}$ and $I_i^k \in \mathbb{R}$ are the mass and inertial parameters of each link for $k = 1,\ldots,6$. Interestingly, equation (3) can be further simplified to the form $\mathbf{x} = \hat{D}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is the minimum set of identifiable parameters, i.e. a linear combination of the elements of vector $\boldsymbol{\eta}$ (see [10] for details).

The vector $\boldsymbol{\phi}$ of the system dynamical parameters can be often retrieved from an accurate model of the robot (e.g. CAD drawings), but this procedure

[5] Each kinematic chain link has an associated reference frame, defined by the Denavit-Hartenberg convention [3] . All the dynamic and kinematic quantities of each link (center of mass, inertia, lengths, etc.) refer to the associated reference frame.

is typically neither feasible nor accurate. Different ways for identifying the dynamic parameters can be found in the literature, but their discussion is out of the scope of this work (the interested reader should refer to [6, 10]). Here we focus on a technique based on a weighted linear least squares solution. Let us define a weighting diagonal matrix ω containing the variances of each component of force (f_x, f_y, f_z) and torque (τ_x, τ_y, τ_z):

$$\omega = \begin{bmatrix} \frac{1}{\sigma^2_{f_x}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma^2_{f_y}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma^2_{\tau_z}} \end{bmatrix} . \tag{9}$$

At the i^{th} time instant, we measure the system position ($\mathbf{q}_i$), velocity ($\dot{\mathbf{q}}_i$) and acceleration ($\ddot{\mathbf{q}}_i$) and the associated force sensor output $\mathbf{x}_i$. After ℓ time samples, a possible estimation for the vector $\boldsymbol{\phi}$ of dynamical parameters is given by the vector $\boldsymbol{\phi}^\circ$ which minimizes the (weighted) norm of the error vectors ($\mathbf{x}_i - \hat{D}_i\boldsymbol{\phi}$), i.e.:

$$\boldsymbol{\phi}^\circ = \arg\min_{\boldsymbol{\phi}} \sum_{i=1}^{\ell} (\mathbf{x}_i - \hat{D}_i\boldsymbol{\phi})^\top \omega (\mathbf{x}_i - \hat{D}_i\boldsymbol{\phi}) \ . \tag{10}$$

where we defined $\hat{D}_i = \hat{D}(\mathbf{q}_i, \dot{\mathbf{q}}_i, \ddot{\mathbf{q}}_i)$. The explicit solution is given by:

$$\boldsymbol{\phi}^\circ = \Delta^\dagger_\Omega \mathbf{Y} = \left[\Delta^\top \Omega \Delta\right]^{-1} \Delta^\top \Omega \mathbf{Y} \ , \tag{11}$$

where $\Omega = diag(\omega)$ and

$$\Delta = \begin{bmatrix} \hat{D}_1 \\ \hat{D}_2 \\ \vdots \\ \hat{D}_\ell \end{bmatrix} \quad \mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_\ell \end{bmatrix} . \tag{12}$$

Remark 2. Given the discussion above, learning the optimal parameters value $\boldsymbol{\phi}^\circ$ consists in a matrix inversion. With simple algebraic simplifications, it can be proved that the dimension of the matrix to be inverted does not depend on the number of acquired data but only on the dimension of the vector $\boldsymbol{\phi}$. Similarly, once the model has been trained, the model prediction (the prediction of $\mathbf{x}$ given $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$) consists in evaluating $\hat{D}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ and the product $\hat{D}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\phi}^\circ$. Therefore, the computational complexity of the prediction depends mainly on the evaluation of the matrix $\hat{D}$ (in our example represented by ~ 1700 multiplications, ~ 700 sums and 8 sine/cosine evaluations).

3.2 Least Squares Support Vector Machines for Regression

Least Squares Support Vector Machines (LS-SVMs) belong to the class of kernel methods, which use a positive definite kernel function to estimate a linear approximator in a (usually) high-dimensional feature space [23]. Its formulation shares similarities with the Support Vector Machine for Regression (SVR) [22]. Let us define the data set $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$, where inputs $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding outputs $y_i \in \mathbb{R}$ for $i = 1, \ldots, \ell$. LS-SVM estimates a linear decision function of the form $f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$, where b is a bias term and $\phi(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^f$ maps samples from the input space into a (usually) high-dimensional feature space. The weight vector $\mathbf{w}$ and bias b are chosen such that both the squared norm of $\mathbf{w}$ and the sum of the squared errors $\epsilon_i = y_i - f(\mathbf{x}_i)$ are minimized. This is described by the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}C\sum_{i=1}^{\ell}\epsilon_i^2 \\ \text{subject to} \quad & y_i = \langle \mathbf{x}_i, \mathbf{w} \rangle + b + \epsilon_i \qquad \text{for } 1 \leq i \leq \ell \ , \end{aligned} \tag{13}$$

where C is a regularization constant. Standard application of the Lagrange method yields the dual optimization problem [23]:

$$\text{maximize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}C\sum_{i=1}^{\ell}\epsilon_i^2 - \sum_{i=1}^{\ell}\alpha_i\left(\langle \mathbf{x}_i, \mathbf{w} \rangle + b + \epsilon_i - y_i\right) \ . \tag{14}$$

Here $\alpha_i \in \mathbb{R}$ are the Lagrange multipliers associated with each sample. Using this dual formulation, the decision function can be rewritten as $f(\mathbf{x}) = \sum_{i=1}^{\ell}\alpha_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b$. One particular advantage of this expansion is that the solution is described in terms of inner products with respect to the training samples $\mathbf{x}_i$. Hence, a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ can be used to implicitly map the data into the feature space. Given a kernel matrix $K = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^{\ell}$, the solution to the optimization problem in (14) is given by a system of linear equations:

$$\begin{bmatrix} \boldsymbol{\alpha} \\ b \end{bmatrix} = \begin{bmatrix} K + C^{-1}I & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \ . \tag{15}$$

Note that solving the LS-SVM optimization problem reduces to a $(\ell+1)\times(\ell+1)$ matrix inversion, which in return can be solved efficiently using Cholesky decomposition [2]. Another advantage of LS-SVM over other kernel methods (e.g. SVR), is that the Leave-One-Out (LOO) error can be computed exactly using a single training run on the complete data set [2]. It is important to note that the final generalization performance of the LS-SVM is strongly dependent on the selection of both C and the kernel function. For our experiments, we consider the commonly used Radial Base Function

(RBF) kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, where parameter γ tunes the radius of the Gaussian. A grid search on the range $C \in \left[2^0, 2^1, \ldots, 2^{16}\right]$ and $\gamma \in \left[2^{-11}, 2^{-10}, \ldots, 2^1\right]$ is used to select "optimal" hyperparameters, where the generalization performance of each configuration is estimated using the LOO error on the training set. Furthermore, we considered the cases that $\mathbf{x} = \{\mathbf{q}, [\mathbf{q}, \dot{\mathbf{q}}], [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}], [\mathbf{q}, \ddot{\mathbf{q}}]\}$. As the output y is limited to scalar values, a distinct machine has to be trained for each output dimension, such that $y = \{f_x, f_y, f_z, \tau_x, \tau_y, \tau_z\}$.

Remark 3. Though LS-SVM has several advantageous properties with respect to SVM, one apparent disadvantage is that it does not produce sparse models. Input samples $\mathbf{x}_i$ can only be removed from the kernel expansion when $\alpha_i = 0$, which in return is only the case if $\epsilon_i = 0$. As a result, on practical problems all input samples will be included in the model. This reflects negatively on the prediction time. For m output dimensions and assuming the RBF kernel function, the prediction of an n dimensional input vector requires $m\,(\ell(n+1)+1)$ sums, $m\ell(n+2)$ products, and $m\ell$ exponentials; where ℓ is the size of the data set the m distinct machines has been trained on.

Remark 4. The scope of this work is limited to batch learning on small to medium sized data sets. Nguyen-Tuong et al. (cf. [18] in this volume) have shown that – on a similar learning problem – Gaussian Processes Regression (GPR) commonly outperforms other methods in this particular context. It is worth noting that the performance of LS-SVM can be expected to be similar to GPR, as both methods share some similarities in the approximation function[6].

3.3 Neural Networks

Lastly, a multiple input - multiple output one-hidden-layer (OHL) feed-forward neural network (NN) is chosen as the second machine learning method, for its generalization and approximation capabilities [9], and denoising property when dealing with experimental data. More specifically, we constrain the approximation function to take on a fixed, parameterized structure, that is a ν "neurons", n inputs, m outputs neural network, $\hat{\boldsymbol{\mu}}(\cdot, \mathbf{w})$, with sigmoidal activation functions σ in both hidden and output layer[7]:

$$\hat{\boldsymbol{\mu}}(\tilde{\mathbf{x}}, \mathbf{w}) = col\left(\tilde{\boldsymbol{\sigma}}_j\left[\sum_{h=1}^{\nu} c_{hj}\sigma(\tilde{\mathbf{x}}, \boldsymbol{\kappa}_h) + b_j\right], j = 1, \ldots, m\right) \tag{16}$$

[6] The main differences between both methods are that LS-SVM includes a bias term and requires less assumptions on the distribution of the data.

[7] We chose a sigmoidal output layer (instead of a classical linear output layer) since it naturally generates bounded values within a specific range, which are consistent with the output ranges after data normalization. This choice allows to remove signal constraints and not to take care of the possibility that the network generates inconsistent values.

where $\hat{\boldsymbol{\mu}}(\cdot, \mathbf{w}) : \mathbb{R}^n \times \mathbb{R}^W \mapsto \mathbb{R}^m$, $c_{hj}, b_j \in \mathbb{R}, \boldsymbol{\kappa}_h \in \mathbb{R}^{n+1}, j = 1, \ldots, m$, being ν the number of *neurons* constituting the network. The vector $\mathbf{w} \in \mathbb{R}^W, W = (n+1)\nu + (\nu+1)2m$ collects all the parameters to be optimized. The notations $\tilde{\boldsymbol{\sigma}}$ and $\tilde{\mathbf{x}}$ account for the output and input normalization.[8] Furthermore, we trained four different type of networks, with $\mathbf{x} = \{\mathbf{q}, [\mathbf{q}, \dot{\mathbf{q}}], [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}], [\mathbf{q}, \ddot{\mathbf{q}}]\}$ and $n = \{4, 8, 12\}$ respectively, for different number of neurons $\nu = 5, 20, 50, 100, 150$ and different training sets. The number of outputs was always fixed to 6 (forces and torques). The training algorithm is based on the well known Levenberg-Marquardt (LM) algorithm [12, 13]. The criterion for training the network (that is to find the optimal parameters $\mathbf{w}^\circ$) is to minimize the mean square error between the estimated and the measured data:

$$\text{minimize} \;\; \Phi(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{\ell} \boldsymbol{\epsilon}_i^\top(\mathbf{w}) \boldsymbol{\epsilon}_i(\mathbf{w}) \;\; , \tag{17}$$

where $\boldsymbol{\epsilon}_i(\mathbf{w}) = \mathbf{y}_i - \hat{\boldsymbol{\mu}}(\tilde{\mathbf{x}}_i, \mathbf{w})$ is the error between the measured and the predicted data. Once all the partial derivatives of the error function Φ are back-propagated, the weights update equation is applied:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - [J^\top(\mathbf{w}_k) J(\mathbf{w}_k) + \mu I]^{-1} J^\top(\mathbf{w}_k) \boldsymbol{\epsilon}(\mathbf{w}_k) \;\; , \tag{18}$$

where $\boldsymbol{\epsilon}(\mathbf{w}_k) = [\boldsymbol{\epsilon}_0(\mathbf{w}_k), \ldots, \boldsymbol{\epsilon}_{N-1}(\mathbf{w}_k)]$, and $J(\mathbf{w}_k) \in \mathbb{R}^{N \times W}$ is the Jacobian matrix of the errors with respect to the parameters of the NN:

$$J(\mathbf{w}) = \begin{bmatrix} \frac{\partial \boldsymbol{\epsilon}_0}{\partial w_0} & \frac{\partial \boldsymbol{\epsilon}_0}{\partial w_1} & \cdots & \frac{\partial \boldsymbol{\epsilon}_0}{\partial w_{W-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{\epsilon}_{N-1}}{\partial w_0} & \frac{\partial \boldsymbol{\epsilon}_{N-1}}{\partial w_1} & \cdots & \frac{\partial \boldsymbol{\epsilon}_{N-1}}{\partial w_{W-1}} \end{bmatrix} . \tag{19}$$

The parameter μ, adjusted iteratively, balances the LM between a steepest descent and a Gauss-Newton algorithm. To improve the training performance we used the Nguyen-Widrow (NW) method [17] to initialize the network (see also [7, 14]).

Remark 5. The proposed neural network training is designed for batch learning, and generically the estimate improves with the growth of both training set and number of parameters[9]. Since the training is performed offline, in the prediction phase the computation is quite fast, consisting only of a single

[8] The input variables are normalized from their original range to $[-1, 1]$, while the network outputs are scaled from $[-1, 1]$ (the output range of a sigmoidal tanh-based neural network) to the forces and torques real ranges.

[9] The number of parameters usually depend on three factors: the complexity of the function to be approximated (i.e. a very smooth function requires fewer neurons than a highly varying one, as more basis function are necessary to approximate the irregular changes of the latter), the dimension of the training set and the quality of the training set (i.e. to which extent the training set is representative for the variable space).

Table 1 Value ranges of the arm joint positions, velocities and accelerations.

	$q[^\circ]$				$\dot{q}[^\circ/s]$				$\ddot{q}[^\circ/s^2]$			
joint #	0	1	2	3	0	1	2	3	0	1	2	3
max	150	60	30	70	18	15	20	15	103	76	248	34
min	50	-100	-60	10	-39	-49	-34	-23	-135	-85	-158	-38

forward pass of the network. More precisely, given the number of neurons ν for a OHL neural network with n inputs, m outputs, sigmoidal activation functions (hyperbolic tangent $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$) in the hidden and output layer, the necessary operations are $\nu(n+m)$ products, $\nu(n+m+1)+m+2(\nu+m)$ sums, $2(\nu+m)$ exponentials and $\nu+m$ divisions, and the flops count is linear with ν. As an example, for 20 neurons, 4 inputs, 6 outputs and both layers with the usual hyperbolic tangent, the flops count (computed with the Lightspeed Matlab toolbox v.2.2. [16]) is 2766.

4 Results and Discussion

The three previously discussed methods have been evaluated experimentally on a common data set that has been gathered during a sequence of random arm movements, performed in joint space. Every movement brings the arm from a starting position $\mathbf{q}_\mathrm{s} \in \mathbb{R}^4$ to a final position $\mathbf{q}_\mathrm{f} \in \mathbb{R}^4$, which subsequently becomes the starting position for the next movement. Each of these positions is defined by a vector of joint angles, which are chosen randomly using a uniform distribution within the admissible range of the respective joint. Joint velocity profiles during motion are *bell-shaped* with a predefined maximum velocity, which causes the absolute velocities to vary from zero to the maximum value during any movement. Trivially, the sign of the velocity depends on the direction of the motion. The joint accelerations (i.e. actual slope of the velocity profiles) depend on the distance between $\mathbf{q}_\mathrm{s}$ and $\mathbf{q}_\mathrm{f}$, since the time duration of the movement is kept constant. Joint positions, velocities and accelerations were retrieved from the DSP boards at 50 Hz. Velocities and accelerations were computed via numerical differentiation on the DSP boards at a higher frequency (1 kHz). A simple collision avoidance strategy was used during the experimental data acquisition, in order to ensure that the arm would not collide with the body or the environment.

The complete data set of 40000 samples has been shuffled and split in two equal parts. The set of the first 20000 samples is used for training and is subsequently subsampled to obtain smaller sized training sets, whereas the second half is used as a common test set. The reported performance measure on the test set is the average Normalized Mean Squared Error (NMSE) over all 6 output dimensions, where the NMSE is defined as the mean squared error divided by the variance of the respective output dimension. For the two machine learning approaches, the input dimensions have been rescaled (see original ranges Table 1) to be approximately within the range $[-1, +1]$, based

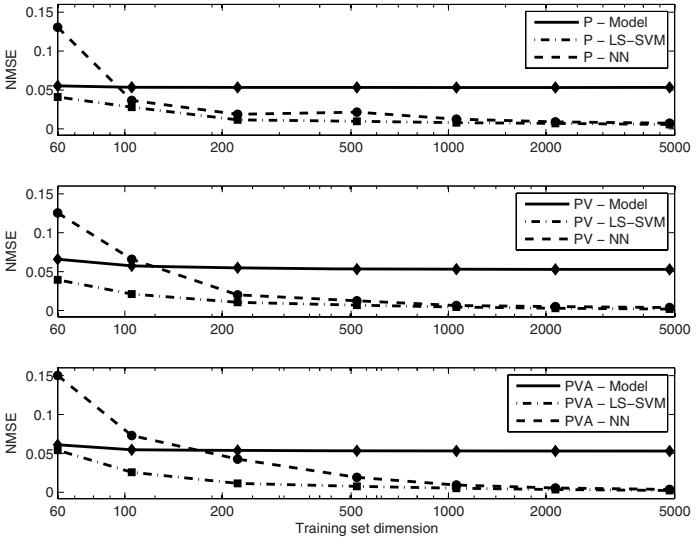


Fig. 3 Comparison of the three methods on random training subsets of increasing dimension and three different input spaces. P denotes the input space containing only joint positions ($\mathbf{q} \in \mathbb{R}^4$), PV contains both joint positions and velocities ($\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^8$), and PVA contains joint positions, velocities and accelerations ($\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^{12}$).

on the maximum and minimum values found in each input dimension in the training set.

4.1 Number of Training Samples

In this initial experiment we measured the performance of each method when increasing the number of training samples. The results in Fig. 3 show clearly that the two learning methods have a strong dependency on the size of the training set. As more samples become available, they consistently continue to improve performance, eventually outperforming the model-based approach by an order of magnitude.

Interestingly, the model-based approach appears to perform at a constant level, regardless of the number of samples. This is confirmed by further analysis on even smaller data sets, as demonstrated in Fig. 4. When considering only the joint positions, it shows the remarkable capability of achieving acceptable performance using only 5 training samples. This means that the model-based approach is the preferred approach when only very few samples are available. The machine learning methods require many more samples to achieve similar performance. This is not surprising considering that the

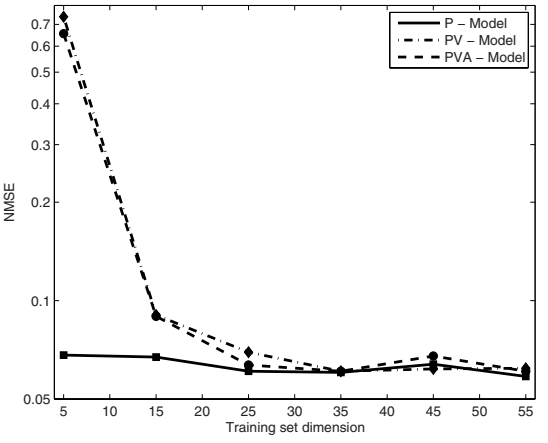


Fig. 4 Performance of the model-based method on very small training sets.

model based approach takes advantage of additional information implicit in the structure of the model.

4.2 Contribution of Velocity and Acceleration on the Estimation

Another observation (Fig. 5) is that inclusion of joint velocities and accelerations does not always improve the generalization performance when training is done on a small number of samples. Intuitively, one might expect that adding relevant information could only improve the estimation. However, learning methods require an increasing amount of training samples to make effective use of this additional information (i.e. the *curse of dimensionality* [4]). This affects particularly the learning methods, since these need to construct their model based solely on training data. Fig. 5 shows that both LS-SVM and NN eventually use joint velocities to improve their predictions, given a sufficiently large training set.

Joint accelerations, however, do not improve prediction performance in any of the cases (cf. Fig. 6). This is probably due to the low signal to noise ratio for the acceleration and, in first place, to the robotic setup used to obtain the data set. In particular, the joint accelerations were not measured directly but were derived from positions. This causes the acceleration measurements to be much less precise and reliable than those for joint velocities and positions. Furthermore, the range of accelerations is relatively small[10] and within this range we observed that the contribution of $M(\mathbf{q})\ddot{\mathbf{q}}$ in equation (2) is relatively small compared to the contribution of the other terms ($C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ and $G(\mathbf{q})$).

[10] The chosen motors produce limited torques, which reflects into relatively low accelerations.

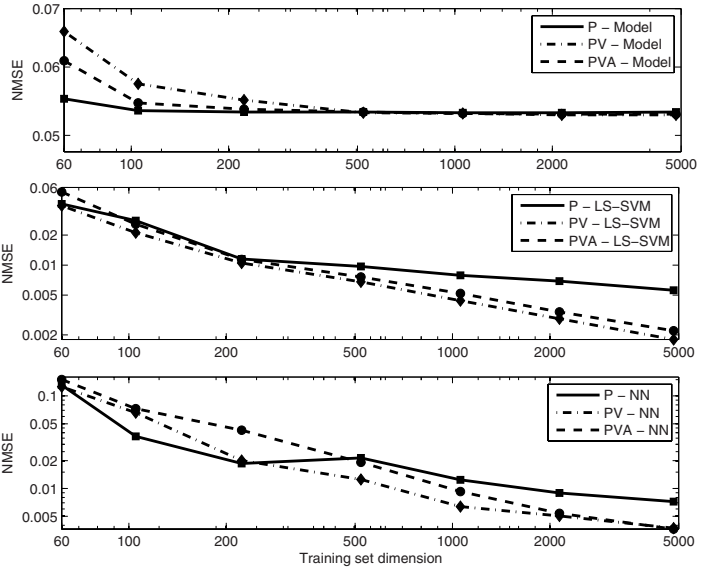


Fig. 5 Comparison of the performance for all methods with increasing input spaces (i.e. P, PV and PVA; defined as in Fig. 3). Note that both axes are in logarithmic scale to accentuate differences in final performance.

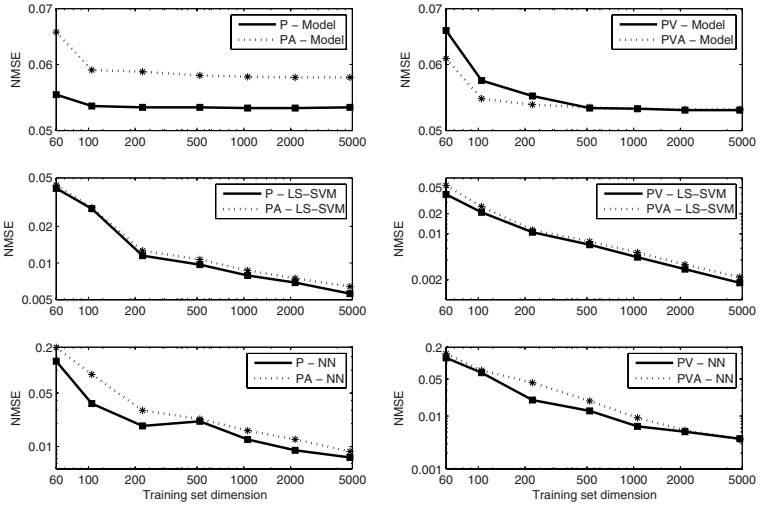


Fig. 6 Performance after inclusion of joint accelerations. The figures on the left hand side compare the performance on P and PA (defined analogously to P, PV and PVA in Fig. 3), while those the right hand side compare PV and PVA.

4.3 Selective Subsampling

The data set we acquired is characterized by the fact that there is an abundance of training samples. Thus far, we have used a uniform random subsampling strategy, as to ensure that the qualitative properties of the subset approximate those of the original data set. However, with such an abundance of samples it is likely that the original data set is *oversampled* (i.e. additional samples do not further increase the generalization performance) and contains samples that are (nearly) identical to each other. This similarity of input samples particularly affects LS-SVM, as this method describes the prediction function in terms of inner products with respect to all training samples.

We can guarantee a certain "sparsity" of the training set by taking a subset, such that the inter-sample distance is at least a threshold t. Let us define a distance measure $D(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)\Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)}$, where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^n$ and Σ is an $n \times n$ matrix containing the variances of all input dimensions on its diagonal. This measure coincides with the Euclidean distance in the standardized input space. In order to construct a Euclidean subset $\mathcal{E}_t$, we iterate over a permutation of the original data set $\mathcal{S}$ using index $i = 1, \ldots, \ell$ and append only those samples to $\mathcal{E}_t$, for which $\min D(\mathbf{x}_i, \mathbf{x}) \geq t \ \forall \mathbf{x} \in \mathcal{E}_t$.

Fig. 7 shows the prediction performance of LS-SVM with random and Euclidean subsampling. The Euclidean subsets were generated by varying the threshold t, such that the size of the subsets were nearly identical to each of the random subsets. Whether selective subsampling based on Euclidean distance outperforms random subsampling depends on the input space that is used to determine the inter-sample distance, and the size of the training set. When this distance is determined solely based on the joint positions, then Euclidean subsampling results in a significant improvement for small data sets. In contrast, random subsampling performs better than Euclidean subsampling based on joint positions, velocities and accelerations. It is our belief that this difference is due to the Euclidean strategy attempting to create a uniform sampling distribution in all dimensions under consideration, effectively forming subsets that contain a wide range of velocities and accelerations (besides positions). Given the relatively low velocities and accelerations of the robot, the force and torques are primarily caused by gravity. In return, gravity is only dependent on the joint positions of the robot. It is therefore beneficial, for limited training sets, to select those samples that help LS-SVM to model this gravity component.

Further, we can note that the different subsampling strategies perform nearly identically for large training sets. This can easily explained by the fact that the size of $\mathcal{E}_t$ is inversely proportional to the chosen distance threshold t and, by definition, $\mathcal{E}_t$ becomes a random permutation of $\mathcal{S}$ as t approaches zero. In short, for large data sets, and thus a small inter-sample threshold, the Euclidean and random subsets have very similar sample distributions and therefore similar performance.

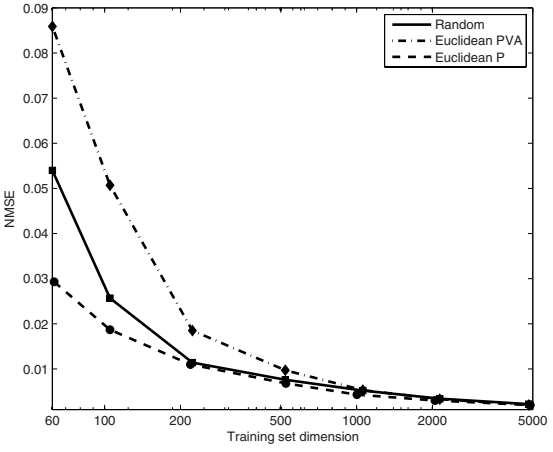


Fig. 7 Comparison of random and selective subsampling based on standardized Euclidean distance. *Euclidean P* and *Euclidean PVA* denote subsampling based on Euclidean distance thresholds $t = \{1.35, 1.15, 0.88, 0.65, 0.5, 0.35, 0.18\}$ using position inputs and thresholds $t = \{6.0, 5.3, 4.5, 3.7, 3.1, 2.5, 1.8\}$ using position-velocity-acceleration inputs, respectively.

5 Conclusions

In this paper, we presented and compared different approaches to force/torque estimation from a FT sensor. Experimental data-driven learning methods are proposed and compared with respect to the classical model-based technique, and advantages and disadvantages of both types of approaches are discussed. Learning algorithms outperform the rigid body dynamic model in terms of prediction accuracy, given that a sufficient amount of training data is available. The generalization performance of these methods improves steadily as more training samples become available. LS-SVM converges slightly faster than Neural Networks, but their final performance on large data sets is nearly identical. The model-based method, on the other hand, requires very few samples (in order of ten) to achieve acceptable predictions, and in practice requires very few samples if the estimate relies only on joint positions. Furthermore, we tested the relevance of velocity and acceleration information when learning the dynamic equation which describes the FT measurement. It was observed that machine learning methods improve their prediction when including velocity data at the cost of requiring larger training sets. On the contrary, acceleration does not significantly improve performance and the reason for this incongruence has to be found in the low signal to noise ratio (positions are double differentiated to obtain accelerations). We also evaluated the impact of training set pre-processing by defining a suitable selective subsampling strategy: a Euclidean distance metric was introduced and the

resulting training set was compared with a random sampling one. For LS-SVM, the resulting spatial distribution of the training samples improves the estimation for small data sets, while its beneficial effect disappear with increasing the size of the training set.

Acknowledgment

This work was supported by European Commission projects RobotCub (IST-004370), ITALK (ICT-214668) and CHRIS (ICT-215805).

References

1. Haddadin, S., De Luca, A., Albu-Schaffer, A., Hirzinger, G.: Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1623–1630 (2006)
2. Cawley, G.C.: Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In: IJCNN 2006: Proceedings of the International Joint Conference on Neural Networks, Vancouver, BC, Canada, July 2006, pp. 1661–1668 (2006)
3. Denavit, J., Hartenberg, R.S.: A kinematic notation for lower-pair mechanisms based on matrices. Journal of Applied Mechanics 23, 215–221 (1955)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience, Hoboken (2001)
5. Metta, G., Cannata, G., Maggiali, M., Sandini: An embedded artificial skin for humanoid robots. In: Proc. of IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems, pp. 434–438 (2008)
6. Dubowsky, S., Liu, G., Iagnemma, K., Morel, G.: A base force/torque sensor approach to robot manipulator inertial parameter estimation. In: Proceedings of the 1998 IEEE International Conference on Robotics and Automation, 1998. ICRA 1998 (1998)
7. Hagan, M.T., Menhaj, M.B.: Training feedforward networks with the marquardt algorithm. IEEE Transactions on Neural Networks 5, 989–993 (1994)
8. Jamone, L., Nori, F., Metta, G., Sandini, G.: James: A humanoid robot acting over an unstructured world. In: International Conference on Humanoid Robots, Genova, Italy (2006)
9. Stinchombe, M., Hornik, K., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2, 359–366 (1989)
10. Kozlowski, K.: Modelling and Identification in Robotics. Springer, Secaucus (1998)
11. Lagarde, M., Andry, P., Gaussier, P., Boucenna, S., Hafemeister, L.: Proprioception and imitation: on the road to agent individuation. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 43–63. Springer, Heidelberg (2010)
12. Levenberg, K.: A method for the solution of certain problems in least squares. Quarterly of Applied Mathematics 2, 164–168 (1944)
13. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. SIAM Journal on Applied Mathematics 11, 431–441 (1963)

14. Ivaldi, S., Fumagalli, M., Jamone, L., Nori, F., Natale, L., Metta, G., Baglietto, M.: Estimation of forces and torques in a humanoid arm: comparison of model based and offline/online learning techniques. Submitted to the 48th IEEE Conference on Decision and Control (2009)
15. ATI mini45. 6 axes f/t sensor, http://www.ati-ia.com/products/ft/ft_models.aspx?id=Mini45
16. Minka, T.: Lightspeed toolbox, http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/
17. Nguyen, D., Widrow, B.: Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In: Proc. of the Int. Joint Conference on Neural Networks, June 1990, vol. 3, pp. 21–26 (1990)
18. Nguyen-Tuong, D., Seeger, M., Peters, J.: Real-time local gp model learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 193–207. Springer, Heidelberg (2010)
19. Chung, J.H., Lu, S., Velinsky, S.A.: Human-robot collision detection and identification based on wrist and base force/torque sensors. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005, April 2005, pp. 3796–3801 (2005)
20. Sciavicco, L., Siciliano, B.: Modeling and control of robot manipulators. MacGraw-Hill, New York (1996)
21. Shinya, M., Kazuhiro, K.: Collision detection system for manipulator based on adaptive impedance control law. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1080–1085 (2003)
22. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing 14(3), 199–222 (2004)
23. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific Publishing Co. Pte. Ltd., Singapore (2002)
24. Swevers, J., Ganseman, C., Tukel, D.B., De Schutter, J., Van Brussel, H.: Optimal robot excitation and identification. IEEE Trans. on Robotics and Automation 3(5), 730–740 (1997)
25. Ting, J., Mistry, M., Peters, J., Schaal, S., Nakanishi, J.: A bayesian approach to nonlinear parameter identification for rigid body dynamics. In: Robotics: Science and Systems, RSS (2006)

Learning Forward Models for the Operational Space Control of Redundant Robots

Camille Salaün, Vincent Padois, and Olivier Sigaud

Abstract. We present an adaptive control approach combining model learning methods with the operational space control approach. We learn the forward kinematics model of a robot and use standard algebraic methods to extract pseudo-inverses and projectors from it. This combination endows the robot with the ability to realize hierarchically organised learned tasks in parallel, using tasks null space projectors built upon the learned models. We illustrate the proposed method on a simulated 3 degrees of freedom planar robot. This system is used as a benchmark to compare our method to an alternative approach based on learning an inverse of the extended Jacobian. We show the better versatility of the retained approach with respect to the latter.

Keywords: learning, redundancy, robotics, inverse velocity kinematics.

1 Introduction

Real-world Robotics applications are evolving from the industrial domain (well-defined tasks in structured environment) to the service domain where it is much harder to model all the aspects of the mission. Service Robotics induces complexity both in terms of the tasks that have to be achieved and in terms of the nature of the environment where robots are supposed to evolve. Part of the answer to the problems raised by this growing complexity lies in the increasing number of sensors with which robots are now equipped as well as in the increasing number of degrees of freedom of the robots themselves (e.g., Mobile manipulators such as the humanoid robot iCub (Metta et al, 2008) or the wheeled assistant PR2 (Willow Garage, 2009)).

Camille Salaün, Vincent Padois, and Olivier Sigaud
Institut des Systèmes Intelligents et de Robotique - CNRS UMR 7222
Université Pierre et Marie Curie, Pyramide Tour 55 - Boîte Courrier 173,
4 Place Jussieu, 75252 Paris CEDEX 5, France
e-mail: firstname.name@upmc.fr

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 169–192.
springerlink.com

As a matter of fact, the motion controllers developed for such robots have to be either highly robust to uncertainties in the model of the robot and its environment or adaptive, i.e., able to build their own model on-line. The former gets more and more difficult as the complexity of the context grows. When the structure of the model is known, the latter can be acheived with classical parametric identification methods (Ljung, 1986). When this structure is more difficult to obtain, due to different physical phenomena such as friction, internal delays, unmodeled nonlinearities, etc., machine learning methods can be more versatile: they learn a model based only on the inputs and outputs of the real system without making assumptions on their physical relationship.

In this context, learning the model of the robot is achieved using specific representations such as Neural Networks (Van der Smagt, 1994) or Radial Basis Function Networks (Sun and Scassellati, 2004), Gaussian Processes (Shon et al, 2005) or (Nguyen-Tuong et al, 2010) in this issue, Gaussian Mixture Models (Calinon et al, 2007), Locally Weighted Projection Regression (LWPR) (D'Souza et al, 2001; Natale et al, 2007; Peters and Schaal, 2008), but the control methods used in the corresponding work do not always take advantage of the state-of-the-art techniques developed in recent Robotics research.

Among these techniques, operational space control Khatib (1983) is a model-based approach which provides a mathematical framework giving rise to an easy definition of the tasks and constraints characterising a robotic mission in a hierarchical manner (Liégeois (1977), Nakamura (1991). Readers can refer to Sentis and Khatib (2005) for a more recent work and Nakanishi et al (2008) for a survey. In order to take advantage of this framework, one must develop learning methods and associated representations which fit the needs of the corresponding control techniques.

Actuators of a robot generally act on joints, but the tasks or constraints associated to a mission cannot often be described in the joint space in a natural way. The operational space (also called task space) provides an alternative, more natural space, for such a definition.

The robot being controlled at the level of joints, the operational or task space control approach requires the knowledge of the mapping between the joint space and the task space. More specifically, it is the inverse mapping which is often of interest: given a task, what are the actions required in the joint space to achieve it. Considering minimum representations for the joint and task spaces, it is important to notice that when the dimension of the joint space is larger than the one of the task space, there is an infinite number of inverse mappings and the robot is said redundant with respect to the task. That is the case we are focusing on in this chapter.

More precisely, we examine how one can combine learning techniques and operational space control in such a way that we can hierarchically deal with several tasks and constraints when the robot is redundant with respect to the task. Our method learns a forward kinematics model using LWPR, a state-of-the-art

method already used in the context of learning robot models (Vijayakumar et al, 2005). We show how we can both carefully derive the forward and inverse mappings at the velocity level and the projectors which are necessary to combine several tasks. We compare this approach to an alternative approach presented in the literature where the inversion problem that arises in the redundant case is solved in less generic manner (D'Souza et al, 2001).

The chapter is organised as follows. In section 2, we give some background on operational space control and the different levels of forward and inverse mappings which can be used to relate the joint space to the task space and vice versa. We also present different contexts in which several tasks can be combined depending on their compatibility. In section 3, we give an overview of the learning methods that have been applied to learn these forward and inverse mappings, before focusing on our approach. In section 4, we introduce our experimental apparatus and protocol, as well as the series of simulated experiments that we perform. The corresponding results are presented in section 5. Finally, section 6 highlights the properties of our approach before concluding on the potential extensions that are unique to the perspectives raised by our work.

2 Background in Operational Space Control

In this section, we give some background information on joint to task space mappings with a focus on the velocity level. We recall the general expression of minimum norm solutions in the redundant case and give an overview of redundancy resolution schemes.

2.1 Joint Space to Task Space Mappings

The joint space is the space of the configuration parameters $\mathbf{q}$ of size n, where n is the number of parameters chosen to describe the robot configuration. In the holonomic, fully actuated, minimum representation case, n is also the number of degrees of freedom of the robot as well as the dimension of the actuation torque vector $\mathbf{\Gamma}$.

As stated in the introduction, the tasks or constraints associated to a mission can rarely be described in the joint space in a natural way. The task space is often associated to the end-effector(s) of the robot but can actually be any point of the robot and more generally any set of parameters of interest which can be described as a function of the robot configuration. This is the case for external collision avoidance where the constraint point can evolve along the robot body. Joint limits avoidance is also a particular case of constraint where the task space is a subset of the joint space itself. Independently from their physical meanings, task spaces can be described by task space parameters $\boldsymbol{\xi}$ of size m where m is, in the case of a minimum representation, the number of degrees of freedom required to achieve the task.

The joint space to task space mapping can be described at three different levels. At the geometric level, the forward kinematics model can be described as a non-linear function $\mathbf{f}$ such as:

$$\boldsymbol{\xi} = \mathbf{f}(\mathbf{q}). \tag{1}$$

As stated before, if the robot is redundant, there is an infinite number of possible inverses for $\mathbf{f}$. However, there is no simple method to span the set of possible solutions at the geometric level and the mapping is often described at the velocity level by the Jacobian matrix $J(\mathbf{q}) = \partial \mathbf{f}(\mathbf{q}) / \partial \mathbf{q}$ such that:

$$\dot{\boldsymbol{\xi}} = J(\mathbf{q})\,\dot{\mathbf{q}}. \tag{2}$$

$J(\mathbf{q})$ is a $m \times n$ matrix and thus can be inverted using linear algebra techniques. Once again, there is an infinity of inverse mappings corresponding to the infinity of possible generalised inverses of $J(\mathbf{q})$ (Ben Israel and Greville, 2003).

The last mapping of interest is the dynamic one. It relates forces applied to the system, among which the control input $\boldsymbol{\Gamma}$, to the resulting acceleration $\ddot{\mathbf{q}}$. It can be written:

$$\boldsymbol{\Gamma} = A(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) + \boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}}) - \boldsymbol{\Gamma}_{ext}, \tag{3}$$

where $A(\mathbf{q})$, $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$, $\mathbf{g}(\mathbf{q})$, $\boldsymbol{\epsilon}(\mathbf{q}, \dot{\mathbf{q}})$ and $\boldsymbol{\Gamma}_{ext}$ are respectively the $n \times n$ inertia matrix of the system, the vector of Coriolis and centrifugal effects, the vector of gravity effects, the vector of unmodeled effects and the torque resulting from external forces applied to the system.

This equation represents a joint space to joint space mapping at the dynamics level with only one solution. It is of course of interest to learn this mapping since it captures a lot of properties of the system among which effects such as friction which cannot always be easily identified using parametric identification techniques. However, the learning of this mapping is out of the scope of the work presented here (interested readers can refer to Peters and Schaal (2008) and Nguyen-Tuong et al (2008)) and it is supposed to be known in the experiments presented here. We rather focus on the velocity kinematics mapping which is sufficient to capture and characterise the redundancy of the system[1].

2.2 Model-Based Control at the Velocity Level

In the redundant and non singular case, i.e., $\text{rank}(J(\mathbf{q})) = m$ and $m < n$, there is an infinite number of generalised inverses of $J(\mathbf{q})$. Among these

[1] A velocity kinematics and dynamic combined mapping known as the Operational Space Formulation is proposed by Khatib (1987). It is of interest if one wants to obtain the dynamic model of the system expressed in the task space.

inverses, weighted pseudoinverses provide minimum norm solutions (Doty et al, 1993) and can be written as

$$J(\mathbf{q})^{\sharp} = W_q^{-1} J(\mathbf{q})^T \left[J(\mathbf{q}) W_q^{-1} J(\mathbf{q})^T \right]^{-1}, \tag{4}$$

where W_q is a symmetric and positive definite matrix of dimension $n \times n$.

Given a desired task velocity $\dot{\boldsymbol{\xi}}^{\star}$, the inverse mapping of Equation (2) which minimises the Euclidean W_q-weighted norm[2] of the solution is given by

$$\dot{\mathbf{q}} = J(\mathbf{q})^{\sharp} \dot{\boldsymbol{\xi}}^{\star}. \tag{5}$$

The Moore-Penrose inverse or pseudoinverse $J(\mathbf{q})^{+}$ of $J(\mathbf{q})$ corresponds to the case where $W_q = I_n$.

The system being redundant with respect to the task, Equation (5) is not the unique solution to the inverse mapping problem and other solutions of interest are those giving rise to internal motions that do not induce any perturbation on the task. This particular subset of solutions corresponds to the nullspace of $J(\mathbf{q})$ and the general form of the minimum norm solutions to Equation (2) can be written

$$\dot{\mathbf{q}} = J(\mathbf{q})^{\sharp} \dot{\boldsymbol{\xi}}^{\star} + P_J(\mathbf{q}) \dot{\mathbf{q}}_0, \tag{6}$$

where $P_J(\mathbf{q})$ is a projector on the nullspace of $J(\mathbf{q})$ and $\dot{\mathbf{q}}_0$ is any vector of dimension n. Equation (6) is the minimum norm solution that minimises $||\dot{\mathbf{q}} - \dot{\mathbf{q}}_0||_{W_q}$. A commonly used expression for $P_J(\mathbf{q})$ is

$$P_J(\mathbf{q}) = \left(I_n - J(\mathbf{q})^{\sharp} J(\mathbf{q}) \right). \tag{7}$$

Efficient computation of $J(\mathbf{q})^{\sharp}$ and $P_J(\mathbf{q})$ can be done using the SVD (Golub and Van Loan, 1996) of $J(\mathbf{q})$. The SVD of $J(\mathbf{q})$ is given by $J = UDV^T$ where U and V are orthogonal matrices with dimensions $m \times m$ and $n \times n$ respectively. D is a $m \times n$ diagonal matrix with a diagonal composed of the m singular values of J in decreasing order. Given this decomposition, the pseudoinverse of J can be computed as follows

$$J^{+} = V D^{+} U^T, \tag{8}$$

where the computation of D^{+} is straightforward given its diagonal nature. Regarding $P_J(\mathbf{q})$, it can be computed using the $m+1$ to n columns of V which form a basis for the nullspace of $J(\mathbf{q})$

$$P_J(\mathbf{q}) = \left[\mathbf{v}_{m+1} \dots \mathbf{v}_n \right] \left[\mathbf{v}_{m+1}^T \dots \mathbf{v}_n^T \right]^T, \tag{9}$$

where $\mathbf{v}_i$ is the i^{th} column of V.

[2] $\sqrt{\dot{\mathbf{q}}^T W_q \dot{\mathbf{q}}}$, also noted $||\dot{\mathbf{q}}||_{W_q}$.

A weighted extension of the SVD can be used in the case where $W_q \neq I_n$. Details about this extension can be found in Ben Israel and Greville (2003).

2.3 Redundancy Resolution Schemes

Different possible redundancy resolution schemes can be applied, depending on the compatibility of the tasks or constraints which have to be solved.

Let us consider two tasks of respective dimensions m_1 and m_2 and with associated Jacobian matrices J_1 and J_2 such as $rank\left(J_1\left(\mathbf{q}\right)\right) = m_1$ and $m_1 \leq n$ and $rank\left(J_2\left(\mathbf{q}\right)\right) = m_2$ and $m_2 \leq n$. These two tasks are said to be compatible if $J_{ext} = \left[J_1^T \; J_2^T\right]^T$ is full row rank. This condition is equivalent to saying that the m_{ext} parameters of the augmented task space are in minimum number and that $rank\left(J_{ext}\right) \leq n$.

Given this definition, one has to consider the underconstrained (compatible, infinity of solutions), fully-constrained (compatible, one solution) and over-constrained (incompatible, no exact solution) cases. In these three cases, one can write the solution to the inverse velocity kinematics problem using the solution proposed initially by Maciejewski and Klein (1985)

$$\dot{\mathbf{q}} = J_1^{\sharp}\dot{\boldsymbol{\xi}}_1^{\star} + \left(J_2 P_{J_1}\right)^{\sharp}\left(\dot{\boldsymbol{\xi}}_2^{\star} - J_2 J_1^{\sharp}\dot{\boldsymbol{\xi}}_1^{\star}\right). \tag{10}$$

In the compatible case, tasks 1 and 2 will be achieved perfectly. In the incompatible case task 1 will also be perfectly achieved whereas the error on the achievement of task 2 will be minimised. This solution can present singularities when tasks are highly incompatible, i.e., m_{ext} is much greater than n, but this can be compensated for using a proper damped-least square regularisation (Chiaverini, 1997). This task projection scheme can be extended to several tasks, interested readers can refer to Mansard and Chaumette (2007).

Another method, originally proposed in Baillieul (1985), consists in writing an extended Jacobian J_{ext} in order to reach the fully constrained case ($m_{ext} = n$ and $rank(J_{ext}) = m_{ext}$) and thus to simplify the inversion problem to a square, regular matrix inversion. In the fully constrained case, this is achieved automatically. However, in the under constrained case, this requires to artificially add tasks whereas in the over constrained one, some projections have to be done in order to ensure both a square Jacobian matrix and priorities between tasks.

Similarly to what is shown in the non learning case literature, we will show hereafter that in the case of complex missions where the tasks and constraints constantly evolve, one cannot ensure compatibility at all time. Thus, the solution provided by Equation (10) is more general and should be preferred.

Finally, in the case of constraints such as joints limits, a possibility consists in choosing $\dot{\mathbf{q}}_0$ in Equation (6) as the opposite of the gradient of a cost function $Q\left(\mathbf{q}\right)$. The resulting solution leads to the local maximisation of

the cost function as long as this secondary constraint does not induce any perturbation on the first task. The general form of this solution is written

$$\dot{\mathbf{q}} = J(\mathbf{q})^{\sharp}\, \dot{\boldsymbol{\xi}}^{\star} - \alpha P_J \nabla Q(\mathbf{q}), \tag{11}$$

where α is a positive scalar used to tune the steepness of the gradient descent Snyman (2005). This method is often used in the incompatible case, i.e., when it is known in advance that the task will not be perfectly achieved, or when only a global trend has to be followed: minimise the kinetic energy of the system, avoid joint limits or collisions, etc.

3 Learning Forward and Inverse Velocity Kinematics Models

Machine learning researchers have developed several families of methods capable of approximating linear and non-linear functions: perceptrons, multilayer perceptrons, radial basis function networks, gaussian processes, support-vector regression, gaussian process regression, learning classifiers systems and locally weighted regression methods such as LWPR Vijayakumar and Schaal (2000). In this section, we briefly explain basic concepts of some of these methods focusing on LWPR which we use and then mention how they are used to learn forward or inverse kinematics model. Then we expose the advantages of learning forward (instead of inverse) velocity kinematics mappings in redundant cases.

3.1 An Overview of Neural Networks Function Approximation

Neural networks Rojas (1996) are a wide class of general function approximators. They are declined under different forms. The general neural network algorithm may be split into three steps. First, compute an excitation level for each neuron depending on the inputs. Second, apply an activation function on this excitation level to determine if the neuron is active or not. Third, compute an error of the global network output to update all weighted connections. A commonly used feedforward neural network is the Multi-Layer Perceptron (MLP) where the activation level of each neuron i is calculated as $z_i(\mathbf{x}) = \sum_{i=1}^{N} w_i x_i$ where N is the number of neurons, x_i is an input of the neuron and w_i is the associated weight. Classical activation functions are $y(z_i) = \tanh(z_i)$ or $y(z_i) = (1 + e^{-z_i})^{-1}$. The weights are updated through a backpropagation error method based on the error e_j between the desired value and the real output of each neuron. This method consists in computing an energy function, such as $\mathbf{E}(k) = 1/2 \sum_j e_j^2(k)$, which is minimised to update each weighted connection with a gradient descent: $\delta w_l(k) = -\gamma \partial \mathbf{E}(k)/\partial w_l(k)$. γ is the learning rate which updates the weights

$w_{1..l}$ of a layer in the opposite direction with respect to the energy gradient. For a general review on learning non-linear models, readers can refer to Jordan and Rumelhart (1992).

Radial basis function networks are another type of neural networks where weighted sum and activation functions are temporally inverted compared to multi-layer perceptrons. The algorithm consists in calculating the function activation $\phi(r) = \exp(-\beta r^2), \beta > 0$ of the norm $r = \|\mathbf{x} - \mathbf{c}_i\|$ which is then weighted and summed: $y(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi(r)$.

A fundamental problem with those approaches is that neural networks are always considered as black boxes and tuning is usually empirical. Radial basis function networks tend to avoid this limitation and could be treated as grey boxes. They have the advantage of generating differentiable continuous approximations (Sun and Scassellati, 2004).

An extension of radial basis function networks are the locally weighted regression methods (Atkeson, 1991) which combine gaussian and linear models. Some of those methods, such as LWPR, make a projection on the relevant subspace to decrease the dimension of the input space. We focus on this method below since it is the one we use.

3.2 Locally Weighted Projection Regression

Locally weighted regressions were first used by Atkeson (1991) to realise supervised learning on robots. As described in Schaal et al (2002), lots of models have followed, such as LWPLS which include partial least square to reduce input dimensionality or RFWR (Schaal and Atkeson, 1997) which transform the algorithm into an incremental regression method, avoiding to store data.

Locally Weighted Projection Regression (LWPR) is an algorithm which perform both incremental regression and inputs projection. It is a function approximator which provides accurate approximation in very large spaces in $O(k)$, where k is the number of data points used to perform this estimation. LWPR uses a combination of linear models that are valid on a zone of the input space. This space, delimited by a gaussian, may change during the training to match the trained data. Each model is called a receptive field. The prediction of an entire LWPR model on an input vector is the weighted sum of the results of all the active surrounding receptive fields. Receptive fields are created or pruned in order to keep an optimal repartition.

Each receptive field first projects the input vector on the most relevant dimensions to estimate the output vector. This is done by using the covariance matrix of the input/output vectors. At each modification, the projector is updated and the algorithm checks if increasing the complexity, by adding another dimension to the input projection, significantly improves the receptive field results and modifies the projector accordingly. The projected vector is then used in the m dimension linear model (m being the output dimension) to give the output of the receptive field. During prediction, only the

significant receptive fields are activated. The algorithm may also computes the gradient of the output with respect to the input.

Different methods listed above have been used to learn various models among the ones listed in Section 2. Before presenting our own approach, we provide an overview of these different works.

3.3 Learning Inverse Kinematics

Some authors learn the inverse kinematics with a neural network which realise the mapping between operational and joint velocities, solving an unconstrained optimisation problem formulated as an energy function. This energy function is minimised during the gradient descent and weights are consequently updated. It is thus possible to obtain a desired articular velocity which correspond to a minimum energy function as

$$\dot{q}^* = \arg\min_{\dot{q}} \left(E\left(\dot{q}\right) \right)$$

where E is an energy function which is based on different errors. Pourboghrat (1989) minimises an energy which leads to learn the Moore-Penrose inverse minimising the weighted norm (as seen in Equation 5). Barhen et al (1989) resolve redundancy in minimising different Lyapunov function with goal attractors. Guez and Ahmad (1988) and Ahmad and Guez (1990) optimise the manipulability criterion: $H = \sqrt{\|JJ^T\|}$ in each configurations. Lee and Kil (1990) minimise an energy function composed of two types of constraints: one is minimising the angle difference of the first joint and another is locating the joints in the middle of joint-limits. They consider the forward kinematics as known. Brüwer and Cruse (1990) keep the system away from joint limits and compare their results to planar human motion. Finally, DeMers and Kreutz-Delgado (1997) includes the topology to bring flexibilities in his redundant kinematic inverse model.

Based on self-organising maps (Kohonen, 2001), Martinetz et al (1990b,a); Walter and Schulten (1993) learn a mapping between end effector position measured by two cameras and joint positions on a three degrees of freedom robot. The mapping is made by a three dimensional self-organising map. They automatically resolve redundancy minimising the variation of the joint angles during the learning process, using motor babbling or just learning along target trajectories.

Closer to our approach, D'Souza et al (2001) et al. learn an inverse kinematic model with LWPR. The learned inverse model is an inverse of the extended Jacobian learned on the task with the input $\left(\mathbf{q}, \dot{\boldsymbol{\xi}}\right)$ and the output $(\dot{\mathbf{q}})$

$$\mathcal{M} = LWPR_{learn}\left(\left[\mathbf{q}, \dot{\boldsymbol{\xi}}\right], \dot{\mathbf{q}}\right).$$

Doing so, no inversion is involved and singularity problems are avoided.

3.4 Learning Forward Kinematics

As forward models of serial robots can easily be computed analytically, there are few papers treating this subject.

A few authors have used multi-layer perceptrons networks or self-organising maps to learn the forward kinematics model of various simple systems (Nguyen et al, 1990; Wang and Zilouchian, 1997; Sadjadian and Taghirad, 2004), but in general the evaluation is based on the accuracy of the model itself rather than on its control capabilities. Boudreau et al (1998) and Sang and Han (1999) learn in a more convenient way the forward models of parallel robots which involves highly coupled nonlinear equations and which are difficult to model analytically.

More related to our work, Sun and Scassellati (2004, 2005) learn a forward geometric model with a radial basis function network. They derive each function to obtain a Jacobian, using classical operational space control techniques similarly to what we propose. They specifically use the two cameras of a humanoid robot to obtain the operational position used in their controller. Their approach is very similar to the one we present hereafter, provided that they use radial basis function networks whereas we use LWPR. On a similar line, Butz and Herbort (2008) learn a forward kinematics model and inverse it, using the learning classifier system XCSF as a model learning tool.

In fact, learning forward models for a redundant robot does not raise particular problems. By contrast, as explained in section 2, there exists an infinity of possible inverse mappings, thus, unless one always wants to use the same inverse mapping, it does not really make sense to directly learn kinematics or velocity kinematics inverse mappings since this leads to a loss of information regarding the redundant nature of the system. Instead, one can learn the forward mappings and invert them with the methods described in section 2.2, keeping the infinity possibilities for the inversion. One may argue that inverting a learned mapping will lead to the amplification of learning errors. This is true. However, the results we present in this chapter demonstrate that this approach actually provides very good results when combined with a closed-loop controller as well as when keeping the learning active while controlling the robot.

Taking into account these considerations and in order to compare the two approaches, in this paper we propose to learn the forward kinematics model in Equation (1) of a 3 degrees of freedom robot, giving as input the joint parameters $\mathbf{q}$ adjusted in $[0, 2\pi[$ and the task space parameters $\boldsymbol{\xi}$ as output

$$\mathcal{M} = LWPR_{learn}\left(\mathbf{q}, \boldsymbol{\xi}\right).$$

LWPR does not return directly the global model, but only the predicted output for a particular input. However, the Jacobian matrix is the first order derivative of the forward kinematics model relatively to joint space parameters $\mathbf{q}$, thus this matrix is provided "for free" while learning the

forward kinematics model. This calculation is made easier by the fact that the learned model is a simple sum of multiple linear functions which are easily differentiated.

4 Experimental Study

In this section, we present simulation based experiments designed to compare the under, fully and over-constrained cases using both the projection and the extended Jacobian approaches. When using the latter, we do not learn the inverse mapping as in D'Souza et al (2001) but rather the forward mapping which we inverse.

4.1 Control Architecture

We have chosen to evaluate the compared approaches on a 3 degrees of freedom planar system, shown in Figure 1. Sticks lengths are 0.50m, 0.40m and 0.20m. To simulate this system, we use Arboris, a dynamic simulator based on Newton-Euler equations which is implemented in matlab (Barthelemy and Bidaud, 2008). The integration step time of the simulator is chosen to be 10 milliseconds.

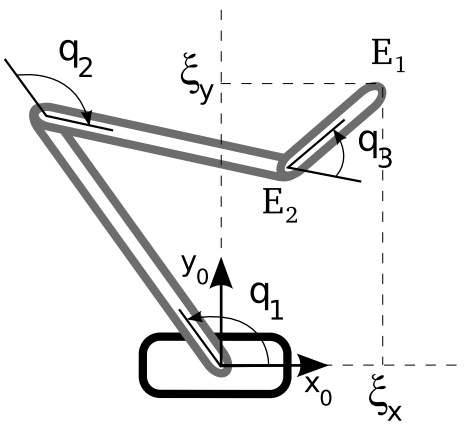


Fig. 1 Schematic view of our simulated system.

Our control scheme uses the resolved motion rate control principle, i.e., the desired task space velocity is computed using the task space parameters error

$$\dot{\boldsymbol{\xi}}^{\star} = K_p \left(\boldsymbol{\xi}^{\star} - \boldsymbol{\xi} \right), \tag{12}$$

when $\boldsymbol{\xi}^{\star}$ denotes the desired value of the task space parameters and K_p is a symmetric positive definite matrix. The actual task space parameters are

obtained from the simulator model and, in the case of a real robot, they would be measured from exteroceptive sensors. One could think about using LWPR forward kinematics prediction however this would lead to a drift with respect to the real target since no external reference would then be used to close the control loop.

Regarding the projection method, the inverse velocity kinematics is done using solution (10) and the estimated Jacobian matrices and projector which can be written as

$$\dot{\mathbf{q}} = \hat{J}_1^{+}\dot{\boldsymbol{\xi}}_1^{\star} + \left(\hat{J}_2 P_{\hat{J}_1}\right)^{+}\left(\dot{\boldsymbol{\xi}}_2^{\star} - \hat{J}_2\hat{J}_1^{+}\dot{\boldsymbol{\xi}}_1^{\star}\right). \tag{13}$$

$\hat{J}_1$ and $\hat{J}_2$ are respectively obtained from LWPR predictions

$$\left[\hat{\boldsymbol{\xi}}_1, \hat{J}_1\right] = LWPR_{predict}\left(\mathbf{q}, \mathcal{M}_1\right)$$

and

$$\left[\hat{\boldsymbol{\xi}}_2, \hat{J}_2\right] = LWPR_{predict}\left(\mathbf{q}, \mathcal{M}_2\right).$$

The extended Jacobian method leads to a solution that can be written:

$$\dot{\mathbf{q}} = \begin{bmatrix}\hat{J}_1 \\ \hat{J}_2\end{bmatrix}^{-1}\begin{bmatrix}\dot{\boldsymbol{\xi}}_1^{\star} \\ \dot{\boldsymbol{\xi}}_2^{\star}\end{bmatrix}. \tag{14}$$

$P_{\hat{J}_1}$ and pseudoinverses of $\hat{J}_1$ and $\hat{J}_2 P_{J_1}$ are obtained using their SVD as presented in Section 2.2.

$\dot{\mathbf{q}}$ obtained from Equations (13) or (14) is then differentiated and the resulting joints acceleration vector is used to compute the actuation torque based on the dynamics model in Equation (3) which we suppose to know and is obtained from Arboris (see above). A short version of this control scheme is presented on Figure 2.

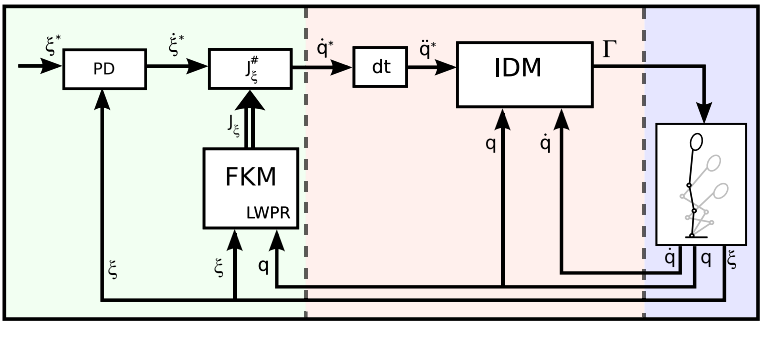


Fig. 2 Our control scheme including a Forward Velocity Kinematics Model learned with LWPR and an Inverse Dynamics Model. The dashed line is executed only during the learning process.

4.2 Choice of Parameters for the LWPR Algorithm

Before performing our experiments, we start with an initial exploration phase that can be seen as motor babbling, to initialise the model, as suggested in Klanke et al (2008). We generate random configurations taking $q_i \in [0, 2\pi[$. Depending on the corresponding configurations, we measure task parameters and feed the LWPR model with the corresponding $(\mathbf{q}, \boldsymbol{\xi})$ pairs.

Then LWPR comes with some parameters that need to be initialised. We initialise LWPR as proposed by Klanke et al (2008). The $init_D$ coefficient corresponds to the initial size of all receptive fields. This coefficient significantly affects the convergence time of LWPR. $init_D$ is tuned experimentally from comparing the performance of a set of motor babbling phases to find the best value corresponding to the minimal prediction error.

Two important parameters for our simulations are w_{gen} and *penalty*. The first one is a threshold responsible for the creation of a new local model if no model responds high enough. The *penalty* coefficient is critical to the evolution of the size of receptive fields. A small *penalty* term increases precision but decreases the smoothness of the model. We have chosen $w_{gen} = 0.5$ and $penalty = 1e^{-6}$ to have the best precision while avoiding "overlearning". Finally, from our experiments, updating D is not so important once the initialisation is well done but we still keep this option. We set $init_\alpha$ to 10000 and activate meta learning (see Klanke et al (2008)).

4.3 Experiments

In this Section, we detail three different constrained cases and associated tasks in order to study the robustness of our control scheme.

4.3.1 Under-Constrained Case

The first studied task is a reaching task. From an initial end-effector position $\boldsymbol{\xi}_1^i = [0.10\ 1.00]^T\ m$, the end-effector (E_1) of the robot has to reach a target $\boldsymbol{\xi}_1^\star = [0.20\ 0.50]^T\ m$ with a specified precision of 0.01 meters. Once the task is achieved, the end-effector is sent back to its initial position with the same controller and the same required precision. This point to point movement is repeated until the end of the simulation.

For this simple reaching task, the task space dimension is 2, thus the Jacobian is redundant and there is an infinity of ways to reach the goal. We compare the projection approach presented in Section 2.3 without any secondary task to the extended Jacobian approach, where the extension is realised by adding a one dimension constraint on point (E_2):

$\xi_{2x}^\star = 0.40\ m.$

4.3.2 Fully Constrained Case

The second experiment consists in reaching $\boldsymbol{\xi}_1^\star = [0.20\ 0.50]^T\ m$ and keeping the end effector in this position while realising a second task. This second task alternatively requires the parameter ξ_{2x} to reach the values $0.10\ m$ and $0.30\ m$ which are accessible. The first task is a two dimensional task whereas the second one is a one dimensional task. The system is thus fully constrained.

For these two tasks, the same redundancy resolution schemes are tested. In the case of the projection method, the second task is projected in the nullspace of the first one accordingly to Equation (13). In the case of the extended Jacobian method, J_{ext} is chosen as in the previous experiment.

4.3.3 Over-Constrained Case

The last experiment is very similar to the previous one. The first task is identical whereas the second one is a two dimensional task for point (E_2) which has to reach $\boldsymbol{\xi}_2^\star = [0.45\ 0.25]^T\ m$. This second task is not compatible with the first one. The system is over constrained.

Regarding this experiment, the projection method is the only one to be tested since the extended Jacobian method would require the same projection in order to obtain a square Jacobian matrix J_{ext}.

5 Results

In this section, results from the babbling phase and the experiments described in Section 4.3 are presented and analysed. Except for the babbling phase where the presented results are an average over 40 trials, the results presented below correspond to representative trials.

5.1 Babbling Phase

To evaluate the effectiveness of the forward velocity kinematics model prediction, we use the Normalised Mean Square Error ($NMSE$) computed as:

$$NMSE = \frac{1}{\sigma^2}\frac{1}{N}\sum_{i}^{N}(\mathbf{y_i} - \hat{\mathbf{y}}_\mathbf{i})^2$$

where N is the number of points used to compute this error. $\mathbf{y_i}$ is the i^{th} value of the data obtained by the real model of the robot, $\hat{\mathbf{y}}_\mathbf{i}$ is the i^{th} predicted value by the learned model and σ^2 is the sample variance of y: $\sigma^2 = \frac{1}{N}\sum_{k=1}^{N}(y_k - \overline{y}_k)^2$.

To actually compute this error, we fixed the velocity of each joint to $1.00\ rad.s^{-1}$. As can be seen on Figure 3, the $NMSE$ of the predicted velocity decreases during motor babbling. A babbling phase with 5000 samples is,

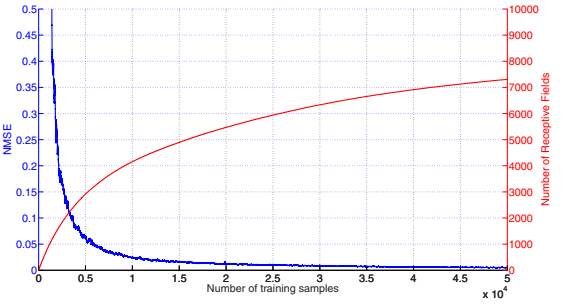


Fig. 3 Evolution of the Normalised Mean Square Error of the LWPR task space velocity prediction (blue, scale left) and of the number of receptive fields for one output (red, scale right) for a 50000 samples babbling phase (average over 40 trials).

in this case, sufficient for LWPR to cover roughly the joint space, having an output, even bad, in each configurations, and to predict an accurate enough Jacobian matrix.

Regarding the number of receptive fields, in the end of each experiment, it varies between 2000 and 8000 for each output depending on the length of the babbling phase. For babbling phases with a large number of samples, this number is almost reached at the end of the babbling phase.

The relatively large number of receptive fields required in these experiments is due to two factors. The first one is the precision of the prediction which is asked for and that can be related to our choice of parameters for the LWPR algorithm. The second factor is more specific to the redundancy of the robot which we wish to exploit. This redundancy induces possible internal motions from secondary tasks which cannot be predicted *a priori* and thus require a good coverage of the joint space in complement to specific trajectory learning.

5.2 Under-Constrained Case

In this experiment, in order to highlight the model adaptation during the control phase, we only realise motor babbling using 2000 samples. Figure 4(a) represents the two first seconds of simulation. The model of the robot is still quite approximative and the resolved motion rate controller is not sufficiently robust to compensate for inaccuracies in the learned model. Figures 4(b) (between 2s and 4s) and 4(c) (between 6s and 8s) show the evolution of the trajectories. It can be noticed that the learned model is being adapted during the control phase. Also, the precision requirements (errors smaller than 0.01m) in terms of the point that has to be reached are met. After 20s (see Figure 4(d)), the precision is improved and the trajectory of the robot trajectory is almost linear as one would expect when using a resolved motion rate controller.

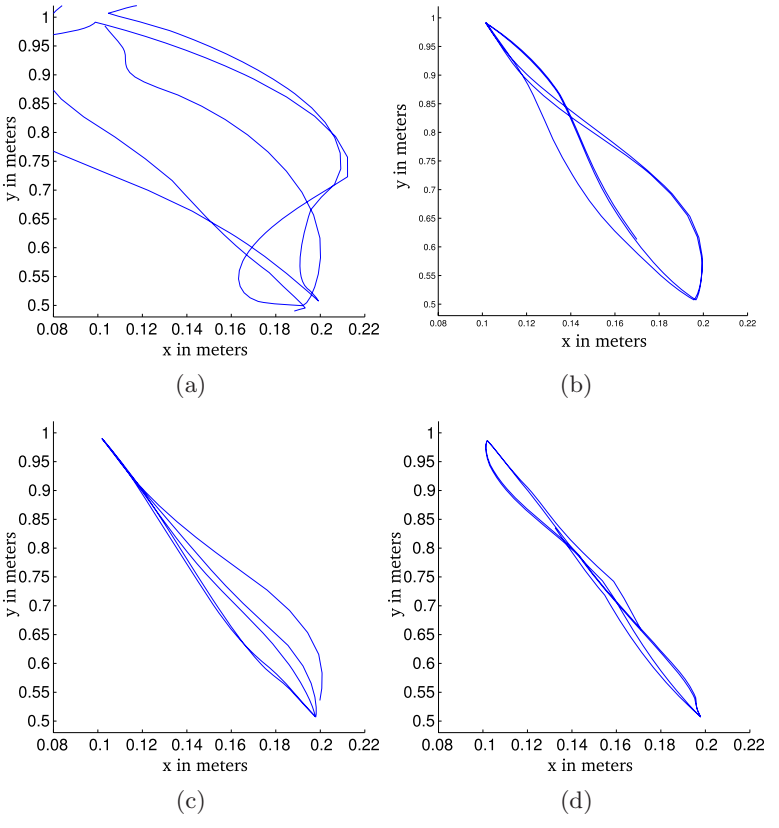


Fig. 4 Evolution of operational trajectories while learning. Each graphic represents two seconds of simulation.(a): 0s to 2s.(b): 2s to 4s.(c): 6s to 8s.(d): 20s to 22s.

This is not illustrated here but, as expected, the results obtained using the projection and the extended Jacobian methods are equivalent in the under constrained case.

5.3 Fully Constrained Case

In the fully constrained case, the precision requirements (0.01m) are also met for the two tasks which respectively constrain the position of the end-effector (point (E_1)) and the position along the $\mathbf{x}_0$ axis (see Figure 1) of the wrist of the robot (point (E_2)). This is illustrated on Figure 5 for the first task. It is shown, that there is no major difference in the precision obtained when controlling redundancy using an extended Jacobian or using the projector approach. From 0 to 1s, the reference task point is not reached yet, which explains the large error (the initial error, not shown on the figure, is 0.65m).

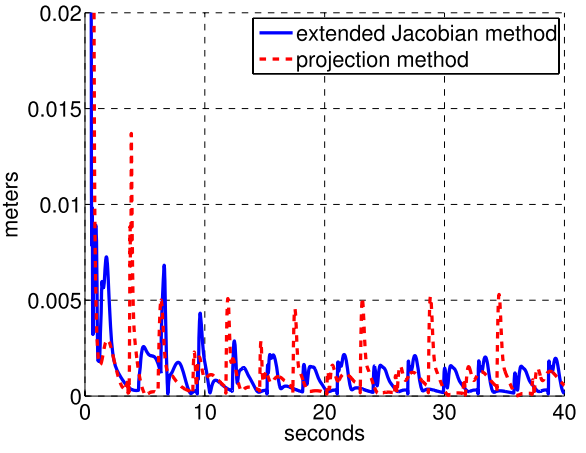


Fig. 5 Norm of the end-effector error induced by a second compatible task for two different controllers.

After 1s, the required precision is obtained and errors are due to the cyclic change of reference point for the second task. These errors decrease with time thanks to the on-line improvement of the learned model.

These errors are due to the fact that a learned model is used as well as to the fact that learning errors are propagated when computing the inverse velocity kinematics mapping from the forward one. Using the extended Jacobian approach or ours, if the Jacobian matrices are not accurately predicted, errors disturbs the tasks. Similarly, when specifically using the projection method, an error in the prediction of the first task Jacobian induces an error in the computation of the associated projector leading to disturbances induced by the second task on the first one.

Despite these error propagation effects, the achieved performances are satisfactory.

5.4 Over-Constrained Case

The results obtained from the last experiment illustrate the effectiveness of the projection method. The controller maintains the distance between the end effector point (E_1) and the desired reference point (A) under 0.01m. In the same time, the second task is partially achieved as expected from the redundancy resolution scheme which was chosen. It is achieved with the minimum possible error and without inducing any disturbance on the first task: the task hierarchy is respected.

These results are illustrated on Figure 6 where the final configuration of the system is shown as well as intermediate configuration, illustrating the convergence of the second task to the best possible result.

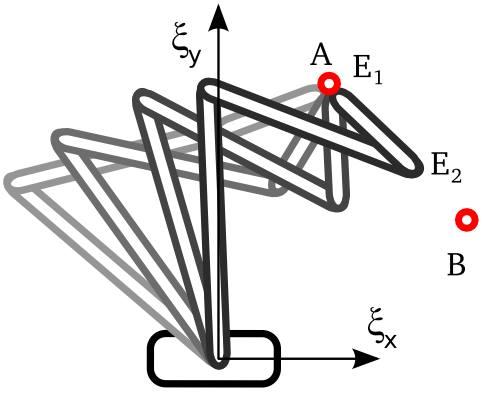


Fig. 6 Stroboscopic view of the robot realising two prioritised tasks in the incompatible case. The task hierarchy is respected as the end effector E_1 reach point A while the distance between point E_2 and point B is minimum.

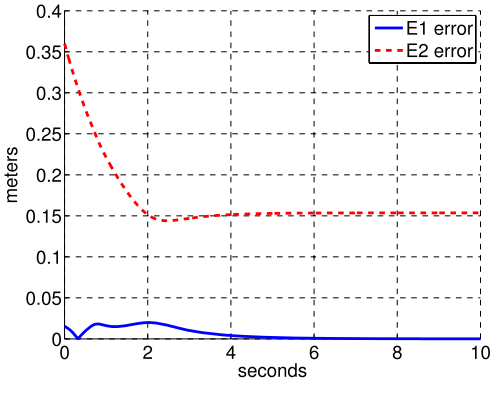


Fig. 7 Reaching errors for the first (blue, plain line) and second task (red, dotted line) in the incompatible case.

Figure 7 gives a view of the positioning errors for both tasks. Similarly to the last experiment, error propagation effects are present but once again the end effector error in position is very low.

6 Discussion

First, our results demonstrate the necessity of a babbling phase in the redundant case. The number of receptive fields associated to the learned model is quite important but this is explained by the required precision as well as by the necessity to cover the joint space appropriately (see Section 5.1).

The results of the experiments for the under constrained case shows that after some initialisation with motor babbling, our method is able to improve the model of the system while controlling it so that a reaching task is achieved with a prescribed precision. The end-effector trajectory converges to what would be expected in the case of resolved motion rate control. A similar result has already been obtained by D'Souza et al (2001), using the extended Jacobian approach and learning directly the inverse velocity kinematics mapping.

The second series of experiments (fully constrained case) convey more original results. To our knowledge, our approach is original in the sense that the forward velocity kinematics mapping is learned (through the learning of the forward kinematics model) and the obtained Jacobian matrix is used to derive the required inverse and projector allowing to combine several learned tasks. We show that this method induces error propagations but the performance of the controller remains satisfactory. This is mostly due to the fact that learning is still active while performing the task, inducing on-line model adaptation. Also, closing the control loop at the task level using exteroceptive sensor information results in the possibility to compensate for model uncertainties.

The last series of experiments (over constrained case) reinforces the results of the fully constrained case by showing that several learned tasks can be combined in a hierarchical manner in the case where those tasks are not compatible. This has been a state-of-the-art result for a while in model-based control in Robotics (Nakamura, 1991). However, to the best of our knowledge, this is the first time that such results are achieved in the case of learned models.

The retained redundancy resolution scheme in that last case is the projection method which leads to the optimal solution for both tasks. In fact, in the over constrained case, the only effective redundancy resolution scheme is the projection method. The extended Jacobian approach can be applied in that case but in a way that requires projections similarly to our approach.

Taking these considerations into account, we draw two conclusions. The first one is that the extended Jacobian approach is not satisfactory in the case where models have to be learned. Combining two tasks in a single one in order to simplify inversion leads to unnecessary constraints on the learning problem to be solved whereas it is simpler to learn elementary tasks separately. Furthermore, tasks combination is easier when the tasks are learned separately. In the extended Jacobian method, the learned inverse velocity kinematics model depends on the supplementary task. The whole model has to be learned again if this task changes whereas learning separately different Jacobian matrices leave them independent and changing one of them does not impact the others.

Our second conclusion is that in the redundant case, learning forward models does not lead to a loss of information about the system. In our method, one can choose to add any secondary task independently from the first one

and any weighting matrix W_q can be used[3] when performing the inverse of the Jacobian (see Equation (4)). That is not the case when inverse models are learned directly since this corresponds to a specific choice of inverse. Also, learning the nullspace of a given mapping at the velocity level would require a complex learning process and thus it sounds more appropriate to compute them from the learned forward velocity kinematics mapping.

7 Conclusion

In the work presented in this paper, we have used a state-of-the-art function approximation technique, LWPR, to learn the forward kinematics and, by extension, forward velocity kinematics models of a simple robotic system. We have shown that this model learning process could be combined with state-of-the-art operational space control techniques to control a robot. In particular, we demonstrated that we can benefit from the hierarchical combination capabilities of the operational space control framework to achieve several learned tasks in parallel even when those tasks are not fully compatible. This is made possible by learning the unique forward mapping for each task and then inversing it instead of directly learning an inverse mapping. Two methods were tested: the extended Jacobian approach and the projection method. The latter is shown to be more versatile than the former.

There are several possible extensions to this work. The most immediate one consists in dealing with the case of trajectory tracking instead of reaching tasks using resolved motion rate control. This may require faster on-line learning capabilities during the control phase and we will have to demonstrate that benefiting from redundancy and combining tasks is still possible in that more complex case. More generally, we should study the performance of our approach in a wider variety of tasks and combinations (joint limits avoidance, external collision avoidance) as well as in the case of using different types of inversion.

A second extension consists in learning the dynamics of the system and studying the behaviour of our approach under perturbations to validate its on-line adaptation capabilities to external forces, that will make it possible to interact with unknown objects and human users.

Longer term perspectives include an extension of our framework to systems with a larger number of degrees of freedom. Even though our example is complex enough to present our approach to learning for the control of redundant systems, model learning is of interest for complex systems. Increasing the number of dimensions leads to more complex learning problems. In that context, replacing LWPR by the Local Gaussian Process approach described in this volume (Nguyen-Tuong et al, 2010) seems a natural choice. Finally, the control framework presented in this paper considers the system as

[3] The use of a proper weighting matrix (different from the Identity) can be crucial in the dynamic case (Khatib, 1987).

deterministic, whereas in a model learning context, regarding it as stochastic seems more adequate. As a consequence, we will examine the option of moving from our deterministic framework to its stochastic equivalent described in (Toussaint and Goerick, 2010) in this volume.

References

Ahmad, Z., Guez, A.: On the solution to the inverse kinematic problem. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 3, pp. 1692–1697 (1990)

Atkeson, C.: Using locally weighted regression for robot learning. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 958–963 (1991)

Baillieul, J.: Kinematic programming alternatives for redundant manipulators. In: Proceedings of the International Conference on Robotics and Automation (ICRA), vol. 2, pp. 722–728 (1985)

Barhen, J., Gulati, S., Zak, M.: Neutral learning of constrained nonlinear transformations. Computer 22(6), 67–76 (1989)

Barthelemy, S., Bidaud, P.: Stability measure of postural dynamic equilibrium based on residual radius. In: Proceedings of the 17th CISM-IFToMM Symposium on Robot Design, Dynamics and Control (RoManSy), Tokyo, Japan (2008)

Ben Israel, A., Greville, T.: Generalized Inverses: Theory and Applications, 2nd edn. Springer, Heidelberg (2003)

Boudreau, R., Darenfed, S., Gosselin, C.: On the computation of the direct kinematics of parallel manipulators using polynomial networks. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 28(2), 213–220 (1998)

Brüwer, M., Cruse, H.: A network model for the control of the movement of a redundant manipulator. Biological Cybernetics 62(6), 549–555 (1990)

Butz, M.V., Herbort, O.: Context-dependent predictions and cognitive arm control with XCSF. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 1357–1364. ACM, New York (2008)

Calinon, S., Guenter, F., Billard, A.: On Learning, Representing and Generalizing a Task in a Humanoid Robot. IEEE Transactions on Systems, Man and Cybernetics, Part B 37(2), 286–298 (2007)

Chiaverini, S.: Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Transactions on Robotics and Automation 13(3), 398–410 (1997)

DeMers, D., Kreutz-Delgado, K.: Inverse kinematics of dextrous manipulators. Neural Systems for Robotics, 75–116 (1997)

Doty, K., Melchiorri, C., Bonivento, C.: A theory of generalized inverses applied to Robotics. The International Journal of Robotics Research 12(1), 1–19 (1993)

D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 1, pp. 298–303 (2001), doi:10.1109/IROS.2001.973374

Golub, G., Van Loan, C.: Matrix computations, 3rd edn. The John Hopkins University Press, Baltimore (1996)

Guez, A., Ahmad, Z.: Solution to the inverse kinematics problem in robotics by neural networks. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 2, pp. 617–624 (1988)

Jordan, M.I., Rumelhart, D.E.: Forward models: Supervised learning with a distal teacher. Cognitive science 16(3), 307–354 (1992)

Khatib, O.: Dynamic control of manipulators in operational space. In: Sixth CISM-IFToMM Congress on Theory of Machines and Mechanisms, pp. 1128–1131 (1983)

Khatib, O.: A unified approach to motion and force control of robot manipulators: The operational space formulation. IEEE Journal on Robotics and Automation 3(1), 43–53 (1987)

Klanke, S., Vijayakumar, S., Schaal, S.: A library for locally weighted projection regression. Journal of Machine Learning Research 9, 623–626 (2008)

Kohonen, T.: Self-Organizing Maps. Springer, Berlin (2001)

Lee, S., Kil, R.: Robot kinematic control based on bidirectional mapping neural network. In: Proceedings of International Joint Conference on Neural Networks (IJCNN), vol. 3, pp. 327–335 (1990)

Liégeois, A.: Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Transactions on Systems, Man and Cybernetics 7(12), 868–871 (1977)

Ljung, L.: System identification: theory for the user. Prentice-Hall, Inc., Upper Saddle River (1986)

Maciejewski, A., Klein, C.: Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. The International Journal of Robotics Research 4(3), 109–117 (1985)

Mansard, N., Chaumette, F.: Task sequencing for sensor-based control. IEEE Transactions on Robotics 23(1), 60–72 (2007)

Martinetz, T., Bitter, H., Schulten, K.: Learning of Visuomotor-Coordination of a robot arm with redundant degrees of freedom. In: Proceedings of the Third International Symposium on Robotics and Manufacturing–Research, Education, and Applications, Amer. Society of Mechanical, p. 521 (1990)

Martinetz, T., Ritter, H., Schulten, K.: Three-dimensional neural net for learning visuomotor coordination of a robot arm. IEEE Transactions on Neural Networks 1(1), 131–136 (1990)

Metta, G., Sandini, G., Vernon, D., Natale, L., Nori, F.: The iCub humanoid robot: an open platform for research in embodied cognition. In: PerMIS: Performance Metrics for Intelligent Systems Workshop, Washington DC, USA (2008)

Nakamura, Y.: Advanced Robotics: redundancy and optimization. Addison Wesley, Reading (1991)

Nakanishi, J., Cory, R., Mistry, M., Peters, J., Schaal, S.: Operational space control: A theoretical and empirical comparison. The International Journal of Robotics Research 27(6), 737–757 (2008)

Natale, L., Nori, F., Metta, G., Sandini, G.: Learning precise 3d reaching in a humanoid robot. In: Proceedings of the IEEE International Conference of Development and Learning (ICDL), London, UK, pp. 1–6 (2007)

Nguyen, L., Patel, R., Khorasani, K.: Neural network architectures for the forward kinematics problem in robotics. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN), vol. 3, pp. 393–399 (1990)

Nguyen-Tuong, D., Peters, J., Seeger, M., Schoelkopf, B.: Learning inverse dynamics: a comparison. In: Proceedings of the European Symposium on Artificial Neural Networks, ESANN (2008)

Nguyen-Tuong, D., Seeger, M., Peters, J.: Real-time local gp model learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 193–207. Springer, Heidelberg (2010)

Peters, J., Schaal, S.: Learning to control in operational space. The International Journal of Robotics Research 27(2), 197–212 (2008)

Pourboghrat, F.: Neural networks for learning inverse-kinematics of redundant manipulators. In: Proceedings of the 32nd Midwest Symposium on Circuits and Systems, vol. 2, pp. 760–762 (1989)

Rojas, R.: Neural networks: a systematic introduction. Springer, Heidelberg (1996)

Sadjadian, H., Taghirad, H.D.: Numerical methods for computing the forward kinematics of a redundant parallel manipulator. In: Proceedings of the IEEE Conference on Mechatronics and Robotics, Aachen, Germany (2004)

Sang, L.H., Han, M.C.: The estimation for forward kinematic solution of stewart platform using the neural network. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 1, pp. 501–506 (1999)

Schaal, S., Atkeson, C.G.: Receptive field weighted regression. ART Human Inf. Process Lab, Kyoto, Japan, Tech. Rep. TR-H-209 (1997)

Schaal, S., Atkeson, C.G., Vijayakumar, S.: Scalable techniques from nonparametric statistics for real time robot learning. Applied Intelligence 17(1), 49–60 (2002)

Sentis, L., Khatib, O.: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. The International Journal of Humanoid Robotics 2(4), 505–518 (2005)

Shon, A., Grochow, K., Rao, R.: Robotic imitation from human motion capture using gaussian processes. In: Proceedings of the IEEE-RAS/RSJ International Conference on Humanoid Robots, Humanoids (2005)

Van der Smagt, P.P.: Minimisation methods for training feedforward neural networks. Neural Networks 7(1), 1–11 (1994)

Snyman, J.A.: Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms. Springer, Heidelberg (2005)

Sun, G., Scassellati, B.: Reaching through learned forward model. In: 4th IEEE/RAS International Conference on Humanoid Robots, vol. 1, pp. 93–112 (2004)

Sun, G., Scassellati, B.: A fast and efficient model for learning to reach. International Journal of Humanoid Robotics 2(4), 391–414 (2005)

Toussaint, M., Goerick, C.: A bayesian view on motor control and planning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 227–252. Springer, Heidelberg (2010)

Vijayakumar, S., Schaal, S.: Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford, CA (2000)

Vijayakumar, S., D'Souza, A., Schaal, S.: LWPR: A Scalable Method for Incremental Online Learning in High Dimensions. Tech. rep. Edinburgh University Press (2005)

Walter, J., Schulten, K.: Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. IEEE Transactions on Neural Networks 4(1), 86–96 (1993)

Wang, D., Zilouchian, A.: Solutions of kinematics of robot manipulators using a kohonen self-organizing neural network. In: Proceedings of the IEEE International Symposium on Intelligent Control, pp. 251–255 (1997)

Willow, G.: Overview of the PR2 robot (2009), http://www.willowgarage.com/pages/robots/pr2-overview

Real-Time Local GP Model Learning

Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters

Abstract. For many applications in robotics, accurate dynamics models are essential. However, in some applications, e.g., in model-based tracking control, precise dynamics models cannot be obtained analytically for sufficiently complex robot systems. In such cases, machine learning offers a promising alternative for approximating the robot dynamics using measured data. However, standard regression methods such as Gaussian process regression (GPR) suffer from high computational complexity which prevents their usage for large numbers of samples or online learning to date. In this paper, we propose an approximation to the standard GPR using local Gaussian processes models inspired by [Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal, Snelson and Ghahramani(2007)]. Due to reduced computational cost, local Gaussian processes (LGP) can be applied for larger samplesizes and online learning. Comparisons with other nonparametric regressions, e.g., standard GPR, support vector regression (SVR) and locally weighted projection regression (LWPR), show that LGP has high approximation accuracy while being sufficiently fast for real-time online learning.

1 Introduction

Precise models of technical systems can be crucial in technical applications [Roberts et al(2010)Roberts, Moret, Zhang, and Tedrake,Fumagalli et al(2010)Fumagalli, Gijsberts, Ivaldi, Jamone, Metta, Natale, Nori, and Sandini]. In robot tracking control, only well-estimated inverse dynamics models allow both high accuracy and compliant, low-gain control. For complex robots such as humanoids or light-weight arms, it is often hard to analytically model the system sufficiently well and, thus, modern

Duy Nguyen-Tuong and Jan Peters
Max Planck Institute for Biological Cybernetics, 72076 Tübingen
e-mail: {duy.nguyen-tuong,jan.peters}@tuebingen.mpg.de

Matthias Seeger
Saarland University, 66123 Saarbrücken
e-mail: mseeger@mmci.uni-saarland.de

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 193–207.
springerlink.com

regression methods can offer a viable alternative [Schaal et al(2002)Schaal, Atkeson, and Vijayakumar, Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal]. However, highly accurate regression methods such as Gaussian process regression (GPR) suffer from high computational cost, while fast real-time learning algorithms such as locally weighted projection regression (LWPR) are not straightforward to use, as they require manual adjustment of many data dependent parameters.

In this paper, we attempt to combine the strengths of both approaches, i.e., the high accuracy and comfortable use of GPR with the fast learning speed of LWPR. We will proceed as follows: firstly, we briefly review both model-based control as well as standard Gaussian process regression. We will discuss the necessity of estimating the inverse dynamics model for compliant, low-gain control. Subsequently, we describe our local Gaussian process models (LGP) approach.

In Section 3, the learning accuracy and performance of the presented LGP approach will be compared with several relevant regression methods, e.g., standard GPR [Rasmussen and Williams(2006)], ν-support vector regression (ν-SVR) [Schölkopf and Smola (2002)], sparse online GP (OGP) [Csato and Opper(2002)] and LWPR [Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal, Schaal et al(2002)Schaal, Atkeson, and Vijayakumar]. The applicability of the LGP for low-gain model-based tracking control and real-time learning is demonstrated on a Barrett whole arm manipulator (WAM). We can show that its tracking performance exceeds analytical models [Craig(2004)] while remaining fully compliant.

1.1 Model-Based Control

Model-based control, e.g., computed torque control [Spong et al(2006)Spong, Hutchinson, and Vidyasagar], enables high speed and compliant robot control while achieving accurate control with small tracking errors for sufficiently precise robot models. The controller is supposed to move the robot that is governed by the system dynamics [Spong et al(2006)Spong, Hutchinson, and Vidyasagar]

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \varepsilon(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) = \mathbf{u}, \tag{1}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ are joint angles, velocities and accelerations of the robot, respectively, $\mathbf{u}$ denotes the applied torques, $\mathbf{M}(\mathbf{q})$ the inertia matrix of the robot and $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ Coriolis and centripetal forces, $\mathbf{G}(\mathbf{q})$ gravity forces and $\varepsilon(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ represents non-linearities of the robot which are not part of the rigid-body dynamics.

The model-based tracking control law determines the joint torques $\mathbf{u}$ necessary for following a desired trajectory $\mathbf{q}_d$, $\dot{\mathbf{q}}_d$, $\ddot{\mathbf{q}}_d$ using a dynamics model while employing feedback in order to stabilize the system. For example, the inverse dynamics model of the robot can be used as a feed-forward model that predicts the joint torques $\mathbf{u}_{\mathrm{FF}}$ required to perform the desired trajectory [Spong et al(2006)Spong, Hutchinson, and Vidyasagar, Craig(2004)], while a feedback term $\mathbf{u}_{\mathrm{FB}}$ ensures the stability of the tracking control with a resulting control law of $\mathbf{u} = \mathbf{u}_{\mathrm{FF}} + \mathbf{u}_{\mathrm{FB}}$. The feedback term can be a linear control law such as $\mathbf{u}_{\mathrm{FB}} = \mathbf{K}_p\mathbf{e} + \mathbf{K}_v\dot{\mathbf{e}}$, where $\mathbf{e} = \mathbf{q}_d - \mathbf{q}$ denotes the tracking error and $\mathbf{K}_p, \mathbf{K}_v$ position-gain and velocity-gain, respectively.

If an accurate model in the form of Equation (1) can be obtained for the feed-forward model, e.g., for negligible unknown nonlinearities ε, the resulting feed-forward term $\mathbf{u}_{\mathrm{FF}}$ will largely cancel the robots nonlinearities [Spong et al(2006)Spong, Hutchinson, and Vidyasagar].

For complex robots such as humanoids or light-weight arms, it is often hard to model the system sufficiently well using the rigid body dynamics. Unknown nonlinearities $\varepsilon(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ such as flexible hydraulic tubes, complex friction, gear boxes, etc, couple several degrees of freedom together and result in highly altered dynamics. Such unknown nonlinearities can dominate the system dynamics and deteriorate the analytical model [Nakanishi et al(2005)Nakanishi, Farrell, and Schaal]. The resulting tracking error needs to be compensated using large gains [Spong et al(2006)Spong, Hutchinson, and Vidyasagar]. However, high feedback gains prohibit compliant control and, thus, make the robot less safe for the environment while causing many practical problems such as actuator saturation, excitation of unmodeled dynamics, may result in large tracking errors in presence of noise, increase energy consumption, etc. To avoid high-gain feedback, it is essential to improve the accuracy of the dynamics model for predicting $\mathbf{u}_{\mathrm{FF}}$. Since $\mathbf{u}_{\mathrm{FF}}$ is a function of $\mathbf{q}_d,\dot{\mathbf{q}}_d,\ddot{\mathbf{q}}_d$, it can be obtained with supervised learning using measured data. The resulting problem is a regression problem that can be solved by learning the mapping $\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}\rightarrow\mathbf{u}$ on sampled data [Schaal et al(2000)Schaal, Atkeson, and Vijayakumar, Nguyen-Tuong et al(2008)Nguyen-Tuong, Peters, and Seeger] and, subsequently, using the resulting mapping for determining the feed-forward motor commands. As trajectories and corresponding joint torques are sampled directly from the real robot, learning the mapping will include all nonlinearities and not only the ones described in the rigid-body model.

1.2 Regression with Standard GPR

As any realistic inverse dynamics is a well-defined functional mapping of continuous, high-dimensional inputs to outputs of the same kind, we can view it as a regression problem. Given the input $\mathbf{x}\in\mathbb{R}^n$ and the target $\mathbf{y}\in\mathbb{R}^n$, the task of regression algorithms is to learn the mapping describing the relationship from input to target using samples. A powerful method for accurate function approximation in high-dimensional space is Gaussian process regression (GPR) [Rasmussen and Williams(2006)]. Given a set of n training data points $\{\mathbf{x}_i,y_i\}_{i=1}^n$, we would like to learn a function $f(\mathbf{x}_i)$ transforming the input vector $\mathbf{x}_i$ into the target value y_i given a model $y_i=f(\mathbf{x}_i)+\varepsilon_i$, where ε_i is Gaussian noise with zero mean and variance σ_n^2 [Rasmussen and Williams(2006)]. As a result, the observed targets can also be described by a Gaussian distribution $\mathbf{y}\sim\mathcal{N}\left(\mathbf{0},\mathbf{K}(\mathbf{X},\mathbf{X})+\sigma_n^2\mathbf{I}\right)$, where $\mathbf{X}$ denotes the set containing all input points $\mathbf{x}_i$ and $\mathbf{K}(\mathbf{X},\mathbf{X})$ the covariance matrix computed using a given covariance function. Gaussian kernels are probably the most frequently used covariance functions [Rasmussen and Williams(2006)] and are given by

$$k(\mathbf{x}_p,\mathbf{x}_q)=\sigma_s^2\exp\left(-\frac{1}{2}(\mathbf{x}_p-\mathbf{x}_q)^T\mathbf{W}(\mathbf{x}_p-\mathbf{x}_q)\right), \tag{2}$$

where σ_s^2 denotes the signal variance and $\mathbf{W}$ represents the widths of the Gaussian kernel. Other choices for possible kernels can be found in [Schölkopf and Smola (2002), Rasmussen and Williams(2006)]. The joint distribution of the observed target values and predicted value $f(\mathbf{x}_*)$ for a query point $\mathbf{x}_*$ is given by

$$\begin{bmatrix} \mathbf{y} \\ f(\mathbf{x}_*) \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_n^2\mathbf{I} & k(\mathbf{X},\mathbf{x}_*) \\ k(\mathbf{x}_*,\mathbf{X}) & k(\mathbf{x}_*,\mathbf{x}_*) \end{bmatrix}\right). \tag{3}$$

Conditioning the joint distribution yields the predicted mean value $f(\mathbf{x}_*)$ with the corresponding variance $V(\mathbf{x}_*)$

$$\begin{aligned} f(\mathbf{x}_*) &= k_*^T \left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y} = k_*^T\alpha\,, \\ V(\mathbf{x}_*) &= k(\mathbf{x}_*,\mathbf{x}_*) - k_*^T\left(\mathbf{K} + \sigma_n^2\mathbf{I}\right)^{-1}k_*\,, \end{aligned} \tag{4}$$

with $k_* = k(\mathbf{X},\mathbf{x}_*)$, $\mathbf{K} = \mathbf{K}(\mathbf{X},\mathbf{X})$ and α denotes the so-called prediction vector. The hyperparameters of a Gaussian process with Gaussian kernel are given by $\theta = [\sigma_n^2, \sigma_f^2, \mathbf{W}]$ and remain the only open parameters. Their optimal value for a particular data set can be automatically estimated by maximizing the log marginal likelihood using standard optimization methods such as Quasi-Newton methods [Rasmussen and Williams(2006)].

2 Local Gaussian Process Regression

Due to high computational complexity of nonlinear regression techniques, inverse dynamics models are frequently only learned offline for pre-sampled desired trajectories [Nguyen-Tuong et al(2008)Nguyen-Tuong, Peters, and Seeger]. In order to take full advantage of a learning approach, online learning is an absolute necessity as it allows the adaption to changes in the robot dynamics, load or the actuators. Furthermore, a training data set will never suffice for most robots with a large number of degrees of freedom and, thus, fast online learning is necessary if the trajectory leads to new parts of the state-space. However, for most real-time applications online model learning poses a difficult regression problem due to three constraints, i.e., firstly, the learning and prediction process should be very fast (e.g., learning needs to take place at a speed of 20-200Hz and prediction may take place at 200Hz up to 5kHz). Secondly, the learning system needs to be capable of dealing with large amounts of data (i.e., with data arriving at 200Hz, less than ten minutes of runtime will result in more than a million sampled data points). And, thirdly, the data arrives as a continuous stream, thus, the model has to be continuously adapted to new training examples over time.

Model learning with GPR suffers from the expensive computation of the inverse matrix $(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}$ which yields a cost of $\mathcal{O}(n^3)$, see Equation (4). Inspired by locally weighted regression [Schaal et al(2002)Schaal, Atkeson, and Vijayakumar, Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal], we propose a method for speed-up the training and prediction process by partitioning the training data in local regions and learning an independent Gaussian process model (as given

in Section 1.2) for each region. The number of data points in the local models is limited, where insertion and removal of data points can be treated in a principled manner. The prediction for a query point is performed by weighted average similar to LWPR [Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal]. For partitioning and weighted prediction we use a kernel as similarity measure. Thus, our algorithm consists out of three stages: (i) clustering of data, i.e., insertion of new data points into the local models, (ii) learning of corresponding local models and (iii) prediction for a query point.

2.1 Partitioning of Training Data

Clustering input data can be performed efficiently using a similarity measure between the input point $\mathbf{x}$ and the centers of the respective local models. From a machine learning point of view, the similarity or proximity of data points can be defined in terms of a kernel. Kernel functions represent the dot product between two vectors in the feature space and, hence, naturally incorporate the similarity measure between data points. The clustering step described in this section results from the basic assumption that nearby input points are likely to have similar target values. Thus, training points that belong to the same local region (represented by a center) are informative about the prediction for query points next to this local region.

A specific characteristic in this framework is that we take the kernel for learning the Gaussian process model as similarity measure w_k for the clustering process. If a Gaussian kernel is employed for learning the model, the corresponding measure will be

$$w_k\left(\mathbf{x},\mathbf{c}_k\right)=\exp\left(-\frac{1}{2}\left(\mathbf{x}-\mathbf{c}_k\right)^T\mathbf{W}\left(\mathbf{x}-\mathbf{c}_k\right)\right), \tag{5}$$

where $\mathbf{c}_k$ denotes the center of the k-th local model and $\mathbf{W}$ a diagonal matrix represented the kernel width. It should be emphasized that for learning the Gaussian process model any admissible kernel can be used. Thus, the similarity measure for the clustering process can be varied in many ways, and, for example, the commonly used Matern kernel [Seeger(2004)] could be used instead of the Gaussian one. For the hyperparameters of the measure, such as $\mathbf{W}$ for Gaussian kernel, we use the *same* training approach as introduced in Section 1.2. Since the hyperparameters of a Gaussian process model can be achieved by likelihood optimization, it is straightforward to adjust the open parameters for the similarity measure. For example, we can subsample the available training data and, subsequently, perform the standard optimization procedure.

After computing the proximity between the new data point $\mathbf{x}_{new}$ and all available centers, the data point will be included to the *nearest* local model, i.e., the one with the maximal value of w_k. As the data arrives incrementally over time, a new model with center $\mathbf{c}_{k+1}$ is created if all similarity measures w_k fall below a threshold w_{gen}. The new data point is then used as new center $\mathbf{c}_{k+1}$ and, thus, the number of local models will increase if previously unknown parts of the state space are visited. When a new data point is assigned to a particular k-th model, i.e., $\max_k w_k(\mathbf{x}) > w_{\text{gen}}$ the center $\mathbf{c}_k$ will be updated to the mean of corresponding local data points.

Algorithm 1: Partitioning the training data with incremental model learning.

Input: new data point $\{\mathbf{x}_{\text{new}}, y_{\text{new}}\}$.
for $k = 1$ **to** number of local models **do**
  Compute proximity to the k-th local model:
    $w_k = k(\mathbf{x}_{\text{new}}, \mathbf{c}_k)$
end for
Take the nearest local model:
  $v = \max_k w_k$
if $v > w_{\text{gen}}$ **then**
  Insert $\{\mathbf{x}_{\text{new}}, y_{\text{new}}\}$ into the nearest local model:
    $\mathbf{X}_{\text{new}} = [\mathbf{X}, \mathbf{x}_{\text{new}}]$, $\mathbf{y}_{\text{new}} = [\mathbf{y}, y_{\text{new}}]$
  Update the corresponding center:
    $\mathbf{c}_{\text{new}} = \text{mean}(\mathbf{X}_{\text{new}})$
  Update the Cholesky matrix and the
  prediction vector of local model:
    Compute l and l_*
    Compute $\mathbf{L}_{\text{new}}$
    If the maximum number of data points is reached
    delete another point by permutation.
    Compute α_{new} by back-substitution
else
  Create new model:
    $\mathbf{c}_{k+1} = \mathbf{x}_{\text{new}}$, $\mathbf{X}_{k+1} = [\mathbf{x}_{\text{new}}]$, $\mathbf{y}_{k+1} = [y_{\text{new}}]$
  Initialize of new Cholesky matrix $\mathbf{L}$ and
  new prediction vector α.
end if

Algorithm 2: Prediction for a query point.

Input: query data point $\mathbf{x}$, M .
Determine M local models closest to $\mathbf{x}$.
for $k = 1$ **to** M **do**
  Compute proximity to the k-th local model:
    $w_k = k(\mathbf{x}, \mathbf{c}_k)$
  Compute local prediction using the k-th local model:
    $\bar{y}_k = \mathbf{k}_k^T \alpha_k$
end for
Compute weighted prediction using M local models:
  $\hat{y} = \sum_{k=1}^{M} w_k \bar{y}_k / \sum_{k=1}^{M} w_k$.

2.2 Incremental Update of Local Models

During online learning, we have to deal with an endless stream of data (e.g., at a 500 Hz sampling rate we get a new data point every 2 ms and have to treat 30 000 data points per minute). In order to cope with the real-time requirements, the maximal number of training examples needs to be limited so that the local models do not end up with the same complexity as a standard GPR regression. Since the number

of acquired data points increases continuously over time, we can enforce this limit by incrementally deleting old data points when newer and better ones are included. Insertion and deletion of data points can be achieved using first order principles, for example, maximizing the information gain while staying within a budget (e.g., the budget can be a limit on the number of data points). Nevertheless, while the update of the target vector $\mathbf{y}$ and input matrix $\mathbf{X}$ can be done straightforwardly, the update of the covariance matrix (and implicitly the update of the prediction vector α, see Equation (4)) is more complicated to derive and requires thorough analysis given here.

The prediction vector α can be updated incrementally by directly adjusting the Cholesky decomposition of the Gram matrix $(\mathbf{K} + \sigma_n^2\mathbf{I})$ as suggested in [M.Seeger(2007)]. For doing so, the prediction vector can be rewritten as $\mathbf{y} = \mathbf{L}\mathbf{L}^T\alpha$, where the lower triangular matrix $\mathbf{L}$ is a Cholesky decomposition of the Gram matrix. Incremental insertion of a new point is achieved by adding an additional row to the matrix $\mathbf{L}$.

Proposition 1. *If* $\mathbf{L}$ *is the Cholesky decomposition of the Gram matrix* $\mathbf{K}$ *while* $\mathbf{L}_{new}$ *and* $\mathbf{K}_{new}$ *are obtained by adding additional row and column, such that*

$$\mathbf{L}_{new} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{l}^T & l_* \end{bmatrix}, \mathbf{K}_{new} = \begin{bmatrix} \mathbf{K} & \mathbf{k}_{new}^T \\ \mathbf{k}_{new} & k_{new} \end{bmatrix}, \tag{6}$$

with $\mathbf{k}_{new} = k(\mathbf{X}, \mathbf{x}_{new})$ *and* $k_{new} = k(\mathbf{x}_{new}, \mathbf{x}_{new})$*, then* $\mathbf{l}$ *and* l_* *can be computed by solving*

$$\mathbf{L}\mathbf{l} = \mathbf{k}_{new} \tag{7}$$

$$l_* = \sqrt{k_{new} - \|\mathbf{l}\|^2} \tag{8}$$

Proof. Multiply out the equation $\mathbf{L}_{new}\mathbf{L}_{new}^T = \mathbf{K}_{new}$ and solve for $\mathbf{l}$ and l_*. □

Since $\mathbf{L}$ is a triangular matrix, $\mathbf{l}$ can be determined from Equation (7) by substituting it back in after computing the kernel vector $\mathbf{k}_{new}$. Subsequently, l_* and the new prediction vector α_{new} can be determined from Equation (8), where α_{new} can be achieved by twice back-substituting while solving $\mathbf{y}_{new} = \mathbf{L}_{new}\mathbf{L}_{new}^T\alpha_{new}$. If the maximal number of training examples is reached, an old data point has to be deleted every time when a new point is being included. The deletion of the m-th data point can be performed efficiently using a permutation matrix $\mathbf{R}$ and solving $\mathbf{y}_{new} = \mathbf{R}\,\mathbf{L}_{new}\mathbf{L}_{new}^T\mathbf{R}\,\alpha_{new}$, where $\mathbf{R} = I - (\delta_m - \delta_n)(\delta_m - \delta_n)^T$ and δ_i is a zero vector whose i-th element is one [M.Seeger(2007)]. In practice, the new data point is inserted as a first step to the *last* row (n-th row) according to Equation (6) and, subsequently, the m-th data point is removed by adjusting $\mathbf{R}$. The partitioning and learning process is summarized in Algorithm 1. The incremental Cholesky update is very efficient and can be performed in a numerically stable manner as discussed in detail in [M.Seeger(2007)].

Due to the Cholesky update formulation, the amount of computation for training can be limited due to the incremental insertion and deletion of data points. The main computational cost for learning the local models is dominated by the incremental update of the Cholesky matrix which yields $\mathcal{O}(N_l^2)$, where N_l presents the number

of data points in a local model. Importantly, N_l can be set in accordance with the computational power of the available real-time computer system.

2.3 Prediction Using Local Models

The prediction for a mean value $\hat{y}$ is performed using weighted averaging over M local GP predictions $\bar{y}_k$ for a query point $\mathbf{x}$ similar to LWPR [Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal]. The weighted prediction $\hat{y}$ is then given by $\hat{y} = \mathbb{E}\{\bar{y}_k|\mathbf{x}\} = \sum_{k=1}^{M} \bar{y}_k p(k|\mathbf{x})$. According to the Bayesian theorem, the probability of the model k given query point $\mathbf{x}$ can be expressed as

$$p(k|\mathbf{x}) = \frac{p(k,\mathbf{x})}{p(\mathbf{x})} = \frac{p(k,\mathbf{x})}{\sum_{k=1}^{M} p(k,\mathbf{x})} = \frac{w_k}{\sum_{k=1}^{M} w_k}. \tag{9}$$

Hence, we have

$$\hat{y} = \frac{\sum_{k=1}^{M} w_k \bar{y}_k}{\sum_{k=1}^{M} w_k}, \tag{10}$$

Thus, each local GP prediction $\bar{y}_k = k(\mathbf{X}_k, \mathbf{x})^T \alpha_k$ is additionally weighted by the similarity $w_k(\mathbf{x}, \mathbf{c}_k)$ between the corresponding center $\mathbf{c}_k$ and the query point $\mathbf{x}$. The search for M local models can be quickly done by evaluating the proximity between the query point $\mathbf{x}$ and all model centers $\mathbf{c}_k$. The prediction procedure is summarized in Algorithm 2.

3 Learning Inverse Dynamics for Model-Based Control

Learning models for control of high-dimensional systems in real-time is a difficult endeavor and requires extensive evaluation. For this reason, we evaluate our algorithm (LGP) using high-dimensional data taken from two real robots, e.g., the 7 degree-of-freedom (DoF) anthropomorphic SARCOS master arm and 7-DoF Barrett WAM both shown in Figure 1. Subsequently, we apply LGP for online learning of inverse dynamics models for robot tracking control. The tracking control task with model online learning is performed on the Barrett WAM in real-time. Finally, we highlight the advantages of online-learned models versus offline approximation and analytical model in a more complex experiment for learning of character writing.

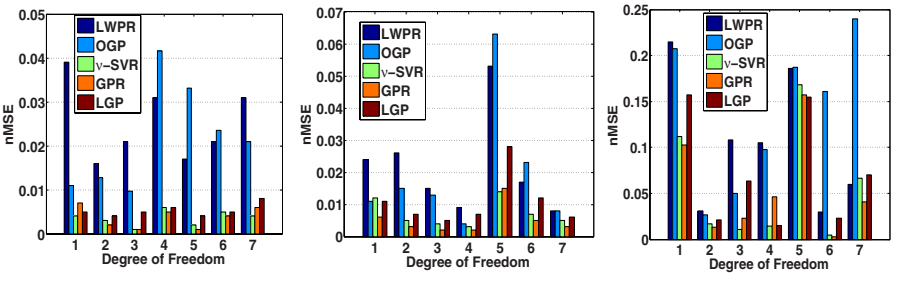
(a) **SARCOS arm** (b) **Barrett WAM**

Fig. 1. Robot arms used for data generation and experiments

3.1 Learning Accuracy Comparison

In this section, we compare the learning performance of LGP with the state-of-the-art in nonparametric regression, e.g., LWPR, ν-SVR [Schölkopf and Smola (2002)], standard GPR and online Gaussian Process Regression (OGP) [Schölkopf and Smola (2002)] in the context of approximating inverse robot dynamics. For evaluating ν-SVR and GPR, we have employed the libraries [Chang and Lin(2001)] and [Seeger(2007)], respectively. The code for LGP contained also parts of the library [Seeger(2007)].

For comparing the prediction accuracy of our proposed method in the setting of learning inverse dynamics, we use three data sets, (i) SL simulation data (SARCOS model) as described in [Nguyen-Tuong et al(2008)Nguyen-Tuong, Peters, and Seeger] (14094 training points and 5560 test points), (ii) data from the SARCOS master arm (13622 training points and 5500 test points) [Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal] as well as (iii) a data set generated from our Barrett arm (13572 training points, 5000 test points). Given samples $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]$ as input, where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ denote the joint angles, velocity and acceleration, respectively, and using the corresponding joint torques $\mathbf{y} = [\mathbf{u}]$ as targets, we have a well-defined, proper regression problem. The considered seven degrees of freedom (DoF) robot arms result in 21 input dimensions (i.e., for each joint, we have an angle, a velocity and an acceleration) and seven target or output dimensions (i.e., a single torque for each joint). The robot inverse dynamics model can be estimated separately for each DoF employing LWPR, ν-SVR, GPR, OGP and LGP, respectively.



(a) **Approximation Error on SL data (SARCOS model)**

(b) **Approximation Error on SARCOS data**

(c) **Approximation Error on Barrett WAM data**

Fig. 2. The approximation error is represented by the normalized mean squared error (nMSE) for each DoF (1–7) and shown for (a) simulated data from physically realistic SL simulation, (b) real robot data from an anthropomorphic SARCOS master arm and (c) measurements from a Barrett WAM. In all cases, LGP outperforms LWPR and OGP in learning accuracy while being competitive to ν-SVR and standard GPR. The small variances of the output targets in the Barrett data results in a nMSE that is a larger scale compared to SARCOS; however, this increase has no practical meaning and only depends on the training data.

The training examples for LGP can be partitioned either in the same input space where the local models are learned or in a subspace that has to be physically consistent with the approximated function. In the following, we localize the data depending on the position of the robot. Thus, the partitioning of training data is performed in a seven dimensional space (i.e., consisting of the seven joint angles). After determining the similarity metric w_k for all k local models in the partitioning space, the input point will be assigned to the *nearest* local model, i.e., the local model with the maximal value of distance measure w_k. For computing the localization, we will use the Gaussian kernel as given in Equation (2) and the corresponding hyperparameters are optimized using a subset of the training set.

Note that the choice of the limit value w_{gen} during the partitioning step is crucial for the performance of LGP and, unfortunately, is an open parameter requiring manual tuning. If w_{gen} is too small, a large number of local models will be generated with small number of training points. As these small models receive too little data for a stable GPR, they do not generalize well to unknown neighboring regions of the state space. If w_{gen} is large, the local models will include too many data points which either results in over-generalization or, if the number of admitted data points is enlarged as well, it will increase the computational complexity. Here, the training data is clustered in about 30 local regions ensuring that each local model has a sufficient amount of data points for high accuracy (in practice, roughly a hundred data points for each local model suffice) while having sufficiently few that the solution remains feasible in real-time (e.g., on the test hardware, an Intel Core Duo at 2GHz, that implies the usage of up to a 1000 data points per local model). On average, each local model includes approximately 500 training examples, i.e., some models will not fill up while others actively discard data. This small number of training data points enables a fast training for each local model using the previously described fast Cholesky matrix updates.

Figure 2 shows the normalized mean squared error (nMSE) of the evaluation on the test set for each of the three evaluated scenarios, i.e., a physically realistic simulation of the SARCOS arm in Figure 2 (a), the real anthropomorphic SARCOS master arm in Figure 2 (b) and the Barrett WAM arm in Figure 2 (c). Here, the normalized mean squared error is defined by nMSE = Mean squared error/Variance of target. During the prediction on the test set using LGP, we take the most activated local models, i.e., the ones which are next to the query point.

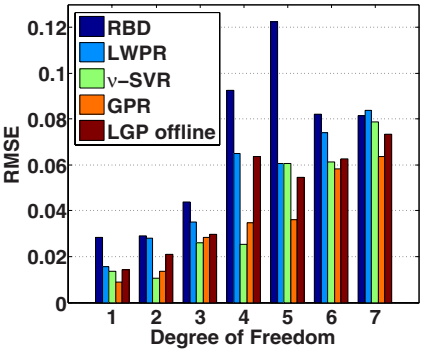
When observing the approximation error on the test set shown in Figure 2(a-c), it can be seen that LGP generalizes well to the test data during prediction. In all cases, LGP outperforms LWPR and OGP while being close in learning accuracy to the offline-methods GPR and ν-SVR. The mean prediction for GPR is determined according to Equation (4) where we pre-computed the prediction vector α from training data. When a query point appears, the kernel vector $\mathbf{k}_*^T$ is evaluated for this particular point.

3.2 Online Learning for Model-Based Control

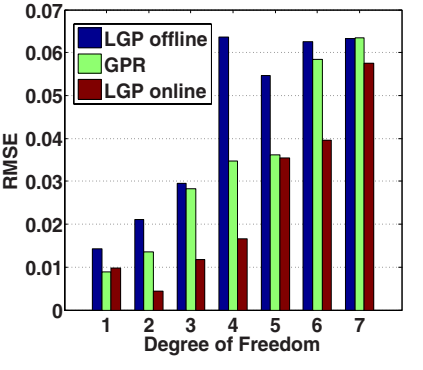
In this section, we apply the inverse dynamics models for a model-based tracking control task [Craig(2004)]. Here, the model is used for predicting the feedforward

torques $\mathbf{u}_{\text{FF}}$ necessary to execute a given the desired trajectory $[\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d]$. First, we compare standard rigid-body dynamics (RBD) models with several models learned offline on training data sets. The RBD-parameters are estimated from the corresponding CAD model. During the control task, the offline-learned models are used for an online torque prediction. For this comparison, we use LWPR, ν-SVR, standard GPR as well as our LGP as compared learning methods. We show that our LGP is competitive when compared with its alternatives. Second, we demonstrate that LGP is capable of online adaptation while being used for predicting the required torques. During online learning, the local GP models are updated in real-time, and the online improvement during a tracking task outperforms the fixed offline model in comparison. Our goal is to achieve compliant tracking in robots without exception handling or force sensing but purely based on using low control gains. Our control gains are three orders of magnitude smaller than the manufacturers in the experiments and we can show that using good, learned inverse dynamics models we can still achieve compliant control. Due to the low feedback gains, the accuracy of the model has a stronger effect on the tracking performance in this setting and, hence, a more precisely learned model will also results in a significantly lower tracking error.

For comparison with offline-learned models, we also compute the feedforward torque using rigid-body (RB) formulation which is a common approach in robot control [Craig(2004)]. The control task is performed in real-time on the Barrett WAM, as shown in Figure 1. As desired trajectory, we generate a test trajectory which is similar to the one used for learning the inverse dynamics models. Figure 3 (a) shows the tracking errors on test trajectory for 7 DoFs using offline-learned models. The error



(a) **Tracking errors on Barrett: comparison of offline-learned models**



(b) **Tracking errors on Barrett: full GPR vs. offline and online-learned models with LGP**

Fig. 3. (a) and **(b)** show the tracking errors (RMSE) on the Barrett WAM. For offline-learned models, LGP is competitive with full GPR and ν-SVR while being better than LWPR and rigid-body model. When employing online-updates, LGP can largely improve the tracking results outperforming the offline-learned models using full GPR. The reported results are computed for a test trajectory executed on the robot.

is computed as root mean square error (RMSE) which is a frequently used measure in time series prediction and tracking control. Here, LGP provides a competitive control performance compared to GPR while being superior to LWPR and the state-of-the art rigid-body model.

Figure 3 (b) shows the tracking error after online learning with LGP in comparison with offline learned models. It can be seen that the errors are significantly reduced for LGP with online updates when compared to both standard GPR and LGP with offline learned models. During online-learning, the local GP models are adapted as new data points arrive. Since the number of training examples in each local model is limited, the update procedure is sufficiently fast for real-time application. For doing so, we employ the joint torques $\mathbf{u}$ and the resulting robot trajectories $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]$ as samples which are added to the LGP models online as described in Section 2.2. New data points are added to the local models until these fill up and, once full, new points replace previously existing data points. The insertion of new data point is performed with information gain [M.Seeger(2005)] while for the deletion we randomly take an old point from the corresponding local model. A new data point is inserted to the local model, if its information gain is larger than a given threshold value. In practice, this value is set such that the model update procedure can be maintained in real-time (the larger the information gain threshold, the more updates will be performed).

3.3 Performance on a Complex Test Setting

In this section, we create a more complex test case for tracking with inverse dynamics models where the trajectories are acquired by kinesthetic teach-in, i.e., we take the Barrett WAM by the end-effector and guide it along several trajectories which are subsequently used both in learning and control experiments. In order to make these trajectories straightforward to understand for humans, we draw all 26 characters of the alphabet in an imaginary plane in task space. An illustration for this data generation process is shown in Figure 4. During the imagined writing, the joint trajectories are sampled from the robot. Afterwards, it will attempt to reproduce that trajectory, and the reproductions can be used to generate training data. Subsequently, we used several characters as training examples (e.g., characters from **D** to **O**) and others, e.g., **A**, as test examples. This setup results in a data set with 10845 samples for training and 1599 for testing.

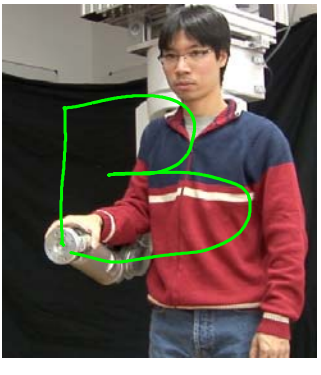
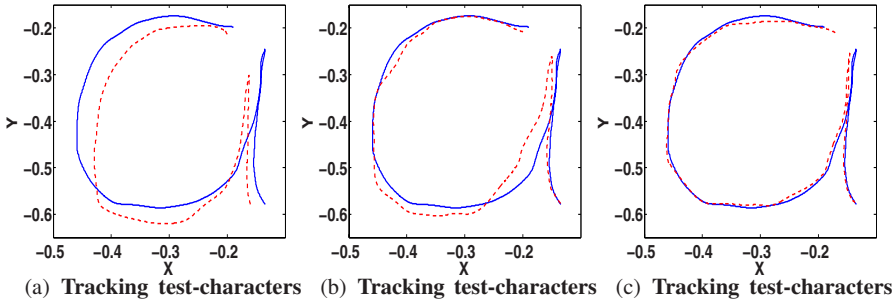


Fig. 4. The figure illustrates the data generation for the learning task.



(a) **Tracking test-characters using rigid-body model**

(b) **Tracking test-characters using offline-learned GP model**

(c) **Tracking test-characters after online-learning with LGP**

Fig. 5. Compliant tracking performance on Barrett WAM for the test character **A**, where the controlled trajectory lies in joint-space while our visualization is in task space for improved comprehensibility. We compare the corresponding rigid body model, an offline trained GP model and an online learning LGP. The thick, blue line denotes the desired trajectory, while the dashed, red line represents the robot trajectory during the compliant tracking task. The results indicate that online learning with LGP outperforms the offline-learned model using full GPR as well as the rigid-body dynamics.

Similar as in Section 3.1, we learn the inverse dynamics models using joint trajectories as input and joint torques as targets. The robot arm is then controlled to perform the joint-space trajectories corresponding to the test characters using the learned models. For LGP, we additionally show that the test characters can be learned online by updating the local models, as described in Section 3.2. The Figure 5 shows the tracking results using online-learning with LGP in comparison to the offline trained model with standard GPR and a traditional rigid body model.

It can be observed that the offline trained models (using standard GPR) can generalize well to unknown characters often having a better tracking performance than the rigid-body model. However, the results can be improved even further if the dynamics model is updated online – as done by LGP. The LGP results are shown in Figure 5 and are achieved after three trials on the test character.

4 Conclusion

The local Gaussian process regression LGP combines the strength of fast computation as in local regression with the potentially more accurate kernel regression methods. As a result, we obtain a real-time capable regression method which is relatively easy to tune and works well in robot application. When compared to locally linear methods such as LWPR, the LGP achieves higher learning accuracy while having less computational cost compared to state of the art kernel regression methods such as GPR and ν-SVR. The reduced complexity allows the application of the LGP for online model learning which is necessary for realtime adaptation of model errors or changes in the system. Model-based tracking control using online learned LGP

models achieves a superior control performance for low gain control in comparison to rigid body models as well as to offline learned models.

Future research will focus on several important extensions such as finding kernels which are most appropriate for clustering and prediction, and how the choice of a similarity can affect the LGP performance. Partitioning in higher dimension space is still a challenging problem, a possible solution is to perform dimensionality reduction during the partitioning step. Furthermore, alternative criteria for insertion and deletion of data points need to be examined more closely. This operation is crucial for online learning as not every new data point is informative for the current prediction task, and on the other hand deleting an old but informative data point may degrade the performance. It also interesting to investigate further applications of the LGP in humanoid robotics with 35 of more DoFs and learning other types of the control such as operational space control.

References

[Chang and Lin(2001)] Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm

[Craig(2004)] Craig, J.J.: Introduction to Robotics: Mechanics and Control, 3rd edn. Prentice Hall, Englewood Cliffs (2004)

[Csato and Opper(2002)] Csato, L., Opper, M.: Sparse online gaussian processes. Neural Computation (2002)

[Fumagalli et al(2010)Fumagalli, Gijsberts, Ivaldi, Jamone, Metta, Natale, Nori, and Sandini] Fumagalli, M., Gijsberts, A., Ivaldi, S., Jamone, L., Metta, G., Natale, L., Nori, F., Sandini, G.: Learning how to exploit proximal force sensing: a comparison approach. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 149–169. Springer, Heidelberg (2010)

[M.Seeger(2005)] MSeeger, Bayesian gaussian process models: Pac-bayesian generalisation error bounds and sparse approximations. PhD thesis, University of Edinburgh (2005)

[M.Seeger(2007)] MSeeger. Low rank update for the cholesky decomposition. Tech. rep., University of California at Berkeley (2007), http://www.kyb.tuebingen.mpg.de/bs/people/seeger/

[Nakanishi et al(2005)Nakanishi, Farrell, and Schaal] Nakanishi, J., Farrell, J.A., Schaal, S.: Composite adaptive control with locally weighted statistical learning. Neural Networks (2005)

[Nguyen-Tuong et al(2008)Nguyen-Tuong, Peters, and Seeger] Nguyen-Tuong, D., Peters, J., Seeger, M.: Computed torque control with nonparametric regression models. In: Proceedings of the 2008 American Control Conference, ACC 2008 (2008)

[Rasmussen and Williams(2006)] Rasmussen, C.E., Williams, C.K.: Gaussian Processes for Machine Learning. MIT-Press, Massachusetts Institute of Technology (2006)

[Roberts et al(2010)Roberts, Moret, Zhang, and Tedrake] Roberts, J.W., Moret, L., Zhang, J., Tedrake, R.: Motor Learning at Intermediate Reynolds Number: Experiments with Policy Gradient on the Flapping Flight of a RigidWing. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 293–309. Springer, Heidelberg (2010)

[Schaal et al(2000)Schaal, Atkeson, and Vijayakumar] Schaal, S., Atkeson, C.G., Vijayakumar, S.: Real-time robot learning with locally weighted statistical learning. In: International Conference on Robotics and Automation (2000)

[Schaal et al(2002)Schaal, Atkeson, and Vijayakumar] Schaal, S., Atkeson, C.G., Vijayakumar, S.: Scalable techniques from nonparameteric statistics for real-time robot learning. In: Applied Intelligence, pp. 49–60 (2002)

[Schölkopf and Smola (2002)] Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. MIT Press, Cambridge (2002)

[Seeger(2004)] Seeger, M.: Gaussian processes for machine learning. International Journal of Neural Systems (2004)

[Seeger(2007)] Seeger, M.: LHOTSE: Toolbox for Adaptive Statistical Model (2007), `http://www.kyb.tuebingen.mpg.de/bs/people/seeger/lhotse/`

[Snelson and Ghahramani(2007)] Snelson, E., Ghahramani, Z.: Local and global sparse gaussian process approximations. Artificial Intelligence and Statistics (2007)

[Spong et al(2006)Spong, Hutchinson, and Vidyasagar] Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot Dynamics and Control. John Wiley and Sons, New York (2006)

[Vijayakumar et al(2005)Vijayakumar, D'Souza, and Schaal] Vijayakumar, S., D'Souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Computation (2005)

Imitation and Reinforcement Learning for Motor Primitives with Perceptual Coupling

Jens Kober, Betty Mohler, and Jan Peters

Abstract. Traditional motor primitive approaches deal largely with open-loop policies which can only deal with small perturbations. In this paper, we present a new type of motor primitive policies which serve as closed-loop policies together with an appropriate learning algorithm. Our new motor primitives are an augmented version version of the dynamical system-based motor primitives [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal] that incorporates perceptual coupling to external variables. We show that these motor primitives can perform complex tasks such as Ball-in-a-Cup or Kendama task even with large variances in the initial conditions where a skilled human player would be challenged. We initialize the open-loop policies by imitation learning and the perceptual coupling with a handcrafted solution. We first improve the open-loop policies and subsequently the perceptual coupling using a novel reinforcement learning method which is particularly well-suited for dynamical system-based motor primitives.

1 Introduction

The recent introduction of motor primitives based on dynamical systems [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal, Schaal et al(2003)Schaal, Peters, Nakanishi, and Ijspeert, Schaal et al (2007)Schaal, Mohajerian, and Ijspeert] have allowed both imitation learning and Reinforcement Learning to acquire new behaviors fast and reliably. Resulting successes have shown that it is possible to rapidly learn motor primitives for complex behaviors such as tennis swings [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal], T-ball batting [Peters and Schaal(2006)], drumming [Pongas et al(2005)Pongas, Billard, and Schaal], biped

Jens Kober, Betty Mohler, and Jan Peters
Max Planck Institute for Biological Cybernetics, Tübingen, Germany
e-mail: {kober,mohler,jrpeters}@tuebingen.mpg.de

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 209–225.
springerlink.com

locomotion [Schaal et al(2003)Schaal, Peters, Nakanishi, and Ijspeert, Nakanishi et al(2004b)Nakanishi, Morimoto, Endo, Cheng, Schaal, and Kawato] and even in tasks with potential industrial application [Urbanek et al(2004)Urbanek, Albu-Schäffer, and van der Smagt]. However, in their current form these motor primitives are generated in such a way that they are either only coupled to internal variables [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal] or only include manually tuned phase-locking, e.g., with an external beat [Pongas et al(2005)Pongas, Billard, and Schaal] or between the gait-generating primitive and the contact time of the feet [Schaal et al(2003)Schaal, Peters, Nakanishi, and Ijspeert, Nakanishi et al(2004b)Nakanishi, Morimoto, Endo, Cheng, Schaal, and Kawato]. Furthermore, they incorporate the possibility to update parameters of a movement in real-time thus enabling perceptual coupling. E.g., changing the the goal of a movement can couple it to a target, i.e., an external variable. However, this perceptual coupling only is effective for the end of the movement and the rest of the movement is not coupled to the external variable. In many human motor control tasks, more complex perceptual coupling is needed in order to perform the task. Using handcrafted coupling based on human insight will in most cases no longer suffice. If changes of the internal variables constantly influences the behavior of the external variable more complex perceptual coupling is required as the coupling needs to incorporate knowledge of the behavior of the external variable. In this paper, it is our goal to augment the Ijspeert-Nakanishi-Schaal approach [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal] of using dynamical systems as motor primitives in such a way that it includes perceptual coupling with external variables. Similar to the biokinesiological literature on motor learning (see e.g., [Wulf(2007)]), we assume that there is an object of internal focus described by a state x and one of external focus y. The coupling between both foci usually depends on the phase of the movement and, sometimes, the coupling only exists in short phases, e.g., in a catching movement, this could be at initiation of the movement (which is largely predictive) and during the last moment when the object is close to the hand (which is largely prospective or reactive and includes movement correction). Often, it is also important that the internal focus is in a different space than the external one. Fast movements, such as a Tennis-swing, always follow a similar pattern in joint-space of the arm while the external focus is clearly on an object in Cartesian space or fovea-space. As a result, we have augmented the motor primitive framework in such a way that the coupling to the external, perceptual focus is phase-variant and both foci y and x can be in completely different spaces.

Integrating the perceptual coupling requires additional function approximation, and, as a result, the number of parameters of the motor primitives grows significantly. It becomes increasingly harder to manually tune these parameters to high performance and a learning approach for perceptual coupling is needed. The need for learning perceptual coupling in motor primitives has long been recognized in the motor primitive community [Schaal et al (2007)Schaal, Mohajerian, and Ijspeert]. However, learning perceptual coupling to an external variable is not as straightforward. It requires many trials in order to properly determine the connections from external to internal focus. It is straightforward to grasp a general movement by

imitation and a human can produce a Ball-in-a-Cup movement or a Tennis-swing after a single or few observed trials of a teacher but he will never have a robust coupling to the ball. Furthermore, small differences between the kinematics of teacher and student amplify in the perceptual coupling. This part is the reason why perceptually driven motor primitives can be initialized by imitation learning but will usually require self-improvement by reinforcement learning. This is analogous to the case of a human learning tennis: a teacher can show a forehand but a lot of self-practice is needed for a proper tennis game.

2 Augmented Motor Primitives with Perceptual Coupling

There are several frameworks for motor primitives used in robotics (e.g., [Kulic and Nakamura(2010)]). In this section, we first introduce the general idea behind dynamic system motor primitives as suggested in [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal] and, subsequently, show how perceptual coupling can be introduced. Subsequently, we show how the perceptual coupling can be realized by augmenting the acceleration-based framework from [Schaal et al (2007)Schaal, Mohajerian, and Ijspeert].

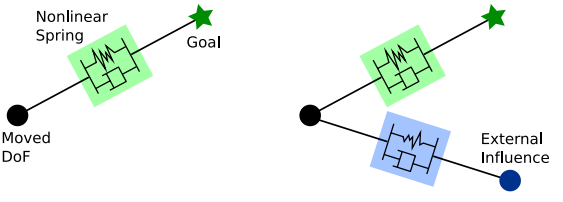


Fig. 1 Illustration of the behavior of the motor primitives (i) and the augmented motor primitives (ii).

2.1 Perceptual Coupling for Motor Primitives

The basic idea in the original work of Ijspeert, Nakanishi and Schaal [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal] is that motor primitives can be parted into two components, i.e., a canonical system h which drives transformed systems g_k for every considered degree of freedom k. As a result, we have a system of differential equations given by

$$\dot{z} = h(z), \tag{1}$$

$$\dot{\mathrm{x}} = \mathrm{g}(\mathrm{x}, z, \mathrm{w}), \tag{2}$$

which determines the variables of internal focus x (e.g., Cartesian or joint positions). Here, z denotes the state of the canonical system, which is indicates the

current phase of the movement, and w the internal parameters for transforming the output of the canonical system. The schematic in Figure 2 illustrates this traditional setup in black. In Section 2.2, we will discuss good choices for these dynamical systems as well as their coupling based on the most current formulation [Schaal et al (2007)Schaal, Mohajerian, and Ijspeert].

When taking an external variable y into account, there are three different ways how this variable influences the motor primitive system which one can consider, i.e., (i) it could only influence Eq.(1) which would be appropriate for synchronization problems and phase-locking (similar as in [Pongas et al(2005)Pongas, Billard, and Schaal, Nakanishi et al(2004a)Nakanishi, Morimoto, Endo, Cheng, Schaal, and Kawato]), (ii) only affect Eq.(2) which allows the continuous modification of the current state of the system by another variable and (iii) the combination of (i) and (ii). While (i) and (iii) are the right solution if phase-locking or synchronization are needed, the coupling in the canonical system will destroy many of the nice properties of the system and make it prohibitively hard to learn in practice. Furthermore, as we focus on discrete movements in this paper, we focus on the case (ii) which has not been used to date. In this case, we have a modified dynamical system

$$\dot{z} = h(z), \tag{3}$$

$$\dot{\mathrm{x}} = \hat{\mathrm{g}}(\mathrm{x}, \mathrm{y}, \bar{\mathrm{y}}, z, \mathrm{v}), \tag{4}$$

$$\dot{\bar{\mathrm{y}}} = \overline{\mathrm{g}}(\bar{\mathrm{y}}, z, \mathrm{w}), \tag{5}$$

where y denotes the state of the external variable, $\bar{\mathrm{y}}$ the expected state of the external variable and $\dot{\bar{\mathrm{y}}}$ its derivative. This architecture inherits most positive properties from the original work while allowing the incorporation of external feedback. We will show that we can incorporate previous work with ease and that the resulting framework resembles the one in [Schaal et al (2007)Schaal, Mohajerian, and Ijspeert] while allowing to couple the external variables into the system.

2.2 Realization for Discrete Movements

The original formulation in [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal] was a major breakthrough as the right choice of the dynamical systems in Equations (1, 2) allows determining the stability of the movement, choosing between a rhythmic and a discrete movement and is invariant under rescaling in both time and movement amplitude. With the right choice of function approximator (in our case locally-weighted regression), fast learning from a teachers presentation is possible. In this section, we first discuss how the most current formulation from the motor primitives as discussed in [Schaal et al (2007)Schaal, Mohajerian, and Ijspeert] is instantiated from Section 2.1. Subsequently, we show how it can be augmented in order to incorporate perceptual coupling.

While the original formulation in [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal] used a second-order

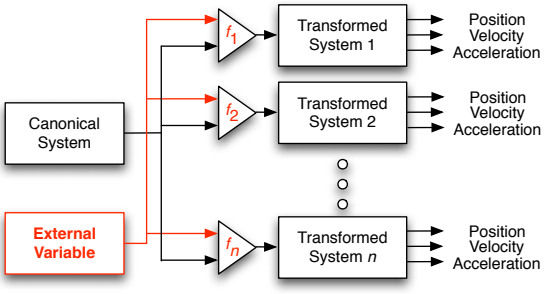


Fig. 2 General schematic illustrating both the original motor primitive framework by [Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal, Schaal et al (2007)Schaal, Mohajerian, and Ijspeert] in black and the augmentation for perceptual coupling in red.

canonical system, it has since then been shown that a single first order system suffices [Schaal et al (2007)Schaal, Mohajerian, and Ijspeert], i.e., we have

$$\dot{z} = h(z) = -\tau \alpha_h z,$$

which represents the phase of the trajectory. It has a time constant $\tau = \frac{1}{T}$ (where T is the movement duration) and a parameter α_h which is chosen such that $z \approx 0$ at T thus ensuring that the influence of the transformation function (8) vanishes. We can now choose our internal state such that position of degree of freedom k is given by $q_k = x_{2k}$, i.e., the $2k$-th component of x, the velocity by $\dot{q}_k = \tau x_{2k+1} = \dot{x}_{2k}$ and the acceleration by $\ddot{q}_k = \tau \dot{x}_{2k+1}$. Upon these assumptions, we can express the motor primitives function g in the following form

$$\dot{x}_{2k+1} = \tau \alpha_g \left(\beta_g \left(t_k - x_{2k} \right) - x_{2k+1} \right) + \tau \left(\left(t_k - x_{2k}^0 \right) + a_k \right) f_k, \tag{6}$$

$$\dot{x}_{2k} = \tau x_{2k+1}. \tag{7}$$

This function has the same time constant τ as the canonical system, parameters α_g, β_g set such that the system is critically damped, a goal parameter t_k corresponding to the final position of x_{2k}, the initial position x_{2k}^0, an amplitude modifier a_k which can be set arbitrarily, and a transformation function f_k. This transformation function transforms the output of the canonical system so that the transformed system can represent complex nonlinear patterns and is given by

$$f_k(z) = \sum_{i=1}^{N} \psi_i(z) w_i z, \tag{8}$$

where w are adjustable parameters and uses normalized Gaussian kernels without scaling such as

$$\psi_i = \frac{\exp\left(-h_i \left(z - c_i \right)^2 \right)}{\sum_{j=1}^{N} \exp\left(-h_j \left(z - c_j \right)^2 \right)} \tag{9}$$

for localizing the interaction in phase space where we have centers c_i and width h_i.

In order to learn a motor primitive with perceptual coupling, we need two components. First, we need to learn the normal or average behavior $\bar{\mathrm{y}}$ of the variable of external focus y (e.g., the relative positions of an object) which can be represented by a single motor primitive $\bar{\mathrm{g}}$, i.e., we can use the same type of function from Equations (2, 5) for $\bar{\mathrm{g}}$ which are learned based on the same z and given by Equations (6, 7). Additionally, we have the system $\hat{\mathrm{g}}$ for the variable of internal focus x which determines our actual movements which incorporates the inputs of the normal behavior $\bar{\mathrm{y}}$ as well as the current state y of the external variable. We obtain the system $\hat{\mathrm{g}}$ by inserting a modified coupling function $\hat{\mathrm{f}}(z, \mathrm{y}, \bar{\mathrm{y}})$ instead of the original $\mathrm{f}(z)$ in g. Function $\mathrm{f}(z)$ is modified in order to include perceptual coupling to y and we obtain

$$\hat{f}_k(z, \mathrm{y}, \bar{\mathrm{y}}) = \sum_{i=1}^{N} \psi_i(z) \hat{w}_i z + \sum_{j=1}^{M} \hat{\psi}_j(z) \left(\kappa_{jk}^{\mathrm{T}}(\mathrm{y} - \bar{\mathrm{y}}) + \delta_{jk}^{\mathrm{T}}(\dot{\mathrm{y}} - \dot{\bar{\mathrm{y}}}) \right), \tag{10}$$

where $\hat{\psi}_j(z)$ denote Gaussian kernels as in Equation (9) with centers $\hat{c}_j$ and width $\hat{h}_j$. Note, that it can be useful to set $N > M$ for reducing the number of parameters. All parameters are given by $\mathrm{v} = [\hat{\mathrm{w}}, \kappa, \delta]$. Here, $\hat{\mathrm{w}}$ are just the standard transformation parameters while κ_{jk} and δ_{jk} are the local coupling factors which can be interpreted as gains acting on the difference between the desired behavior of the external variable and its actual behavior. Note that for noise-free behavior and perfect initial positions, such coupling would never play a role; thus, the approach would simplify to the original approach. However, in the noisy, imperfect case, this perceptual coupling can ensure success even in extreme cases.

3 Learning for Perceptually Coupled Motor Primitives

While the transformation function $f_k(z)$ (8) can be learned from few or even just a single trial, this simplicity no longer transfers to learning the new function $\hat{f}_k(z, \mathrm{y}, \bar{\mathrm{y}})$ (10) as perceptual coupling requires that the coupling to an uncertain external variable is learned. While imitation learning approaches are feasible, they require larger numbers of presentations of a teacher with very similar kinematics for learning the behavior sufficiently well. As an alternative, we could follow "Nature as our teacher", and create a concerted approach of imitation and self-improvement by trial-and-error. For doing so, we first have a teacher who presents several trials and, subsequently, we improve our behavior by reinforcement learning.

3.1 Imitation Learning with Perceptual Coupling

Imitation learning is applied to a large number of problems in robotics (e.g., [Howard et al(2009b)Howard, Klanke, Gienger, Goerick, and Vijayakumar, Howard et al (2010) Howard, Klanke, Gienger, Goerick, and Vijayakumar, Ratliff et al(2009)Ratliff, Silver, and Bagnell]). Here we can largely follow the original work

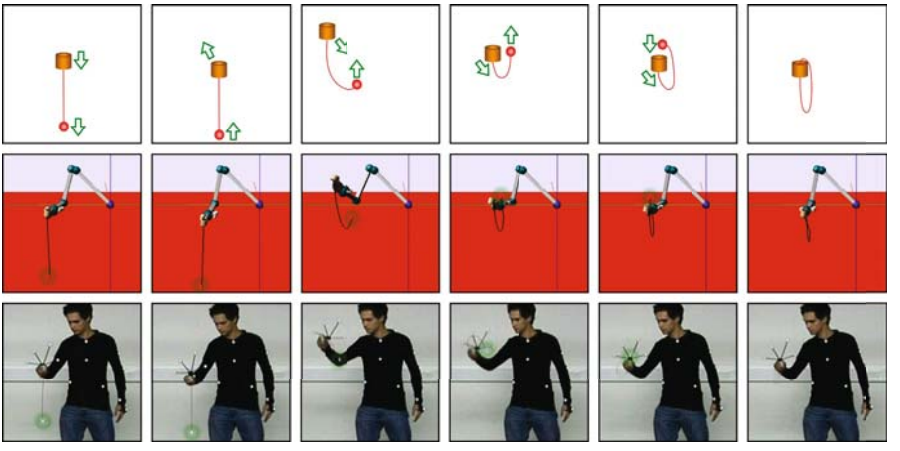


Fig. 3 This figure shows schematic drawings of the Ball-in-a-Cup motion, the final learned robot motion as well as a motion-captured human motion. The green arrows show the directions of the momentary movements. The human cup motion was taught to the robot by imitation learning with 91 parameters for 1.5 seconds. Please also refer to the video on the first author's website.

in [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal, Schaal et al (2007)Schaal, Mohajerian, and Ijspeert] and only need minor modifications. We also make use of locally-weighted regression in order to determine the optimal motor primitives, use the same weighting and compute the targets based on the dynamical systems. However, unlike in [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal, Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal], we need a bootstrapping step as we determine first the parameters for the system described by Equation (5) and, subsequently, use the learned results in the learning of the system in Equation (4). These steps can be performed efficiently in the context of dynamical systems motor primitives as the transformation functions (8) of Equations (4) and (5) are linear in parameters. As a result, we can choose the weighted squared error

$$\varepsilon_m^2 = \sum_{i=1}^{n} \psi_i^m \left(f_i^{\text{ref}} - \mathrm{z}_i^{\mathrm{T}} \mathrm{w}^m \right)^2 \tag{11}$$

as cost function and minimize it for all parameter vectors w^m with $m \in \{1, 2, \ldots, M\}$. Here, the corresponding weighting function are denoted by ψ_i^m and the basis functions by $\mathrm{z}_i^{\mathrm{T}}$. The reference or target signal f_i^{ref} is the desired transformation function and $i \in \{1, 2, \ldots, n\}$ indicates the number of the sample. The error in Equation (11) can be rewritten as

$$\varepsilon_m^2 = \left(\mathrm{f}^{\text{ref}} - \mathrm{Z}\mathrm{w}^m \right)^{\mathrm{T}} \Psi \left(\mathrm{f}^{\text{ref}} - \mathrm{Z}\mathrm{w}^m \right) \tag{12}$$

with $\mathrm{f}^{\mathrm{ref}}$ giving the value of f_i^{ref} for all samples i, $\Psi = \mathrm{diag}\left(\psi_i^m, \ldots, \psi_n^m\right)$ and $\mathrm{Z}_i = \mathrm{z}_i^{\mathrm{T}}$. As a result, we have a standard locally-weighted linear regression problem that can be solved straightforwardly and yields the unbiased estimator

$$\mathrm{w}^m = \left(\mathrm{Z}^{\mathrm{T}}\Psi\mathrm{Z}\right)^{-1}\mathrm{Z}^{\mathrm{T}}\Psi\mathrm{f}^{\mathrm{ref}}. \tag{13}$$

This general approach has originally been suggested in [Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal]. Estimating the parameters of the dynamical system is slightly more daunting, i.e., the movement duration is extracted using motion detection (velocities are zero at the start and at the end) and the time-constant is set accordingly.

This local regression yields good values for the parameters of $f_k(z)$. Subsequently, we can perform the exact same step for $\hat{f}_k(z, \mathrm{y}, \bar{\mathrm{y}})$ where only the number of variables has increased but the resulting regression follows analogously. However, note that while a single demonstration suffices for the parameter vector w and $\hat{\mathrm{w}}$, the parameters κ and δ cannot be learned by imitation as these require deviation from the nominal behavior for the external variable.

However, as discussed before, pure imitation for perceptual coupling can be difficult for learning the coupling parameters as well as the best nominal behavior for a robot with kinematics different from the human, many different initial conditions and in the presence of significant noise. Thus, we suggest to improve the policy by trial-and-error using reinforcement learning upon an initial imitation.

3.2 Reinforcement Learning for Perceptually Coupled Motor Primitives

Reinforcement learning [Sutton and Barto(1998)] is widely used in robotics (e.g., [Riedmiller et al(2009)Riedmiller, Gabel, Hafner, and Lange]) but reinforcement learning of discrete motor primitives is a very specific type of learning problem where it is hard to apply generic reinforcement learning algorithms [Peters and Schaal(2006), Peters and Schaal(2007)]. For this reason, the focus of this paper is largely on domain-appropriate reinforcement learning algorithms which operate on parametrized policies for episodic control problems.

3.2.1 Reinforcement Learning Setup

When modeling our problem as a reinforcement learning problem, we always have a state $\mathrm{s} = [z, \mathrm{y}, \bar{\mathrm{y}}, \mathrm{x}]$ with high dimensions (as a result, standard RL methods which discretize the state-space can no longer be applied), and the action $\mathrm{a} = [\mathrm{f}(z) + \varepsilon, \hat{\mathrm{f}}(z, \mathrm{y}, \bar{\mathrm{y}}) + \hat{\varepsilon}]$ is the output of our motor primitives. Here, the exploration is denoted by ε and $\hat{\varepsilon}$, and we can give a stochastic policy $\mathrm{a} \sim \pi(\mathrm{s})$ as distribution over the states with parameters $\theta = [\mathrm{w}, \mathrm{v}] \in \mathbb{R}^n$. After a next time-step δt, the actor transfers to a state s_{t+1} and receives a reward r_t. As we are interested in learning complex motor tasks consisting of a single stroke [Wulf(2007), Schaal et al

(2007)Schaal, Mohajerian, and Ijspeert], we focus on finite horizons of length T with episodic restarts [Sutton and Barto(1998)] and learn the optimal parametrized policy for such problems. The general goal in reinforcement learning is to optimize the *expected return* of the policy with parameters θ defined by

$$J(\theta) = \int_{\mathbb{T}} p(\tau) R(\tau) d\tau, \tag{14}$$

where $\tau = [\mathrm{s}_{1:T+1}, \mathrm{a}_{1:T}]$ denotes a sequence of states $\mathrm{s}_{1:T+1} = [\mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_{T+1}]$ and actions $\mathrm{a}_{1:T} = [\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_T]$, the probability of an episode τ is denoted by $p(\tau)$ and $R(\tau)$ refers to the return of an episode τ. Using Markov assumption, we can write the path distribution as $p(\tau) = p(\mathrm{x}_1) \prod_{t=1}^{T+1} p(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t) \pi(\mathrm{a}_t|\mathrm{s}_t, t)$ where $p(\mathrm{s}_1)$ denotes the initial state distribution and $p(\mathrm{s}_{t+1}|\mathrm{s}_t, \mathrm{a}_t)$ is the next state distribution conditioned on last state and action. Similarly, if we assume additive, accumulated rewards, the return of a path is given by $R(\tau) = \frac{1}{T} \sum_{t=1}^{T} r(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1}, t)$, where $r(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1}, t)$ denotes the immediate reward.

While episodic Reinforcement Learning (RL) problems with finite horizons are common in motor control, few methods exist in the RL literature (c.f., model-free method such as Episodic REINFORCE [Williams(1992)] and the Episodic Natural Actor-Critic eNAC [Peters and Schaal(2006)] as well as model-based methods, e.g., using differential-dynamic programming [Atkeson(1994)]). In order to avoid learning of complex models, we focus on model-free methods and, to reduce the number of open parameters, we rather use a novel Reinforcement Learning algorithm which is based on expectation-maximization. Our new algorithm is called Policy learning by Weighting Exploration with the Returns (PoWER) and can be derived from the same higher principle as previous policy gradient approaches, see [Kober and Peters(2008)] for details.

3.2.2 Policy Learning by Weighting Exploration with the Returns (PoWER)

When learning motor primitives, we intend to learn a deterministic mean policy $\bar{\mathrm{a}} = \theta^{\mathrm{T}} \mu(\mathrm{s}) = \mathrm{f}(\mathrm{z})$ which is linear in parameters θ and augmented by additive exploration $\varepsilon(\mathrm{s}, t)$ in order to make model-free reinforcement learning possible. As a result, the explorative policy can be given in the form $\mathrm{a} = \theta^{\mathrm{T}} \mu(\mathrm{s}, t) + \varepsilon(\mu(\mathrm{s}, t))$. Previous work in [Peters and Schaal(2006), Peters and Schaal(2007)], with the notable exception of [Rückstieß et al(2008)Rückstieß, Felder, and Schmidhuber], has focused on state-independent, white Gaussian exploration, i.e., $\varepsilon(\mu(\mathrm{s}, t)) \sim \mathcal{N}(0, \Sigma)$, and has resulted into applications such as T-Ball batting [Peters and Schaal(2006)] and constrained movement [Guenter et al(2007)Guenter, Hersch, Calinon, and Billard]. However, from our experience, such unstructured exploration at every step has several disadvantages, i.e., (i) it causes a large variance in parameter updates which grows with the number of time-steps, (ii) it perturbs actions too frequently, as the system acts as a low pass filter the perturbations average out and thus, their effects are 'washed' out and (iii) can damage the system executing the trajectory.

Alternatively, as introduced by [Rückstieß et al(2008)Rückstieß, Felder, and Schmidhuber], one could generate a form of structured, state-dependent exploration $\varepsilon(\mu(\mathrm{s},t)) = \varepsilon_t^{\mathrm{T}}\mu(\mathrm{s},t)$ with $[\varepsilon_t]_{ij} \sim \mathcal{N}(0,\sigma_{ij}^2)$, where σ_{ij}^2 are meta-parameters of the exploration that can be optimized in a similar manner. Each σ_{ij}^2 corresponds to one θ_{ij}. This argument results into the policy $\mathrm{a} \sim \pi(\mathrm{a}_t|\mathrm{s}_t,t) = \mathcal{N}(\mathrm{a}|\mu(\mathrm{s},\mathrm{t}),\hat{\Sigma}(\mathrm{s},t))$. This form of policies improves upon the shortcomings of directly perturbed policies mentioned above. Based on the EM updates for Reinforcement Learning as suggested in [Peters and Schaal(2007),Kober and Peters(2008)], we can derive the update rule

$$\theta' = \theta + \frac{E_\tau\left\{\sum_{t=1}^{T}\varepsilon_t Q^\pi(\mathrm{s}_t,\mathrm{a}_t,t)\right\}}{E_\tau\left\{\sum_{t=1}^{T}Q^\pi(\mathrm{s}_t,\mathrm{a}_t,t)\right\}}, \tag{15}$$

where

$$Q^\pi(\mathrm{s},\mathrm{a},t) = E\left\{\sum_{\tilde{t}=t}^{T} r(\mathrm{s}_{\tilde{t}},\mathrm{a}_{\tilde{t}},\mathrm{s}_{\tilde{t}+1},\tilde{t})\,|\mathrm{s}_t = \mathrm{s},\mathrm{a}_t = \mathrm{a}\right\}$$

is the state-action value function. Note that this algorithm does not need the learning rate as a meta-parameter.

In order to reduce the number of trials in this on-policy scenario, we reuse the trials through importance sampling [Andrieu et al(2003)Andrieu, de Freitas, Doucet, and Jordan,Sutton and Barto(1998)]. To avoid the fragility sometimes resulting from importance sampling in reinforcement learning, samples with very small importance weights are discarded.

The more shape parameters w are used the more details can be captured in a motor primitive and it can ease the imitation learning process. However, if the motor primitives need to be refined by RL, each additional parameter slows down the learning process The parameters σ_{ij}^2 determine the exploration behavior where larger values lead to greater changes in the mean policy and, thus, may lead to faster convergence but can also drive the robot in unsafe regimes. The optimization of the parameters decreases the exploration during convergence.

4 Evaluation and Application

In this section, we demonstrate the effectiveness of the augmented framework for perceptually coupled motor primitives as presented in Section 2 and show that our concerted approach of using imitation for initialization and reinforcement learning for improvement works well in practice, particularly with our novel PoWER algorithm from Section 3. We show that this method can be used in learning a complex, real-life motor learning problem Ball-in-a-Cup in a physically realistic simulation of an anthropomorphic robot arm. This problem is a good benchmark for testing the motor learning performance and we show that we can learn the problem roughly at the efficiency of a young child. This algorithm successfully creates a perceptual coupling even to perturbations that are very challenging for a skilled adult player.

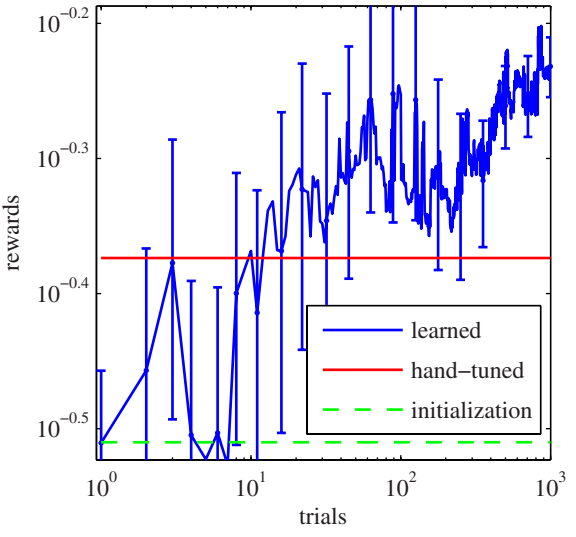


Fig. 4 This figure shows the expected return for one specific perturbation of the learned policy in the Ball-in-a-Cup scenario (averaged over 3 runs with different random seeds and the standard deviation indicated by the error bars). Convergence is not uniform as the algorithm is optimizing the returns for a whole range of perturbations and not for this test case. Thus, the variance in the return as the improved policy might get worse for the test case but improve over all cases. Our algorithm rapidly improves, regularly beating a hand-tuned solution after less than fifty trials and converging after approximately 600 trials. Note that this plot is a double logarithmic plot and, thus, single unit changes are significant as they correspond to orders of magnitude.

4.1 Robot Application: Ball-in-a-Cup

We have applied the presented algorithm in order to teach a physically-realistic simulation of an anthropomorphic SARCOS robot arm how to perform the traditional American children's game Ball-in-a-Cup, also known as Balero, Bilboquet or Kendama [Wikipedia(2008)]. The toy has a small cup which is held in one hand (or, in our case, is attached to the end-effector of the robot) and the cup has a small ball hanging down on a string (the string has a length of 40cm for our toy). Initially, the ball is hanging down vertically in a rest position. The player needs to move fast in order to induce a motion in the ball through the string, toss it up and catch it with the cup, a possible movement is illustrated in Figure 3 in the top row.

Note that learning Ball-in-a-Cup and Kendama have previously been studied in robotics and we are going to contrast a few of the approaches here. While we learn directly in the joint space of the robot, Takenaka et al. [Takenaka(1984)] recorded planar human cup movements and determined the required joint movements for a planar, three degree of freedom (DoF) robot so that it could follow the trajectories while visual feedback was used for error compensation. Both Sato

et al. [Sato et al(1993)Sato, Sakaguchi, Masutani, and Miyazaki] and Shone [Shone et al(2000)Shone, Krudysz, and Brown] used motion planning approaches which relied on very accurate models of the ball while employing only one DoF in [Shone et al(2000)Shone, Krudysz, and Brown] or two DoF in [Sato et al(1993)Sato, Sakaguchi, Masutani, and Miyazaki] so that the complete state-space could be searched exhaustively. Interestingly, exploratory robot moves were used in [Sato et al(1993)Sato, Sakaguchi, Masutani, and Miyazaki] to estimate the parameters of the employed model. The probably most advanced preceding work on learning Kendama was done by Miyamoto [Miyamoto et al(1996)Miyamoto, Schaal, Gandolfo, Gomi, Koike, Osu, Nakano, Wada, and Kawato] who used a seven DoF anthropomorphic arm and recorded human motions to train a neural network to reconstruct via-points. Employing full kinematic knowledge, the authors optimize a desired trajectory. We previously learned a policy without perceptual coupling on a real seven DoF anthropomorphic Barrett WAM$^{\text{TM}}$ [Kober and Peters(2008)] developing the method used below to get the initial success.

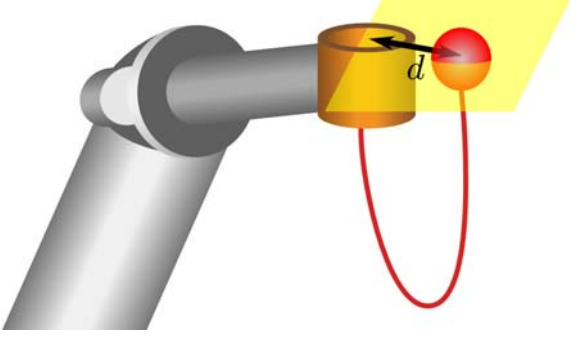


Fig. 5 This figure illustrates how the reward is calculated. The plane represents the level of the upper rim of the cup. For a successful rollout the ball has to be moved above the cup first. The reward is then calculated as the distance of the center of the cup and the center of the ball on the plane at the moment the ball is passing the plane in a downward direction.

The state of the system is described in Cartesian coordinates of the cup (i.e., the operational space) and the Cartesian coordinates of the ball. The actions are the cup accelerations in Cartesian coordinates with each direction represented by a motor primitive. An operational space control law [Nakanishi et al(2007)Nakanishi, Mistry, Peters, and Schaal] is used in order to transform accelerations in the operational space of the cup into joint-space torques. All motor primitives are perturbed separately but employ the same joint reward which is $r_t = \exp(-\alpha(x_c - x_b)^2 - \alpha(y_c - y_b)^2)$ the moment where the ball passes the rim of the cup with a downward direction and $r_t = 0$ all other times (see Figure 5). The cup position is denoted by $[x_c, y_c, z_c] \in \mathbb{R}^3$, the ball position $[x_b, y_b, z_b] \in \mathbb{R}^3$ and a scaling parameter $\alpha = 10000$. The task is quite complex as the reward is not modified solely by the movements of the cup but foremost by the movements of the ball and the movements of the ball are

very sensitive to perturbations. A small perturbation of the initial condition or the trajectory will drastically change the movement of the ball and hence the outcome of the trial if we do not use any form of perceptual coupling to the external variable "ball".

Due to the complexity of the task, Ball-in-a-Cup is even a hard motor task for children who only succeed at it by observing another person playing or deducing from similar previously learned tasks how to maneuver the ball above the cup in such a way that it can be caught. Subsequently, a lot of improvement by trial-and-error is required until the desired solution can be achieved in practice. The child will have an initial success as the initial conditions and executed cup trajectory fit together by chance, afterwards the child still has to practice a lot until it is able to get the ball in the cup (almost) every time and so cancel various perturbations. Learning the necessary perceptual coupling to get the ball in the cup on a consistent basis is even a hard task for adults, as our whole lab can testify. In contrast to a tennis swing, where a human just needs to learn a goal function for the one moment the racket hits the ball, in Ball-in-a-Cup we need a complete dynamical system as cup and ball constantly interact. Mimicking how children learn to play Ball-in-a-Cup, we first initialize the motor primitives by imitation and, subsequently, improve them by reinforcement learning in order to get an initial success. Afterwards we also acquire the perceptual coupling by reinforcement learning.

We recorded the motions of a human player using a VICON^{TM} motion-capture setup in order to obtain an example for imitation as shown in Figure 3(c). The extracted cup-trajectories were used to initialize the motor primitives using locally-weighted regression for imitation learning. The simulation of the Ball-in-a-Cup behavior was verified using the tracked movements. We used one of the recorded trajectories for which, when played back in simulation, the ball goes in but does not pass the center of the opening of the cup and thus does not optimize the reward. This movement is then used for initializing the motor primitives and determining their parametric structure where cross-validation indicates that 91 parameters per motor primitive are optimal from a bias-variance point of view. The trajectories are optimized by reinforcement learning using the PoWER algorithm on the parameters w for non perturbed initial conditions. The robot constantly succeeds at bringing the ball into the cup after approximately 60-80 iterations given no noise and perfect initial conditions.

One set of the found trajectories is then used to calculate the baseline $\bar{y} = (h - b)$ and $\dot{\bar{y}} = (\dot{h} - \dot{b})$, where h and b are the hand and ball trajectories. This set is also used to set the standard cup trajectories.

Without perceptual coupling the robot misses for even tiny perturbations of the initial conditions. Hand tuned coupling factors work quite well for small perturbations. In order to make them more robust we use reinforcement learning using the same joint reward as before. The initial conditions (positions and velocities) of the ball are perturbed completely randomly (no PEGASUS Trick) using Gaussian random values with variances set according to the desired stability region. The PoWER algorithm converges after approximately 600-800 iterations. This is roughly comparable to the learning speed of a 10 year old child (Figure 4). For the training we

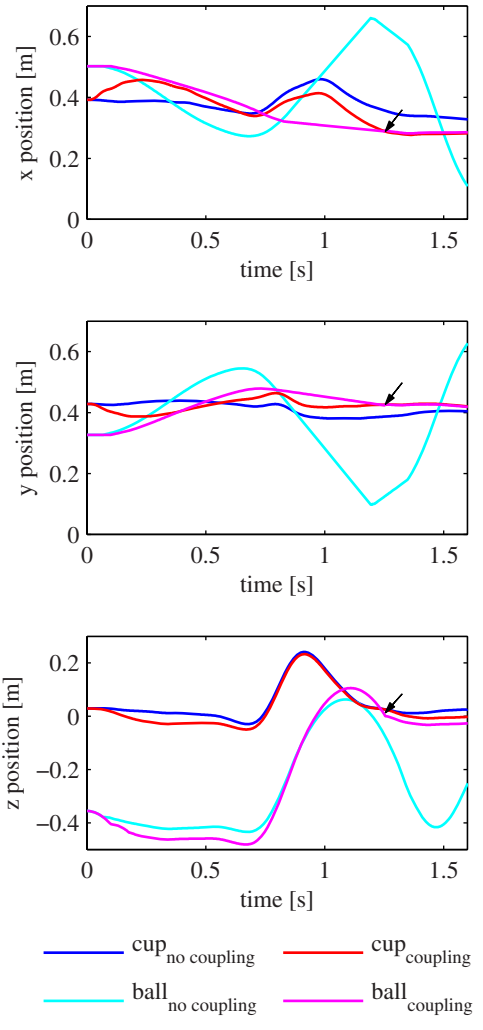


Fig. 6 This figure compares cup and ball trajectories with and without perceptual coupling. The trajectories and different initial conditions are clearly distinguishable. The perceptual coupling cancels the swinging motion of the string and ball "pendulum" out. The successful trial is marked by black arrows at the point where the ball enters the cup.

used concurrently standard deviations of 0.01m for x and y and of 0.1 m/s for $\dot{x}$ and $\dot{y}$. The learned perceptual coupling gets the ball in the cup for all tested cases where the hand-tuned coupling was also successful. The learned coupling pushes the limits of the canceled perturbations significantly further and still performs consistently well for double the standard deviations seen in the reinforcement learning process.

Figure 6 shows an example of how the visual coupling adapts the hand trajectories in order to cancel perturbations and to get the ball in the cup.

The coupling factors represent the actions to be taken in order to get back to the desired relative positions and velocities of the ball with respect to the hand. This corresponds to an implicit model of how cup movements affect the ball movements. The factors at the beginning of the motion are small as there is enough time to correct the errors later on. At the very end the hand is simply pulled directly under the ball so it can fall into the cup. The perceptual coupling is robust to small changes of the parameters of the toy (string length, ball weight). We also learned the coupling directly in joint-space in order to show, that the augmented motor primitives can handle perception and action in different spaces (perception in task space and action in joint space, for our evaluation). For each of the seven degrees of freedom a separate motor primitive is used, $\bar{y}$ and $\dot{\bar{y}}$ remain the same as before. Here we were not able to find good coupling factors by hand-tuning. Reinforcement learning finds working parameters but they do not perform as well as the Cartesian version. These effects can be explained by two factors: the learning task is harder as we have a higher dimensionality. Furthermore, we are learning the inverse kinematics of the robot implicitly. If the perturbations are large, the perceptual coupling has to do large corrections. These large corrections tend to move the robot in regions where the inverse kinematics differ from the ones for the mean motion and, thus, the learned implicit inverse kinematics no longer perform well. This behavior leads to even larger deviations and the effects accumulate.

5 Conclusion

Perceptual coupling for motor primitives is an important topic as it results in more general and more reliable solutions while it allows the application of the dynamical systems motor primitive framework to many other motor control problems. As manual tuning can only work in limited setups, an automatic acquisition of this perceptual coupling is essential.

In this paper, we have contributed an augmented version of the motor primitive framework originally suggested by [Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal,Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal,Schaal et al (2007)Schaal, Mohajerian, and Ijspeert] such that it incorporates perceptual coupling while keeping a distinctively similar structure to the original approach and, thus, preserving most of the important properties. We present a concerted learning approach which relies on an initialization by imitation learning and, subsequent, self-improvement by reinforcement learning. We introduce a particularly well-suited algorithm for this reinforcement learning problem called PoWER. The resulting framework works well for learning Ball-in-a-Cup on a simulated anthropomorphic SARCOS arm in setups where the original motor primitive framework would not suffice to fulfill the task.

References

[Andrieu et al(2003)Andrieu, de Freitas, Doucet, and Jordan] Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.I.: An introduction to MCMC for machine learning. Machine Learning 50(1), 5–43 (2003)

[Atkeson(1994)] Atkeson, C.G.: Using local trajectory optimizers to speed up global optimization in dynamic programming. In: Hanson, J.E., Moody, S.J., Lippmann, R.P. (eds.) Advances in Neural Information Processing Systems 6 (NIPS), pp. 503–521. Morgan Kaufmann, Denver (1994)

[Guenter et al(2007)Guenter, Hersch, Calinon, and Billard] Guenter, F., Hersch, M., Calinon, S., Billard, A.: Reinforcement learning for imitating constrained reaching movements. Advanced Robotics, Special Issue on Imitative Robots 21(13), 1521–1544 (2007)

[Howard et al (2010) Howard, Klanke, Gienger, Goerick, and Vijayakumar] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Methods for learning control policies from variable-constraint demonstrations. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 253–291. Springer, Heidelberg (2010)

[Howard et al(2009b)Howard, Klanke, Gienger, Goerick, and Vijayakumar] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: A novel method for learning policies from variable constraint data. Autonomous Robots (2009b)

[Ijspeert et al(2002)Ijspeert, Nakanishi, and Schaal] Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), Washington, DC, pp. 1398–1403 (2002)

[Ijspeert et al(2003)Ijspeert, Nakanishi, and Schaal] Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems 16 (NIPS), vol. 15, pp. 1547–1554. MIT Press, Cambridge (2003)

[Kober and Peters(2008)] Kober, J., Peters, J.: Policy search for motor primitives in robotics. In: Advances in Neural Information Processing Systems, NIPS (2008)

[Kulic and Nakamura(2010)] Kulic, D., Nakamura, Y.: Incremental learning of full body motion primitives. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 383–406. Springer, Heidelberg (2010)

[Miyamoto et al(1996)Miyamoto, Schaal, Gandolfo, Gomi, Koike, Osu, Nakano, Wada, and Kawato] Miyamoto, H., Schaal, S., Gandolfo, F., Gomi, H., Koike, Y., Osu, R., Nakano, E., Wada, Y., Kawato, M.: A kendama learning robot based on bi-directional theory. Neural Networks 9(8), 1281–1302 (1996)

[Nakanishi et al(2004a)Nakanishi, Morimoto, Endo, Cheng, Schaal, and Kawato] Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., Kawato, M.: A framework for learning biped locomotion with dynamic movement primitives. In: Proc. IEEE-RAS Int. Conf. on Humanoid Robots (HUMANOIDS), Santa Monica, CA, November 10-12. IEEE, Los Angeles (2004)

[Nakanishi et al(2004b)Nakanishi, Morimoto, Endo, Cheng, Schaal, and Kawato] Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., Kawato, M.: Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems (RAS) 47(2-3), 79–91 (2004)

[Nakanishi et al(2007)Nakanishi, Mistry, Peters, and Schaal] Nakanishi, J., Mistry, M., Peters, J., Schaal, S.: Experimental evaluation of task space position/orientation control towards compliant control for humanoid robots. In: Proc. IEEE/RSJ 2007 Int. Conf. on Intell. Robotics Systems, IROS (2007)

[Peters and Schaal(2006)] Peters, J., Schaal, S.: Policy gradient methods for robotics. In: Proc. IEEE/RSJ 2006 Int. Conf. on Intell. Robots and Systems (IROS), Beijing, China, pp. 2219–2225 (2006)

[Peters and Schaal(2007)] Peters, J., Schaal, S.: Reinforcement learning for operational space. In: Proc. Int. Conference on Robotics and Automation (ICRA), Rome, Italy (2007)

[Pongas et al(2005)Pongas, Billard, and Schaal] Pongas, D., Billard, A., Schaal, S.: Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In: Proc. IEEE 2005 Int. Conf. on Intell. Robots and Systems (IROS), vol. 2005, pp. 2911–2916 (2005)

[Ratliff et al(2009)Ratliff, Silver, and Bagnell] Ratliff, N., Silver, D., Bagnell, J.: Learning to search: Functional gradient techniques for imitation learning. Autonomous Robots 27(1), 25–53 (2009)

[Riedmiller et al(2009)Riedmiller, Gabel, Hafner, and Lange] Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement learning for robot soccer. Autonomous Robots 27(1), 55–73 (2009)

[Rückstieß et al(2008)Rückstieß, Felder, and Schmidhuber] Rückstieß, T., Felder, M., Schmidhuber, J.: State-dependent exploration for policy gradient methods. In: Proceedings of the European Conference on Machine Learning (ECML), pp. 234–249 (2008)

[Sato et al(1993)Sato, Sakaguchi, Masutani, and Miyazaki] Sato, S., Sakaguchi, T., Masutani, Y., Miyazaki, F.: Mastering of a task with interaction between a robot and its environment: "kendama" task. Transactions of the Japan Society of Mechanical Engineers C 59(558), 487–493 (1993)

[Schaal et al(2003)Schaal, Peters, Nakanishi, and Ijspeert] Schaal, S., Peters, J., Nakanishi, J., Ijspeert, A.J.: Control, planning, learning, and imitation with dynamic movement primitives. In: Proc. Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE 2003 Int. Conf. on Intell. Robots and Systems (IROS), Las Vegas, NV, October 27-31 (2003)

[Schaal et al (2007)Schaal, Mohajerian, and Ijspeert] Schaal, S., Mohajerian, P., Ijspeert, A.J.: Dynamics systems vs. optimal control — a unifying view. Progress in Brain Research 165(1), 425–445 (2007)

[Shone et al(2000)Shone, Krudysz, and Brown] Shone, T., Krudysz, G., Brown, K.: Dynamic manipulation of kendama. Tech. rep., Rensselaer Polytechnic Institute (2000)

[Sutton and Barto(1998)] Sutton, R., Barto, A.: Reinforcement Learning. MIT Press, Cambridge (1998)

[Takenaka(1984)] Takenaka, K.: Dynamical control of manipulator with vision: "cup and ball" game demonstrated by robot. Transactions of the Japan Society of Mechanical Engineers C 50(458), 2046–2053 (1984)

[Urbanek et al(2004)Urbanek, Albu-Schäffer, and van der Smagt] Urbanek, H., Albu-Schäffer, A., van der Smagt, P.: Learning from demonstration repetitive movements for autonomous service robotics. In: Proc. IEEE/RSL 2004 Int. Conf. on Intell. Robots and Systems (IROS), Sendai, Japan, pp. 3495–3500 (2004)

[Wikipedia(2008)] Wikipedia (2008), http://en.wikipedia.org/wiki/Ball_in_a_cup

[Williams(1992)] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8, 229–256 (1992)

[Wulf(2007)] Wulf, G.: Attention and motor skill learning. Human Kinetics, Urbana Champaign (2007)

A Bayesian View on Motor Control and Planning

Marc Toussaint and Christian Goerick

Abstract. The problem of motion control and planning can be formulated as an optimization problem. In this paper we discuss an alternative view that casts the problem as one of probabilistic inference. In simple cases where the optimization problem can be solved analytically the inference view leads to equivalent solutions. However, when approximate methods are necessary to tackle the problem, the tight relation between optimization and probabilistic inference has fruitfully lead to a transfer of methods between both fields. Here we show that such a transfer is also possible in the realm of robotics. The general idea is that motion can be generated by fusing motion objectives (task constraints, goals, motion priors) by using probabilistic inference techniques. In realistic scenarios exact inference is infeasible (as is the analytic solution of the corresponding optimization problem) and the use of efficient approximate inference methods is a promising alternative to classical motion optimization methods. In this paper we first derive Bayesian control methods that are directly analogous to classical redundant motion rate control and optimal dynamic control (including operational space control). Then, by extending the probabilistic models to be Markovian models of the whole trajectory, we show that approximate probabilistic inference methods (message passing) efficiently compute solutions to trajectory optimization problems. Using Gaussian belief approximations and local linearization the algorithm becomes related to Differential Dynamic Programming (DDP) (aka. iterative Linear Quadratic Gaussian (iLQG)).

1 Introduction

Bayesian Networks and inference methods like message passing algorithms are a basic computational paradigm for information processing on coupled random

Marc Toussaint
Technical University Berlin, Franklinstr. 28/29, 10587 Berlin
e-mail: mtoussai@cs.tu-berlin.de

Christian Goerick
Honda Research Institute Europe, Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany
e-mail: christian.goerick@honda-ri.de

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 227–252.
springerlink.com

variables. Inference methods compute the posterior distribution over random variables when all couplings are taken into account (we will be more formal later). In this view, it is not surprising that there are strong relations between the fields of optimization and probabilistic inference: In the context of optimization, the coupling of variables is typically described by additive decomposable functions; which is in analogy to the factorization of a joint distribution as described by a Bayesian network or factor graph (with the typical identification of cost with neg-log probability). Consequently methods that originated in optimization can be translated to solve inference problems and vice versa: Message passing algorithms can be used to address satisfiability problems [3]; graph cut algorithms are used to address inference problems (MAP estimation) in Markov-Random-Fields [15]. The key of efficient methods is that local computations (e.g., local message passing equations) are used to achieve global coherence.[1]

Motion control and optimization are fundamental and very interesting problems in robotics. The problem can be formalized as an optimization problem: devising an appropriate cost function we can derive classical solutions (e.g., motion rate control, stochastic optimal control) which provide the basis of modern robot control [13]. Complementary to the optimization view on robot control, we can also address the problem from the point of view of probabilistic inference. The classical cost function is replaced by a joint distribution over coupled random variables (e.g., via a neg-log transform), and the classical solution methods are replaced by methods of probabilistic inference. In simple cases, in particular those where an exact solution can efficiently be computed in the optimization framework, the inference approach will only reproduce the same solution. As in the field of optimization, the transfer of methods becomes interesting when the optimization problem becomes hard and exact algorithms are computationally expensive. Inference methods like message passing algorithms are promising candidates to yield (approximate) solutions to the optimization problem.

The problem of motion control and planning typically involves "solving" a system of many coupled variables: transition or control costs couple the state variable in two consecutive time steps, task constraints couple a task variable with the state variable within a time slice. Classically, these couplings are implicit in the cost function. In the inference view, these couplings are explicitly formulated as conditional dependencies in a joint distribution. This view naturally extends to more complex and structured robotic systems where the state of the system is represented by a number of state variables instead of a single state variable. An example is hierarchical control, or decoupled (or weakly coupled) control problems, where we maintain separate state variables for the left and the right effector of a robot and their control and plans (posterior distributions) are only weakly coupled. In such cases we can use probabilistic inference methods that exploit structured (factored) representations of the problem. Generally, while Bayesian methods have become a standard tool for sensor processing and fusion, we investigate them to solve a fusion problem on the

[1] More formally, "coherence" could denote the marginal consistency in the context of inference, or the consistency with constraints in the context of optimization.

motor level, namely the problem of fusing motion objectives like task constraints, goals and motion priors into a posterior distribution over trajectories.

In this chapter we give an introduction to Bayesian motor control and planning, i.e., methods to compute motions *given* a (probabilistic) model. Although our focus is not directly on learning such models, the methods are interesting also from the point of view of "motor and interaction learning": For instance, in [22] we show how inference methods for model-based planning can be translated to model-free Reinforcement Learning algorithms using stochastic sampling methods. [2] show how probabilistic motion inference is useful in the context of imitation learning. Generally, in a model-based learning approach (where the behavior learning problem is decomposed in a first stage of learning a model and a second stage of using the model to generate behavior) one should always expect the learned model to be uncertain (see other chapters of this book, such as [14, 4]). The Bayesian methods we propose here naturally address such uncertainty. The Bayesian framework also motivates new interesting learning problems in the context of motion, for instance, learning motion priors from data.

This chapter is organized as follows. In the next section we first address the kinematic and dynamic control problem, derive Bayesian control equations and highlight the close relation to classical control equations. Section 3 introduces analogous probabilistic models that represent the motion planning problem. Approximate inference methods in these models yield new algorithms. When we approximate the system locally these new equations are related to the Ricatti equation of the Linear Quadratic Gaussian (LQG) case. In the general non-LQG case we need approximate inference methods to solve the planning problem, for which we derive local message update equations. Section 4 presents experiments that illustrate the methods. Additionally, we discuss hierarchical planning (where one alternates between planning in the task space and planning in the q-space, see also [7, 16]). This paper is an extension of the work presented earlier in [20]; see also the more theoretical discussion [18] of the relation between iLQG and inference in general stochastic optimal control scenarios.

2 A Bayesian View on Classical Control

2.1 Kinematic Case

We first address the case of kinematic control, i.e., the problem of deciding on the control signal given a desired constraint at the *next* time step. Throughout the derivation we will make use of identities for Gaussians which are summarized in the appendix. Let $q_t \in \mathbb{R}^n$ be a random variable referring to the robot's joint state at time t. And let x_t be a random variable referring to a task space of the robot (e.g. an endeffector state) at time t. Consider the joint probability distribution

$$P(x_t, q_t, q_{t\text{-}1}) = P(x_t \,|\, q_t)\; P(q_t \,|\, q_{t\text{-}1})\; P(q_{t\text{-}1}) \tag{1}$$

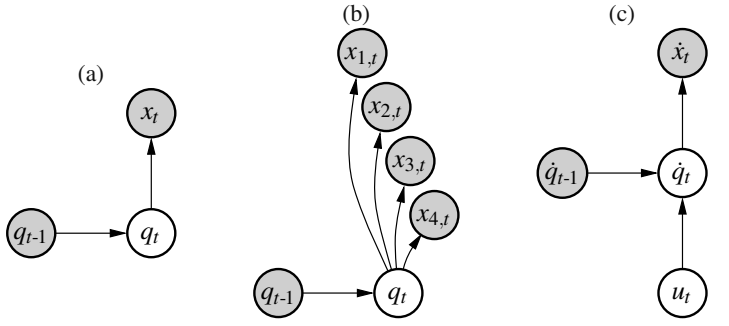


Fig. 1 Graphical models of Bayesian kinematic control, (a) for motion rate control, (b) under multiple task constraints, (c) for Bayesian dynamic control.

as also illustrated by the graphical model in Figure 1(a). Here, we call $P(q_t \,|\, q_{t\text{-}1})$ the *motion prior* and assume

$$P(q_t \,|\, q_{t\text{-}1}) = \mathcal{N}(q_t \,|\, q_{t\text{-}1} + h, W^{\text{-}1}) \;, \tag{2}$$

where $h \in \mathbb{R}^n$ is a vector that induces an *asymmetry in the motion prior* and W is the *motion metric* which enters this prior in terms of its covariance $W^{\text{-}1}$. Further, we call $P(x_t \,|\, q_t)$ the *task coupling* and assume

$$P(x_t \,|\, q_t) = \mathcal{N}(x_t \,|\, \phi(q_t), C) \;. \tag{3}$$

Here, ϕ is a non-linear function (the task kinematics) with Jacobian $J = \frac{\partial \phi}{\partial q}$ and C denotes the covariance in this coupling (inversely, $C^{\text{-}1}$ denotes the precision or tolerance of this coupling).

Given this model we can compute the posterior motion conditioned on a desired task constraint. That is, given x_t and $q_{t\text{-}1}$ we compute $P(q_t \,|\, x_t, q_{t\text{-}1})$. We can derive the following result.

Theorem 1. *Given equations (1-3), the posterior motion is*

$$P(q_t \,|\, x_t, q_{t\text{-}1}) \approx \mathcal{N}(q_t \,|\, q_t^{MAP}, (J^T C^{\text{-}1} J + W)^{\text{-}1}) \tag{4}$$

with the MAP motion

$$\begin{aligned} q_t^{MAP} &= q_{t\text{-}1} + (J^T C^{\text{-}1} J + W)^{\text{-}1} [J^T C^{\text{-}1} (x_t - \phi(q_{t\text{-}1})) + W h] && (5) \\ &= q_{t\text{-}1} + J^\sharp_{WC} (x_t - \phi(q_{t\text{-}1})) + (\mathbf{I}_n - J^\sharp_{WC} J)\, h \;, && (6) \\ J^\sharp_{WC} &:= W^{\text{-}1} J^T (J W^{\text{-}1} J^T + C)^{\text{-}1} \;. \end{aligned}$$

The approximation refers to the linearization of $\phi(q_t) \approx \phi(q_{t\text{-}1}) + J(q_t - q_{t\text{-}1})$.

Proof. We have

$$P(q_t \,|\, x_t, q_{t\text{-}1}) \propto P(x_t \,|\, q_t)\; P(q_t \,|\, q_{t\text{-}1}) = \mathcal{N}(x_t \,|\, \phi(q_t), C)\; \mathcal{N}(q_t \,|\, q_{t\text{-}1} + h, W^{\text{-}1})$$

Using the linearization $\phi(q_t) \approx \phi(q_{t\text{-}1}) + J(q_t - q_{t\text{-}1})$ we get

$$P(q_t \,|\, x_t, q_{t\text{-}1}) \approx \mathcal{N}(x_t \,|\, \phi(q_{t\text{-}1}) + J(q_t - q_{t\text{-}1}), C)\; \mathcal{N}(q_t \,|\, q_{t\text{-}1} + h, W^{\text{-}1})$$

Applying the Gaussian identities given in the appendix we have

$$\begin{aligned}
&P(q_t \,|\, x_t, q_{t\text{-}1}) \\
&\quad = \mathcal{N}(Jq_t \,|\, Jq_{t\text{-}1} + x_t - \phi(q_{t\text{-}1}), C)\; \mathcal{N}(q_t \,|\, q_{t\text{-}1} + h, W^{\text{-}1}) \\
&\quad \propto \mathcal{N}[q_t \,|\, J^T C^{\text{-}1} J q_{t\text{-}1} + J^T C^{\text{-}1}(x_t - \phi(q_{t\text{-}1})), J^T C^{\text{-}1} J]\, \mathcal{N}[q_t \,|\, W q_{t\text{-}1} + Wh, W] \\
&\quad \propto \mathcal{N}[q_t \,|\, (J^T C^{\text{-}1} J + W) q_{t\text{-}1} + J^T C^{\text{-}1}(x_t - \phi(q_{t\text{-}1})) - Wh, J^T C^{\text{-}1} J + W] \\
&\quad = \mathcal{N}(q_t \,|\, q_t^{\text{MAP}}, A) \\
&\qquad\qquad A = (J^T C^{\text{-}1} J + W)^{\text{-}1} \\
&\qquad\qquad q_t^{\text{MAP}} = q_{t\text{-}1} + A[J^T C^{\text{-}1}(x_t - \phi(q_{t\text{-}1})) + Wh]
\end{aligned}$$

To rewrite q_t^{MAP} we can use the Woodbury identity

$$(J^T C^{\text{-}1} J + W)^{\text{-}1} J^T C^{\text{-}1} = W^{\text{-}1} J^T (J W^{\text{-}1} J^T + C)^{\text{-}1} \tag{7}$$

and get

$$q_t^{\text{MAP}} = q_{t\text{-}1} + J_{WC}^{\sharp}(x_t - \phi(q_{t\text{-}1})) + (J^T C^{\text{-}1} J + W)^{\text{-}1} W\; h\;.$$

Further, using the identity

$$\begin{aligned}
&\mathbf{I}_n = (J^T C^{\text{-}1} J + W)^{\text{-}1} (J^T C^{\text{-}1} J + W) \\
&\Rightarrow (J^T C^{\text{-}1} J + W)^{\text{-}1} W = \mathbf{I}_n - (J^T C^{\text{-}1} J + W)^{\text{-}1} J^T C^{\text{-}1} J = \mathbf{I}_n - J_{WC}^{\sharp} J
\end{aligned} \tag{8}$$

we get

$$q_t^{\text{MAP}} = q_{t\text{-}1} + J_{WC}^{\sharp}(x_t - \phi(q_{t\text{-}1})) + (\mathbf{I}_n - J_{WC}^{\sharp} J)\; h\;. \qquad \blacksquare$$

The theorem gives two expressions (5) and (6) to compute q_t^{MAP}, related by the Woodbury identity. Note that the second expression (6) includes only d-dimensional matrix inversions rather than n-dimensional (neglecting $W^{\text{-}1}$, which can be precomputed). Thus, in practice the second expression will be more efficient to implement.

The second expression (6) also shows that the MAP motion q_t^{MAP} is very similar to classical kinematic motion rate control using the pseudo-inverse Jacobian [11] – in the tight constraint limit $C \to 0$, $J_{WC}^{\sharp}$ coincides with the standard pseudo inverse $J_W^{\sharp} = W^{\text{-}1} J^T (J W^{\text{-}1} J^T)^{\text{-}1}$ and the two are equivalent. For non-zero covariance C (i.e.,

Table 1 Correspondences between the classical and Bayesian view.

classical view	Bayesian view
metric W of the pseudo-inverse $J_W^\sharp = W^{-1}J^T(JW^{-1}J^T)^{-1}$	covariance W^{-1} of the motion prior $\mathcal{N}(q_{t+1} \mid q_t + h, W^{-1})$
nullspace motion $(\mathbf{I} - J_W^\sharp J)\, h$	asymmetry h of the motion prior $\mathcal{N}(q_{t+1} \mid q_t + h, W^{-1})$
regularizer in the singularity-robust $\tilde{J}_W^\sharp = W^{-1}J^T(JW^{-1}J^T + k\mathbf{I}_d)^{-1}$	covariance C of the task coupling $\mathcal{N}(x_t \mid \phi(q_t), C)$

when loosening the task constraint) the MAP motion q_t^{MAP} corresponds to classical control with regularization in the computation of the pseudo-inverse Jacobian. In fact, a standard approach to deal with singularities (where JJ^T is not invertible) is to consider the so-called singularity-robust inverse [11] $\tilde{J}_W^\sharp = W^{-1}J^T(JW^{-1}J^T + k\mathbf{I}_d)^{-1}$, which directly corresponds to the regularized pseudo-inverse $J_{WC}^\sharp$ as defined in (6). The regularizer can be interpreted as measuring the tolerance of the task constraint. The asymmetry h of the motion prior $\mathcal{N}(q_{t+1} \mid q_t + h, W^{-1})$ is the Bayesian counterpart of nullspace motion. Table 1 summarizes the relations between classical quantities and their Bayesian counterparts.

2.2 Multiple Task Variables

Theorem 1 directly extends to the case when we have multiple task variables $x_1, .., x_m$, with x_i d_i-dimensional, corresponding to different task mappings $\phi_i : \mathbb{R}^n \to \mathbb{R}^{d_i}$. The full joint (see Figure 1(b)) then reads

$$P(x_t, q_t, q_{t\text{-}1}) = \Big[\prod_{i=1}^{m} P(x_{i,t} \mid q_t)\Big]\; P(q_t \mid q_{t\text{-}1})\; P(q_{t\text{-}1}) \tag{9}$$

where the motion prior is as before and for each task coupling

$$P(x_{i,t} \mid q_t) = \mathcal{N}(x_{i,t} \mid \phi_i(q_t), C_i) \tag{10}$$

we have a different task covariance C_i.

The extension can be subsumed in the previous derivation by introducing the joint random variable $x = (x_1, .., x_m)$ (d-dimensional with $d = \sum_i d_i$) and defining the covariance matrix $C = \text{diag}(C_1, .., C_m)$ to be the block matrix with sub-matrices C_i. Nevertheless, the explicit derivation allows us to establish interesting relations to classical prioritized inverse kinematics [12, 1].

Corollary 1. *In the case of multiple task variables, as given by equations (9,10,2), the motion posterior is*

$$P(q_t \mid x_t, q_{t\text{-}1}) \approx \mathcal{N}(q_t \mid q_t^{MAP}, (\sum_{i=1}^{m} J_i^T C_i^{-1} J_i + W)^{-1}) \tag{11}$$

with the MAP motion

$$q_t^{MAP} = q_{t\text{-}1} + \Big[\sum_{i=1}^{m} J_i^T C_i^{-1} J_i + W\Big]^{-1} \Big[\sum_{i=1}^{m} J_i^T C_i^{-1}(x_{i,t} - \phi_i(q_{t\text{-}1})) + Wh\Big] \qquad (12)$$

The corollary follows directly from equation (5). The question of the classical limit is particularly interesting in the case of multiple variables. Let us investigate the case when we hierarchically require tightness in the task constraints. More specifically, one can iteratively take the limit $C_i \to 0$ starting with $i = 1$ up to $i = m$; in other terms this limit can be generated when defining $C_i = \varepsilon^{m-i}\mathbf{I}_{d_i}$ and simultaneously taking the limit $\varepsilon \to 0$. It turns out that this limit is exactly equivalent to prioritized inverse kinematics [12, 1].

For $m = 2$ task variables one can prove the equivalence between prioritized inverse kinematics and the hierarchical classical limit of the MAP motion exactly (by directly applying the Woodbury identity). For $m > 2$ we could not find an elegant proof but we numerically confirmed this limit for up to $m = 4$.

Non-zero task variances can again be interpreted as regularizers. Note that without regularizers the standard prioritized inverse kinematics is numerically brittle. Handling many control signals (e.g., the over-determined case $\sum d_i > n$) is problematic since the nullspace-projected Jacobians will become singular (with rank $< d_i$). For non-zero C_i the computations in equation (12) are rather different to iterative nullspace projections and numerically robust.

2.3 Dynamic Case

We address the case of dynamic motion control by moving to velocity space and considering the random variables $\dot{q}_t \in \mathbb{R}^n$ and $\dot{x}_t \in \mathbb{R}^d$, which refer to the joint velocities and task velocities, respectively. In addition to these variables, let $u_t \in \mathbb{R}^n$ be a random variable that refers to a (torque) control signal we apply to the actuators. Consider the joint probability distribution

$$P(\dot{x}_t, \dot{q}_t, u_t, \dot{q}_{t\text{-}1}) = P(\dot{x}_t \,|\, \dot{q}_t)\; P(\dot{q}_t \,|\, u_t, \dot{q}_{t\text{-}1})\; P(u_t)\; P(q_{t\text{-}1}) \qquad (13)$$

as also illustrated by the graphical model in Figure 1(c). Here, we call $P(u_t)$ the *control prior* and assume

$$P(u_t) = \mathcal{N}(u_t \,|\, h, H^{-1}) \;, \qquad (14)$$

where $h \in \mathbb{R}^n$ is a vector that induces an *asymmetry in the control prior* and H is a *control metric* which enters this prior in terms of its covariance H^{-1}. Further, $P(\dot{q}_t \,|\, u_t, \dot{q}_{t\text{-}1})$ is the *system dynamics* and we assume

$$P(\dot{q}_t \,|\, \dot{q}_{t\text{-}1}, u_t) = \mathcal{N}(\dot{q}_t \,|\, \dot{q}_{t\text{-}1} + M^{-1}(u_t + F), Q) \;, \qquad (15)$$

where M is the generalized mass matrix, $F \in \mathbb{R}^n$ the generalized force, and Q describes the control stochasticity. Finally, for the task coupling we assume

$$P(\dot{x}_t \,|\, \dot{q}_t) = \mathcal{N}(\dot{x}_t \,|\, J\dot{q}_t, C) \;, \tag{16}$$

where the Jacobian $J = \frac{\partial \phi}{\partial q}$ relates the task and joint space velocities and, as before, C denotes the covariance in this coupling.

Bayesian dynamic control now computes the posterior control $P(u_t \,|\, \dot{x}_t, \dot{q}_{t\text{-}1})$ conditioned on the desired task velocity $\dot{x}_t$. We can derive the following result

Theorem 2. *Given equations (13-16), the posterior control is*

$$P(u_t \,|\, \dot{x}_t, \dot{q}_{t\text{-}1}) = \mathcal{N}(u_t \,|\, u_t^{MAP}, (T^T A^{\text{-}1} T + H)^{\text{-}1}) \tag{17}$$
$$T := JM^{\text{-}1} \;, \quad A := JQJ^T + C \;,$$

and the MAP control

$$u_t^{MAP} = (T^T A^{\text{-}1} T + H)^{\text{-}1} [T^T A^{\text{-}1} (\dot{x}_t - J\dot{q}_{t\text{-}1} - TF) + Hh] \tag{18}$$
$$= T_{HA}^{\sharp} (\dot{x}_t - J\dot{q}_{t\text{-}1} - TF) + (\mathbf{I}_n - T_{HA}^{\sharp} T)\, h \;, \tag{19}$$
$$T_{HA}^{\sharp} := H^{\text{-}1} T^T (T H^{\text{-}1} T^T + A)^{\text{-}1} \;. \tag{20}$$

Proof. We have

$$\begin{aligned}
&P(u_t \,|\, \dot{x}_t, \dot{q}_{t\text{-}1}) \propto \textstyle\int_{\dot{q}_t} \mathrm{d}\dot{q}_t \; P(\dot{x}_t \,|\, \dot{q}_t)\; P(\dot{q}_t \,|\, \dot{q}_{t\text{-}1}, u_t)\; P(u_t) \\
&= \textstyle\int_{\dot{q}_t} \mathrm{d}\dot{q}_t \; \mathcal{N}(\dot{x}_t \,|\, J\dot{q}_t, C)\; \mathcal{N}(\dot{q}_t \,|\, \dot{q}_{t\text{-}1} + M^{\text{-}1}(u_t + F), Q)\; \mathcal{N}(u_t \,|\, h, H^{\text{-}1}) \\
&= \textstyle\int_{\dot{q}_t} \mathrm{d}\dot{q}_t \; \mathcal{N}[\dot{q}_t \,|\, J^T C^{\text{-}1} \dot{x}_t, J^T C^{\text{-}1} J]\, \mathcal{N}[\dot{q}_t \,|\, Q^{\text{-}1} \dot{q}_{t\text{-}1} + Q^{\text{-}1} M^{\text{-}1}(u_t + F), Q^{\text{-}1}]\; \mathcal{N}(u_t \,|\, h, H^{\text{-}1}).
\end{aligned}$$

Applying the product rule (37) produces a Gaussian over $\dot{q}_t$ which integrates to 1. Using the short hand $A := JQJ^T + C$ we get

$$\begin{aligned}
&P(u_t \,|\, \dot{x}_t, \dot{q}_{t\text{-}1}) \\
&\quad = \mathcal{N}\Big[\dot{q}_{t\text{-}1} + M^{\text{-}1}(u_t + F) \,|\; J^T A^{\text{-}1} \dot{x}_t,\; J^T A^{\text{-}1} J\Big]\; \mathcal{N}[u_t \,|\, Hh, H] \\
&\quad \propto \mathcal{N}\Big[u_t \,|\; M^{-\top} J^T A^{\text{-}1} (\dot{x}_t - J\dot{q}_{t\text{-}1} - JM^{\text{-}1} F),\; M^{-\top} J^T A^{\text{-}1} J M^{\text{-}1}\Big]\; \mathcal{N}[u_t \,|\, Hh, H] \;.
\end{aligned}$$

Again applying the product rule (37) and using the short hand $T := JM^{\text{-}1}$ we get

$$\begin{aligned}
P(u_t \,|\, \dot{x}_t, \dot{q}_{t\text{-}1}) &\propto \mathcal{N}\Big[u_t \,|\; T^T A^{\text{-}1} (\dot{x}_t - J\dot{q}_{t\text{-}1} - TF) + Hh,\; T^T A^{\text{-}1} T + H\Big] \\
&= \mathcal{N}(u_t \,|\, u_t^{\mathrm{MAP}}, B) \\
&\qquad B = (T^T A^{\text{-}1} T + H)^{\text{-}1} \\
&\qquad u_t^{\mathrm{MAP}} = B[T^T A^{\text{-}1} (\dot{x}_t - J\dot{q}_{t\text{-}1} - TF) + Hh]
\end{aligned}$$

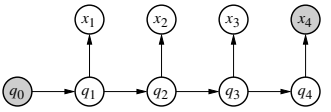


Fig. 2 The same graphical model as for redundant motion control (Figure 1(a)) but for multiple time slices.

Using the Woodbury identity as in (7) and (8) we can rewrite this expression as

$$u_t^{\text{MAP}} = T_{HA}^{\sharp}(\dot{x}_t - J\dot{q}_{t\text{-}1} - TF) + (\mathbf{I}_n - T_{HA}^{\sharp}T)\, h\,. \qquad \blacksquare$$

Again, the theorem provides two expressions (18) and (19) for u_t^{MAP}. In the classical limit $C \to 0$ and $Q \to 0$ (tight task constraint and zero control noise) the second expression directly retrieves the general optimal dynamic control law presented in [13],

$$u_t^{\text{MAP}} \overset{\substack{Q\to 0\\ C\to 0}}{=} T_W^{\sharp}(\dot{x}_t - J\dot{q}_{t\text{-}1} - TF) + (\mathbf{I}_n - T_W^{\sharp}T)\, h\,. \tag{21}$$

Again, C can be understood as a regularizer for a singularity-robust matrix inversion. As discussed in [13], special choices of the control metric H in the dynamic control, e.g., $H = M^{-1}$, $H = M^2$, or $H = \mathbf{I}_n$ correspond to special classical control strategies. For instance, Khatib's operational space control follows from choosing $H = M^{-1}$.

3 A Bayesian View on Motion Planning

From the last two theorems we may conclude that the Bayesian approach applied to control in a single time-slice largely reproduces regularized classical solutions in a different interpretation. In particular, in these 'single time slice' models one is typically only interested in the MAP solutions. The additional covariances we derived do not seem of much practical relevance.

The situation changes when we move from single time slice models of the immediate control to time extended models of the whole trajectory. The probabilistic inference approach naturally extends to inference over time and will become a planning method. For instance, we will consider inference in the model of Figure 2 as the direct temporal extension to Figure 1(a). The resulting posterior over $q_{1:4}$ will describe the set of likely trajectories starting in q_0 that are consistent with the constraint x_4.

The inference techniques in such temporal models typically have the flavor of forward-backward algorithms, similar to inference in HMMs or Kalman smoothing in state-space models. A flexible description of such inference techniques is in terms of message passing algorithms. These algorithms also extend to more structured temporal models with many random variables in one time slice, as we will investigate them later. In most realistic cases exact inference is infeasible because the shape of the exact probability distributions would be very complex (not part of

a simple parametric family of distributions). Again, message passing is a convenient framework to handle approximations systematically by maintaining approximate belief representations in the spirit of assumed density filtering or Expectation Propagation [9].

A more detailed description of message passing in general factor graphs is given in [17], see also [23, 10, 8]. Here we only give the message equations in pair-wise coupled networks: Given two random variables X_i and X_j coupled via a potential $f_{ij}(X_i, X_j)$, the message passing equations are

$$\mu_{j\to i}(X_i) = \sum_{X_j} f_C(X_i, X_j) \prod_{k:k\neq i} \mu_{k\to j}(X_j) \;, \tag{22}$$

where k indicates variables coupled to j other than i and, roughly speaking, the message $\mu_{j\to i}(X_i)$ is a distribution over the random variable X_i which encodes the information over X_i that results from its coupling to X_j. Given all incoming messages to a variable, the posterior marginal belief is given as the product of these,

$$b_i(X_i) := \prod_j \mu_{j\to i}(X_i) \;. \tag{23}$$

In the continuous case (as in the following) summations are replaced by integrals.

3.1 Kinematic Case

We can now derive the messages for inference in the motion planning scenario. As before, let $q_t \in \mathbb{R}^n$ be a random variable referring to the robot's joint state at time t. And let x_t be a random variable referring to a task space of the robot (e.g. an endeffector state) at time t. We consider the joint probability distribution

$$P(x_{1:T}, q_{0:T}) = \Big[\prod_{t=1}^{T} P(x_t \,|\, q_t)\; P(q_t \,|\, q_{t\text{-}1})\Big] P(q_0) \tag{24}$$

as also illustrated by the graphical model in Figure 2. We choose the motion prior and the task coupling exactly as before

$$P(q_t \,|\, q_{t\text{-}1}) = \mathcal{N}(q_t \,|\, q_{t\text{-}1} + h_t, W^{-1}) \;, \tag{25}$$

$$P(x_t \,|\, q_t) = \mathcal{N}(x_t \,|\, \phi(q_t), C_t) \;. \tag{26}$$

Please note that we have indexed the task covariance C_t with time – this means we can choose for each time t separately how tight we want to follow a given task constraint. In particular, in the typical planning case we might be interested only in the final endeffector position x_T; in this formalism this corresponds to choosing $C_T \to 0$ very tight while choosing all other covariances $C_{1:T\text{-}1} \to \infty$ very large (they will drop out of the messages). The messages take the following form

Theorem 3. *Given equations (24 - 26) and using a local linearization of ϕ at $\hat{q}_t$ in time slice t, the message update equations read*

$$\begin{aligned}
\mu_{q_{t\text{-}1}\to q_t}(q_t) &= \mathcal{N}(q_t \,|\, s_t, S_t) \;, \\
s_t &= h_t + (S_{t\text{-}1}^{-1} + R_{t\text{-}1})^{-1}(S_{t\text{-}1}^{-1} s_{t\text{-}1} + r_{t\text{-}1}) \\
S_t &= W^{-1} + (S_{t\text{-}1}^{-1} + R_{t\text{-}1})^{-1} \qquad (27) \\
\mu_{q_{t+1}\to q_t}(q_t) &= \mathcal{N}(q_t \,|\, v_t, V_t) \;, \\
v_t &= -h_t + (V_{t+1}^{-1} + R_{t+1})^{-1}(V_{t+1}^{-1} v_{t+1} + r_{t+1}) \\
V_t &= W^{-1} + (V_{t+1}^{-1} + R_{t+1})^{-1} \qquad (28) \\
\mu_{x_t\to q_t}(q_t) &\approx \mathcal{N}[q_t \,|\, r_t, R_t] \;, \\
r_t &= J^T C_t^{-1}(x_t - \phi(\hat{q}) + J\hat{q}) \\
R_t &= J^T C_t^{-1} J \qquad (29)
\end{aligned}$$

Further, given these messages, the belief $b_t(q_t)$ over the posture at time t reads

$$\begin{aligned}
b_t(q_t) &= \mathcal{N}[q_t \,|\, b_t, B_t] \;, \\
b_t &= S_t^{-1} s_t + V_t^{-1} v_t + r_t \;,\; B_t = S_t^{-1} + V_t^{-1} + R_t \qquad (30)
\end{aligned}$$

Proof. Since all factors are pairwise we can use the expression (22) for the messages. We have

$$\begin{aligned}
\mu_{q_{t\text{-}1}\to q_t}(q_t) &= \textstyle\int_{q_{t\text{-}1}} \mathrm{d}q_{t\text{-}1}\; P(q_t \,|\, q_{t\text{-}1})\; \mu_{q_{t\text{-}2}\to q_{t\text{-}1}}(q_{t\text{-}1})\; \mu_{x_{t\text{-}1}\to q_{t\text{-}1}}(q_{t\text{-}1}) \\
&= \textstyle\int_{q_{t\text{-}1}} \mathrm{d}q_{t\text{-}1}\; \mathcal{N}(q_t \,|\, q_{t\text{-}1} + h_t, W^{-1})\; \mathcal{N}(q_{t\text{-}1} \,|\, s_{t\text{-}1}, S_{t\text{-}1})\; \mathcal{N}[q_{t\text{-}1} \,|\, r_{t\text{-}1}, R_{t\text{-}1}]
\end{aligned}$$

Using the product rule (38) on the last two terms gives a Gaussian $\mathcal{N}(s_{t\text{-}1} \,|\, R_{t\text{-}1}^{-1} r_{t\text{-}1}, S_{t\text{-}1} + R_{t\text{-}1}^{-1})$ independent of q_t which we can subsume in the normalization. What remains is

$$\mu_{q_{t\text{-}1}\to q_t}(q_t) \propto \textstyle\int_{q_{t\text{-}1}} \mathrm{d}q_{t\text{-}1}\; \mathcal{N}[q_{t\text{-}1} \,|\, W(q_t - h_t), W]\; \mathcal{N}[q_{t\text{-}1} \,|\, S_{t\text{-}1}^{-1} s_{t\text{-}1} + r_{t\text{-}1}, S_{t\text{-}1}^{-1} + R_{t\text{-}1}]$$

Using the product rule (37) and integrating over $q_{t\text{-}1}$ we finally get

$$\mu_{q_{t\text{-}1}\to q_t}(q_t) = \mathcal{N}\big(\, q_t - h_t \,|\, [S_{t\text{-}1}^{-1} + R_{t\text{-}1}]^{-1}[S_{t\text{-}1}^{-1} s_{t\text{-}1} + r_{t\text{-}1}],\; W^{-1} + [S_{t\text{-}1}^{-1} + R_{t\text{-}1}]^{-1} \,\big) \;.$$

The message $\mu_{q_{t+1}\to q_t}(q_t)$ can be derived in exactly the same way. Concerning $\mu_{x_t\to q_t}(q_t)$ equation (22) simplifies to

$$\mu_{x_t\to q_t}(q_t) = P(x_t \,|\, q_t) = \mathcal{N}(x_t \,|\, \phi(q_t), C_t)$$

Linearizing the task coupling at $\hat{q}_t$ we have

$$\begin{aligned}
\mu_{x_t\to q_t}(q_t) &\approx \mathcal{N}(x_t \,|\, \phi(\hat{q}_t) + J(q_t - \hat{q}_t), C_t) \\
&= \frac{1}{|J|}\mathcal{N}[q_t \,|\, J^T C_t^{-1}(x_t - \phi(\hat{q}_t) + J\hat{q}_t), J^T C_t^{-1} J)
\end{aligned}$$

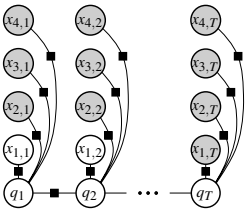


Fig. 3 Factor graph for Bayesian kinematic motion planning under multiple constraints.

Concerning the belief $b_t(q_t)$, by definition (23) we have

$$\begin{aligned} b_t(q_t) &= \mu_{q_{t\text{-}1}\to q_t}\ \mu_{q_{t+1}\to q_t}\ \mu_{x_t\to q_t} \\ &= \mathcal{N}[q_t \,|\, S_t^{\text{-}1} s_t, S_t^{\text{-}1}]\ \mathcal{N}[q_t \,|\, V_t^{\text{-}1} v_t, V_t^{\text{-}1}]\ \mathcal{N}[q_t \,|\, r_t, R_T] \end{aligned}$$

Applying the product rule (37) twice and neglecting normalization terms we have

$$b_t(q_t) \propto \mathcal{N}[q_t \,|\, S_t^{\text{-}1} s_t + V_t^{\text{-}1} v_t + r_t, S_t^{\text{-}1} + V_t^{\text{-}1} + R_t] \ . \qquad \blacksquare$$

The theorem provides two recursive equations for the forward messages (27) and the backward messages (28). A standard algorithm would resolve the recursive equations by first iterating forward over time to compute the forward messages, and then iterate backward over time to compute the backward messages. However, an extra complication in our case is that in (29) we used a linearization of ϕ at $\hat{q}_t$ in each time slice. We will choose this point of linearization as follows: in the first forward iteration we use the previous time step's MAP belief $\hat{q} = b_{t\text{-}1}(q_{t\text{-}1})$ while in subsequent backward and forward iterations we linearize at the old MAP belief $\hat{q}_t = b_t^{\text{old}}(q_t)$. Since the messages depend on the point of linearization we have to iterate the forward and backward sweeps until convergence. The algorithm is analogous to extended Kalman smoothing where observations are replaced by task constraints. The pseudo code is given in table 1.

As in the kinematic control case it is straightforward to extend these equations to the case of multiple task variables. The task coupling message then reads

$$r_t = R_t \hat{q} + \sum_i J_i^T C_{i,t}^{\text{-}1} (x_{i,t} - \phi_i(\hat{q})) \ , \quad R_t = \sum_i J_i^T C_{i,t}^{\text{-}1} J_i \ .$$

In this case we can choose different task variances $C_{i,t}$ for each task variables and in each time slot. This flexibility allows one to determine when and which task constraint has to be fulfilled by which precision. One can also follow the cascaded limit approach we mentioned in section 2.1 which in effect leads to a prioritization of the tasks, which might vary over time.

Finally, we note that the Bayesian motion planning scheme can be extended to the dynamic case in the same way as we extended the Bayesian control to the dynamic case. Due to the limited space we omit this derivation here; a general derivation can be found in [18].

Algorithm 1. Bayesian kinematic motion planning.

Input: start posture q_0, metric W, motion prior h_t, task targets $x_{0:T}$, kinematic and Jacobian functions ϕ, J
Output: trajectory $q_{0:T}$
initialize $s_0 = q_0$, $S_0^{-1} = 10^{10}$, $v_{0:T} = 0$, $V_{0:T}^{-1} = 0$, $r_{0:T} = 0$, $R_{0:T} = 0$, $k = 0$
repeat
  for $t = 1 : T$ **do** *// forward sweep*
    update s_t and S_t using (27)
    update v_t and V_t using (28)
    if $k = 0$ **then**
      $\hat{q}_t \leftarrow s_t$
    else
      $\hat{q}_t \leftarrow (1-\alpha)\hat{q}_t + \alpha b_t$
    end if
    update r_t and R_t using (29)
    update b_t and B_t using (30)
    if $|\hat{q}_t - b_t|^2 > \theta$ **then**
      $t \leftarrow t-1$ *// repeat this time slice*
    end if
  end for
  for $t = T-1 : 0$ **do** *// backward sweep*
    ..same updates as above...
  end for
  $k \leftarrow k+1$
until convergence

4 Experiments

4.1 Kinematic Control

Concerning motion control, we showed theoretically that Bayesian motion control is equivalent to regularized classical pseudo-inverse control. Our experiments confirm the close similarity to classical control. For illustration, we give examples on the simple snake benchmark proposed by [1] and specifically focus on "critical" situations where the regularization implicit in the Bayesian equations becomes apparent in comparison to non-regularized hierarchical control. The problem is to control a simulated planar snake configuration composed of 20 segments under four constraints: (1) the center of gravity should always remain vertically aligned with the foot, (2) the goal for the first segment (foot) is to be oriented with 30° from the ground, (3) the positional goal for the last segment (finger) is at $(.5, 0, 1)$, (4) the orientational goal for the last segment (finger) is to be oriented parallel to the ground. Figure 4(a) shows the snake in a valid target configuration. In [1] the problem is solved using prioritized inverse kinematics (pIK) with priority as listed above. We solve the problem by defining four variables $x_1, .., x_4$ according to the constraints above. For all task variables x_i we define that we want to follow a target that

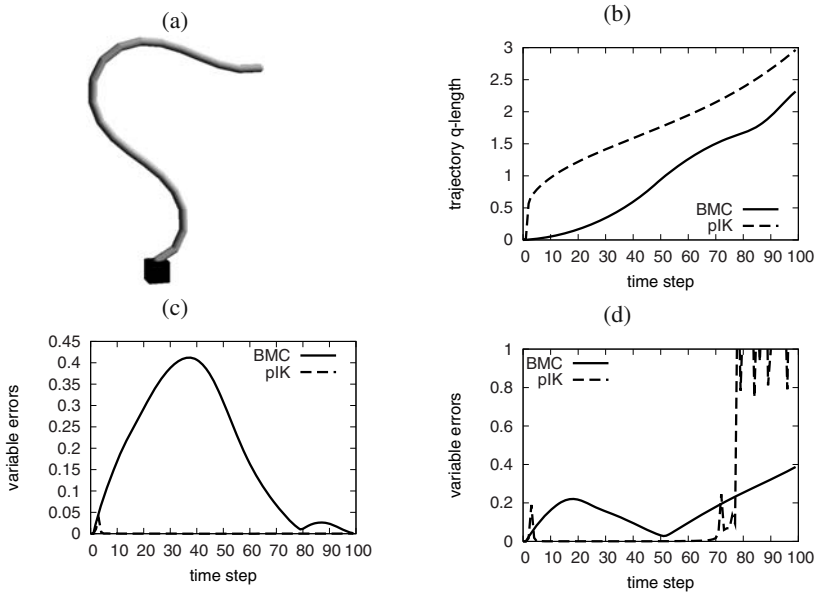


Fig. 4 Snake benchmark with four concurrent constraints. (b) Accumulative trajectory length $L = \sum_t |q_t - q_{t\text{-}1}|$ in q-space over time using prioritized inverse kinematic (pIK, dashed) and a soft Bayesian motion control (BMC, solid). (c) Total error $\sum_i e_i$ during the movement. (d) Errors during a movement towards an unreachable target.

linearly interpolates from the start position x_i^0 (straight upward posture) to the goal position x_i^*,

$$x_{i,t} = \frac{t}{T} x_i^0 + \frac{T-t}{T} x_i^* \,.$$

In the first experiment we assumed rather tight and prioritized precisions $C_i^{-1} = \nu_i \mathbf{I}$ with $\nu_{1:4} = (10^6, 10^5, 10^4, 10^3)$. As a result, the joint trajectories generated with BMC and the classical prioritized inverse kinematics are virtually indistinguishable: the max norm $\|q_{1:T} - q'_{1:T}\|^\infty$ between the two trajectories is < 0.01.

In the second experiment we assume that we require high task precision only at the final time step T. An efficient way to realize this with BMC is to start with rather soft precisions $\nu_{2:4} = 1$ at $t = 0$ and then slowly increasing them to the desired precision $\nu_{2:4} = (10^5, 10^4, 10^3)$ at $t = T$. As an exception we always keep the precision of the balance constraint high, $\nu_1 = 10^6$. The trajectory generated by BMC is now quite different from the previous one: it is much smoother. We can measure the quality of the trajectory in terms of the integrated length of the joint trajectory $L = \sum_t |q_t - q_{t\text{-}1}|$. Figure 4(b) shows pIK and BMC behavior very differently w.r.t. this measure. Nevertheless, all target variables meet the final goal constraint with high precision. This can be seen in Figure 4(b) which shows the errors $e_i = |x_i - x_i^*|$ in the control variables and the total error $E = \sum_i e_i$ during the movement for both approaches. In effect, the BMC tolerates larger errors during the movement (where

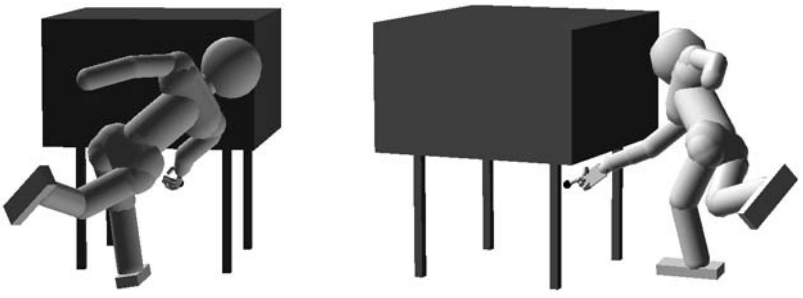


Fig. 5 Reaching test scenarios. The two images display the goal posture, the start posture is upright.

we have only required loose task coupling) in favor of a shorter trajectory – but the final task constraints at time $t = T$ are met precisely.

As a final illustration we address conflicting and infeasible constraints. Assume we want to generate trajectories where the snake curvature is minimal, as measured by a fifth variable $x_5 = \sum_{i=1}^{n} q_i^2$ and a target $x_5^* = 0$. This curvature constraint is in conflict with most other constraints. As a result, the prioritized IK numerically breaks down when adding this fifth variable without further regularization. In contrast, BMC (with constant $\nu_{1:5} = (10^6, 10^1, 10^1, 10^1, 10^0)$) yields a smooth movement which eventually fulfills the targets for $x_{1:4}$ but additionally realizes a final curvature $e_5 = 1.57494$ much lower than without this additional constraint ($e_5 = 3.11297$). In another case, assume we set a target $(1, 0, .5)$ for the last segment (finger) which is actually out of reach. BMC (with constant $\nu_{1:4} = (10^6, 10^1, 10^1, 10^1)$) yields smooth and balanced postures where necessarily the error e_2 increases. As can be seen in Figure 4(d), classical pIK drastically diverges as soon as the hard constraint in finger position becomes infeasible (at $t = 75$).

4.2 Kinematic Motion Planning

We first investigate the kinematic motion planning algorithm on a reaching problem with a simulated humanoid figure (39 DoFs) as illustrated in Figure 5. We consider a

Table 2 Trajectory length and control errors for Bayesian motion planning.

fwd-bwd iterations k	$\int \lvert\dot{q}\rvert\, dt$	$E = \int \sum_i e_{i,t}\, dt$
½ (reactive controller)	13.0124	0.0873
1	7.73616	0.1366
1½	7.70018	0.0071
2	7.68302	0.0027
5	7.65795	0.0013
10	7.62888	0.0012

trajectory of length $T = 200$. Starting from an upright posture (right image) the goal is to reach a target point (black dot) with the right finger while avoiding collisions and keeping balance on the one foot rigidly anchored to the ground. We introduce three control variables for the finger tip (endeffector) position, the center of gravity, and a global collision cost. The desired motion is defined by trajectories $x_{i,1:T}$ for each control variable x_i. We defined these to be linear interpolations from the start state to the target with $T = 100$, while keeping the precisions $\nu_{1:3} = (10^3, 10^3, 10^6)$ constant over time. Table 2 displays the trajectory length and control errors after different numbers of forward-backward iterations of belief propagation for Bayesian motion planning. $k = ½$ means a single forward pass and corresponds to the reactive application of the single time-slice Bayesian motion control. $k = 1$ additionally includes a single backward smoothing. For instance, if we fix the total computational cost to 3 times that of the Bayesian forward controller ($k = 1½$ iterations) we find an improvement of 40.8% w.r.t. the trajectory length and 91.9% w.r.t. control errors as compared to the forward controller. These improvements are due to the fact that the forward controller chooses a non-efficient path which first moves straight towards the obstacle and then needs a longer path to circumvent the obstacle. In contrast, the probabilistic smoothing of extended BMC leads to early nullspace movements (leaning to the right) which make the later circumvention of the obstacle more efficient.

4.3 Planning in More Structured Models

Consider the model in Figure 6 with factors

$$\begin{aligned} f_1(q_{t+1}, q_t) &= \mathcal{N}(q_{t+1} \,|\, q_t, W^{-1}) \\ f_2(q_t) &= \mathcal{N}(q_t \,|\, 0, 1.0) \\ f_3(x_{t+1}, x_t) &= \mathcal{N}(x_{t+1} \,|\, x_t, 0.1) \\ f_4(x_t, q_t) &= \mathcal{N}(x_t \,|\, \phi(q_t), 0.001) \\ f_5(x_T) &= \mathcal{N}(x_T \,|\, x_T^*, 0.001) \end{aligned} \tag{31}$$

The first factor is the usual motion prior in the joint state. The second factor places a prior $P(q_t)$ which limits the joint range – for simplicity we use again a Gaussian potential ($q = 0$ indicates the joint centers). The third factor expresses a prior about the endeffector movements – since we do not have a strong prior about endeffector movements we assume a weak potential with standard deviation of endeffector movements of 0.1. The fourth factor is the usual coupling between joint state and endeffector. Generally, in factor graphs conditioning a variable can be expressed as including a Kronecker factor. Hence, the fifth factor represents the goal constraint, conditioning the target of the endeffector to be close to x_T^*.

This model is different to the one of the previous section in two respects: We condition only on the final task state x_T, and we included a motion prior also within the task space. We investigate this graph because it allows for a qualitative new

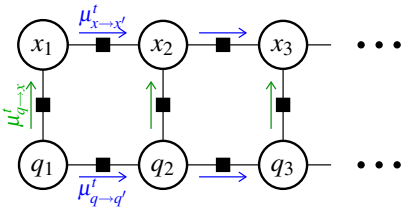


Fig. 6 The factor graph for the decomposed planning scenario.

approach to decompose planning. In [7, 16] an algorithm was proposed for hierarchical planning. In the first stage it first computes an optimal trajectory only in the task space. In the second stage, constrained on this task space trajectory, it computes an optimal trajectory in joint space. Clearly, this approach is limited since the task space trajectory was computed without any concern whether this task space trajectory might lead to high costs when realized in joint space.

In the factor graph 6 we can follow a very similar approach to hierarchically decompose planning – but fully account for the mutual coupling of task and joint variables. Given the explicit model we can derive all necessary messages for belief propagation from equation (23). The algorithm we propose is given by the following message passing scheme:

1. Initialize all beliefs uniformly, except for q_0, x_0 and x_T.
2. Update the task space beliefs

$$b(x_t) = \mu_{x_{t-1}\to x_t}\ \mu^t_{x_{t+1}\to x_t}\ \mu^t_{q_t\to x_t}\ ,$$

   first iterating forward for $t = 1,..,T$, then iterating backward for $t = T\text{-}1,..,1$. This will yield a *preliminary belief* over possible trajectories *in task space*.
3. Update the q-space beliefs

$$b(q_t) = \mu^{q_{t-1}\to q_t}\ \mu_{q_{t+1}\to q_t}\ \mu^t_{x_t\to q_t}\ ,$$

   first iterating forward for $t = 1,..,T$, then iterating backward for $t = T\text{-}1,..,1$. This procedure is exactly as described in the previous section, using local linearizations of the kinematics at $\hat{q}_t = \langle b(q_t)\rangle$. This generates a belief over possible trajectories in q-space.
4. Iterate steps (ii) and (iii).

Conceptually, the most interesting aspect is that in step (ii) we do not compute a *single* task space trajectory, but rather represent the *whole variety of possible task space trajectories* by the beliefs. The coupling to the q-space then narrows down this variety according to the prior in q-space. Iterating steps (ii) and (iii) means to propagate up ($\mu^t_{q_t\to x_t}$) and down ($\mu^t_{x_t\to q_t}$) messages between the x-level and the q-level until coherence between both levels is achieved.

4.3.1 Illustration on a Planar Arm

We would first like to illustrate the approach on a simple 3-link planar arm described by the joint state $q \in \mathbb{R}^3$ and the endeffector position $x \in \mathbb{R}^2$. We are given the initial configuration $q_0 = (.5, .5, .5)$, the endeffector target $x_T^* = (-1.5, .5)$ and $T = 20$.

Figure 7(a) displays the preliminary belief over endeffector states after the first stage of inference (step (ii)). We find that at this stage, the belief over the endeffector trajectory is simply a straight line with quite a large standard deviation associated with each via-point. This "Gaussian worm" can be interpreted as the range of possible endeffector trajectories neglecting the coupling to any other variables or constraints. All subsequent inference steps will further refine and narrow down this initial belief by fusing it with messages from the q-space. Figure 7(b) displays the belief over endeffector trajectories after a cycle of inference steps (ii), (iii), (ii), i.e., the coupling to the joint state has been accounted for. Also the MAP joint configuration is displayed at each time step. As expected from the discussion above, the MAP endeffector trajectory is not anymore a straight line. The reason is that the constraints we induced via the prior joint transition probability (31) favors small steps in joint space.

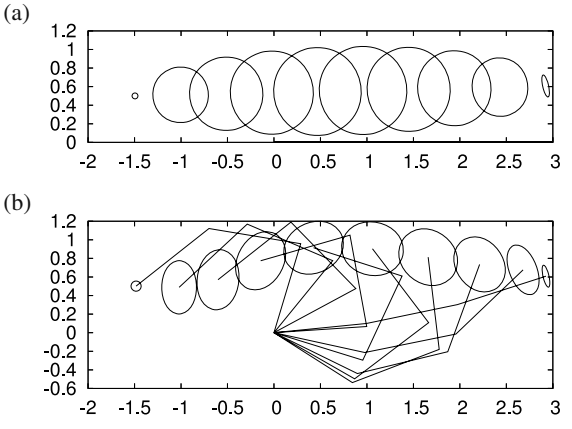


Fig. 7 (a) The belief over the endeffector trajectory after the first stage of inference – neglecting the coupling to the joint state. (b) The belief over the endeffector trajectory after coupling to the joint state; also the MAP joint configurations for each time step are displayed.

4.3.2 Illustration with a Humanoid Upper Body

As another illustration, consider the $n = 13$ joint humanoid upper body displayed in Figure 8. We take the right hand as the endeffector and plan a target reaching trajectory (of length $T = 50$) to a goal in the upper left working domain of the robot.

Figures 9(a&b) display the result after 2 iterations of the inference steps (1-4), which provided sufficient convergence. The figures display the maximum a posteriori joint configuration (MAP, the maximum of the posterior joint state distribution)

and the standard deviations of the endeffector distribution at different time steps. The MAP endeffector trajectory is not a straight line. To give a quantitative measure of the quality of the trajectory we compare the MAP joint trajectory computed via probabilistic inference with the joint trajectory that results from a standard redundant control approach. More precisely, the redundant control approach first presumes a straight endeffector line equally divided in $T = 50$ segments and then uses standard motion rate control.

We define a global trajectory cost using the q-space metric W,

$$C(q_{1,..,T}) = \sum_{t=1}^{t\text{-}1} ||q_{t+1} - q_t||_W \ .$$

Table 3 displays the trajectory costs for the trajectories computed via the forward controller and the MAP trajectory computed via probabilistic inference. The MAP trajectory is clearly more efficient in terms of this cost. This stems from the fact that equation (31) imposes a prior transition likelihood $f_1(q_{t+1}, q_t) \propto \mathcal{N}(q, W^{-1})$ which

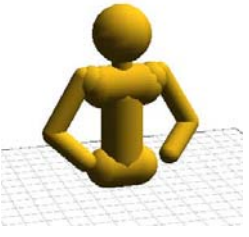


Fig. 8 A humanoid upper body with $n = 13$ hinge joints. The hip is fixed, the right hand serves as endeffector.

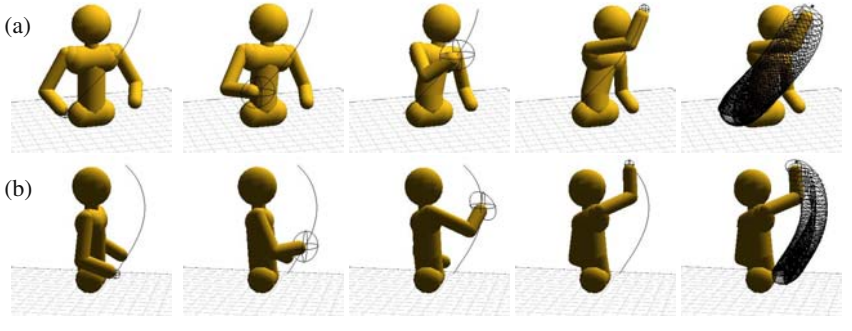


Fig. 9 Results of probabilistic inference planning with an humanoid upper body. Reaching to a target without obstacles, displayed from two different perspectives. We see the MAP joint configuration and the Gaussian endeffector distribution (indicated as ellipsoid) at different intermediate time steps. The optimal trajectory in joint space leads to a curve trajectory in endeffector space.

Table 3 Global cost of joint space trajectories.

	trajectory cost $C(q_{1,..,T})$
forward controller	11.19
MAP trajectory	8.14

reflects exactly this metric in joint space. The curve of the endeffector trajectory is a result of this efficiency.

4.4 Coupling with Collision Constraints

In [7, 16] the idea of hierarchical decomposition of planning was considered, where a planning problem is first solved on a reduced task. This results is then coupled into a second-stage planning problem on the q-space level. This strict two-stage procedure neglects that constraints (or priors) on the q-space level should eventually also influence the optimal task space plan. Intuitively one might come up with an algorithm that reiterates planning on both levels until their mutual interaction leads to a coherent plan on both levels. This is exactly what belief propagation does automatically in our framework.

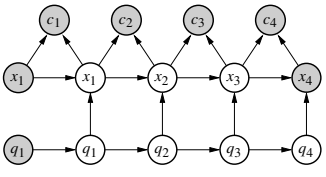


Fig. 10 Model for decomposed planning under collision constraints.

Let q_t and x_t be as before. In addition, let c_t be a binary random variable that indicates collision of the endeffector with an obstacle (a table in the following experiment). Consider the model

$$P(c_{1:T}, x_{0:T}, q_{0:T}) = P(x_0)\, P(q_0) \\ \Big[\prod_{t=1}^{T} P(c_t \,|\, x_t, x_{t\text{-}1})\, P(x_t \,|\, q_t, x_{t\text{-}1})\, P(q_t \,|\, q_{t\text{-}1})\Big] \tag{32}$$

as illustrated in Figure 10. We assume the motion prior $P(q_t \,|\, q_{t\text{-}1})$ as before and

$$P(c_t \,|\, x_t, x_{t\text{-}1}) = \mathcal{N}(c_t \,|\, \phi^c(x_t, x_{t\text{-}1}), D)\ , \tag{33}$$

$$P(x_t \,|\, q_t, x_{t\text{-}1}) \propto \mathcal{N}(x_{t+1} \,|\, x_t, C^{xx})\, \mathcal{N}(x_t \,|\, \phi(q_t), C^{xq})\ . \tag{34}$$

Here, ϕ^c is a function that determines the maximal penetration depth with the obstacle when the endeffector moves from $x_{t\text{-}1}$ to x_t. We compute gradients for this as

in [19]. Further, note that we have now included a *task motion prior* $\mathcal{N}(x_{t+1} \,|\, x_t, C^{xx})$ as an additional factor in the model, even though this might be a rather week prior (we will choose $C^{xx} = .1$ in the experiments).

In the specific experiment we condition on the final endeffector position $x_T = x^*$ and we condition on each collision variable $c_t = 0$ being zero. We perform inference in this model by a message passing scheme that effectively alternates between forward-backward inference on the task level and on the q-space level until a coherent posterior over both is found. We apply the following message passing scheme:

1. propagate forward & backward on x: first compute the messages $\mu_{x_{t-1} \to x_t}$ for $t = 1, .., T$, then the messages $\mu_{x_{t+1} \to x_t}$ for $t = T\text{-}1, .., 1$
2. compute all the messages $\mu_{c \to (x_{t-1} x_t)}$ using local linearizations of ϕ^c at the current MAP task belief $b_t(x_t)$
3. propagate down: compute all the messages $\mu_{x_t \to q_t}$
4. propagate forward & backward on q: first compute the messages $\mu_{q_{t-1} \to q_t}$ for $t = 1, .., T$, then the messages $\mu_{q_{t+1} \to q_t}$ for $t = T\text{-}1, .., 1$
5. propagate up: compute all the messages $\mu_{q_t \to x_t}$
6. iterate steps (i)-(v)

In the first iteration, step (i) will compute a preliminary task space belief neglecting the collision constraint. In step (ii) the collision constraint is then coupled into the task space belief which is then in step (iii) propagated to the q-space belief, et cetera.

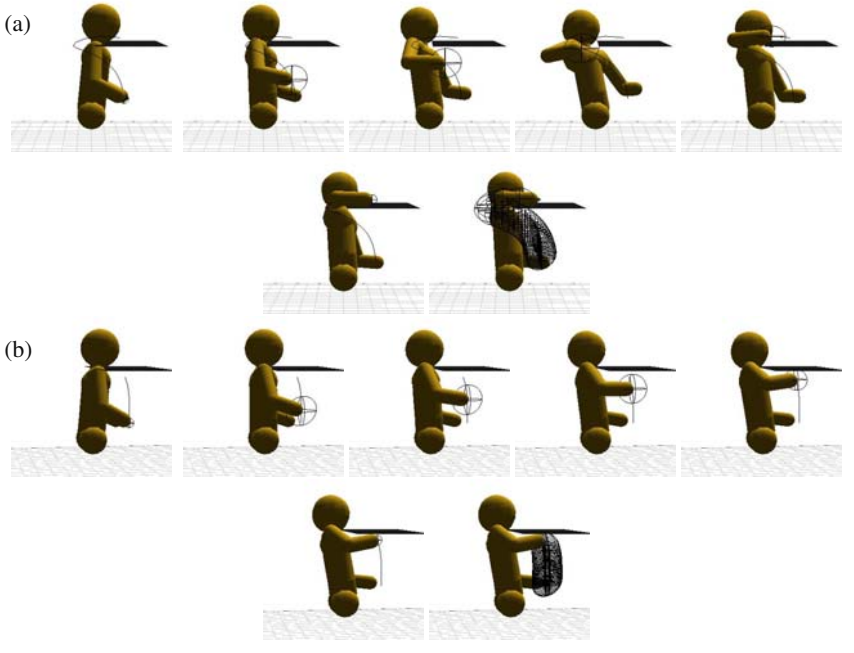


Fig. 11 Reaching to a target above (a) the table and below (b) the table whilst avoiding collision.

Figures 11(a&b) display the result after two iterations of this message passing scheme for $T = 30$. In the first case (Figure (a)) the target endeffector position is slightly above the table and the generated movement avoids the obstacle. In the second case, the target is only slightly displaced but now below the table. Here, the result is a rather straight trajectory. A standard forward controller that follows a gradient of a superimposed target potential and obstacle avoidance potential would typically get trapped under the table in the first case. Also, the local target potential of a reactive controller would hardly distinguish between the first and the second case.

The experiments ran on a laptop with a 1.1GHz Centrino Mobile processor. The first experiment ($T = 50$, without constraints, $k = 2$ inference sweeps) takes 3.56 seconds, the second experiment ($T = 50$, with constraints, $k = 2$ sweeps) takes 3.97 seconds.

5 Discussion

Let us discuss here specific aspects in relation to the derived algorithms and the Bayesian approach in general.

Local vs. Global Planning

An important aspect often discussed in the context of robotic planning is locality. Many classical approaches like RRTs [6, 5] try to tackle global planning problems, for instance where one first has to move away from a goal before moving towards the goal becomes possible. Theoretically, planning based on *exact* inference would also generate globally correct posterior distributions about *all* possible trajectories and thereby perform global planning. For discrete MDPs this is feasible and has been demonstrated [21]. However, in almost any realistic robotics scenario exact inference is infeasible since this would require to represent very complex distributions (beliefs) which are not part of a feasible parametric family of distributions. In the algorithms we derived we assumed Gaussian belief representations and used local linearizations to stay in this family of belief representations. If we tried exact inference in domains with collision constraints, the exact beliefs had very complex, discontinuous forms. The Gaussian belief approximations effectively introduce a kind of "locality" since the set of likely trajectories is assumed close to a mean trajectory. Other kinds of belief representations would give a more global character to planning, e.g. sample based representations (particle filters) or mixture of Gaussians. In conclusion, it is very much a matter of which approximations and belief representations are chosen which determine how global the inference approach is.

Computational Complexity

The complexity of inference is linear in the number of message computations needed; each message computation requires operations on *symmetric* matrices that

approximately with n^2. A single forward-backward pass along one variable is linear in T. If the number of neighbors to each variable is bounded, then the total number of edges in the factor graph is $O(TK)$, i.e., linear in the number K of variables. For complex (e.g., hierarchically deep) DBNs it is however an open question how many iterations of inference sweeps one needs until convergence.

Handling Delayed Feedback in Control

In realistic cases the sensor feedback on the precise robot configuration (q_t and $\dot{q}_t$) is delayed. Thus, for instance in the case of torque control, the direct Bayesian dynamic control law (19) is not applicable. However, it is straight-forward to explicitly include the delay in the probabilistic model and then apply Bayesian inference as we discussed it in the planning case. Figure 12 is an example for representing delayed feedback in the model – here the feedback is delayed for 2 time steps. The Bayesian control law is then given by computing $P(u_t \,|\, \bar{q}_{t3}, u_{t1}, u_{t2}, \dot{x}_t)$. In a sense, this approach naturally combines a probabilistic state estimation of $\bar{q}_{t1}$ with the ordinary Bayesian dynamic control.

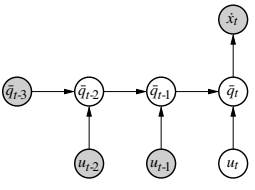


Fig. 12 Bayesian dynamic control in the case of delayed feedback. Here, $\bar{q}_t = (q_t, \dot{q}_t)$ subsumes positions and velocities.

6 Conclusion

Bayesian motion control and planning is based on the idea of *fusing motion objectives* (task constraints, goals, motion priors, etc) in a way similar to Bayesian sensor fusing. We formulate probabilistic models which represent these objectives, condition on constraints and goals, and use probabilistic inference to compute a posterior distribution over the possible trajectories and control signals. This distribution is a representation of the variety of likely trajectories and the corresponding control signals given that constraints and goals must be met.

The main contribution of this paper are derivations of explicit Bayesian control and planning algorithms. In the case of control we have addressed the problems of kinematic and dynamic task control by deriving a posterior distribution over the posture q_t and the control u_t, respectively. A straight-forward way to apply these results is to choose the MAP posture q_t^{MAP} as the next kinematic control point, or the MAP control u_t^{MAP} as the current control signal. We have shown that these MAP control laws are closely related to the classical control laws. More specifically, q_t^{MAP} corresponds to classical motion rate control including nullspace movement and a

regularized pseudo-inverse Jacobian – where the regularizer is now interpreted as the tightness of the task constraint and the nullspace motion as the asymmetry of the motion prior. And u_t^{MAP} correspond to the classical dynamic control (as given, e.g., in [13]) generalized to include also a non-zero task variance C and a stochastic control variance Q.

To solve planning problems we extended the approach to multiple time slices, i.e. we proposed to use Bayesian inference to compute posterior trajectories conditioned on (future) constraints and goals. We derived explicit message passing algorithms for the basic cases of kinematic and dynamic planning. However, the general approach of inference and message passing algorithms can also be applied to more structured representations. Structure means that either the state q_t is factored into multiple variables (e.g., when we need to include variables describing objects or other properties of the environment) or that the task x_t is factored in multiple variables (the multi task variable scenario we mentioned is an example). In particular when the dependencies between such variables are sparse inference and message passing algorithms can exploit this structure to increase computational efficiency. This leads us away from the classical bottleneck of planning: the need to represent state in one big state space. Rather than deriving more explicit message passing algorithms for specific cases of structured models, the theorems showed how message equations can systematically be derived for specific graphical models. Our last experimental scenario addressed such an alternative model structure which is related to hierarchical planning where one alternates between planning in the task space and planning in the q-space. In section 5 we have also discussed in what sense *exact* inference would correspond to optimal global planning, whereas the more realistic case of approximate inference corresponds to more local planning. Future research on Bayesian motion planning should focus on exactly these two points: (1) exploiting the benefits of message passing algorithms in more structured models and (2) exploring different belief representation techniques (such as particle representations) for better approximations during inference.

Acknowledgments

This work was supported by the German Research Society (DFG), Emmy Noether fellowship TO 409/1-3.

Used Gaussian Identities

We define a Gaussian over x with mean a and covariance matrix A as the function

$$\mathcal{N}(x\,|\,a,A) = \frac{1}{|2\pi A|^{1/2}}\ \exp\{-\frac{1}{2}(x-a)^T\ A^{-1}\ (x-a)\} \tag{35}$$

with property $N(x\,|\,a,A) = N(a|\ x,A)$. We also define the canonical representation

$$\mathcal{N}[x\,|\,a,A] = \frac{\exp\{-\frac{1}{2}a^T A^{-1} a\}}{|2\pi A^{-1}|^{1/2}}\ \exp\{-\frac{1}{2}x^T\ A\ x + x^T a\} \tag{36}$$

with properties

$$\mathcal{N}[x\,|\,a,A] = \mathcal{N}(x\,|\,A^{-1}a,A^{-1})\ ,\ \mathcal{N}(x\,|\,a,A) = \mathcal{N}[x\,|\,A^{-1}a,A^{-1}]\ .$$

The product of two Gaussians can be expressed as

$$\mathcal{N}[x\,|\,a,A]\ \mathcal{N}[x\,|\,b,B] = \mathcal{N}[x\,|\,a+b,A+B]\ \mathcal{N}(A^{-1}a\,|\,B^{-1}b,A^{-1}+B^{-1})\ , \tag{37}$$

$$\mathcal{N}(x\,|\,a,A)\ \mathcal{N}(x\,|\,b,B) = \mathcal{N}[x\,|\,A^{-1}a+B^{-1}b,A^{-1}+B^{-1}]\ \mathcal{N}(a\,|\,b,A+B)\ , \tag{38}$$

$$\mathcal{N}(x\,|\,a,A)\ \mathcal{N}[x\,|\,b,B] = \mathcal{N}[x\,|\,A^{-1}a+b,A^{-1}+B]\ \mathcal{N}(a\,|\,B^{-1}b,A+B^{-1})\ . \tag{39}$$

Linear transformations in x imply the following identities,

$$\mathcal{N}(Fx+f\,|\,a,A) = \frac{1}{|F|}\ \mathcal{N}(x|\ F^{-1}(a-f),\ F^{-1}AF^{-\top})\ , \tag{40}$$

$$= \frac{1}{|F|}\ \mathcal{N}[x|\ F^{T}A^{-1}(a-f),\ F^{T}A^{-1}F]\ , \tag{41}$$

$$\mathcal{N}[Fx+f\,|\,a,A] = \frac{1}{|F|}\ \mathcal{N}[x|\ F^{T}(a-Af),\ F^{T}AF]\ . \tag{42}$$

The joint Gaussian of two linearly dependent Gaussian variables reads

$$\mathcal{N}(x\,|\,a,A)\ \mathcal{N}(y\,|\,b+Fx,B) = \mathcal{N}\left(\binom{x}{y}\,|\,\binom{a}{b+Fa},\ \begin{pmatrix} A & A^{T}F^{T} \\ FA & B+FA^{T}F^{T}\end{pmatrix}\right) \tag{43}$$

References

1. Baerlocher, P., Boulic, R.: An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. In: The Visual Computer (2004)
2. Bui, H., Venkatesh, S., West, G.: Policy recognition in the abstract hidden markov models. Journal of Artificial Intelligence Research 17, 451–499 (2002)
3. Culotta, A., McCallum, A., Selman, B., Sabharwal, A.: Sparse message passing algorithms for weighted maximum satisfiability. In: New England Student Colloquium on Artificial Intelligence, NESCAI (2007)
4. Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Methods for learning control policies from variable-constraint demonstrations. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 253–291. Springer, Heidelberg (2010)
5. Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., Inoue, H.: Motion planning for humanoid robots. In: Proc. 20th Int. Symp. Robotics Research, ISRR 2003 (2003)
6. Kuffner, J.J., LaValle, S.M.: RRT-connect: An efficient approach to single-query path planning. In: Proc. of IEEE Int'l Conf. on Robotics and Automation (2000)
7. Li, W., Todorov, E., Pan, X.: Hierarchical optimal control of redundant biomechanical systems. In: 26th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society (2004)
8. Minka, T.: A family of algorithms for approximate bayesian inference. PhD thesis, MIT (2001)
9. Minka, T.P.: Expectation propagation for approximate Bayesian inference. In: Proc. of the 17th Annual Conf. on Uncertainty in AI (UAI 2001), pp. 362–369 (2001)

10. Murphy, K.: Dynamic bayesian networks: Representation, inference and learning. PhD Thesis, UC Berkeley, Computer Science Division (2002)
11. Nakamura, Y., Hanafusa, H.: Inverse kinematic solutions with singularity robustness for robot manipulator control. Journal of Dynamic Systems, Measurement and Control 108 (1986)
12. Nakamura, Y., Hanafusa, H., Yoshikawa, T.: Task-priority based redundancy control of robot manipulators. Int. Journal of Robotics Research 6 (1987)
13. Peters, J., Mistry, M., Udwadia, F.E., Cory, R., Nakanishi, J., Schaal, S.: A unifying framework for the control of robotics systems. In: IEEE Int. Conf. on Intelligent Robots and Systems (IROS 2005), pp. 1824–1831 (2005)
14. Salaun, C., Padois, V., Sigaud, O.: Learning forward models for the operational space control of redundant robots. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 169–192. Springer, Heidelberg (2010)
15. Tappen, M.F., Freeman, W.T.: Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters. In: IEEE Intl. Conference on Computer Vision, ICCV (2003)
16. Todorov, E., Li, W.: Hierarchical optimal feedback control of redundant systems. In: Advances in Computational Motor Control IV, Extended Abstract (2004)
17. Toussaint, M.: Lecture notes: Factor graphs and belief propagation (2008), `http://ml.cs.tu-berlin.de/~mtoussai/notes/`
18. Toussaint, M.: Robot trajectory optimization using approximate inference. In: Proc. of the 26rd Int. Conf. on Machine Learning, ICML 2009 (2009)
19. Toussaint, M., Gienger, M., Goerick, C.: Optimization of sequential attractor-based movement for compact behaviour generation. In: 7th IEEE-RAS Int. Conf. on Humanoid Robots, Humanoids 2007 (2007)
20. Toussaint, M., Goerick, C.: Probabilistic inference for structured planning in robotics. In: Int. Conf. on Intelligent Robots and Systems (IROS 2007), pp. 3068–3073 (2007)
21. Toussaint, M., Harmeling, S., Storkey, A.: Probabilistic inference for solving (PO)MDPs. Tech. Rep. EDI-INF-RR-0934, University of Edinburgh, School of Informatics (2006)
22. Vlassis, N., Toussaint, M.: Model-free reinforcement learning as mixture learning. In: Proc. of the 26rd Int. Conf. on Machine Learning, ICML 2009 (2009)
23. Yedidia, J., Freeman, W., Weiss, Y.: Understanding belief propagation and its generalizations (2001)

Methods for Learning Control Policies from Variable-Constraint Demonstrations

Matthew Howard, Stefan Klanke, Michael Gienger, Christian Goerick, and Sethu Vijayakumar

Abstract. Many everyday human skills can be framed in terms of performing some task subject to constraints imposed by the task or the environment. Constraints are usually not observable and frequently change between contexts. In this chapter, we explore the problem of learning control policies from data containing variable, dynamic and non-linear constraints on motion. We discuss how an effective approach for doing this is to learn the *unconstrained policy* in a way that is *consistent with the constraints*. We then go on to discuss several recent algorithms for extracting policies from movement data, where observations are recorded under variable, unknown constraints. We review a number of experiments testing the performance of these algorithms and demonstrating how the resultant policy models *generalise over constraints* allowing prediction of behaviour under unseen settings where new constraints apply.

1 Introduction

A wide variety of everyday human skills can be framed in terms of performing some task subject to a set of constraints. Constraints may be imposed either by the environment [37], the task [6] or, more commonly, both. For example, when opening a door, the door acts as an environmental constraint that restricts the movement of ones hand along the opening arc of the door. When stirring soup in a saucepan, the sides of the pan prevent the spoon moving beyond their radius. Many tasks require self-imposed task constraints to be fulfilled in order to achieve adequate performance. For example when pouring

Matthew Howard, Stefan Klanke, and Sethu Vijayakumar
Institute of Perception Action and Behaviour, University of Edinburgh, Scotland, UK
e-mail: matthew.howard@ed.ac.uk

Michael Gienger and Christian Goerick
Honda Research Institute Europe (GmBH), Offenbach, Germany

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 253–291.
springerlink.com

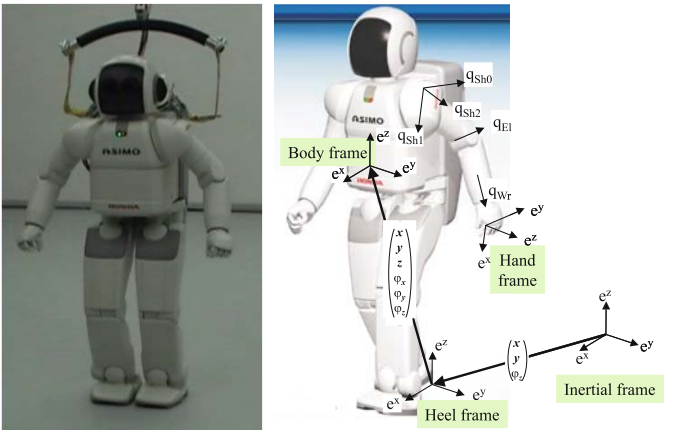


Fig. 1 ASIMO humanoid robot (left) and kinematic model used for whole body motion control (right) [15]. In our experiments 22 upper body degrees of freedom were used (2×7 DOF arms, 2 DOF head, 6 DOF torso), with the heel frame fixed.

water from a bottle to a cup, the orientation of the bottle must be constrained so that the stream of water falls within the mouth of the cup. When wiping a window, ones hand should be constrained to maintain contact with the wiping surface [38] and when climbing a ladder constraints may be applied to the centre of mass or the tilt of the torso of the climber to prevent overbalancing [27]. When manipulating or grasping solid objects the motion of ones fingers is constrained by the presence of the object [43]. Consider the task of running or walking on uneven terrain: the cyclic movement of the runner's legs is constrained by the impact of the feet on the ground in a dynamic, discontinuous and unpredictable way. In short, constraints that may be non-linear, spatially or temporally varying, or even discontinuous are ubiquitous in our everyday behaviour [50, 49, 15, 44, 48].

A promising approach to providing robots with abilities such as the above is to take examples of motion from existing skilled demonstrators (e.g. humans) and attempt to learn a control policy that somehow captures the desired behaviour [3, 4, 47]. Such techniques offer (i) a simple, intuitive interface for programming robots [4], (ii) effective methods for motion recognition and classification [26], and; (iii) accelerated optimisation of movements by using demonstration data to seed the solution [45]. However, while a wide variety of approaches for learning and representing movements have been proposed in recent years [3, 4, 47, 14], relatively few have explicitly considered the problem of dealing with constraints on motion in learning. An important component of this is the ability to deal with the apparent variability in movements *induced by varying constraints*. For example, one wishes to learn a policy that allows one not only to open a specific door of a particular size (e.g. constraining the

hand to a curve of a particular radius), but rather to open many doors of varying sizes (radii).

In this chapter we will review recent works that deal precisely with this problem, that is learning from movement data that implicitly contains dynamic and uncertain constraints. We will primarily focus on two methods recently proposed [22, 19], that allow the effect of constraints to be dealt with in an appropriate way during learning. We will compare and contrast the two methods, consider their relative strengths, and in particular assess their suitability for improving existing policy learning methods that currently rely on traditional supervised learning techniques.

The novel techniques we consider are aimed at learning from demonstrated movements that are subject to variable, dynamic constraints with the goal of finding policies that can *generalise over constraints*. The two approaches both use a similar strategy to do this; namely they both attempt to consolidate movement observations under different constraints in order to model the underlying *unconstrained policy* common to all. Learning the latter enables generalisation since we can apply new constraints to predict behaviour in novel scenarios. This is inspired by work in analytical dynamics (e.g. see [55]) where an effective and intuitive strategy for analytically solving constrained motion problems is to consider the effect constraints have in modifying the fundamental equations of motion of the unconstrained system. The difference here is that we attempt to do this in an automated, data-driven way.

In general, we will see that learning (unconstrained) policies from constrained motion data is a formidable task. This is due to several problems, such as (i) *unobservability* of constraints (ii) *non-convexity* of observations under different constraints, and; (iii) *degeneracy* in the set of possible policies that could have produced the observed movement under the constraint [19]. We will discuss at length how these problems arise when learning in the constrained setting, and then look at how the two methods overcome them, first for the special case of potential-based policies, and later for learning generic, arbitrary policies. Using these *constraint-consistent* approaches to learning it has been shown [22, 19] that given observations (i) under a sufficiently rich set of constraints it is possible to reconstruct the fully unconstrained policy; (ii) under an impoverished set of constraints we can learn a policy that generalises well to constraints of a similar class, and; (iii) under 'pathological' constraints (e.g. those that constrain the same part of the policy in all observations, effectively rendering it unobservable) we can learn a policy that at least reproduces behaviour subject to those same constraints. Furthermore, achieving these is possible without the need for explicit knowledge of the constraints in force at the time of observation.

An extensive set of experiments are have been reported [22, 19] in order to validate the methods and to assess their performance. Here, we will compare and review some of these for learning on data from several policies on complex, high-dimensional movement systems, subject to various realistic

constraints, with a view to illustrating the utility of the approaches for transferring behaviour to robots such as the ASIMO humanoid (Fig. 1).

2 Effect of Variable Dynamic Constraints on Learning

In this section, we characterise the general problem of learning control policies from data, and discuss the problems encountered when variable constraints are applied to motion.

2.1 Learning Control Policies from Data

Following Schaal et al. [47], we consider the direct learning of autonomous policies

$$\mathbf{u}(t) = \boldsymbol{\pi}(\mathbf{x}(t)) \ , \qquad \boldsymbol{\pi} : \mathbb{R}^n \mapsto \mathbb{R}^d, \tag{1}$$

from data, where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^d$ are appropriately[1] chosen state- and action-spaces, respectively. The goal of learning is to approximate the policy (1) as closely as possible [47] using a supervised learning approach, that is, we are given observations of $\mathbf{u}(t)$, $\mathbf{x}(t)$ (often in the form of trajectories) and from these we wish to learn the mapping $\boldsymbol{\pi}$. In previous work this has been done by fitting parametrised models in the form of dynamical systems [25, 24], non-parametric modelling [40, 6, 12], probabilistic Bayesian approaches [17, 16] and hidden Markov models [53, 26].

An implicit assumption found in policy learning approaches to date is that the data used for training comes from behavioural observations of some *unconstrained* or *consistently constrained* policy. By this it is meant that the policy is observed either under no constraint (e.g. movements in free space such as gestures or figure drawing), or under constraints consistent over observations (e.g. interacting with the same objects or obstacles in each case). However, in many everyday behaviours, there is variability in the constraints, such as when opening doors of varying sizes or walking on uneven terrain. This *variability in the constraints* cannot be accounted for by standard learning approaches.

[1] It should be noted that, as with all policy-based learning approaches, the choice of state- and action-space is problem specific [47] and, when used for imitation learning, depends on the *correspondence* between demonstrator and imitator. For example if we wish to learn the policy a human demonstrator uses to wash a window, and transfer that behaviour to an imitator robot, an appropriate choice of $\mathbf{x}$ would be the Cartesian coordinates of the hand, which would correspond to the end-effector coordinates of the robot. Transfer of behaviour across non-isomorphic state- and action-spaces, for example if the demonstrator and imitator have different embodiments, is also possible by defining an appropriate state-action metric [1].

Example: Finger Extension with Contact Constraints

As an example, consider the learning of a simple policy to extend a jointed finger. In Fig. 2(a) the finger is unconstrained and the policy simply moves the joints towards the zero (outstretched) position. On the other hand, in Fig. 2(b), an obstacle lies in the path of the finger, so that the finger movement is constrained – it is not able to penetrate the obstacle, so moves along the surface. The vector field representation of the two behaviours is shown in Fig. 2(c).

Given the task of learning in this scenario, applying traditional learning approaches would result in one of two possibilities. The first is that if the observations are *labelled with respect to the constraint* (here, the orientation, position and shape of the obstacle) one could learn a separate policy model for the behaviour in each of the settings. However this is clearly unsatisfactory, since each model would only be valid for the specific setting, and we would need increasing numbers of models as we observed the policy under new constraints (for example different shaped obstacles at different positions and orientations). The second possibility is that the data is *unlabelled* with respect to the constraint. In this case, one might try to perform regression directly on the observations, that is observations from both vector fields (cf. Fig. 2(c), black and red vectors). However, this presents the problem that *model averaging* would occur across observations under different constraints, resulting in a poor representation of the movement in terms of the magnitude and direction of the predictions (see Sec. 2.3).

We can avoid the need for multiple policy models if we relax our assumptions on the form (1) of the observed commands, and allow for an additional transformation of $\boldsymbol{\pi}(\mathbf{x})$. We thus model both the red and black observations as stemming from the same policy ('extend the finger'), and attribute its different appearance to the transformations as induced by the constraints. With a restriction on the class of possible transformations, as will be detailed in the next section, we can model the unconstrained policy even if we only observed constrained movements, and we can apply new constraints to adapt the policy to novel scenarios.

2.2 Formal Constraint Model

In the remainder of this chapter we will focus on constraints which act as hard restrictions on the actions available to the policy. Specifically, we consider policies subject to a set of k-dimensional ($k \leq n$) Pfaffian constraints [55]

$$\mathbf{A}(\mathbf{x}, t)\mathbf{u} = \mathbf{0}. \tag{2}$$

Under these constraints, the policy is projected into the nullspace of $\mathbf{A}(\mathbf{x}, t)$:

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{N}(\mathbf{x}, t)\boldsymbol{\pi}(\mathbf{x}(t)) \tag{3}$$

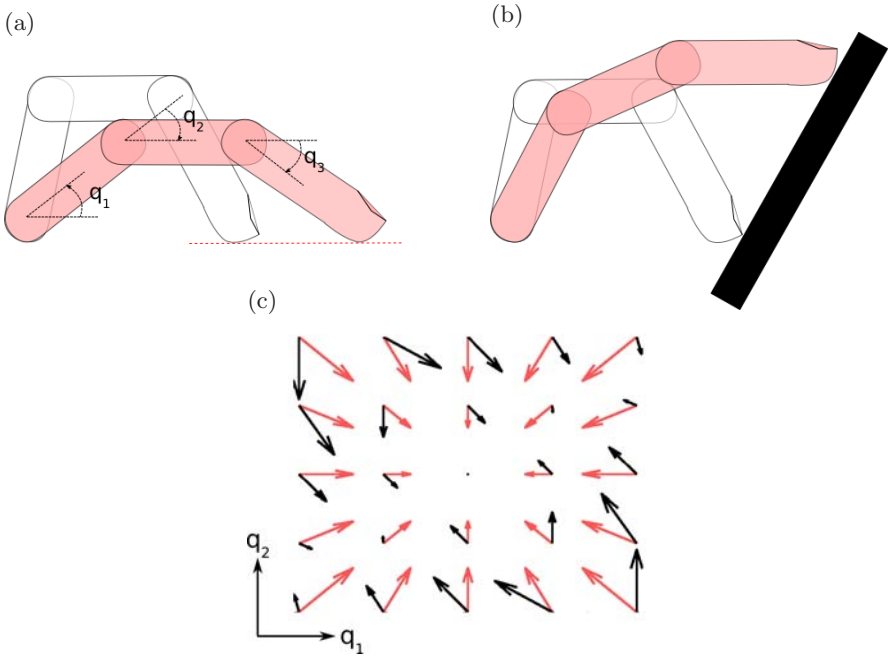


Fig. 2 Illustration of two apparently different behaviours from the same policy: (a) unconstrained movement (b) movement constrained by an obstacle (black box) (c) vector field visualisation of the unconstrained (red) and constrained (black) policy for two of the finger joints as a function of their angles.

where $\mathbf{N}(\mathbf{x}, t) \equiv (\mathbf{I} - \mathbf{A}^{\dagger}\mathbf{A}) \in \mathbb{R}^{d \times d}$ is a projection matrix that in general will have a non-linear dependence on state and time[2], $\mathbf{A}(\mathbf{x}, t) \in \mathbb{R}^{k \times d}$ is some matrix describing the constraint and $\mathbf{I} \in \mathbb{R}^{d \times d}$ is the identity matrix. Constraints of the form (2) commonly appear in scenarios where manipulators interact with solid objects, for example when grasping a tool or turning a crank or a pedal, that is, contact constraint scenarios [38, 35, 34]. Such constraints are also common in the control of redundant degrees of freedom in high-dimensional manipulators [33, 30, 39], where policies such as (3) are used, for example, to aid joint stabilisation [39], or to avoid joint limits [9], kinematic singularities [59] or obstacles [10, 29] under task constraints. As an example: Setting $\mathbf{A}$ to the Jacobian that maps from joint-space to end-effector position increments would allow any motion in the joint space provided that the end-effector remained stationary. The same formalism applies equally to policies controlling dynamic quantities such as forces and momentum, for example see [39] and [27] respectively for constrained control schemes where the formalism applies directly. It has also found more exotic

[2] Here, $\mathbf{A}^{\dagger}$ denotes the (unweighted) Moore-Penrose pseudoinverse of the matrix $\mathbf{A}$.

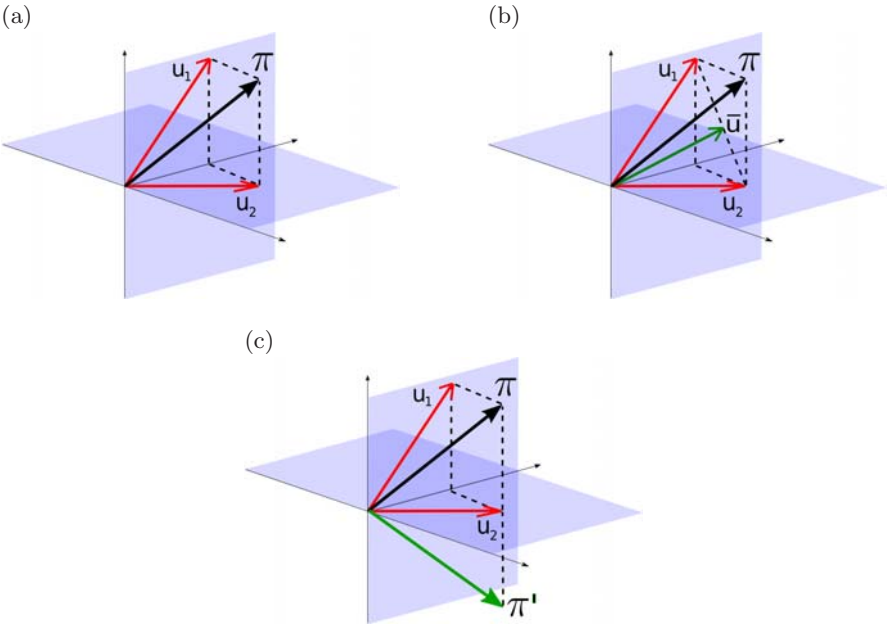


Fig. 3 Illustration of the effect of constraints on the unconstrained policy, the averaging effect of direct regression and the degeneracy problem. (a) Two constraints applied to the policy $\boldsymbol{\pi}$ result in projected observations $\mathbf{u}_1, \mathbf{u}_2$. (b) Direct regression results in averaging of the two movements $\bar{\mathbf{u}}$ in a way that cannot explain the observations. (c) Two policies $\boldsymbol{\pi}, \boldsymbol{\pi}'$ that both may be constrained in such a way as to produce the observation $\mathbf{u}_2$.

applications; for example Antonelli et al. [2] apply it to team coordination in mobile robots.

In general the effect of constraints (2)-(3) is to disallow commands in some sub-space of the system, specifically the space orthogonal to the image of $\mathbf{N}(\mathbf{x}, t)$. In essence these components of motion are *projected out* of the observed movement. For example, as illustrated in Fig. 3(a), a policy $\boldsymbol{\pi}$ is constrained in two different ways corresponding to two different projections of the unconstrained movement. In the first observation $\mathbf{u}_1$, the constraint prevents movement in the direction normal to the vertical plane[3]. For the second observation $\mathbf{u}_2$, the constraint only allows movement in the horizontal plane.

[3] Note that if the constraint has a non-linear dependence on time or state position the orientation of the constraint plane onto which the policy is projected will vary according to that dependence.

2.3 Learning from Constrained Motion Data

From the viewpoint of learning, constraints as described in the previous section present problems for traditional policy learning approaches. Specifically, there are two issues in particular that must be dealt with; that of *non-convexity* of observations and *degeneracy* between policies [20].

The *non-convexity* problem comes from the fact that between observations, or even during the course of a single observation, constraints may change. For example consider Fig. 3(b). There, the two policy observations under the different constraints, $\mathbf{u}_1$ and $\mathbf{u}_2$, appear different depending on the constraint. To the learner, this means that the data from the two scenarios will appear *non-convex*, in the sense that for any given point in the input ($\mathbf{x}$) space multiple outputs ($\mathbf{u}$) may exist. This causes problems for supervised learning algorithms, for example directly training on these observations may result in model-averaging. Here, averaging of $\mathbf{u}_1, \mathbf{u}_2$ results in the prediction $\bar{\mathbf{u}}$ that clearly does not match the unconstrained policy $\boldsymbol{\pi}$, either in direction or magnitude (ref. Fig. 3(b)).

The *degeneracy* problem stems from the fact that for any given set of projected (constrained) policy observations, there exist multiple candidate policies that could have produced that movement. This is due to the projection eliminating components of the unconstrained policy that are orthogonal to the image of $\mathbf{N}(\mathbf{x}, t)$ so that the component of $\boldsymbol{\pi}$ in this direction is undetermined by the observation. For example consider the constrained observation $\mathbf{u}_2$ in Fig. 3(c). There motion in the y direction is restricted, meaning that that component of $\boldsymbol{\pi}$ is not seen in this observation. Given only $\mathbf{u}_2$ we cannot determine if the policy $\boldsymbol{\pi}$ or an alternative, such as $\boldsymbol{\pi}'$ (ref. Fig. 3(c)) produced the observation. In effect we are not given sufficient information about the unconstrained policy to guarantee that it is fully reconstructed.

Despite these restrictions, we wish to do the best we can with the data available. We adopt a strategy whereby we look for policies that are, as a minimum, consistent with the constrained observations $\mathbf{u}$. For example, returning to Fig. 3, if we only observe $\mathbf{u}_2$, (that is the policy under a single, specific constraint) the simplest (and safest) strategy would be to use that same vector as our prediction. In this way we can at least accurately predict the policy under that constraint (albeit only under that particular constraint). If we are given further observations under new constraints we can recover more information about the unconstrained policy $\boldsymbol{\pi}$. For instance, observing $\mathbf{u}_1$ eliminates the possibility that $\boldsymbol{\pi}'$ underlies the movements since it cannot project onto both $\mathbf{u}_1$ and $\mathbf{u}_2$. Applying this strategy for increasing numbers of observations, our model will not only generalise over the constraints seen, but also come closer to the unconstrained policy $\boldsymbol{\pi}$.

Finally, it should be noted that, if in all observations, certain components of the policy are constrained, then we can never hope to uncover those components. However, in such cases it is reasonable to assume that, if these

components are always eliminated by the constraints, then they are not relevant for the scenarios in which movements were recorded.

Despite the problems that such constraints cause, in recent studies two approaches have been proposed that make considerable progress in learning in a constraint-consistent way [22, 19]. The first of these provided a solution to the problem for the important special class of *potential-based* policies[4] [19]. Using this approach it was shown that learning consistent policies can be efficiently learnt from variable-constraint data using an indirect learning approach that models the policy's underlying potential function as its representation.

Following this, a second approach was proposed, aimed at removing some of the restrictive assumptions used by the earlier potential-based approach. The key to the second method was to use a small but significant modification to the empirical risk function used by standard regression techniques. It was found that using this approach policies of arbitrary complexity, including rotational policies (i.e. policies that cannot be described by a potential) can also be efficiently learnt [22]. In the next two sections we describe the details of the two approaches.

3 Constraint-Consistent Learning of Potential-Based Policies

In this section we discuss constraint-consistent learning for the special case that the policy is potential-based. We first give a precise definition of such policies and describe the kinds of behaviour that they can be used to represent. We then go on to discuss how such policies are particularly amenable to constraint-consistent learning and describe a method recently proposed for doing this [19].

3.1 Potential-Based Policies

A potential-based policy is a policy defined through the gradient of a scalar potential function $\phi(\mathbf{x})$

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}). \tag{4}$$

Such policies can be thought of as greedily optimising the potential function at every time step [36] and thus encode attractor landscapes where the minima of the potential correspond to stable attractor points. An example is given in Fig. 4 where a potential function with three maxima (repellors) and two minima (attractors) is shown and the corresponding policy is overlaid (black vectors).

A wide variety of behaviours may be represented as potential-based. For example, reaching behaviours may be represented by a potential defined in hand space, with a single minimum at the target. Furthermore decision-based

[4] We precisely define this class of policies in the next section.

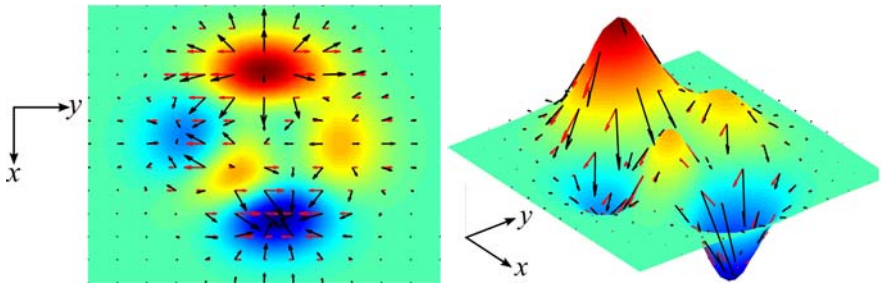


Fig. 4 Potential function with three maxima (repellors) and two minima (attractors). Overlaid are the corresponding unconstrained policy vectors (black) and a set of constrained policy vectors (red).

behaviours may be encoded as potentials [32, 31, 7, 8]. For example when reaching for an object, a potential may be defined with two minima, one corresponding to reaching with the right hand, the other reaching with the left. The decision of which hand to use for reaching would thus be determined by the start state (e.g. reach with the closest hand) and the relative offset of the two minima (e.g. right-handedness would imply a lower minimum for that hand). Potential-based policies are also extensively used as nullspace policies in control of redundant manipulators [15, 13, 9, 10, 36, 59], and for navigation and obstacle avoidance problems in mobile robotics [41, 11, 42]. Furthermore, in reinforcement learning and optimal control [52, 54], policies that are greedy with respect to the value function can be thought of as potential-based, in the sense that the policy does a gradient descent on the value function.

3.2 Learning from Constrained Potential-Based Policies

If the policy under observation is potential-based, an elegant solution to solving the non-convexity and degeneracy problems is to model the policy's *potential function* [20, 23] rather than modelling it directly. This is due to a special property of constrained potential-based policies, namely that observations of the constrained movements give us information about the shape of the underlying potential, up to a translation in ϕ corresponding to constants of integration for the observations.

In Fig. 4 this is shown for a potential function defined over a two-dimensional state-space (top and 3-D perspective views). The potential function (colours) and unconstrained policy (black vectors) is shown, along with the policy subject to a constraint (red vectors). For the case of potential-based policies the policy vectors are given by the gradient vector of the potential (as expressed in (4)). This means that the (unconstrained) policy vectors point

in the direction of steepest descent, with magnitude equal to the slope in that direction (Fig. 4, black vectors).

Now, if a constraint is applied, the direction and magnitude of the vectors change. In the example in Fig. 4 the constraint prevents movement in one dimension (x dimension in Fig. 4, left) so that only motion corresponding the the second dimension (y dimension in Fig. 4, left) is observed. The vectors now point in the direction of steepest descent *subject to the constraint*, with magnitude equal to the slope of the potential in that direction, as can be seen from Fig. 4, right. In other words the projected vectors correspond to the *directional derivatives* of the potential, in the direction parallel to the observations.

This lends us the opportunity of modelling the unconstrained policy, by piecing together information about the slope of the potential in different directions. For each observation (e.g. $\mathbf{u}_1$ in Fig. 3) we get information about the directional derivative in that direction (i.e. the direction parallel to $\mathbf{u}_1$). This means we transform the problem of combining these n-dimensional vector observations (ref. Fig. 3) to one of 'piecing together' local estimates of the slope of the potential.

A convenient method for doing this in the case of constrained kinematic policies is to use line integration to accurately estimate the form of the potential along trajectories [20, 23] and then use these local estimates to build a global model of the potential. We outline this approach in the next section.

3.3 Learning the Potential through Local Model Alignment

In the following we describe a method for modelling the potential from constrained motion data. Given observations of constrained trajectories, we first model the potential on a trajectory-wise basis using numerical line integration. We then consolidate these trajectory-wise models using results from recent work in dimensionality reduction [56, 57] to ensure consistency. Finally, we use these consistent models to learn a global model of the potential function, and thus the policy, for use in control.

Estimating the Potential Along Single Trajectories

As has been described in [20, 23], it is possible to model the potential along sampled trajectories from a constrained kinematic policy ($\mathbf{u} \equiv \dot{\mathbf{x}}$) using a form of line integration. Specifically, combining (3) and (4), the (continuous time) state evolution of the system under such policies is given by

$$\dot{\mathbf{x}} = \mathbf{N}(\mathbf{x}, t)\boldsymbol{\pi}(\mathbf{x}) = -\mathbf{N}(\mathbf{x}, t)\nabla_{\mathbf{x}}\phi(\mathbf{x}) \tag{5}$$

Let $\bar{\mathbf{x}}(t)$ be the solution of (5). If we line-integrate along $\bar{\mathbf{x}}(t)$ we have

$$\int_{\bar{\mathbf{x}}} (\nabla_{\mathbf{x}}\phi)^T \, d\mathbf{x} = \int_{t_0}^{t_f} (\nabla_{\mathbf{x}}\phi)^T \dot{\mathbf{x}} \, dt = -\int_{t_0}^{t_f} (\nabla_{\mathbf{x}}\phi)^T \mathbf{N}(\mathbf{x},t) \nabla_{\mathbf{x}}\phi(\mathbf{x}) \, dt, \quad (6)$$

where t_0 and t_f are the start and finishing instants of $\bar{\mathbf{x}}(t)$. We assume that we have recorded trajectories $\mathbf{x}(t), \dot{\mathbf{x}}(t)$ of length T sampled at some sampling rate $1/\delta t$ Hz so that for each trajectory we have a tuple of points $\mathbf{X}_k = \mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,T\delta t}$. Now, assuming the sampling rate to be sufficiently high, we can make a linear approximation to (5)

$$\mathbf{x}_{i+1} \approx \mathbf{x}_i + \delta t \, \mathbf{N}_i \boldsymbol{\pi}_i = \mathbf{x}_i - \delta t \, \mathbf{N}_i \nabla_{\mathbf{x}}\phi(\mathbf{x}_i) \quad (7)$$

and (6) can be approximated using an appropriate numerical integration scheme. An example of such a scheme is Euler integration, which involves the first order approximation

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) + \frac{1}{\delta t}(\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathbf{N}_i \nabla_{\mathbf{x}}\phi(\mathbf{x}_i). \quad (8)$$

Since the effect of the time constant δt is simply to scale the discretised policy vectors, we can neglect it by scaling time units such that $\delta t = 1$. This comes with the proviso that for implementation on the imitator robot, the learnt policy may need to be scaled back to ensure that the correct time correspondence is kept. For steps $\mathbf{x}_i \to \mathbf{x}_{i+1}$ that follow the projected policy (3) we can rearrange (7) with the scaled time coordinates, and substitute into (8) to yield

$$\phi(\mathbf{x}_{i+1}) \approx \phi(\mathbf{x}_i) - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2, \quad (9)$$

where the negative sign reflects our assumption (as expressed in (4)) that attractors are minima of the potential. We use this approximation to generate estimates $\hat{\phi}(\mathbf{x}_i)$ of the potential along any given trajectory $\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_N$ in the following way: We set $\hat{\phi}_1 = \hat{\phi}(\mathbf{x}_1)$ to an arbitrary value and then iteratively assign $\hat{\phi}_{i+1} := \hat{\phi}_i - \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2$ for the remaining points in the trajectory.

Note that an arbitrary constant can be added to the potential function without changing the policy. Therefore, 'local' potentials that we estimate along different trajectories need to be *aligned* in a way that their function value matches in intersecting regions. We'll turn to this problem in the next section.

Constructing the Global Potential Function

Let us assume we are given K trajectories $\mathbf{X}_k = (\mathbf{x}_{k1}, \mathbf{x}_{k2} \ldots \mathbf{x}_{kN_k})$ and corresponding point-wise estimates $\hat{\boldsymbol{\Phi}}_k = (\hat{\phi}_{k1}, \hat{\phi}_{k2} \ldots \hat{\phi}_{kN_k})$ of the potential, as provided from the Euler integration just described. In a first step, we fit a function model $f_k(\mathbf{x})$ of the potential to each tuple $(\mathbf{X}_k, \hat{\boldsymbol{\Phi}}_k)$, such

that $f_k(\mathbf{x}_i) \approx \hat{\phi}_{ki}$. Although in principle any regression method could be applied here, our options are somewhat limited by the fact that these possibly non-linear models have to be acquired from the few data points available in each trajectory. To avoid unnecessary complications, we choose a nearest-neighbour (NN) regression model, i.e.,

$$f_k(\mathbf{x}) = \Phi_{ki^*} \quad , \quad i^* = \arg\min_i \|\mathbf{x} - \mathbf{x}_{ki}\|^2. \tag{10}$$

Since we wish to combine the models to a global potential function, we need to define some function for weighting the outputs of the different models. For the NN algorithm, we choose to use a Gaussian kernel

$$w_k(\mathbf{x}) = \exp\left[-\frac{1}{2\sigma^2}\min_i \|\mathbf{x} - \mathbf{x}_{ki}\|^2\right]. \tag{11}$$

From these weights we can calculate responsibilities

$$q_k(\mathbf{x}) = \frac{w_k(\mathbf{x})}{\sum_{i=1}^K w_i(\mathbf{x})} \tag{12}$$

and a (naive) global prediction $f(\mathbf{x}) = \sum_{k=1}^K q_k(\mathbf{x}) f_k(\mathbf{x})$ of the potential at $\mathbf{x}$. However, as already stated, the potential is only defined up to an additive constant, and most importantly this constant can vary from one local model to another. This means that we first have to shift the models by adding some *offset* to their estimates of the potential, such that all local models are *in good agreement* about the global potential at any number of states $\mathbf{x}$.

Fortunately, a similar problem has already been tackled in the literature: In the field of non-linear dimensionality reduction, Verbeek et al. [57] have shown how to align multiple local PCA models into a common low-dimensional space. In particular, they endowed each local PCA model with an additional affine mapping $\mathbf{g}_k(\mathbf{z}) = \mathbf{A}_k\mathbf{z} + \mathbf{b}_k$, which transformed the coordinates $\mathbf{z}_k$ of a data point *within* the k-th PCA model into the desired global coordinate system. Verbeek et al. [57] retrieved the parameters of the optimal mappings $\mathbf{g}_k$ by minimising the objective function

$$E = \frac{1}{2}\sum_{m=1}^M \sum_{k=1}^K \sum_{j=1}^K q_{km} q_{jm} \|\mathbf{g}_{km} - \mathbf{g}_{jm}\|^2, \tag{13}$$

where $\mathbf{g}_{km}$ denotes the coordinate of the m-th data vector, as mapped through the k-th PCA model, and q_{km} is the corresponding responsibility of that model. The objective can easily be interpreted as the 'disagreement' between any two models, summed up over all data points, and weighted by the responsibilities of two models each. That is, the disagreement for any combination of m, k and j only really counts, if the responsibility of both the k-th and the j-th model is sufficiently high for the particular query point.

Notably, E is convex and can be minimised by solving a generalised eigenvalue problem of moderate dimensions, that is, there are no local minima, and the solution can be found efficiently.

In analogy to the PCA-alignment method [57], we augment our local potential models $f_k(\cdot)$ by a scalar offset b_k and define the corresponding objective function as

$$E(b_1 \dots b_K) = \frac{1}{2} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{j=1}^{K} q_k(\mathbf{x}_m) q_j(\mathbf{x}_m) \times \left((f_k(\mathbf{x}_m) + b_k) - (f_j(\mathbf{x}_m) + b_j) \right)^2, \tag{14}$$

or, in a slightly shorter form,

$$E(\mathbf{b}) = \frac{1}{2} \sum_{m,k,j} q_{km} q_{jm} \left(f_{km} + b_k - f_{jm} - b_j \right)^2. \tag{15}$$

Here, $\sum_m$ denotes a summation over the complete data set, that is, over all points from all trajectories ($M = \sum_{k=1}^{K} N_k$). Using the symmetry in $j \leftrightarrow k$ and $\sum_k q_{kn} = 1$, we split (15) into terms that are constant, linear, or quadratic in b_k, yielding

$$\begin{aligned} E(\mathbf{b}) &= \sum_{m,k} q_{km} f_{km}^2 - \sum_{m,j,k} q_{km} q_{jm} f_{km} f_{jm} \\ &\quad + 2 \sum_{m,k} q_{km} f_{km} b_k - 2 \sum_{m,k} q_{km} q_{jm} f_{jm} b_k \\ &\quad + \sum_{m,k} q_{km} b_k^2 - \sum_{m,k,j} q_{km} q_{jm} b_k b_j \\ &= E_0 + 2\mathbf{a}^T \mathbf{b} + \mathbf{b}^T \mathbf{H} \mathbf{b}. \end{aligned} \tag{16}$$

Here, we introduced E_0 as a shortcut for the terms independent of $\mathbf{b}$, the vector $\mathbf{a} \in \mathbb{R}^K$ with elements $a_k = \sum_m q_{km} f_{km} - \sum_{m,j} q_{km} q_{jm} f_{jm}$, and the Hessian matrix $\mathbf{H} \in \mathbb{R}^{K \times K}$ with elements $h_{ij} = \delta_{ij} \sum_m q_{jm} - \sum_m q_{im} q_{jm}$. The objective function is quadratic in $\mathbf{b}$, so we retrieve the optimal solution by setting the derivatives to zero, which yields the equation $\mathbf{H}\mathbf{b} = -\mathbf{a}$.

However, note that a common shift of all offsets b_k does not change the objective (14), which corresponds to the shift-invariance of the global potential. Therefore, the vector $(1, 1, \dots, 1)^T$ spans the nullspace of $\mathbf{H}$, and we need to use the pseudo-inverse of $\mathbf{H}$ to calculate the optimal offset vector

$$\mathbf{b}_{opt} = -\mathbf{H}^{\dagger} \mathbf{a}. \tag{17}$$

Compared to aligning PCA models, the case we handle here is simpler in the sense that we only need to optimise for scalar offsets b_k instead of affine mappings. On the other hand, our local potential models are non-linear, have

to be estimated from relatively little data, and therefore do not extrapolate well, as will be discussed in the following section.

Smoothing Parameter Selection

Since we restrict ourselves to using simple NN regression for the local potential models, the only open parameter of the algorithm is σ^2, i.e., the kernel parameter used for calculating the responsibilities (11). A too large choice of this parameter will over-smooth the potential, because the NN regression model basically predicts a locally constant potential, but at the same time trajectories will have relatively high responsibilities even for far apart points $\mathbf{x}$ in state space.

On the other hand, a too small value of σ^2 might lead to *weakly connected trajectories*: If a particular trajectory does not make any close approach to other trajectories in the set, the quick drop-off of its responsibility implies that it will not contribute to the alignment error (based on pairs of significant responsibility), which in turn implies that its own alignment – the value of its offset – does not matter much. The same reasoning applies to groups of trajectories that are close to each other, but have little connection to the rest of the set.

Such trajectories can cause problems when attempting to learn a global model of the potential using the output of the optimisation (17), since if their influence on the overall alignment is negligible, their own alignment can be poor. Fortunately, this situation can be detected automatically by looking for small eigenvalues of $\mathbf{H}$: In the same way as adding the same offset to all trajectories leads to a zero eigenvalue, further very small eigenvalues and the corresponding eigenvectors indicate indifference towards a shift of some subset of trajectories versus the rest of the set. In practice, we look for eigenvalues $\lambda < 10^{-8}$, and use a recursive bi-partitioning algorithm in a way that is very similar to spectral clustering [28] (please refer to [19] for details on this step). We can then either discard all trajectories apart from those in the largest 'connected' group (treating the weakly connected trajectories as outliers) or recursively repeat the alignment process on the larger groups of aligned trajectories.

Finally, with these considerations in mind, we select the smoothing parameter σ^2 to match the scale of typical distances in the data sets. In the experiments presented in [19] this parameter was selected heuristically by first calculating the distances between any two trajectories $k, j \in \{1 \dots K\}$ in the set as the distances between their closest points

$$d_{kj} = \min \left\{ \|\mathbf{x}_{kn} - \mathbf{x}_{jm}\|^2 \mid n, m \in \{1 \dots N\} \right\}, \tag{18}$$

and also the distances to the closest trajectory

$$d_k^{min} = \min \left\{ d_{kj} \mid j \neq k \right\}. \tag{19}$$

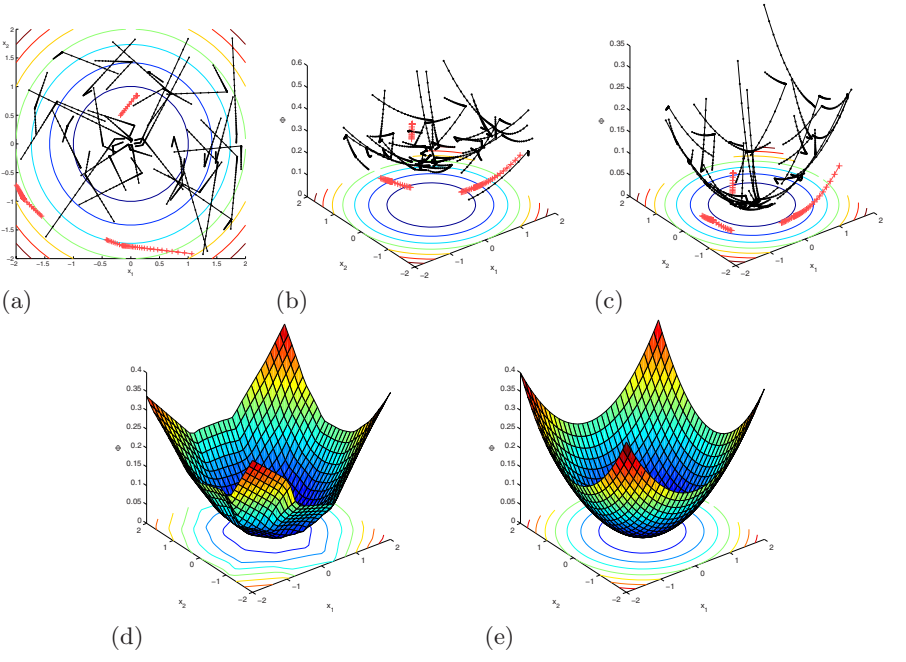


Fig. 5 The alignment algorithm at work for a set of $K = 40$ trajectories of length $N = 40$ sampled from a parabolic potential ($\phi(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x}$, $\mathbf{W} = 0.05\mathbf{I}$) with randomly switching constraints ($\mathbf{A}(\mathbf{x}, t) = (\alpha_1, \alpha_2) \equiv \boldsymbol{\alpha}$, $\alpha_i = N(0, 1)$). (a) Raw data (constrained trajectories through the two-dimensional state space) and contour of the potential. (b) Local potential models estimated by Euler integration prior to alignment. (c) Local potential models after alignment, already revealing the structure of the parabola. (d) Global model $f(\mathbf{x})$ trained on the aligned trajectories (here, trained with LWPR [58]). (e) True parabolic potential shown for comparison. The weakly connected 'outlier' trajectories (here, discarded prior to learning the global model) are shown in red.

Then three choices for σ^2, were considered, referred to as 'narrow', 'wide' and 'medium':

$$\sigma^2_{nar} = \text{median}\left\{d^{min}_k \mid k \in \{1 \dots K\}\right\} \tag{20}$$

$$\sigma^2_{wid} = \text{median}\left\{d_{jk} \mid j, k \in \{1 \dots K\}, j \neq k\right\} \tag{21}$$

$$\sigma^2_{med} = \sqrt{\sigma^2_{nar}\sigma^2_{wid}}. \tag{22}$$

As shown by experiment [19, 20], the choice σ^2_{med} usually represents a reasonable balance between smoothing and alignment performance.

Algorithm 1. PolicyAlign

1: Estimate $\mathbf{X}_k$, $\hat{\mathbf{\Phi}}_k, \{k = 1 \dots K\}$ using Euler integration. Calculate σ^2.

2: Alignment:

- Calculate prediction and responsibility of each local model f_k on each data point $\mathbf{x}_m,\ m = 1 \dots M$:
  $f_{km} = f_k(\mathbf{x}_m);\ \ q_{km} = w_k(\mathbf{x}_m) / \sum_i w_i(\mathbf{x}_m)$
- Construct $\mathbf{H}, \mathbf{a}$ with elements
  $h_{ij} = \delta_{ij} \sum_m q_{jm} - \sum_m q_{im} q_{jm};\ \ a_k = \sum_m q_{km} f_{km} - \sum_{m,j} q_{km} q_{jm} f_{jm}$
- Find optimal offsets $\mathbf{b}_{opt} = -\mathbf{H}^{\dagger}\mathbf{a}$

3: Discard outliers ($\mathbf{H}$ eigenvalues, $\lambda < 10^{-8}$).

4: Train global model on data tuples $(\mathbf{x}_{kn}, \hat{\phi}_{kn} + b_k^{opt})$

Learning the Global Model

After calculating optimal offsets $\mathbf{b}_{opt}$ and cleaning the data set from outliers, we can learn a global model $f(\mathbf{x})$ of the potential using any regression algorithm. For example, in the experiments presented in Sec. 5, we will use Gaussian Radial Basis Function (RBF) models, and in [19, 20] Locally Weighted Projection Regression (LWPR) [58] was used. As the training data for these models, we use all non-outlier trajectories and their estimated potentials as given by the Euler integration *plus* their optimal offset, that is, the input-output tuples

$$\left\{ (\mathbf{x}_{kn}, \hat{\phi}_{kn} + b_k^{opt}) \mid k \in \mathcal{K}, n \in \{1 \dots N_k\} \right\}, \tag{23}$$

where $\mathcal{K}$ denotes the set of indices of non-outlier trajectories. Once we have learnt the model $f(\mathbf{x})$ of the potential, we can take derivatives to estimate the unconstrained policy $\hat{\boldsymbol{\pi}}(\mathbf{x}) = -\nabla_{\mathbf{x}} f(\mathbf{x})$. For convenience, the complete procedure is summarised in Algorithm 1 and illustrated pictorially in Fig. 5 for an example parabolic potential with randomly switching constraints.

4 Constraint-Consistent Learning of Generic Policies

In the previous section we outlined a method for learning in a constraint-consistent manner based on indirect modelling of the policy through its potential. As discussed in Sec. 3.2, potential-based policies are particularly amenable to learning in the constrained setting since observations under different constraints correspond to the directional derivatives of the potential in the different (unconstrained) directions. This allows us to model the shape of the potential to find a policy that is consistent with the observations.

While this approach has been shown to greatly outperform direct learning approaches in an number of experiments on constrained systems [19], it still suffers from some restrictions due to the assumption of a potential-based policy. While this is not a problem when learning from systems such as those described in Sec. 3.1, it can cause difficulties when the policy under observation encodes periodic or rotational behaviour (precisely speaking, when the *curl* of the observed policy is non-zero).

In order to avoid this restriction an alternative approach to learning must be taken. In [22], a new method was proposed that enables learning of generic policies from variable constraint data. This method was based on a small but significant modification of the empirical risk function used for learning. In the following we consider several candidate risk functions that could be used for learning and assess their suitability with respect to the data we are assumed given. We will then discuss the novel risk function proposed in [22] that both satisfies our original assumptions, and has been shown to be effective for learning from variable constraint data [22].

4.1 Optimisation of the Standard Risk, UPE and CPE

As already outlined in Sec. 2.3, throughout this chapter we are targeting problems where we are given data in the form of tuples $(\mathbf{x}_n, \mathbf{u}_n)$ of observed states and constrained actions, where we assume that all commands $\mathbf{u}$ are generated from some underlying policy $\boldsymbol{\pi}(\mathbf{x})$, which for a particular observation might have been constrained. For constrained systems (2)-(3), this means that we observe $\mathbf{u}_n = \mathbf{N}_n\boldsymbol{\pi}(\mathbf{x}_n)$ for some projection matrix $\mathbf{N}_n$. We assume that the projection matrix for any given observation is not explicitly known, i.e. our data is unlabelled with respect to the constraints in force at the time of observation.

Given this data, the first possibility that springs to mind is to perform direct least-squares regression for learning. In this approach one would attempt to estimate the policy $\tilde{\boldsymbol{\pi}}(\cdot)$ by minimising the *standard risk*

$$E_{direct}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2. \tag{24}$$

As already mentioned in Sec. 2 this is an effective approach for learning from unconstrained data or data where the same constraint appears in all observations (i.e. the constraint matrix $\mathbf{A}(\mathbf{x}, t)$ is the same static function of state for all observations). In the former case, one would obtain the best fit to the *unconstrained policy*, and in the latter one would find the best fit to the *constrained policy under that particular set of constraints*. For example if one had several observations of an system opening some particular door and

in every observation the door was the same, then optimisation of (24) would be effective for learning the policy for that particular door.

The problem with this approach, however, is that it cannot handle data where commands are observed under variable constraints. As discussed in Sec. 2.3, if we consider an example where multiple observations are given under different constraints, optimisation of (24) would result in a naive averaging of commands from different circumstances (cf. Fig. 3(b)). In terms of our door-opening example, if we observed the agent opening a new door and attempted to incorporate that into our policy model, we would either get the average door opening action, or have to start a new policy model for the new door. We can therefore rule out (24) for learning in this setting, since it does not meet our requirements for accuracy and generalisation.

An alternative approach then, might be to target error measures that directly measure performance in terms of our objectives. For example, we could attempt to optimise our model with respect either to the *unconstrained policy error* (UPE)

$$E_{upe}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\boldsymbol{\pi}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2 \tag{25}$$

or the *constrained policy error* (CPE)

$$E_{cpe}[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \mathbf{N}_n\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2. \tag{26}$$

Optimising the former would directly give us the best fit to the policy, while optimising the latter would give the best fit that is consistent with the constrained observations. The optimal model with respect to either of these would satisfy our accuracy and generalisation requirements. For example in the former case we could apply any projection (constraint) to the model and still achieve a good CPE under the new constraint.

However, the problem here is that by assumption we do not have access to samples of either (i) the (unconstrained) policy $\boldsymbol{\pi}_n = \boldsymbol{\pi}(\mathbf{x}_n)$, or (ii) the projection matrices $\mathbf{N}_n$ needed for calculating these quantities. This is because in most problems of interest, *constraints are not directly observable* and there is *ambiguity* in what features of motion are due to constraints and what are implicit in the policy itself.

For example consider a contact control scenario such as wiping a window. There, we can identify the surface of the window as an *environmental constraint*[5] preventing the wiping hand from penetrating the surface. We may also identify a task constraint preventing the hand from lifting from the surface, since contact must be maintained for successful wiping. However, while this is one reasonable analysis of the system, there also exist other

[5] Note that would in fact be an inequality constraint since only movement into the surface is restricted, while movement away is unconstrained.

possibilities. For example, it may be that the *unconstrained policy itself* exactly encodes a wiping movement parallel to the surface, so that the presence of the surface is incidental. Alternatively, there could be an *additional task constraint* applied that prevents the hand from pressing hard against the surface. Note that we cannot directly determine which is the correct analysis simply by observing the given movement: If the window surface (environmental constraint) was removed in both of these cases the wiping would still appear to go on exactly as before. In this example then, there is ambiguity in what features of movement are due to the policy, what are due to the constraints, and exactly what constraints (if any) are in force. Since none of these can be resolved by the given observations, we cannot in general use either of these functionals for learning.

4.2 Learning Generic Policies by Minimising Inconsistency

Having ruled out the use of (24)-(26) for learning in this setting we must look for alternative approaches. Our aim is to try to estimate a policy $\tilde{\boldsymbol{\pi}}(\cdot)$ that is *consistent* with our observed $\mathbf{u}_n$, only using quantities that we can derive from the data. That is, we wish to reconstruct the policy, knowing that it may be projected in some way by the constraints. At this point a key observation can be made: in order to uncover the unconstrained policy we must find a policy model that can be *projected in such a way that the observed commands are recovered*. In other words, we require

$$\mathbf{u}(\mathbf{x}) := \mathbf{P}\boldsymbol{\pi}(\mathbf{x})$$

for an appropriate projection matrix $\mathbf{P}$, that either projects onto the same space as the (unknown) $\mathbf{N}(\mathbf{x})$ (i.e. the image of $\mathbf{N}$), or an (even smaller) subspace of that. One such projection, which we know to lie within this subspace, is the 1-D projection onto the observed command itself, that is $\mathbf{P} = \hat{\mathbf{u}}\hat{\mathbf{u}}^T$, with $\hat{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|$ (ref. Fig. 6). Furthermore, since $\mathbf{u}$ is given, we have all the information we need to calculate this projection and use it for learning, neatly side-stepping the need to explicitly model the full constraint matrix $\mathbf{N}$.

With this as motivation, we propose to replace $\mathbf{N}_n$ in (26) by a projection onto $\mathbf{u}_n$ and minimise the *inconsistency* which we define as the functional

$$E_i[\tilde{\boldsymbol{\pi}}] = \sum_{n=1}^{N} \|\mathbf{u}_n - \hat{\mathbf{u}}_n\hat{\mathbf{u}}_n^T\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2 = \sum_{n=1}^{N} \left(r_n - \hat{\mathbf{u}}_n^T\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\right)^2 \qquad (27)$$

with $r_n = \|\mathbf{u}_n\|$, $\hat{\mathbf{u}}_n = \frac{\mathbf{u}_n}{r_n}$. Since $\mathbf{u}_n = \mathbf{N}_n\boldsymbol{\pi}_n$ we can write $\|\mathbf{u}_n - \mathbf{N}_n\tilde{\boldsymbol{\pi}}(\mathbf{x}_n)\|^2 = \|\mathbf{N}_n(\boldsymbol{\pi}_n - \tilde{\boldsymbol{\pi}}(\mathbf{x}_n))\|^2$ and recognise that the CPE is always less than or equal to the UPE, because the projections $\mathbf{N}_n$ can only decrease

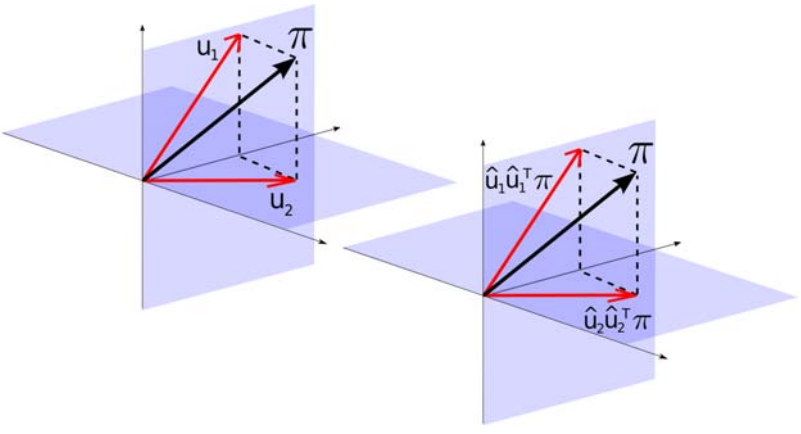


Fig. 6 Illustration of our learning scheme. The projection of the correct policy $\boldsymbol{\pi}$ onto the observations matches those observations.

the norm of the difference between true and predicted policy. The same argument holds for the inconsistency error (27) where the projection onto the 1-D subspace spanned by $\hat{\mathbf{u}}_n$, possibly takes away even more of the error. So we can establish the inequality

$$E_i[\tilde{\boldsymbol{\pi}}] \leq E_{cpe}[\tilde{\boldsymbol{\pi}}] \leq E_{upe}[\tilde{\boldsymbol{\pi}}].$$

Naturally, for estimating the correct policy, we would rather like to minimise an *upper bound* of E_{upe}, but it is unclear how such a bound could be derived from the data we are assumed given. However, note that by framing our learning problem as a risk minimisation task, we can apply standard regularisation techniques such as adding suitable penalty terms to prevent over-fitting due to noise.

The proposed risk functional (27) can be used in conjunction with many standard regression techniques. In the following we derive training rules for two classes of function approximator for learning the (unconstrained) policy to demonstrate how the risk functional can be used. The example function approximators we use are (i) simple parametric models with fixed basis functions (Sec. 4.3), and (ii) locally linear models (Sec. 4.4). We turn to this in the next section.

4.3 Parametric Policy Models

A particularly convenient model of the policy is given by $\tilde{\boldsymbol{\pi}}(\mathbf{x}) = \mathbf{W}\mathbf{b}(\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{d \times M}$ is a matrix of weights, and $\mathbf{b}(\mathbf{x}) \in \mathbb{R}^M$ is a vector of fixed basis functions. This notably includes the case of (globally) linear models where we set $\mathbf{b}(\mathbf{x}) = \bar{\mathbf{x}} = (\mathbf{x}^T, 1)^T$, or the case of normalised radial basis

functions (RBFs) $b_i(\mathbf{x}) = \frac{K(\mathbf{x}-\mathbf{c}_i)}{\sum_{j=1}^{M} K(\mathbf{x}-\mathbf{c}_j)}$ calculated from Gaussian kernels $K(\cdot)$ around M pre-determined centres $\mathbf{c}_i$, $i = 1 \dots M$. With this model, the *inconsistency* error from (27) becomes

$$\begin{aligned} E_i(\mathbf{W}) &= \sum_{n=1}^{N} \left(r_n - \hat{\mathbf{u}}_n^T \mathbf{W} \mathbf{b}(\mathbf{x}_n)\right)^2 \\ &= \sum_{n=1}^{N} \left(r_n - \mathbf{v}_n^T \mathbf{w}\right)^2 = E_i(\mathbf{w}), \end{aligned}$$

where we defined $\mathbf{w} \equiv vec(\mathbf{W})$ and $\mathbf{v}_n \equiv vec(\hat{\mathbf{u}}_n \mathbf{b}(\mathbf{x}_n)^T) = \mathbf{b}(\mathbf{x}_n) \otimes \hat{\mathbf{u}}_n$ in order to retrieve a simpler functional form. Since our objective function is quadratic in $\mathbf{w}$, we can solve for the optimal weight vector easily:

$$\begin{aligned} E_i(\mathbf{w}) &= \sum_n r_n^2 - 2 \sum_n r_n \mathbf{v}_n^T \mathbf{w} + \mathbf{w}^T \sum_n \mathbf{v}_n \mathbf{v}_n^T \mathbf{w} \\ &= E_0 - 2\mathbf{g}^T \mathbf{w} + \mathbf{w}^T \mathbf{H} \mathbf{w} \end{aligned}$$

yielding

$$\mathbf{w}^{opt} = \arg\min E_i(\mathbf{w}) = \mathbf{H}^{-1}\mathbf{g} \tag{28}$$

with $\mathbf{H} = \sum_n \mathbf{v}_n \mathbf{v}_n^T$ and $\mathbf{g} = \sum_n r_n \mathbf{v}_n$. For regularisation, we use a simple weight-decay penalty term, that is, we select $\mathbf{w}_{reg}^{opt} = \arg\min(E_i(\mathbf{w}) + \lambda \|\mathbf{w}\|^2)$. This only requires modifying the Hessian to $\mathbf{H}^{reg} = \sum_n \mathbf{v}_n \mathbf{v}_n^T + \lambda \mathbf{I}$.

Please note that the projection onto $\mathbf{u}$ introduces a coupling between the different components of $\tilde{\boldsymbol{\pi}}$, which prevents us from learning those independently as is common in normal regression tasks. For the same reason, the size of the Hessian scales with $O(d^2 M^2)$.

4.4 Locally Linear Policy Models

The basis function approach quickly becomes non-viable in high-dimensional input spaces. Alternatively, we can fit multiple locally weighted linear models $\tilde{\boldsymbol{\pi}}_m(\mathbf{x}) = \mathbf{B}_m \bar{\mathbf{x}} = \mathbf{B}_m (\mathbf{x}^T, 1)^T$ to the data, learning each local model independently [46]. For a linear model centred at $\mathbf{c}_m$ with an isotropic Gaussian receptive field with variance σ^2, we would minimise

$$\begin{aligned} E_i(\mathbf{B}_m) &= \sum_{n=1}^{N} w_{nm} \left(r_n - \hat{\mathbf{u}}_n^T \mathbf{B}_m \bar{\mathbf{x}}_n\right)^2 \\ &= \sum_{n=1}^{N} w_{nm} \left(r_n - \mathbf{v}_n^T \mathbf{b}_m\right)^2 = E_i(\mathbf{b}_m), \end{aligned}$$

where we defined $\mathbf{b}_m = vec(\mathbf{B}_m)$ and $\mathbf{v}_n \equiv vec(\hat{\mathbf{u}}_n \bar{\mathbf{x}}_n^T)$ similarly to the parametric case. The factors

$$w_{nm} = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_n - \mathbf{c}_m\|^2)$$

weight the importance of each observation $(\mathbf{x}_n, \mathbf{u}_n)$, giving more weight to nearby samples. The optimal slopes $\mathbf{B}_m$ in vector form are retrieved by

$$\mathbf{b}_m^{opt} = \arg\min E_i(\mathbf{b}_m) = \mathbf{H}_m^{-1}\mathbf{g}_m \tag{29}$$

with $\mathbf{H}_m = \sum_n w_{nm}\mathbf{v}_n\mathbf{v}_n^T$ and $\mathbf{g}_m = \sum_n w_{nm}r_n\mathbf{v}_n$.

For predicting the global policy, we combine the local linear models using the convex combination

$$\tilde{\boldsymbol{\pi}}(\mathbf{x}) = \frac{\sum_{m=1}^{M} w_m \mathbf{B}_m \bar{\mathbf{x}}}{\sum_{m=1}^{M} w_m}$$

where $w_m = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{c}_m\|^2\right)$. For extensive experiments assessing the performance of learning with parametric and local linear models using the novel risk function, we refer the reader to the original experiments reported in [22, 21].

5 Constraint-Consistent Learning Performance

To explore the performance of the two approaches, a number of experiments have been performed, learning on data from autonomous kinematic control policies from different plants, including learning from human demonstration data to enable the ASIMO humanoid robot to learn a realistic car washing task [22, 19]. In this section, we briefly review some of these results to provide the reader with a view of the comparative performance the two approaches. For this, we first discuss learning on data from a simple, two-dimensional system controlled according to the framework outlined in Sec. 2. We then discuss an example scenario in which the algorithm is used to enable ASIMO to learn a realistic bi-manual grasping task from observations from a constrained demonstrator. We then give a brief discussion on how constraint-consistent learning has been applied for learning from human demonstration data for transferring skills to the ASIMO humanoid.

5.1 Two-Dimensional Constrained System

In this section we compare the performance of the constraint-consistent learning approaches described in Sec. 3 and Sec. 4 on a simple two-dimensional system $(\mathbf{x}, \mathbf{u} \equiv \dot{\mathbf{x}} \in \mathbb{R}^2$ with policies subject to discontinuously switching constraints. Specifically, the constraints are given by

$$\mathbf{A}(\mathbf{x}, t) = (\alpha_1, \alpha_2) \equiv \boldsymbol{\alpha} \tag{30}$$

where the $\alpha_{1,2}$ are drawn from a normal distribution, $\alpha_i = N(0,1)$. Here, the constraints mean that motion is constrained in the direction orthogonal to the vector $\boldsymbol{\alpha}$ in state space. To increase the complexity of the problem, the constraints are randomly switched during trajectories by re-sampling $\boldsymbol{\alpha}$ twice at regular intervals during the trajectory. This switches the direction in which motion is constrained, causing sharp turns in the trajectories (for example, see Fig. 5(a)).

To compare the methods, we investigate learning on data from three policies of differing complexity. These were (i) a policy defined by a quadratic potential function

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi_q(\mathbf{x}) = (\mathbf{x}-\mathbf{x}_c)^T\mathbf{W}(\mathbf{x}-\mathbf{x}_c) \tag{31}$$

where we chose $\mathbf{x}_c = 0$ and $\mathbf{W} = \alpha\mathbf{I}$; (ii) a sinusoidal potential

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi_s(\mathbf{x}) = \alpha\sin(x_1)\cos(x_2), \tag{32}$$

where we set $\alpha = 0.1$ and (iii) a limit cycle policy

$$\dot{r} = r(\rho - r^2), \quad \dot{\theta} = \omega. \tag{33}$$

where r, θ are the polar representation of the Cartesian state space coordinates (i.e. $x_1 = r\sin\theta$, $x_2 = r\cos\theta$), ρ is the radius of the attractor and $\dot{\theta}$ is the angular velocity. For the experiments, the latter two parameters were set to $\rho = 0.5\ m$ and $\omega = 1\ rad\ s^{-1}$. Here, the two potential-based policies act as attractor landscapes with, for example, a single point attractor at $\mathbf{x}_c$ for the quadratic potential and multiple point attractors for the sinusoidal potential. Note that the limit cycle policy is a rotational policy and therefore cannot be defined by a potential.

Data was collected by sampling $K = 40$ trajectories at a rate of 50 Hz from the policies, starting from random states. This resulted in $N = 40$ data points per trajectory. We then attempted to learn on this data using (i) the alignment approach (ref. Sec. 3), (ii) optimisation of the inconsistency (ref. Sec. 4), and (iii) direct regression (i.e. training directly on the tuples $(\mathbf{x}_i, \mathbf{u}_i), i = 1, \ldots K \times N$ and optimising the risk function (24)). For each of the methods we learnt models consisting of a set of normalised Gaussian RBFs with centres arranged on a 6×6 grid, and with the kernel widths fixed to yield suitable overlap. For the latter two approaches the model represented the mapping $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \mathbf{u} \in \mathbb{R}^2 \mapsto \mathbb{R}^2$ and for the alignment approach it represented the mapping $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \phi \in \mathbb{R}^2 \mapsto \mathbb{R}$. For a thorough comparison, we learnt models of each of the three policies using the three different learning approaches. We repeated this experiment on 100 data sets and evaluated the normalised UPE and CPE, that is, the functionals from (25) and (26) divided by the number of data points and the variance of the policy $\boldsymbol{\pi}_n$ on a subset held out for testing.

Table 1 Normalised UPE and CPE for learning the three policies from the toy example using (i) direct regression, (ii) the alignment approach and (iii) the inconsistency approach. All errors are (mean±s.d.)$\times 10^{-2}$ over 100 data sets.

Policy	Alg.	nUPE	nCPE
Quad. Pot.	direct	54.727 ± 6.218	10.732 ± 2.010
	align.	1.158 ± 1.561	0.443 ± 0.588
	incon.	0.001 ± 0.001	0.001 ± 0.001
Sin. Pot	direct	40.478 ± 4.789	12.354 ± 1.097
	align.	5.020 ± 5.395	2.162 ± 2.536
	incon.	0.003 ± 0.003	0.001 ± 0.004
Lim. Cyc.	direct	43.225 ± 6.599	10.034 ± 1.678
	align.	291.233 ± 156.180	126.902 ± 80.364
	incon.	0.024 ± 0.040	0.003 ± 0.002

Table 1 summarises the results. Firstly, looking at the results for using direct regression to learning the three policies, we see uniformly poor performance both in terms of the normalised UPE and CPE. Because the direct approach to learning is naive to the effect of the constraints, model averaging results. This causes the predictions for each of the three polices to be poor, even under the training constraints.

In contrast to this, looking at the results for the potential-based policies, the alignment approach performs approximately an order of magnitude better both in terms of the UPE and the CPE. Comparing errors for the quadratic and sinusoidal potential-based policies we also see that the latter, more complex potential (with multiple sinks) is more difficult to learn with a data set of this size. However, as expected, the alignment approach performs very badly when training on the limit cycle data: The potential-based representation is not appropriate in this case since the policy is rotational.

Looking at the errors for the inconsistency approach, however, we see improved performance on all three policies, including the rotational limit cycle data. Comparing results for the three policies we see that the sinusoidal potential-based policy and the limit-cycle policy are more difficult to learn due to their increased complexity. However, despite this, the increase in error for this approach is still relatively small.

Finally, in all of the results, the nCPE is always much lower than the nUPE, which follows naturally from the fact that the projection induced by the constraints projects out some of the error in the models, as discussed in Sec. 4. Still, the results show that with a reasonable amount of data, the unconstrained policy can be modelled with remarkable accuracy using the two constraint-consistent approaches.

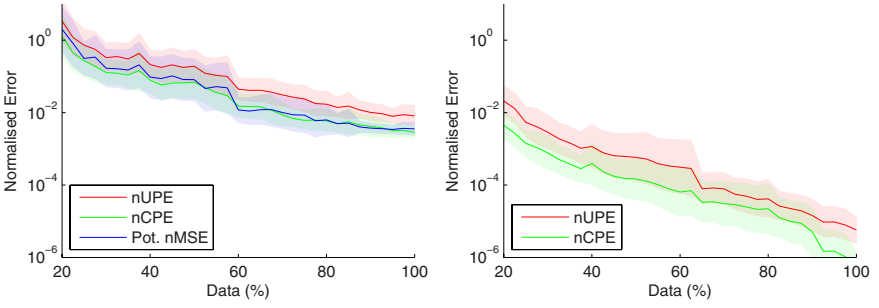


Fig. 7 Learning performance on the quadratic potential (31) with varying data set sizes for the alignment approach (left) and the inconsistency approach (right). Normalised CPE and UPE versus data set size as a percentage of the full $K = 40$ trajectories of length $N = 40$ are shown. For the alignment approach the normalised error in the potential is also plotted.

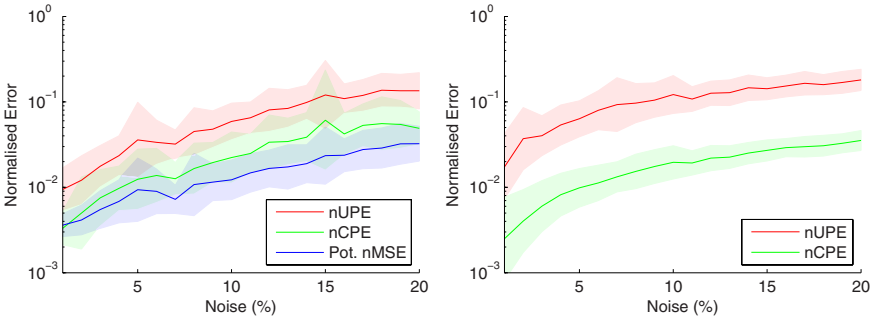


Fig. 8 Learning performance on the quadratic potential (31) with varying noise levels for the alignment approach (left) and the inconsistency approach (right). Normalised CPE and UPE versus noise in the observed $(\mathbf{x}_n, \mathbf{u}_n)$ as a percentage of the variance of the data are shown. For the alignment approach the normalised error in the potential is also plotted.

As a further test, we can also look at the performance in terms of the amount of data required to find a good approximation. In Fig. 7 we show the error curves for the two constraint-consistent learning approaches when learning on different-sized subsets of the data from the quadratic potential (31). As can be seen (ref. Fig. 7), the performance of the two approaches improves with increasing quantities of data in terms of UPE and CPE, with the inconsistency approach generally achieving a lower error than the alignment approach for this data set.

Finally, in Fig. 8 we characterise the noise robustness of the two constraint-consistent approaches when learning, again using the same data, but this time with the the observed states $\mathbf{x}_n$ and actions $\mathbf{u}_n$ contaminated with Gaussian

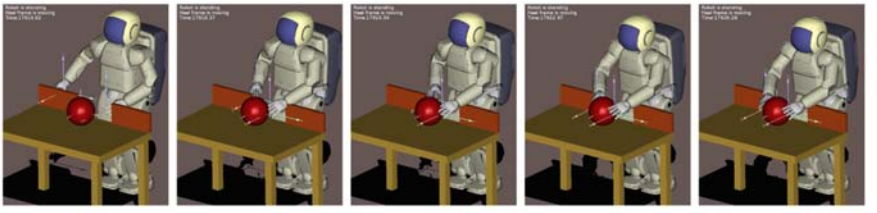


Fig. 9 Example constrained trajectory used as training data in the ball-reaching experiment. Starting with hands at the sides, the demonstrator robot reaches between the barriers to get the ball. Note that the width of the gap in the barriers was randomly altered for each trajectory recorded.

noise, the scale of which was varied to match up to 20% of the scale of the data. We see that the performance of the two approaches approximately follows the noise levels in the data (ref. Fig. 8), although there is slightly more variance in the performance of the alignment approach. This latter effect can be explained by the fact that the alignment uses the nearest neighbour trajectories for the alignment, the measurement of which becomes increasingly unreliable as the noise in $\mathbf{x}_n$ increases. However, despite this, the performance of the two approaches can be seen to decline smoothly as the amount of noise increases.

5.2 Reaching for a Ball

In this section we characterise (i) how well the two approaches scale to more complex, realistic constraints and policies and (ii) how well the policies learnt with these approaches generalise over different constraints. For this, we use data from an example scenario, in which a set of observations of a demonstrator performing the task of reaching for a ball on a table are given, and the student is expected to learn a policy to enable it to reproduce this task [22, 19]. The learning problem is complicated however, by the presence of different obstacles on the table for each of the example trajectories, constraining the possible motions of the hands. The goal is to uncover a policy that accurately predicts the demonstrator's (unconstrained) behaviour and generalises to predict the behaviour under novel constraints.

The example scenario was implemented [22, 19] using the whole body motion (WBM) controller of the 27-DOF humanoid robot ASIMO (for details on the controller see [15]). For this, data was recorded from a 'demonstrator' robot that followed a policy defined by an inverted Gaussian potential

$$\boldsymbol{\pi}(\mathbf{x}) = -\nabla_{\mathbf{x}}\phi(\mathbf{x}); \quad \phi(\mathbf{x}) = \alpha\left(1 - e^{\|\mathbf{x}-\mathbf{x}_c\|^2/2\sigma^2}\right), \tag{34}$$

where $\mathbf{x} \in \mathbb{R}^6$ corresponds to the Cartesian position of the two hands (hereafter, the 'task space') and the actions $\mathbf{u} = \dot{\mathbf{x}} = \boldsymbol{\pi}(\mathbf{x})$ correspond to the hand velocities. We chose $\sigma^2 = 2$, $\alpha = 0.25$ and the target point $\mathbf{x}_c \in \mathbb{R}^6$ to correspond to a reaching position, with the two hands positioned on either side of the ball. Following the policy (34) with this set of parameters, the demonstrator was able to reach the ball under each of the constraints considered in this experiment (see below). Inverse kinematics via the WBM controller was used to map the desired task space policy motion into the appropriate joint-space velocity commands for sending to the robot.

The demonstrator's movements were constrained by the presence of a barrier on the table with a gap in it, placed so that the demonstrator robot had to reach through the gap to get the ball (ref. Fig. 9). The barriers acted as inequality constraints on each of the hands so that motion in the direction normal to the barrier surface was prevented if a hand came too close. Specifically, the constraints took the form

$$\mathbf{A}(\mathbf{x}, t) = \left(\begin{array}{c|c} \mathbf{A}_{[1,1]} & \mathbf{0} \\ \hline \mathbf{A}_{[1,2]} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{A}_{[2,1]} \\ \hline \mathbf{0} & \mathbf{A}_{[2,2]} \end{array}\right) \tag{35}$$

where

$$\begin{aligned} &\mathbf{A}_{[i,j]}(\mathbf{x}, t) = \hat{\mathbf{n}}_j^T \;; \quad d_{i,j} \leq d_{min} \;\; and \;\; \hat{\mathbf{u}}_{[i]}^T \hat{\mathbf{n}}_j > 0 \\ &\mathbf{A}_{[i,j]}(\mathbf{x}, t) = \mathbf{0} \quad ; \quad otherwise. \end{aligned}$$

Here, $d_{i,j}$ is the distance of the ith hand (where $i \in \{1, 2\}$, i.e. left and right hands respectively) to the closest point on the jth barrier (where $j \in \{1, 2\}$, i.e. left and right barriers respectively), $\hat{\mathbf{n}}_j \in \mathbb{R}^3$ is the normal to the barrier surface[6] at that point and $\hat{\mathbf{u}}_{[i]} \in \mathbb{R}^3$ is the normalised command for the ith hand (i.e. the ith 3-vector block of the command vector $\mathbf{u}$ corresponding to that hand; for example for the right hand ($i = 2$) this was $\mathbf{u}_{[2]} \equiv (u_4, u_5, u_6)^T$ with $\hat{\mathbf{u}}_{[2]} = \mathbf{u}_{[2]} / \|\mathbf{u}_{[2]}\|$). Here, the full constraint matrix $\mathbf{A}(\mathbf{x}, t) \in \mathbb{R}^{4\times 6}$ was constructed by assigning 3-vectors to the appropriate matrix blocks $\mathbf{A}_{[i,j]}$, according to the system state. For example, if the left hand ($i = 1$) approached the left barrier ($j = 1$) to a distance of $d_{1,1} < d_{min}$, and if the next commanded movement would bring the hand toward that barrier (i.e. $\hat{\mathbf{u}}_{[1]}^T \hat{\mathbf{n}}_1 > 0$), then the elements of the constraint matrix corresponding to that hand/barrier pair were updated (in this example the first row of the matrix would be updated, $\mathbf{A}_{1,:} = (\hat{\mathbf{n}}_1^T, 0, 0, 0)$, constraining the left hand). Note that under this setup the constraints are highly nonlinear (due to the complex dependence on state) and have discontinuously switching dimensionality (i.e.

[6] Note that in order to ensure smooth, natural-looking trajectories the barriers were modelled as planes with smooth 'swept-sphere' edges, similar to those described in [51].

the rank of $\mathbf{A}(\mathbf{x}, t)$ switches) when either of the hands approaches or recedes from the barrier.

Data was collected by recording $K = 100$ trajectories of length $2s$ at 50 Hz, (i.e. $N = 100$ points per trajectory) from the demonstrator following the policy (34) under the constraints (35). Start states were sampled from a Gaussian distribution over joint configurations $\mathbf{q} \sim N(\mathbf{q}_0, 0.1\mathbf{I})$ (where $\mathbf{q}_0$ corresponds to the default standing position) and using forward kinematics to calculate the corresponding hand positions. The joint vector $\mathbf{q}$ was clipped where necessary to avoid joint limits and self collisions, and to ensure the start postures looked natural. For each trajectory the constraints were varied by randomly changing the width of the gap in the barriers. The gap widths were sampled from a Gaussian distribution $d_{gap} \sim N(\mu_{gap}, \sigma_{gap})$ where $\mu_{gap} = 0.25m$, $\sigma_{gap} = 0.1m$ and the diameter of the ball was $0.15m$. The hand-barrier distance at which the constraints came into force was fixed at $d_{min} = 0.05m$. Fig. 9 shows an example trajectory under this set-up.

We used the three algorithms (the direct, alignment and inconsistency approaches) to perform learning on 50 such data sets using 150 Gaussian RBF models, with centres placed using k-means. For comparison, we repeated the experiment on the same data with the same model (i.e. same number and placement of centres) with the three approaches. Please note that (similar to the experiments in the preceding section) the model for the direct and inconsistency approaches corresponded to the $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \mathbf{u} \in \mathbb{R}^6 \mapsto \mathbb{R}^6$ mapping, whereas for the alignment approach it represented the mapping $\tilde{\boldsymbol{\pi}} : \mathbf{x} \rightarrow \phi \in \mathbb{R}^6 \mapsto \mathbb{R}$.

To assess the performance for the methods we evaluated the errors in predicting the policy subject to (i) the training data constraints (nCPE), (ii) no constraints (nUPE), and (iii) a novel constraint, unseen in the training data, on a set of test data. For the latter, a barrier was placed centrally between the robot and the ball, so that the robot had to reach around the barrier to reach the ball (see Fig. 11). Specifically, the constraint took a form similar to (35) but this time with only one barrier present (i.e. $j \equiv 1$), so that the constraint matrix $\mathbf{A}(\mathbf{x}, t) \in \mathbb{R}^{2\times 6}$ had attained a maximum rank of $k = 2$ when both hands approached the barrier. The width of the new barrier was fixed at $0.5m$.

Comparing the numerical errors (ref. Table 2) for the two constraint-consistent learning methods (i.e. the alignment and inconsistency approaches) with those of the direct approach we see that the former perform several orders of magnitude better under each of the constraint settings considered, with the inconsistency approach performing marginally better. However, the real difference between the constraint-consistent learning methods and the direct approach is best highlighted if we compare trajectories generated by the policies under different constraint settings.

Firstly, Fig. 10 shows example trajectories for the *unconstrained* reaching movements produced by the demonstrator ('expert'), and the policies learnt by (i) the direct approach, (ii) the alignment approach, and (iii) the

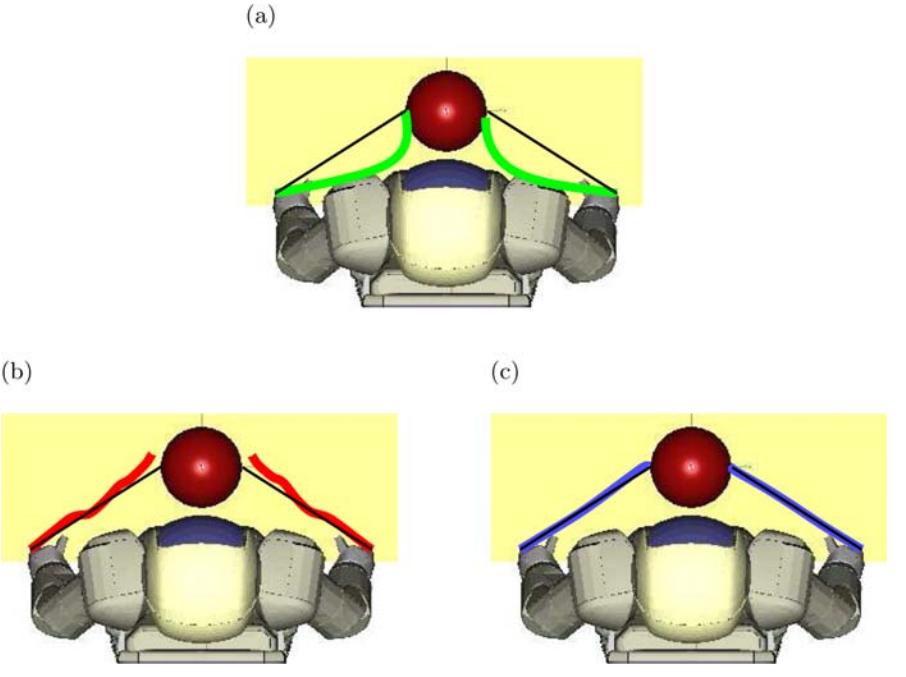


Fig. 10 Unconstrained reaching movement for the policies learnt with (a) direct regression (green) (b) the alignment approach (red), and (c) the inconsistency approach (blue). In each figure the demonstrator's movement is overlaid in black.

Table 2 Normalised errors for policies learnt by the three methods, evaluated on (i) training constraints, (ii) no constraints, and (iii) an unseen test constraint on the ball-reaching task. Values are mean±s.d. over 50 data sets.

Constraint	Direct	Align.	Incon.
Training	0.0531 ± 0.0068	0.0092 ± 0.0021	0.0052 ± 0.0022
Unseen Barrier	0.4630 ± 0.0350	0.0101 ± 0.0023	0.0052 ± 0.0022
Unconstrained	0.9216 ± 0.0625	0.0106 ± 0.0024	0.0052 ± 0.0022

inconsistency approach. In the former the hands always take a curved path to the ball (Fig. 10(a)), reproducing the average behaviour of the (constrained) demonstrated trajectories. The direct approach, being naive to the effect of the constraints is unable to extract the underlying task (policy) from the observed paths around the obstacles. In contrast, the policies learnt with the constraint-consistent approaches better predict the unconstrained policy, enabling them to take a direct route to the ball that closely matches that of the demonstrator (Fig. 10(b),(c)).

Secondly, Fig. 11 shows example trajectories when the policies are again constrained. Figure 11 (top) shows the movement from the policy learnt by the inconsistency approach under a similar constraint as in the training data. Under this constraint the policies learnt by the three methods all take a similar path to that of the demonstrator: The hands move in first, then forward to the ball. Note that under this constraint the movement of the directly learnt policy is noticeably slower due to averaging of the constrained observations.

Finally, under the unseen barrier constraint, there is a marked difference in behaviour. Under this constraint, the demonstrator (still following the policy (34)) reaches around the barrier to get the ball. This behaviour is reproduced by the policy learnt with the two constraint-consistent approaches (Fig. 11, middle row, shows the movement for the policy learnt by the inconsistency approach). In contrast however, the directly learnt policy does not generalise to the new constraint and gets trapped behind the barrier, eventually dislodging it[7] (Fig. 11, bottom).

5.3 Washing a Car

In this section we discuss an application of constraint-consistent learning to the the problem of learning to wash a car from human demonstration data [22]. This is an example of a task which can be intuitively described in terms of a simple movement policy ('wiping') subject to contact constraints that vary depending on the different surfaces of the car to be wiped. Due to the different shapes and orientations of these surfaces, complex, non-linear constraints are imposed on the motion. While the resultant trajectories remain periodic, they are perturbed in different ways by the constraints. The goal of this experiment then, was to learn a policy that captured the periodic nature of the movements, while eliminating artifacts induced by the constraints.

In [22] an experiment was performed to evaluate the performance of constraint-consistent learning on data from human demonstrations of wiping. In this experiment a set of demonstrations of wiping on different surfaces (i.e. on surfaces with different tilts and orientations, see Fig. 12) were presented to the ASIMO humanoid robot by a human demonstrator. The robot used on-board stereo cameras to track the three-dimensional coordinates of the sponge (for details on the ASIMO vision system please see [5]) and the resultant data was used for constraint-consistent learning. The resultant data was used to train a policy model representing the $\mathbb{R}^3 \mapsto \mathbb{R}^3$ mapping from hand

[7] Note that the collision of the hands with the barrier in fact violates the constraint. The reason for this is that on the real robot, under this constraint, the directly learnt policy forces the robot into a self-collision (of the robot's arms with the torso). To prevent damage to the robot, an on-board safety mechanism then kicks in and pushes the hands away from the body, causing collision with the barrier.

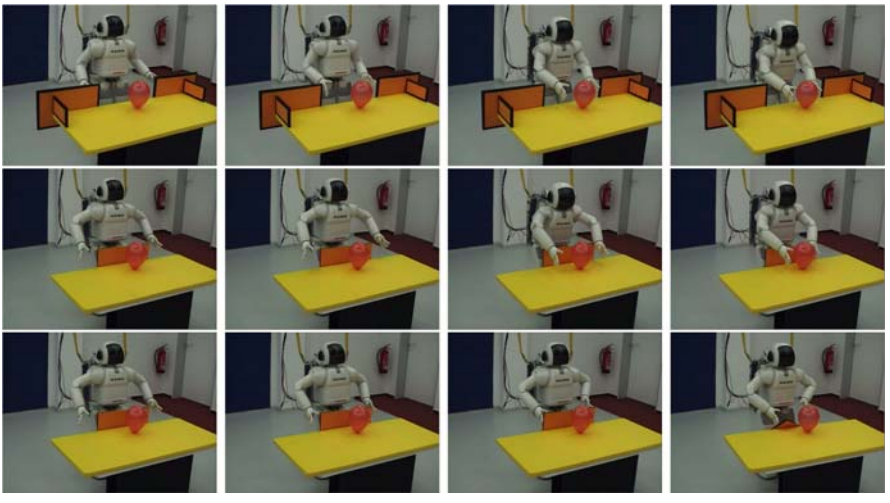


Fig. 11 Reaching movements produced by the learnt policies under different constraints. Shown are trajectories from (i) the policy learnt by the inconsistency approach under a similar constraint as in the training data (top row); (ii) the same policy under a new, unseen barrier constraint (middle row), and; (iii) the policy learnt with direct regression under the new constraint.

Fig. 12 Human wiping demonstrations on surfaces of varying tilt and rotations. The ASIMO stereo vision system was used to track the 3-D coordinates of the sponge (coloured rectangles show the estimated position). Tilts of $\pm 16^{\circ}$ and $+27^{\circ}$ about the x-axis are shown.

(sponge) positions to velocities, consisting of a set of 300 Gaussian RBFs with centres placed by k-means.

Since the ground truth (i.e. the true unconstrained policy and the exact constraints in force) is not known for the human data, performance was evaluated on a behavioural level. For this, the policies were implemented on the ASIMO humanoid robot and an approximation of the human's constraints based on an analysis of the hand-sponge system (for full details on the constraints used to approximate the human constraints, please refer to [22]) were applied to the robot during movement reproduction. Under this set-up,

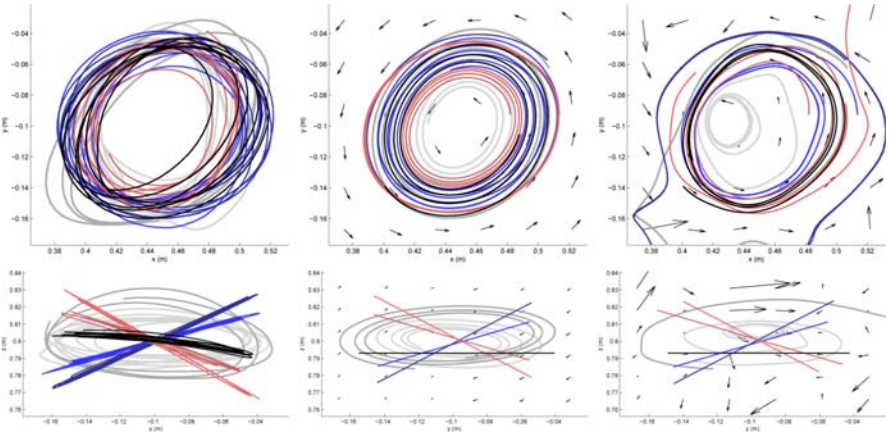


Fig. 13 Learning from human wiping demonstrations. Left: Trajectories of the sponge when wiping on the surface when flat (black), tilted $+16^\circ$ and $+27^\circ$ about the x-axis (red), -16° and -27° about the x-axis (blue), and $\pm 16^\circ$ about the y-axis (grey). Centre and right: Reproduced trajectories using the policies (black arrows) learnt with the inconsistency and direct approaches, respectively. In each case the same example trajectory is highlighted (thick black). The top and front views are shown (top and bottom rows).

constraint-consistent learning with the inconsistency approach was compared to that of direct regression (since in this case the task is clearly periodic, the inconsistency approach is the appropriate choice of constraint-consistent method.)

The results are shown in Fig. 13, where we show the demonstrated trajectories (left), those produced by the constraint-consistent policy (centre) and those learnt by direct regression (right) under the different constraints (tilts of the surface). Looking at the learnt policies, we see that the constraint-consistent approach learns a smooth policy and the trajectories under each of the constraints are smooth periodic movements, similar to those of the human. On the ASIMO robot these produced smooth, natural wiping movements (see Fig. 14).

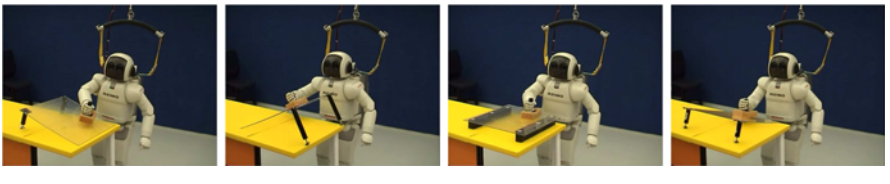


Fig. 14 Reproduced movements on the ASIMO robot for the surface tilted 0°, $+16^\circ$, -27° about the x-axis, and $+16^\circ$ about the y-axis.

The policy learnt with direct regression also captured the periodicity to some extent. However, it appears highly irregular in several regions and the trajectories are unstable, with some spiralling in to the centre, and others diverging to other parts of the state space. By attempting to learn all of the artifacts induced by the constraints, the direct approach, naive to the constraints, learns an unstable policy that cannot be safely used for movement reproduction on the robot.

6 Discussion

In this chapter, we reviewed recent work in the area of policy-based learning of demonstrated movements, where those movements are subject to variable, dynamic, non-linear constraints. We discussed the problems encountered when learning in this setting and showed, through analysis and experimentation, how approaches based on standard supervised learning techniques come into difficulties when presented with data containing variable constraints. We then went on to describe two new learning methods for learning in a constraint-consistent manner. The first, earlier approach solved the learning problem for constrained potential-based policies. The second approach, based on a modification in the calculation of an empirical risk, was shown to be effective for arbitrary policies, including those with a rotational component. It was seen that both methods were capable of recovering the unconstrained policy from arbitrarily constrained observations, without the need for explicit knowledge of the constraints. This allows us to learn policies that generalise over constraints, including novel constraints, unseen in the training data. Furthermore, the comparative performance of the methods was reviewed for learning policies on systems of varying size and complexity.

How far do the policies generalise?

The results presented here and in [22, 19] clearly show the efficacy of learning from constrained demonstrations using the two approaches, and then applying the resultant policies to new constraint scenarios. However, in terms of lessons learnt from these studies there are also some bigger issues raised. One such issue is the question of when, faced with a new constraint, the learnt policy will fail at the desired task. For example, in the ball grasping experiment, under certain configurations of the constraints (e.g. if the barriers were placed exactly on either side of the ball, or a much larger barrier was placed between the robot and the ball) the learnt policy would fail at the task of grasping. This may be due to several factors, for instance if the control vector happens to be orthogonal to the nullspace of the constraint, deadlock would occur (this is similar to the problem of local minima in many gradient-based controllers, e.g. see [11]). While problems such as these are in general unavoidable when dealing with constrained systems, one of the nice properties of the

constraint-consistent approaches is that they learn policies that are successful *under the same constraints that the demonstrator is successful*. So, although the learnt policy for the grasping task is not guaranteed to successfully get the ball in the presence of any arbitrary barrier (constraint), it successfully reaches the ball whenever (i.e. with whatever barriers) the demonstrator does. In some sense we can say the *robustness* of the demonstrator's policy against different constraints was transferred to the learner.

Adaptation to constraints: Re-planning vs. re-using policies?

A second, related issue concerns the role of adaptation of policies in response to constraints. Clearly there are circumstances in which it is desirable to re-plan the policy to cope with certain sets of constraints, especially if the learner's existing policy (here, learnt from demonstration) fails under those constraints (and in some cases the learner may even take advantage of certain types of constraint to improve performance). However, here a balance must be struck. On the one hand re-planning the policy will likely improve performance under any given set of constraints; but on the other hand the adapted policy will also become more specialised to that particular set of constraints (and may even lead to degraded performance for other constraints). In other words we lose the *generalisation to other constraints* that here we attempt to extract from the demonstrator. Furthermore, due to the inherent uncertainty in the constraints in most real world problems, it may not be feasible to explicitly incorporate all of the constraints when re-planning. For example consider planning a policy for walking on uneven terrain; to explicitly incorporate the constraints involved here would require a detailed model of the terrain, which is rarely available. The constraint-consistent approaches, however, allow us to sidestep this, providing a shortcut to uncovering the policy used by the demonstrator[8] (who, if observed to use the same policy under a number of constraint settings, presumably finds it sufficiently successful for those and similar settings). Therefore in this sense, with these approaches, we may now envisage a move away from the traditional approach of planning explicitly with respect to all possible constraints that is typically only possible in highly controlled, structured environments.

References

[1] Alissandrakis, A., Nehaniv, C.L., Dautenhahn, K.: Correspondence mapping induced state and action metrics for robotic imitation. IEEE Transactions on Systems, Man and Cybernetics 37(2), 299–307 (2007)

[8] It should also be noted that our approach may also be combined with adaptive methods, for example using the policy learnt from demonstration to initialise further optimisation of the policy, similar to e.g.,[18].

[2] Antonelli, G., Arrichiello, F., Chiaverini, S.: The null-space-based behavioral control for soccer-playing mobile robots. In: IEEE International Conference Advanced Intelligent Mechatronics, pp. 1257–1262 (2005)
[3] Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. In: Robotics and Autonomous Systems (2008) (in press) (Corrected Proof)
[4] Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. In: Handbook of Robotics, ch. 59. MIT Press, Cambridge (2007)
[5] Bolder, B., Dunn, M., Gienger, M., Janssen, H., Sugiura, H., Goerick, C.: Visually guided whole body interaction. In: IEEE International Conference on Robotics and Automation, pp. 3054–3061 (2007)
[6] Calinon, S., Billard, A.: Learning of gestures by imitation in a humanoid robot. In: Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions (2007)
[7] Chajewska, U., Koller, D., Ormoneit, D.: Learning an agent's utility function by observing behavior. In: International Conference on Machine Learning (2001)
[8] Chajewska, U., Getoor, L., Norman, J., Shahar, Y.: Utility elicitation as a classification problem. In: Uncertainty in Artificial Intelligence, pp. 79–88. Morgan Kaufmann Publishers, San Francisco (1998)
[9] Chaumette, F., Marchand, A.: A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. IEEE Trans. Robotics and Automation 17(5), 719–730 (2001)
[10] Il Choi, S., Kim, B.K.: Obstacle avoidance control for redundant manipulators using collidability measure. Robotica 18(2), 143–151 (2000)
[11] Conner, D.C., Rizzi, A.A., Choset, H.: Composition of local potential functions for global robot control and navigation. In: IEEE International Conference on Intelligent Robots and Systems, October 27-31, vol. 4, pp. 3546–3551 (2003)
[12] D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: IEEE International Conference on Intelligent Robots and Systems (2001)
[13] English, J.D., Maciejewski, A.A.: On the implementation of velocity control for kinematically redundant manipulators. IEEE Transactions on Systems, Man and Cybernetics 30(3), 233–237 (2000)
[14] Fumagalli, M., Gijsberts, A., Ivaldi, S., Jamone, L., Metta, G., Natale, L., Nori, F., Sandini, G.: Learning how to exploit proximal force sensing: A comparison approach. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 149–167. Springer, Heidelberg (2010)
[15] Gienger, M., Janssen, H., Goerick, C.: Task-oriented whole body motion for humanoid robots. In: IEEE International Conference on Humanoid Robots, December 5, pp. 238–244 (2005)
[16] Grimes, D.B., Chalodhorn, R., Rajesh, P.N.R.: Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In: Robotics: Science and Systems. MIT Press, Cambridge (2006)
[17] Grimes, D.B., Rashid, D.R., Rajesh, P.N.R.: Learning nonparametric models for probabilistic imitation. In: Advances in Neural Information Processing Systems. MIT Press, Cambridge (2007)

[18] Guenter, F., Hersch, M., Calinon, S., Billard, A.: Reinforcement learning for imitating constrained reaching movements. RSJ Advanced Robotics, Special Issue on Imitative Robots 21(13), 1521–1544 (2007)
[19] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Behaviour generation in humanoids by learning potential-based policies from constrained motion. Applied Bionics and Biomechanics 5(4), 195–211 (2008) (in press)
[20] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Learning potential-based policies from constrained motion. In: IEEE International Conference on Humanoid Robots (2008)
[21] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: A novel method for learning policies from constrained motion. In: IEEE International Conference on Robotics and Automation (2009)
[22] Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: A novel method for learning policies from variable constraint data. In: Autonomous Robots (submitted, 2009)
[23] Howard, M., Vijayakumar, S.: Reconstructing null-space policies subject to dynamic task constraints in redundant manipulators. In: Workshop on Robotics and Mathematics (September 2007)
[24] Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: IEEE International Conference on Robotics and Automation, pp. 1398–1403 (2002); ICRA 2002 best paper award
[25] Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems, pp. 1523–1530. MIT Press, Cambridge (2003)
[26] Inamura, T., Toshima, I., Tanie, H., Nakamura, Y.: Embodied symbol emergence based on mimesis theory. The International Journal of Robotics Research 23(4), 363–377 (2004)
[27] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In: IEEE Int. Conf. on Intelligent Robots and Systems (2003)
[28] Kannan, R., Vempala, S., Vetta, A.: On clusterings: Good, bad and spectral. Journal of the ACM 51(3), 497–515 (2004)
[29] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: IEEE International Conference on Robotics and Automation, vol. 1, pp. 428–436 (1985)
[30] Khatib, O.: A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE Journal of Robotics and Automation RA-3(1), 43–53 (1987)
[31] Körding, K.P., Fukunaga, I., Howard, I.S., Ingram, J.N., Wolpert, D.M.: A neuroeconomics approach to inferring utility functions in sensorimotor control. PLoS Biolology 2(10), 330 (2004)
[32] Körding, K.P., Wolpert, D.M.: The loss function of sensorimotor learning. Proceedings of the National Academy of Sciences 101, 9839–9842 (2004)
[33] Liégeois, A.: Automatic supervisory control of the configuration and behavior of multibody mechanisms. IEEE Trans. Sys., Man and Cybernetics 7, 868–871 (1977)

[34] Mattikalli, R., Khosla, P.: Motion constraints from contact geometry: Representation and analysis. In: IEEE International Conference on Robotics and Automation (1992)
[35] Murray, R.M., Li, Z., Sastry, S.S.: A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton (1994)
[36] Nakamura, Y.: Advanced Robotics: Redundancy and Optimization. Addison Wesley, Reading (1991)
[37] Ohta, K., Svinin, M., Luo, Z., Hosoe, S., Laboissiere, R.: Optimal trajectory formation of constrained human arm reaching movements. Biological Cybernetics 91, 23–36 (2004)
[38] Park, J., Khatib, O.: Contact consistent control framework for humanoid robots. In: IEEE International Conference on Robotics and Automation (May 2006)
[39] Peters, J., Mistry, M., Udwadia, F.E., Nakanishi, J., Schaal, S.: A unifying framework for robot control with redundant dofs. Autonomous Robots 24, 1–12 (2008)
[40] Peters, J., Schaal, S.: Learning to control in operational space. The International Journal of Robotics Research 27(2), 197–212 (2008)
[41] Ren, J., McIsaac, K.A., Patel, R.V.: Modified Newton's method applied to potential field-based navigation for mobile robots. In: IEEE Transactions on Robotics (2006)
[42] Rimon, E., Koditschek, D.E.: Exact robot navigation using artificial potential functions. IEEE Transactions on Robotics and Automation 8(5), 501–518 (1992)
[43] De Sapio, V., Khatib, O., Delp, S.: Task-level approaches for the control of constrained multibody systems (2006)
[44] De Sapio, V., Warren, J., Khatib, O., Delp, S.: Simulating the task-level control of human motion: a methodology and framework for implementation. The Visual Computer 21(5), 289–302 (2005)
[45] Schaal, S.: Learning from demonstration. In: Mozer, M.C., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems, pp. 1040–1046. MIT Press, Cambridge (1997)
[46] Schaal, S., Atkeson, C.G.: Constructive incremental learning from only local information. Neural Computation 10, 2047–2084 (1998)
[47] Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philosophical Transactions: Biological Sciences 358(1431), 537–547 (2003)
[48] Sentis, L., Khatib, O.: Task-oriented control of humanoid robots through prioritization. In: IEEE International Conference on Humanoid Robots (2004)
[49] Sentis, L., Khatib, O.: Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. International Journal of Humanoid Robotics 2(4), 505–518 (2005)
[50] Sentis, L., Khatib, O.: A whole-body control framework for humanoids operating in human environments. In: IEEE International Conference on Robotics and Automation (May 2006)
[51] Sugiura, H., Gienger, M., Janssen, H., Goerick, C.: Real-time collision avoidance with whole body motion control for humanoid robots. In: IEEE International Conference on Intelligent Robots and Systems, pp. 2053–2058 (2007)

[52] Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction. MIT Press, Cambridge (1998)
[53] Takano, W., Yamane, K., Sugihara, T., Yamamoto, K., Nakamura, Y.: Primitive communication based on motion recognition and generation with hierarchical mimesis model. In: IEEE International Conference on Robotics and Automation (2006)
[54] Todorov, E.: Optimal control theory. In: Doya, K. (ed.) Bayesian Brain. MIT Press, Cambridge (2006)
[55] Udwadia, F.E., Kalaba, R.E.: Analytical Dynamics: A New Approach. Cambridge University Press, Cambridge (1996)
[56] Verbeek, J.: Learning non-linear image manifolds by combining local linear models. IEEE Transactions on Pattern Analysis & Machine Intelligence 28(8), 1236–1250 (2006)
[57] Verbeek, J., Roweis, S., Vlassis, N.: Non-linear cca and pca by alignment of local models. In: Advances in Neural Information Processing Systems (2004)
[58] Vijayakumar, S., D'Souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Computation 17(12), 2602–2634 (2005)
[59] Yoshikawa, T.: Manipulability of robotic mechanisms. The International Journal of Robotics Research 4(2), 3–9 (1985)

Motor Learning at Intermediate Reynolds Number: Experiments with Policy Gradient on the Flapping Flight of a Rigid Wing

John W. Roberts, Lionel Moret, Jun Zhang, and Russ Tedrake

Abstract. This work describes the development of a model-free reinforcement learning-based control methodology for the heaving plate, a laboratory experimental fluid system that serves as a model of flapping flight. Through an optimized policy gradient algorithm, we were able to demonstrate rapid convergence (requiring less than 10 minutes of experiments) to a stroke form which maximized the propulsive efficiency of this very complicated fluid-dynamical system. This success was due in part to an improved sampling distribution and carefully selected policy parameterization, both motivated by a formal analysis of the signal-to-noise ratio of policy gradient algorithms. The resulting optimal policy provides insight into the behavior of the fluid system, and the effectiveness of the learning strategy suggests a number of exciting opportunities for machine learning control of fluid dynamics.

1 Introduction

The possible applications of robots that swim and fly are myriad, and include agile UAVs, high-maneuverability AUVs and biomimetic craft such as ornithopters and robotic fish. However, controlling robots whose behavior is heavily dependent upon their interaction with a fluid can be particularly challenging. Whereas in many regimes robots are able to make use of either accurate dynamical models or easy-to-stabilize dynamics, in the case of flapping and swimming robots neither of these conditions apply. The models available to swimming and flying robots tend to be very limited in their region of validity (such as quasi-steady models of fixed-wing aircraft), very expensive to evaluate (such as direct numerical simulation of the governing equations) or almost completely unavailable (as is the case with interactions between a complex flow and a compliant body).

Here we investigate the problem of designing a controller for a specific experimental system: the heaving plate (see Figure 1). This setup is a model of forward flapping flight developed by the Applied Math Lab (AML) of the Courant Institute

John W. Roberts and Russ Tedrake
Massachusetts Institute of Technology, 32 Vassar St. Room 32-380, Cambridge, MA 02139

Jun Zhang and Lionel Moret
New York University, 4 Washington Place, New York, NY 10003

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 293–309.
springerlink.com

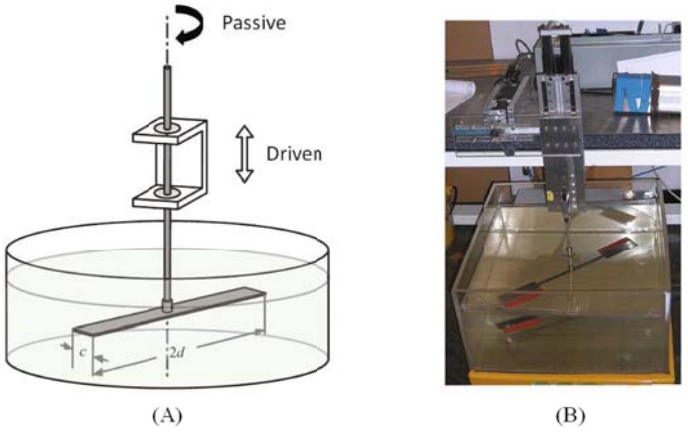


Fig. 1 (A) Schematic of experimental flapping system. The white arrow shows the driven vertical motion determined by the controller, while the black arrow shows the passive rotational motion resulting from the fluid forces. Figure from (Vandenberghe et al., 2006), with slight modifications. (B) Original experimental flapping system. The wing shown is an earlier design used to study the fluid system, while the wing used for this work was a simple rigid rectangle.

of Mathematical Sciences at New York University (Vandenberghe et al., 2004; Vandenberghe et al., 2006). This system consists of a symmetric rigid horizontal plate that is driven up and down along its vertical axis and is free to rotate in the horizontal plane. Previous work demonstrated that driving the vertical motion with a sinusoidal waveform caused the system to begin rotating in a stable "forward flight" (Vandenberghe et al., 2004). Here we will investigate a richer class of waveforms in an attempt to optimize the efficiency of that forward flight. This system is an excellent candidate for control experiments as it is perhaps the simplest experimental model of flapping flight, is experimentally convenient for learning experiments[1], and captures the essential qualities of the rich fluid dynamics in fluid-body interactions at intermediate Reynolds numbers. While accurate Navier-Stokes simulations of this system do exist for the case of a rigid wing (Alben & Shelley, 2005), they require a dramatic amount of computation[2]. As such, model-based design of an effective controller for this system is a daunting task.

Model-free reinforcement learning algorithms, however, offer an avenue by which controllers can be designed for this system despite the paucity of the available models by evaluating the effectiveness of a controller directly on the physical system (Peters et al., 2003; Tedrake et al., 2004). In fact, learning through

[1] Particularly when compared to a flapping machine that falls out of the sky repeatedly during control design experiments.

[2] At the time, this simulation required approximately 36 hours to simulate 30 flaps on specialized hardware (Shelley, 2007).

experimentally collected data can be far more efficient than learning via simulation in these systems, as high-fidelity simulations can take orders of magnitude longer than an experiment to obtain the same data. One of the limitations of this experimental
approach, however, is the potential difficulty in directly measuring the state of the fluid, which, naively, is infinite-dimensional (a continuum model). Therefore, the problem is best formulated as a partially-observable Markov decision process (POMDP) (Kaelbling et al., 1998). In the experiment described here, we solve the POMDP with a pure policy gradient approach, choosing not to attempt to approximate a value function due to our poor understanding of even the dimensionality of the fluid state.

The difficulty in applying policy gradient techniques to physical systems stems from the fact that model-free algorithms often suffer from high variance and relatively slow convergence rates (Greensmith et al., 2004), resulting in the need for many evaluations. As the same systems on which one wishes to use these algorithms tend to have a high cost of policy evaluation, much work has been done on maximizing the policy improvement from any individual evaluation (Meuleau et al., 2000; Williams et al., 2006). Techniques such as Natural Gradient (Amari, 1998; Peters et al., 2003) and GPOMDP (Baxter & Bartlett, 2001) have become popular through their ability to converge on locally optimal policies using fewer policy evaluations.

During our experiments with policy gradient algorithms on this system, we developed a number of optimizations to the vanilla policy gradient algorithm which provided significant benefits to learning performance. The most significant of these is the use of non-Gaussian sampling distributions on the parameters, a technique which is appropriate for systems with parameter-independent additive noise (a common model for sensor-driven observation noise). We make these observations concrete by formulating a signal-to-noise ratio (SNR) analysis of the policy gradient algorithms, and demonstrate that our modified sampling distributions improve the SNR.

With careful experiments and improvements to the policy gradient algorithms, we were able to reliably optimize the stroke form (i.e., the periodic vertical trajectory of the plate through time) of the heaving plate for propulsive efficiency to the same optima from different initial conditions (in policy space) in less than ten minutes of trial-and-error experiments on the system. The remainder of this chapter details these experiments and the theory behind them.

2 Experimental Setup

The heaving plate is an ideal physical system to use as a testbed for developing control methodologies for the broader class of problems involving fluid-robot interactions. The system consists of a symmetric rigid titanium plate pinned in the horizontal plane. It is free to rotate about the point at which it is pinned with very little non-fluid related friction. The control input is the vertical position of the plate z (note that the input is the kinematic position, not the force on the plate). This vertical driven flapping motion is coupled through the fluid with the passive angular

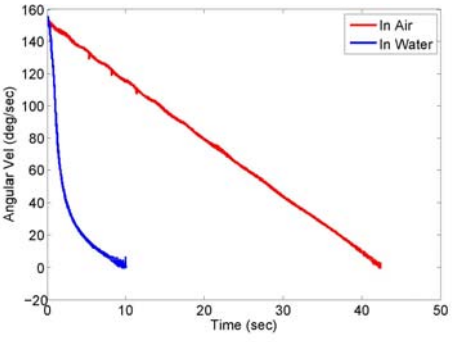


Fig. 2 Comparison of fluid and bearing friction for large rigid wing. The foil (not undergoing a heaving motion) was spun by hand in both air and water, with the angular velocity measured as a function of time. Five curves for both cases were truncated to begin at the same speed then averaged and filtered with a zero-phase low-pass filter to generate these plots. The quick deceleration of the wing in water as compared to air indicates that fluid viscous losses are much greater than bearing losses. At the speeds achieved at the convergence of the learning, the frictional forces in water were over six times that seen in air.

rotation such that once the system begins to flap, fluid forces cause it to spin. To ensure that the rotational dynamics are dominated by fluid forces, and not friction due to the slip ring and bearings used to mount the wing, the decay in angular speed was measured for both a submerged and non-submerged wing. Figure 2 shows the result of this experiment, demonstrating that the fluid forces are far more significant than the bearing friction in the regime of operation.

The only measurements taken were the vertical position of the plate (z) and angular position of the plate (x) by optical encoders and the axial force applied to the plate (F_z), obtained by an analog tension-compression load cell. While these measurements were sufficient for formulating interesting optimization problems, note that the fluid was never measured directly. Indeed, the hidden state of the fluid, including highly transient dynamic events such as the vortex pairs created on every flapping cycle, provide the primary mechanism of thrust generation in this regime (Vandenberghe et al., 2004). Sensing the state of the flow is possible using local flow sensors or even real-time far-field optical flow measurement (Bennis et al., 2008), but experimentally more complex. However, this work demonstrates such sensing is not necessary for the purpose of learning to move effectively in the fluid, despite the critical role the fluid state plays in the relevant dynamics.

The setup was originally used to study the fluid dynamics of forward flapping flight, and possess a Reynolds number of approximately 16,000, putting it in the same regime as dragonflies and other small biological flapping fliers[3]. The plate is unalloyed titanium (chosen for its corrosion resistance), 72 cm long, 5 cm wide

[3] This Reynolds number is determined using the forward flapping motion of the wing, rather than the vertical heaving motion. The vertical heaving motion possesses a Re of approximately 3,000.

and .3175 cm thick. The vertical motion was produced by a scotch yoke, which converted the rotational motion of the motor into the desired linear driven motion of the plate. Due to the high gear ratio between the motor and the scotch yoke, the system was not back-drivable, and thus the plate was controlled be specifying a desired kinematic trajectory which was followed closely using tuned high-gain linear feedback. While the trajectories were not followed perfectly (e.g., there is some lag in tracking the more violent motions), the errors between specified trajectory and followed trajectory were small (on the order of 5% of the waveform's amplitude).

3 Optimal Control Formulation

We formulate the goal of control as maximizing the propulsive efficiency of forward flight by altering the plate's stroke form. We attempt to maximize this efficiency within the class of strokeforms that can be described by a given parameterization. In this section we discuss both the reward function used to measure performance, and the parameterization used to represent policies (i.e., stroke forms).

3.1 Reward Function

We desire our reward function to capture the efficiency of the forward motion produced by the stroke form. To this end, we define the (mechanical) cost-of-transport over one period T as:

$$c_{mt} = \frac{\int_T |F_z(t)\dot{z}(t)|dt}{mg\int_T \dot{x}(t)dt}. \tag{1}$$

where x is the angular position, z is the vertical position, F_z is the vertical force, m is the mass of the body and g is gravitational acceleration. The numerator of this quantity is the energy used, while the denominator is the weight times distance traveled. It was computed on our system experimentally by filtering and integrating the force measured by the load cell and dividing by the measured angular displacement, all over one period. This expression is the standard means of measuring transport cost for walking and running creatures (Collins et al., 2005), and thus seems a sensible place to start when measuring the performance of a swimming system.

This cost has the advantage of being dimensionless, and thus invariant to the units used. The non-dimensionalization is achieved by dividing by a simple scalar (in this case mg), and thus does not change as the policy changes. While alternatives to the mass such as using the difference in mass between the plate and the displaced fluid were debated (to take into account the importance of the fluid to the motion of the plate), changes such as these would affect the magnitude of the cost, but not the optima and learning behavior, as these are invariant to a scaling. Therefore, implementing this change would not effect the found optimal policy.

Another possibility would be non-dimensionalizing by dividing by expected energy to travel through the fluid (as opposed to weight times distance), but this would depend upon the speed of travel as drag is velocity dependent, and thus would have

a more complicated form. While this could obviously still be implemented, and new behavior and optima may be found, rewarding very fast flapping gaits strongly (as this would tend to do) was undesirable simply because the experimental setup struggled mechanically with the violent motions found when attempting to maximize speed. The cost function selected often produced relatively gentle motions, and as such put less strain on the setup.

Finally, note that our learning algorithm attempts to maximize the cost of transport's *inverse* (turning it from a cost into a reward), which is equivalent to minimizing the energy cost of traveling a given distance. This was done for purely practical considerations, as occasionally when very poor policies were tried the system would not move significantly despite flapping, which resulted in an infinite or near-infinite cost of transport. The inverse, however, remained well-behaved.

3.2 Policy Parameterization

The parameterization chosen took the following form: the vertical heaving motion of the wing was represented by a 13-point periodic cubic spline with fixed amplitude and frequency, giving height z as a function of time t (see Figure 3). There were five independent parameters, as the half-strokes up and down were constrained to be symmetric about the t axis (i.e., the first, seventh and last points were fixed at zero, while points 2 and 8, 3 and 9 etc. were set to such that they were equal in absolute value but opposite in sign which were determined by the control parameters).

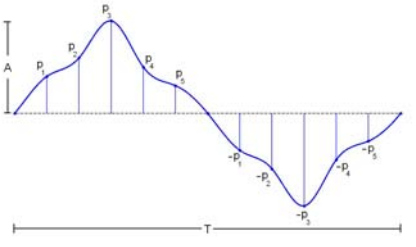


Fig. 3 A schematic of the parameterization of policies used in this work. Note the symmetry of the up and down strokes, and the fact that five independent parameters are used to encode the shape of the waveform.

This parameterization represented an interesting class of waveforms, had a relatively small number of parameters, and was both smooth and periodic. We will also see that many of the properties are desirable when viewed through the SNR (these advantages are discussed in greater detail in Section 6.1).

4 The Learning Algorithm

In light of the challenges faced by learning on a system with such dynamic complexity and partial observability, we made use of the weight-perturbation (WP)

algorithm: a model-free policy gradient method that has been shown empirically to be well-suited to these sorts of problems due to its robustness to noise and insensitivity to the complexity of the system dynamics.

4.1 The Weight Perturbation Update

Consider minimizing a scalar function $J(\mathbf{w})$ with respect to the parameters $\mathbf{w}$ (note that it is possible that $J(\mathbf{w})$ is a long-term cost and results from running a system with the parameters $\mathbf{w}$ until conclusion). The weight perturbation algorithm (Jabri & Flower, 1992) performs this minimization with the update:

$$\Delta\mathbf{w} = -\eta\left(J(\mathbf{w}+\mathbf{z}) - J(\mathbf{w})\right)\mathbf{z}, \tag{2}$$

where the components of the "perturbation", $\mathbf{z}$, are drawn independently from a mean-zero distribution, and η is a positive scalar controlling the magnitude of the update (the "learning rate"). Performing a first-order Taylor expansion of $J(\mathbf{w}+\mathbf{z})$ yields:

$$\Delta\mathbf{w} = -\eta\left(J(\mathbf{w}) + \sum_i \frac{\partial J}{\partial \mathbf{w}_i} z_i - J(\mathbf{w})\right)\mathbf{z} = -\eta\sum_i \frac{\partial J}{\partial \mathbf{w}_i} z_i \cdot \mathbf{z}. \tag{3}$$

In expectation, this becomes the gradient times a (diagonal) covariance matrix, and reduces to

$$E[\Delta\mathbf{w}] = -\eta\sigma^2\frac{\partial J}{\partial \mathbf{w}}, \tag{4}$$

an unbiased estimate of the gradient, scaled by the learning rate and σ^2, the variance of the perturbation. However, this unbiasedness comes with a very high variance, as the direction of an update is uniformly distributed. It is only the fact that updates near the direction of the true gradient have a larger magnitude than do those nearly perpendicular to the gradient that allows for the true gradient to be achieved in expectation. Note also that all samples parallel to the gradient are equally useful, whether they be in the same or opposite direction, as the sign of the change in cost does not affect the resulting update.

The WP algorithm is one of the simplest examples of a policy gradient reinforcement learning algorithm, and in the special case when $\mathbf{z}$ is drawn from a Gaussian distribution, weight perturbation can be interpreted as a REINFORCE update (Williams, 1992).

4.2 The Shell Distribution

Rather than the Gaussian noise which is most commonly used for sampling, in our work we used a distribution in which $\mathbf{z}$ (the perturbation) is uniformly distributed in direction, but always has a fixed magnitude. We call this the shell distribution. This style of sampling was originally motivated by the intuitive realization that when a Gaussian distribution produced a small noise magnitude, the inherent noise in the

system effectively swamped out the change in cost due to the policy perturbation, preventing any useful update from taking place. When the SNR was studied in this domain (noisy policy evaluations and possibly poor baselines), it was found to support these conclusions (as discussed in Section 5.4). For these reasons, the shell distribution was used throughout this work, and as Section 5.5 demonstrates, tangible benefits were obtained.

5 Signal-to-Noise Ratio Analysis

Our experiments with sampling distributions quickly revealed that significant performance benefits could be realized through a better understanding of the effect of sampling distributions and measurement noise on learning performance. In this section we formulate a signal-to-noise ration (SNR) analysis of the policy gradient algorithms. This analysis formalized a number of our empirical observations about WP's performance, and gave insight in several improvements that offered real benefits to the speed of convergence.

5.1 Definition of the Signal-to-Noise Ratio

The SNR is the expected power of the signal (update in the direction of the true gradient) divided by the expected power of the noise (update perpendicular to the true gradient). Taking care to ensure that the magnitude of the true gradient does not effect the SNR, we have:

$$\text{SNR} = \frac{E\left[\Delta \mathbf{w}_{\|}^T \Delta \mathbf{w}_{\|}\right]}{E\left[\Delta \mathbf{w}_{\perp}^T \Delta \mathbf{w}_{\perp}\right]}, \tag{5}$$

$$\Delta \mathbf{w}_{\|} = \left(\Delta \mathbf{w}^T \frac{\mathbf{J_w}}{\|\mathbf{J_w}\|}\right) \frac{\mathbf{J_w}}{\|\mathbf{J_w}\|}, \quad \Delta \mathbf{w}_{\perp} = \Delta \mathbf{w} - \mathbf{w}_{\|}, \tag{6}$$

and using $\mathbf{J_w}(\mathbf{w}_0) = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\Big|_{(\mathbf{w}=\mathbf{w}_0)}$ for convenience.

Intuitively, this expression measures how large a proportion of the update is "useful". If the update is purely in the direction of the gradient the SNR would be infinite, while if the update moved perpendicular to the true gradient, it would be zero. As such, all else being equal, a higher SNR should generally perform as well or better than a lower SNR, and result in less violent swings in cost and policy for the same improvement in performance. For a more in depth study of the SNR's relationship to learning performance, see (Roberts & Tedrake, 2009).

5.2 Weight Perturbation with Gaussian Distributions

Evaluating the SNR for the WP update in Equation 2 with a deterministic $J(\mathbf{w})$ and $\mathbf{z}$ drawn from a Gaussian distribution yields a surprisingly simple result. If one first considers the numerator:

$$E\left[\Delta\mathbf{w}_{\parallel}^{T}\Delta\mathbf{w}_{\parallel}\right] = E\left[\frac{\eta^2}{\|\mathbf{J}_{\mathbf{w}}\|^4}\left(\sum_{i,j}J_{w_i}J_{w_j}z_iz_j\right)\mathbf{J}_{\mathbf{w}}{}^{T}\cdot\left(\sum_{k,p}J_{w_k}J_{w_p}z_kz_p\right)\mathbf{J}_{\mathbf{w}}\right]$$
$$= E\left[\frac{\eta^2}{\|\mathbf{J}_{\mathbf{w}}\|^2}\sum_{i,j,k,p}J_{w_i}J_{w_j}J_{w_k}J_{w_p}z_iz_jz_kz_p\right] = Q, \tag{7}$$

where we have named this term Q for convenience as it occurs several times in the expansion of the SNR. We now expand the denominator as follows:

$$E\left[\Delta\mathbf{w}_{\perp}^{T}\Delta\mathbf{w}_{\perp}\right] = E\left[\Delta\mathbf{w}^{T}\Delta\mathbf{w} - 2\Delta\mathbf{w}_{\parallel}^{T}(\Delta\mathbf{w}_{\parallel}+\Delta\mathbf{w}_{\perp}) + \Delta\mathbf{w}_{\parallel}^{T}\Delta\mathbf{w}_{\parallel}\right] = E\left[\Delta\mathbf{w}^{T}\Delta\mathbf{w}\right] - 2Q + Q \tag{8}$$

Substituting Equation (2) into Equation (8) and simplifying results in:

$$E\left[\Delta\mathbf{w}_{\perp}^{T}\Delta\mathbf{w}_{\perp}\right] = \frac{\eta^2}{\|\mathbf{J}_{\mathbf{w}}\|^2}E\left[\sum_{i,j,k}J_{w_i}J_{w_j}z_iz_jz_k^2\right] - Q. \tag{9}$$

We now assume that each component z_i is drawn from a Gaussian distribution with variance σ^2. Taking the expected value, it may be further simplified to:

$$Q = \frac{\eta^2}{\|\mathbf{J}_{\mathbf{w}}\|^4}\left(3\sigma^4\sum_i {J_{w_i}}^4 + 3\sigma^4\sum_i {J_{w_i}}^2\sum_{j\neq i}{J_{w_j}}^2\right) = \frac{3\sigma^4}{\|\mathbf{J}_{\mathbf{w}}\|^4}\sum_{i,j}{J_{w_i}}^2{J_{w_j}}^2 = 3\sigma^4, \tag{10}$$

$$E\left[\Delta\mathbf{w}_{\perp}^{T}\Delta\mathbf{w}_{\perp}\right] = \frac{\eta^2\sigma^4}{\|\mathbf{J}_{\mathbf{w}}\|^2}\left(2\sum_i {J_{w_i}}^2 + \sum_{i,j}{J_{w_i}}^2\right) - Q = \sigma^4(2+N) - 3\sigma^4 = \sigma^4(N-1), \tag{11}$$

where N is the number of parameters. Canceling σ results in:

$$\mathrm{SNR} = \frac{3}{N-1}. \tag{12}$$

Thus, for small noises and constant σ the SNR and the parameter number have a simple inverse relationship. This is a particularly concise model for performance scaling in PG algorithms.

5.3 SNR with Parameter-Independent Additive Noise

In many real world systems, the evaluation of the cost $J(\mathbf{w})$ is not deterministic, a property which can significantly affect learning performance. In this section we investigate how additive noise in the function evaluation affects the analytical expression for the SNR. We demonstrate that for very high noise WP begins to behave like a random walk, and we find in the SNR the motivation for the shell distribution; an improvement in the WP algorithm that will be examined in Section 5.4.

Consider modifying the update seen in Equation (2) to allow for a parameter-independent additive noise term v and a more general baseline $b(\mathbf{w})$, and again perform the Taylor expansion. Writing the update with these terms gives:

$$\Delta \mathbf{w} = -\eta \left(J(\mathbf{w}) + \sum_i J_{w_i} z_i - b(\mathbf{w}) + v \right) \mathbf{z} = -\eta \left(\sum_i J_{w_i} z_i + \xi(\mathbf{w}) \right) \mathbf{z}. \quad (13)$$

where we have combined the terms $J(\mathbf{w})$, $b(\mathbf{w})$ and v into a single random variable $\xi(\mathbf{w})$. The new variable $\xi(\mathbf{w})$ has two important properties: its mean can be controlled through the value of $b(\mathbf{w})$, and its distribution is independent of parameters $\mathbf{w}$, thus $\xi(\mathbf{w})$ is independent of all the z_i.

We now essentially repeat the calculation seen in Section 5.2, with the small modification of including the noise term. When we again assume independent z_i, each drawn from identical Gaussian distributions with standard deviation σ, we obtain the expression:

$$\text{SNR} = \frac{\phi + 3}{(N-1)(\phi+1)}, \quad \phi = \frac{(J(\mathbf{w}) - b(\mathbf{w}))^2 + \sigma_v^2}{\sigma^2 \|\mathbf{J}_\mathbf{w}\|^2} \quad (14)$$

where σ_v is the standard deviation of the noise v and we have termed the error component ϕ. This expression depends upon the fact that the noise v is mean-zero and independent of the parameters, although the assumption that v is mean-zero is not limiting. It is clear that in the limit of small ϕ the expression reduces to that seen in Equation (12), while in the limit of very large ϕ it becomes the expression for the SNR of a random walk (see (Roberts & Tedrake, 2009)). This expression makes it clear that minimizing ϕ is desirable, a result that suggests two things: (1) the optimal baseline (from the perspective of the SNR) is the value function (i.e., $b^*(\mathbf{w}) = J(\mathbf{w})$) and (2) higher values of σ are desirable as they reduce ϕ by increasing the size of its denominator. However, there is clearly a limit on the size of σ due to higher-order terms in the Taylor expansion; very large σ will result in samples which do not represent the local gradient. Thus, in the case of noisy measurements, there is some optimal sampling distance that is as large as possible without resulting in poor sampling of the local gradient. This is explored in the next section.

5.4 Non-Gaussian Distributions

The analysis in Section 5.3 suggests that for a function with noisy measurements there is an optimal sampling distance which depends upon the local noise and gradient as well as the strength of higher-order terms in that region. For a two-dimensional cost function that takes the form of a quadratic bowl in parameter space, Figure 4 shows the SNR's dependence upon the radius of the shell distribution (i.e., the magnitude of the sampling). For various levels of additive mean-zero noise the SNR was computed for a distribution uniform in angle and fixed in its distance from the mean (this distance is the "sampling magnitude"). The fact that there is a unique maximum for each case suggests the possibility of sampling *only* at that maximal magnitude,

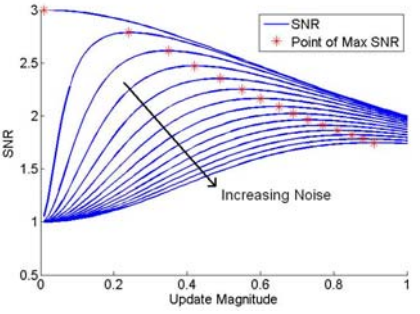
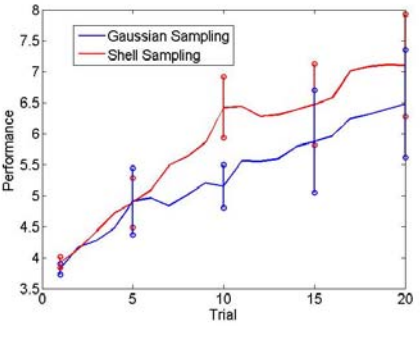


Fig. 4 SNR vs. update magnitude for a 2D quadratic cost function. Mean-zero measurement noise is included with variances from 0 to .65 (the value function's Hessian was diagonal with all entries equal to 2). As the noise is increased, the sampling magnitude producing the maximum SNR is larger and the SNR achieved is lower. Note that the highest SNR achieved is for the smallest sampling magnitude with no noise where it approaches the theoretical value (for 2D) of 3. Also note that for small sampling magnitudes and large noises the SNR approaches the random walk value of $1/N - 1$ (see (Roberts & Tedrake, 2009)).

rather than over all magnitudes as is done with a Gaussian, and thus improving SNR and performance. While determining the exact magnitude of maximum SNR may be impractical, by choosing a distribution with uniformly distributed direction and a constant magnitude close to this optimal value, performance can be improved.

5.5 Experimental Evaluation of Shell Distributions

To provide compelling evidence that the shell distribution could improve convergence in problems of interest, the shell distribution was implemented directly on the



(a)

Fig. 5 Five averaged runs on the heaving plate using Gaussian or Shell distributions for sampling. The error bars represent one standard deviation in the performance of different runs at that trial.

heaving plate, and the resulting learning curves were compared to those obtained using Gaussian sampling. For the purposes of this comparison, policy evaluations were run for long enough to reach steady state (to eliminate any issues relating to the coupling between consecutive evaluations). As can be seen in Figure 5, the shell distribution provided a real advantage in convergence rate on this system of interest, when dealing with the full dynamical complexity of a laboratory experimental system.

5.6 Implications for Learning at Intermediate Reynolds Numbers

The SNR demonstrates many of the properties of WP that make it so well suited to learning on partially observable and dynamically complicated system, such as fluid systems in the intermediate Reynolds number regime. The SNR shows that the system's dynamical complexity does not (locally) effect the difficulty of learning, as the dynamics appear nowhere in the expression. Instead, learning performance is locally effected by the number of parameters in the policy (N), the level of stochasticity in policy evaluations (σ_v), the quality of the baseline and the steepness of local gradients.

The SNR does not take into account the effects of higher-order behavior such as the roughness of the value function in policy space, which is in general a function of the system, the choice of parameterization and the choice of the cost function. These properties can be extremely important to the performance of WP, affecting both number and depth of local minima and the rate of learning, but are not analytically tractable in general.

6 Learning Results

6.1 Policy Representation Viewed through SNR

Due to the importance of the policy parameterization to the performance of the learning, it is important to pick the policy class carefully. Now armed with knowledge of the SNR, some basic guidelines for choosing the parameterization, previously justified heuristically, become more precise. Finding a rich representation with a small number of parameters can greatly improve convergence rates. Furthermore, certain parameterizations can be found with fewer local minima and smoother gradients than others, although determining these properties a priori is often impractical. A parameterization in which all parameters are reasonably well-coupled to the cost function is beneficial, as this will result in less anisotropy in the magnitude of the gradients, and thus larger sampling magnitudes and greater robustness to noise can be achieved. Prior to the periodic cubic spline, several parameterizations were tried, with all taking the form of encoding the z height of the wing as a function of time t over one period T, with the ends of the waveform constrained to be periodic (i.e., the beginning and end have identical values and first derivatives).

Parameterizing the policy in the seemingly natural fashion of a finite Fourier series was ruled out due to the difficulty in representing many intuitively useful waveforms (e.g., square waves) with a reasonable number of parameters. A parameterization using the sum of base waveforms (i.e., a smoothed square wave, a sine wave and a smoothed triangle wave) was used and shown to learn well, but was deemed a too restrictive class which predetermined many features of the waveform. Learning the base period T and the amplitude A of the waveform was also tried, and shown to perform well without significant difficulty. However, it was discovered that periods as long as possible and amplitudes as small as possible were selected, and thus these extra parameters were determined to not be of interest to the learning (this result was useful in determining how the system's dynamics related to the reward function, and is discussed in detail in Section 7).

6.2 Reward Function Viewed through SNR

The SNR also gives some greater insight into the formulation of the reward function. The form of the reward was discussed in Section 3.1, and justified by its connection to previous work in locomotive efficiency. We may now, however, understand more of what makes it advantageous from the perspective of learning performance and SNR. Its integral form performs smoothing on the collected data, which effectively reduces the noise level for a given trial, and intuitively it should differentiate meaningfully between different policies (e.g., a reward function that has no gradient with respect to the parameters in some region of policy space will find it very difficult to learn in that region).

6.3 Implementation of Online Learning

The SNR analysis presented here is concerned with episodic trials (i.e., a series of distinct runs), but this does not preclude online operation, in which trials follow one another immediately and the system runs continuously. While at first we ran longer policy evaluations (on the order of 40 or more flaps, averaged together) to reduce noise, and gave the system plenty of time (20 or more flaps) between trials to reach steady state and avoid inter-trial coupling, as we became more proficient at dealing with the high noise levels of the system we began to attempt to learn more aggressively. Through techniques such as sampling from the shell distribution, we were able to reduce trial length to a single flap (requiring just 1 second), and by reducing noise levels (ultimately choosing a shell distribution with a radius of approximately 4% of the parameter value) were able to eliminate the inter-trial settling time. Inter-trial correlation was explored, and while present, we found it did not significantly hamper learning performance, and that including eligibility traces did not greatly improve the rate of convergence.

6.4 Performance of Learning

In its final form, the SNR-optimized WP update was able to learn extremely efficiently. Using the policy parameterization described above, along with shell sampling and one-flap trials, the optimal policy was found within 7 minutes (around 400 flaps) even when starting far away in state space (see Figure 6). This quick convergence in the face of inter-trial coupling and high variance in policy evaluations (resulting from running them for such a short period of time) demonstrates the WP algorithm's robustness to these complex but very common difficulties.

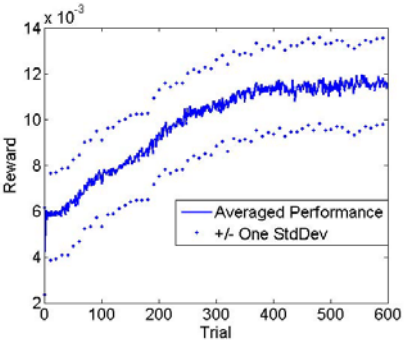


Fig. 6 The average of five learning curves using online learning (an update every second, after each full flapping cycle), with markers for +/- one standard deviation. The high variance is the result of large inter-trial variance in the cost, rather than large differences between different learning curves.

7 Interpretation of Optimal Solution

Once the learning had been successfully implemented, and repeatable convergence to the same optimum was achieved, it is interesting to investigate what the form of the solution suggests about the physical system. Figure 7 shows an example of an initial, intermediate and final waveform from a learning trial, starting at a smoothed out square wave and ending at the triangle wave which was found to be optimal.

The result is actually quite satisfying from a fluid dynamics point of view, as it is consistent with our theoretical understanding of the system, and indeed was the predicted solution given our reward function by experts in the field of flapping flight. If one considers the reward function used in this work (see 3.1), the basis of this behavior's optimality becomes clear.

Consider the cost of transport c_{mt}. As drag force is approximately quadratic with speed in this regime, the numerator behaves approximately as:

$$\int_T |F_z(t)\dot{z}(t)|dt \sim \frac{1}{2}\rho C_d \langle V^2 \rangle T, \tag{15}$$

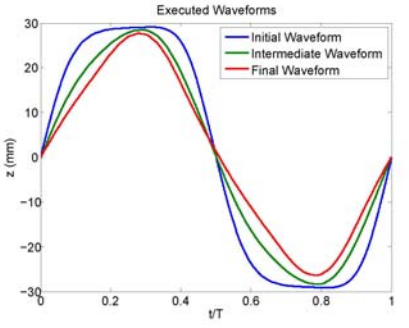


Fig. 7 A series of waveforms (initial, intermediate and final) seen during learning on the rigid plate.

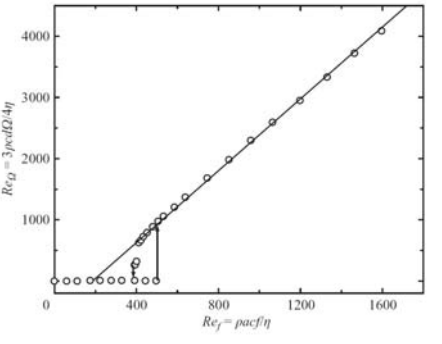


Fig. 8 Linear growth of forward speed with flapping frequency. The axes of this curve have been non-dimensionalized as shown, and the data was taken for a sine wave. Figure from (Vandenberghe et al., 2004).

where C_d is the coefficient of drag and $\langle V^2 \rangle$ is the mean squared heaving speed. However, forward rotational speed was found to grow linearly with flapping frequency (see (Vandenberghe et al., 2004) and Figure 8), thus the denominator can be written approximately as:

$$mg \int_T \dot{x}(t)dt \sim C_f \langle V \rangle T, \tag{16}$$

where C_f is a constant relating vertical speed to forward speed, and $\langle V \rangle$ is the mean vertical speed. Therefore, higher speeds result in quadratic growth of the numerator of the cost and linear growth of the cost's denominator. This can be seen as the reward r (the inverse of c_{mt}) having the approximate form:

$$r \sim C \frac{\langle V \rangle}{\langle V^2 \rangle}, \tag{17}$$

with C a constant. This results in lower speeds being more efficient, causing lower frequencies and amplitudes to be preferred. If period and amplitude are fixed, however, the average speed is fixed (assuming no new extrema of the stroke form are produced during learning, a valid assumption in practice). A triangle wave, then, is the means of achieving this average speed with the minimum average *squared* speed.

The utility of this control development method now becomes more clear. For systems that, despite having complicated dynamics, can be reasonably described with lumped-parameter or quasi-steady models, learning the controller directly serves dual purposes: it suggests lines of reasoning that inform the creation of relatively simple models, and it gives confidence that the modeling captures the aspects of the system relevant to the optimization. Furthermore, on systems for which tractable models are unavailable, the learning methodology can be applied just as easily while model-centric techniques will begin to fail.

8 Conclusion

This work has presented a case study in how to produce efficient, online learning on a complicated fluid system. The techniques used here were shown to be effective, with convergence being achieved on the heaving plate in approximately seven minutes. The algorithmic improvements presented have additional applications to many other systems, and by succeeding on a problem possessing great dynamic complexity, a reasonably large dimensionality, partial observability and noisy evaluations, they have been shown to be robust and useful. We believe the complexity of flow control systems is an excellent match for the capabilities of learning control, and expect to see many more applications for this domain in the future.

References

Alben, S., Shelley, M.: Coherent locomotion as an attracting state for a free flapping body. Proceedings of the National Academy of Science 102, 11163–11166 (2005)

Amari, S.: Natural gradient works efficiently in learning. Neural Computation 10, 251–276 (1998)

Baxter, J., Bartlett, P.: Infinite-horizon policy-gradient estimation. Journal of Artificial Intelligence Research 15, 319–350 (2001)

Bennis, A., Leeser, M., Tadmor, G., Tedrake, R.: Implementation of a highly parameterized digital PIV system on reconfigurable hardware. In: Proceedings of the Twelfth Annual Workshop on High Performance Embedded Computing (HPEC), Lexington, MA (2008)

Collins, S.H., Ruina, A., Tedrake, R., Wisse, M.: Efficient bipedal robots based on passive-dynamic walkers. Science 307, 1082–1085 (2005)

Greensmith, E., Bartlett, P.L., Baxter, J.: Variance reduction techniques for gradient estimates in reinforcement learning. Journal of Machine Learning Research 5, 1471–1530 (2004)

Howard, M., Klanke, S., Gienger, M., Goerick, C., Vijayakumar, S.: Methods for learning control policies from variable-constraint demonstrations. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 253–291. Springer, Heidelberg (2010)

Jabri, M., Flower, B.: Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks. IEEE Trans. Neural Netw. 3, 154–157 (1992)

Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence, 101 (1998)

Kober, J., Mohler, B., Peters, J.: Imitation and reinforcement learning for motor primitives with perceptual coupling. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 209–225. Springer, Heidelberg (2010)

Meuleau, N., Peshkin, L., Kaelbling, L.P., Kim, K.-E.: Off-policy policy search. In: NIPS (2000)

Peters, J., Vijayakumar, S., Schaal, S.: Policy gradient methods for robot control (Technical Report CS-03-787). University of Southern California (2003)

Roberts, J.W., Tedrake, R.: Signal-to-noise ratio analysis of policy gradient algorithms. In: Advances of Neural Information Processing Systems (NIPS), vol. 21, p. 8 (2009)

Shelley, M.: Personal Communication (2007)

Tedrake, R., Zhang, T.W., Seung, H.S.: Stochastic policy gradient reinforcement learning on a simple 3D biped. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan, pp. 2849–2854 (2004)

Vandenberghe, N., Childress, S., Zhang, J.: On unidirectional flight of a free flapping wing. Physics of Fluids, 18 (2006)

Vandenberghe, N., Zhang, J., Childress, S.: Symmetry breaking leads to forward flapping flight. Journal of Fluid Mechanics 506, 147–155 (2004)

Williams, J.L., Fisher III, J.W., Willsky, A.S.: Importance sampling actor-critic algorithms. In: Proceedings of the 2006 American Control Conference (2006)

Williams, R.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8, 229–256 (1992)

Abstraction Levels for Robotic Imitation: Overview and Computational Approaches

Manuel Lopes, Francisco Melo, Luis Montesano, and José Santos-Victor

Abstract. This chapter reviews several approaches to the problem of learning by imitation in robotics. We start by describing several cognitive processes identified in the literature as necessary for imitation. We then proceed by surveying different approaches to this problem, placing particular emphasys on methods whereby an agent first learns about its own body dynamics by means of self-exploration and then uses this knowledge about its own body to recognize the actions being performed by other agents. This general approach is related to the motor theory of perception, particularly to the mirror neurons found in primates. We distinguish three fundamental classes of methods, corresponding to three abstraction levels at which imitation can be addressed. As such, the methods surveyed herein exhibit behaviors that range from raw sensory-motor trajectory matching to high-level abstract task replication. We also discuss the impact that knowledge about the world and/or the demonstrator can have on the particular behaviors exhibited.

Manuel Lopes
Instituto de Sistemas e Robótica, Instituto Superior Técnico
Lisbon, Portugal
e-mail: macl@isr.ist.utl.pt

Francisco Melo
Carnegie Mellon University
Pittsburgh, PA, USA
e-mail: fmelo@cs.cmu.edu

Luis Montesano
Universidad de Zaragoza
Zaragoza, Spain
e-mail: lmontesa@unizar.es

José Santos-Victor
Instituto de Sistemas e Robótica, Instituto Superior Técnico
Lisbon, Portugal
e-mail: jasv@isr.ist.utl.pt

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 313–355.
springerlink.com

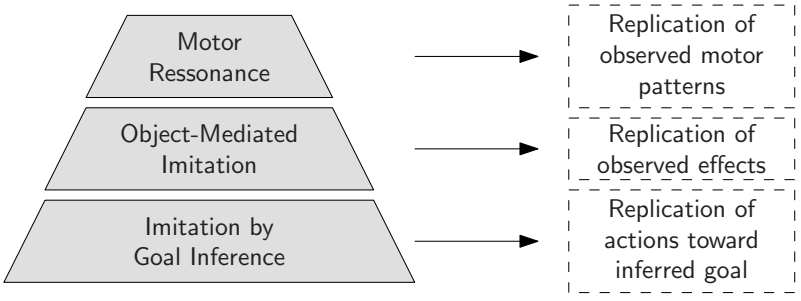


Fig. 1 Approaches to imitation at the three levels of abstraction discussed in this chapter.

1 Introduction

In this chapter we study several approaches to the problem of imitation in robots. This type of skill transfer is only possible if the robots have several cognitive capabilities that, in turn, pose multiple challenges in terms of modeling, perception, estimation and generalization. Throughout the chapter, we survey several methods that allow robots to learn from a demonstration. Several other surveys cover different aspects of imitation, including [6, 11, 16, 128].

Rather than providing another extensive survey of learning from demonstration, in this chapter we review some recent developments in imitation in biological systems and focus on robotics works that consider self-modeling as a fundamental part of the cognitive processes involved in and required for imitation. Self-modeling, in this context, refers to the learning processes that allow the robot to understand its own body and its interaction with the environment.

In this survey, we distinguish three fundamental classes of methods, each addressing the problem of learning by imitation at different levels of abstraction. Each of these levels of abstraction focuses on a particular aspect of the demonstration, giving rise to different imitative behaviors ranging from motor resonance to a more abstract imitation of inferred goals. This hierarchy of behaviors is summarized in the diagram of Fig. 1. It is interesting to note that the approaches at these different levels of abstraction, rather than being mutually exclusive, actually provide a natural hierarchical decomposition, in which approaches at the more abstracted levels can build on the outcome of methods in less abstract levels (see, for example, [80, 84] for an example of such integration).

Why Learn by Imitation?

The impressive research advances in robotics and autonomous systems in the past years have led to the development of robotic platforms of increasingly complex motor, perceptual and cognitive capabilities. These achievements open the way for new applications that require these systems to interact with other robots and/or human users during extended periods of time. Traditional programming methodologies and

robot interfaces will no longer suffice, as these systems need to *learn* to execute new complex tasks and improve their performance throughout its lifetime.

Learning by imitation is likely to become one primary form of teaching such complex robots [9, 127]. Paralleling the ability of human infants to learn through (extensive) imitation, an artificial system can retrieve a large amount of task related information simply by observing other individuals, humans or robots, perform that same task. Such a system would ideally be able to observe humans and learn how to solve similar tasks by imitation only. To be able to achieve such capability there are several other skills that must be developed first [84].

The ability to imitate has also been used in combination with other learning mechanisms. For instance, it can speed up learning either by providing an initial solution for the intended task that can then be improved by trial-and-error [109] or by guiding exploration [112, 114]. It also provides more intuitive and acceptable human-machine interactions due to its inherent social component [20, 79]. Learning by imitation has been applied before the advent of humanoid robots and in several different applications, including robotics [75], teleoperation [153], assembly tasks [149], game characters [139], multiagent systems [113], computer programming [49] and others.

What Is Imitation?

In biological literature, many behaviors have been identified under the general label of "social learning". Two such social learning mechanisms have raised particular interest among the research community, these being *imitation* and *emulation* [148]. In both the agent tries to replicate the effects achieved by the demonstrator but in imitation the agent also replicates the motor behavior used to achieve such goal, while in emulation only the effects are replicated (the agent achieves the effect by its own means).

In robotic research the word *imitation* is also used to represent many different behaviors and methodologies. Some works seek to clarify and distinguish several such approaches, either from a purely computational point-of-view [84, 89, 104, 127] or taking inspiration in the biological counterparts [25, 79, 140, 143, 146, 148, 154]. The taxonomy depicted in Fig. 2 provides one possible classification of different social learning mechanisms that takes into account three sources of information, namely *goals*, *actions* and *effects*.

In this paper, we define imitation in its daily use meaning and use the designations *imitation* and *learning/programming by demonstration* interchangeably. Taking into account the previous taxonomy the works presented may be classified under other labels. Roughly speaking we can consider the three levels as going from mimicking, through (goal) emulation and finally imitation. This division is clear if we consider methods that make an explicit inference about the goal as imitation, but not that clear in the cases where the trajectory generalization is performed using an implicit goal inference.

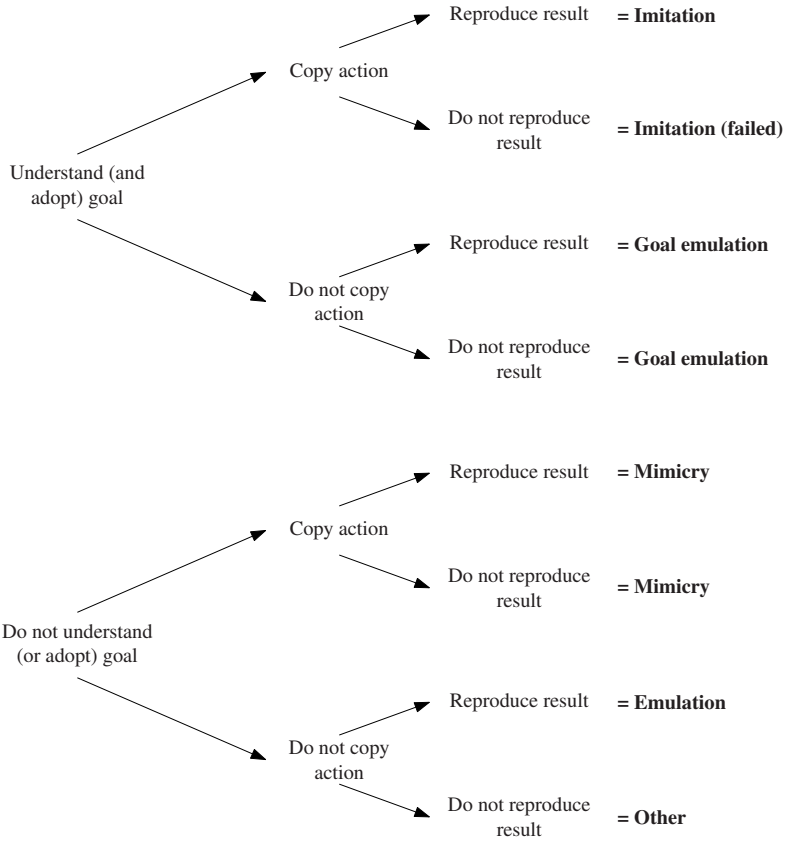


Fig. 2 Behavior classification in terms of goals, actions and effects (reproduced from [25]).

Organization of the Chapter

In the continuation, and before entering into the different computational approaches to imitation, Section 2 briefly outlines relevant aspects from psychology and neurophysiology on the topic of imitation in biological systems. Section 3 then discusses imitation in artificial systems, by pointing out the main scientific challenges that have been identified and addressed in the literature on imitation learning.

The remainder of the chapter is divided into three main sections, each addressing imitation from a specific perspective:

- Section 4 addresses imitation from a motor resonance perspective, namely *trajectory matching and generalization*. It discusses approaches that work at the trajectory level (either joint or task space). These approaches can be interpreted as performing regression (at the trajectory level) using the observed demonstration, and including additional steps to allow the learner to generalize from it.

- Section 5 discusses imitation by *replication of observed* (world) *events*. In this section, the learner focuses on replicating observed effects in the world, mainly effects on objects.
- *Goal inference* is finally presented in Section 6. We survey approaches in which the learner explicitly tries to infer the goal of the demonstrator and then uses this goal to guide its action-choice.

We note that such division is not strict and some of the approaches share ideas across several of the perspectives above. Also, depending on the application and context, one particular perspective might be more appropriate for imitation than the others. We conclude the paper in Sections 7 and 8 by discussing other approaches to imitation and providing some concluding remarks.

2 Imitation in Natural Systems

The idea of learning by imitation has a clear inspiration in the way humans and other animals learn. Therefore, results from neurophysiology and psychology on imitation in humans, chimpanzees and other primates are a valuable source of information to better understand, develop and implement artificial systems able to learn and imitate. Section 2.1 details information from neurophysiology and Section 2.2 presents evidence from psychology and biology. Such results illustrate the highly complex task that imitation is. The brief literature review in this section identifies some of the the problems that must be addressed before robots can learn (efficiently) by imitation and some works in the robotic literature that seek to model/test cognitive hypothesis.

2.1 Neurophysiology

Neurophysiology identified several processes involved in action understanding that, in turn, contributed differently to the development of learning approaches in robotics. For example, mirror neurons [51, 106] provided an significant motivation for using motor simulation theories in robotics. Similar ideas were suggested in speech recognition [50, 77]. Also the existence of forward and backward models in the cerebellum gave further evidence that the production system is involved in the perception, although it is not clear if it is necessary [152]. Most of these methods consider already known actions and not the way such knowledge can be acquired (we refer to [106] for further discussion). For the case of novel actions there is evidence that the mirror system is not sufficient to explain action understanding and a reasoning mechanism must be involved [19]. We now discuss several of these views in more detail.

Motor Theories of Perception

Several theories already claimed that the motor system is involved in perception. An example is the motor theory of speech perception [77]. The three main claims in this theory are: "(a) speech processing is special, (b) perceiving speech is perceiving

gestures, and (c) the motor system is recruited for perceiving speech". In [50], the authors revisit such theory taking into account the results from the last 50 years. The authors argue that although claim (a) is likely false, claims (b) and (c) are still likely to be true, although they admit that most of the findings supporting such claims may be explained by alternative accounts.

One evidence in favor of the theory that the motor system is involved in perception is the existence of several mechanisms in the brain involved in motor prediction and reconstruction. One such mechanism depends on the existence of several pairs of forward and backward models in the brain [152]. The forward model codes the perceptual effects of motor actions, while the backward model represents the inverse relation, *i.e.,* the motor actions that might cause a given percept. These models provide the agent with "simulation capabilities" for its own body dynamics, and are thus able to adapt to perturbations. They are also general enough to take into account task restrictions.

Mirror and Canonical Neurons

The discovery of *mirror neurons* [51, 96, 106] fostered a significant interest on the brain mechanisms involved in action understanding. These neurons are located in the F5 area of the macaque's brain and discharge during the execution of hand/mouth movements. In spite of their localization in a pre-motor area of the brain, mirror neurons fire both when the animal performs a specific goal-oriented grasping action and when it sees that same action being performed by another individual. This observation suggests that the motor system responsible for triggering an action is also involved in the recognition of the action. In other words, recognition may also involve motor information, rather than purely visual information. Furthermore, by establishing a direct connection between gestures performed by a subject and similar gestures performed by others, mirror neurons may be related to the ability to imitate found in some species [117], establishing an implicit level of communication between individuals.

Canonical neurons [96] have the intriguing characteristic of responding when objects that afford a *specific* type of grasp are present in the scene, even if the grasp action is not performed or observed. Thus, canonical neurons may encode object affordances, as introduced in [55], and may help distinguishing ambiguous gestures during the process of recognition. In fact, many objects are grasped in very precise ways that allow the object to be used for specific purposes. A pen is usually grasped in a way that affords writing and a glass is held in such a way that we can use it to drink. Hence, by recognizing an object that is being manipulated, it is also possible to attain information about the most likely grasping possibilities (expectations) and hand motor programs, simplifying the task of gesture recognition.

Reasoning Processes

Even if there is strong evidence that the motor system is involved in perception, it is not clear how fundamental it is and many claims on the mirror system are unlikely to

hold [56]. For instance, mirror neurons are not strictly necessary for action production as their temporal deactivation does not impair grasping control but only slows it down [47, 106]. On the other hand, more complex mechanisms than mirroring are necessary to understand unexpected behaviors of an agent. In [19] an experiment is presented where a person turns a light on using its knee. Similar demonstrations are shown where the person has the arms occupied with a folder, many folders or none. Results from an *fMRI* scan showed that the mirror mechanism is active during the empty arms situation (expected behavior) but it is not active during the other situation (unexpected behaviour). This and other similar results suggest that action understanding in unexpected situations is achieved by an inference-based mechanism taking the contextual constraints into account. In turn, this indicates that there may exist a reasoning mechanism to understand/interpret the observed behaviors.

2.2 Psychology

Studies in behavioral psychology have evidenced the ability of both children and chimpanzees to use different "imitative" behaviors. Individuals of both species also seem to switch between different such behaviors depending on perceived cues about the world [54, 62]. These cues include, for example, the inferred purpose of the observed actions [13, 14, 67] even when the action fails [67, 91, 145]. Other social learning mechanisms are analyzed in [154] under the more general designation of *social influence/learning*. In the continuation, we discuss some examples of behavior switching identified in the literature.

Imitation Capabilities and Behaviour Switching

Imitation and *emulation* are two classes of social learning mechanisms observed in both children and apes [138, 146, 148].[1] In imitation, the learning individual adheres to the inferred goal of the demonstrator, eventually adopting the same action choice. In emulation, on the other hand, the individual focuses on the observed *effects* of the actions of the demonstrator, possibly reaching these using a different action choice. The predisposition of an individual to imitate or emulate can thus be confirmed in tasks where the same effect can be achieved using different actions/motor patterns. For example, both chimpanzees and children are able to copy the choice of a push or twist movement in opening a box [147].

Children, in particular, can be selective about which parts of a demonstration to imitate [54, 150], but are generally more prone to imitate than to emulate. For example, children can replicate parts of a demonstration that are clearly not necessary to achieve the most obvious goal – a phenomenon known as *over-imitation* [62]. Over-imitation can be diminished by reducing the social cues or by increasing the urgency of task completion [21, 85, 88]. It has also been argued that over-imitation

[1] Other species, such as dogs, have also been shown to switch strategies after having observed a demonstration, as seen in [118].

can occur for a variety of social reasons [103] or because the individuals interpret the actions in the demonstration as causally meaningful [85].

Sensitivity to Task Constraints

Social animals also exhibit some sensitivity to the context surrounding the task execution, particularly task constraints. For example, in [90] 14-month-olds were shown a box with a panel that lit up when the demonstrator touched it with his forehead.The results showed that most infants reproduced the use of the forehead rather than using their hand when presented with the object a week later. This experiment was further extended in [54] by including a condition in which the demonstrator was restricted and could not use her hands. It was observed that only 21% of the infants copied the use of the forehead, against the 69% observed in a control condition replicating the [90] study. It was argued that, in the latter condition, infants recognize no constraints upon the demonstrator and thus encode the use of the forehead as a specific part of the intention. In the restricted case, they recognize the constraint as a extraneous reason for the use of the forehead and do not encode the specific action as part of the intention.

We return to this particular experiment in Section 6, in the context of computational models for social learning.

Imperfect Knowledge

Several experiments were conducted to investigate how the knowledge about the world dynamics influences social learning mechanisms. In one archetypical experiment, an individual observes a sequence of actions, not all of which are actually necessary to achieve the outcome. For example, in [62], preschoolers and chimpanzees were presented with two identical boxes, one opaque and one transparent. A demonstrator then inserted a stick into a hole on the top of the box and then into another hole on the front of the box. It was then able to retrieve of a reward from the box. In this experiment, the insertion of the stick into the top hole was unnecessary in order to obtain the reward, but this was only perceivable in the transparent box. The results showed that 3 and 4-year-old children tended to always imitate both actions. On the contrary, chimpanzees were able to switch between emulation and imitation if causal information was available: after having observed demonstrations in a transparent box, the chimpanzees were much less prone to insert the stick into the upper (useless) hole.

Goal Inference

Finally, it has been showed that some species exhibit imitative behavior beyond simple motion mimicry. For example, primates tend to interpret and actually reproduce observed actions in a teleological manner – that is, in terms of the inferred goals of the action [33]. In an experiment designed to test this hypothesis, 3 to 6-year-old children observed a demonstrator reaching across her body to touch a dot

painted on a table to one side of her, using the hand on her other side [13]. When prompted to reproduce the observed demonstration, children tended to copy the dot-touching action, but not the use of the contra-lateral hand. However, when the same demonstration was performed without a dot, children tended to imitate the use of the contra-lateral hand. It was argued that, in the first case, children interpreted the dot touching as the intention, choosing their own (easier) way to achieve it, while in the second case there was no clear target of the action but the action itself. As such, children interpreted the use of the contra-lateral hand as the intention and imitated it more faithfully. Results in experiments adapted for older children infants are similar [28].

2.3 Remarks

The experiments described above show that imitation behaviors result from several complex cognitive skills such as action understanding, reasoning and planning. Each of them depends on the physical and social context and also the knowledge of the agent. Partial world knowledge and contextual restrictions all influence the way an action is understood and replicated. A robot that is able to imitate in a flexible way should thus be able to consider all of such aspects.

3 Imitation in Artificial Systems

Imitation learning brings the promise of making the task of programming robots much easier [127]. However, to be able to imitate, robots need to have several complex skills that must be previously implemented or developed [84].

In Section 2 we discussed some of the complexities involved in the process of learning by imitation in natural systems, as well as all the contextual information taken into account when interpreting actions. Now, we replicate this discussion for artificial systems, outlining some of the issues that must be dealt with when developing an artificial system (*e.g.,* a robot) that can learn by imitation.

In [151], the authors identified three subproblems (or classes thereof) to be addressed in developing one such system:

- Mapping the perceptual variables (*e.g.,* visual and auditory input) into corresponding motor variables;
- Compensating for the difference in the physical properties and control capabilities of the demonstrator and the imitator; and
- Understanding the intention/purpose/reason behind an action (*e.g.,* the cost function to be minimized in optimal control that determines the action to be taken in each situation) from the observation of the resulting movements.

If we further take into account that the perceptual variables from the demonstrator must also be mapped from an allo- to an ego- frame of reference, the first of the above subproblems further subdivides into two other sub-problems: view-point transformation and sensory-motor matching [8, 22, 82, 83, 123]. The second of

the problems referred above is usually known as the *body correspondence problem* [4, 99, 100] and is, in a sense, closely related and dependent on the first problem of mapping between perception and action.

Data Acquisition

The way to address the issues discussed above will largely depend on the context in which imitation takes place. When used for robot programming, it is possible to use *data acquisition* systems that simplify the interpretation and processing of the input data, thus reducing partial observability issues. The latter is important since the learner will seldom be able to unambiguously observe all the relevant aspects of the demonstration. In particular, this can allow more robust and efficient algorithms to tackle the allo-ego transformation, the perception-to-action mapping and the body correspondence. Examples of such systems include exoskeletons, optical trackers or kinesthetic demonstrations [16].

Other applications of imitation occur in less controlled environments, for example as a result of the natural interaction between a robot and a human. In such contexts, perceptual problems must be explicitly addressed. Some authors address this problem adopting a computer vision perspective [82], modeling partial observability [40] or being robust to noise in the demonstration [26, 80, 116].

Mapping of Perceptual Variables and Body Correspondence

Many approaches do not consider a clear separation between data acquisition and learning by demonstration. One way to deal with the lack of information/data is to use prior knowledge to interpret the demonstration. *Action interpretation* strongly depends on the knowledge about how the world evolves as well as on the capabilities of both the learner and the demonstrator to interact with it [54, 62, 79]. This process is closely related with the two first issues pointed out in [151], since it provides a way to map external inputs to internal motor representations (*e.g.,* to robot actions). Therefore, imitation learning algorithms will typically benefit from prior knowledge about the environment, specially when data acquisition cannot provide a full description of the demonstration.

Knowledge about the agent's own body and its interaction with the world simplifies some of the difficulties found in imitation. On one hand, for the perception-to-action mapping, the recognition of others' actions can rely on the learner's model of the world dynamics, *e.g.,* by inferring the most probable state-action sequence given this model. This idea draws inspiration from psychological and neurophysiological theories of motor perception, where recognition and interpretation of behavior are performed using an internal simulation mechanism [5, 51, 83]. As seen in Section 2, mirror neurons are one of such mechanisms [51], and a significant amount of research in imitation learning in robotics flourished from this particular discovery.

On the other hand, this type of knowledge also allows action recognition and matching to occur with an implicit body correspondence, even if the bodies of the learner and demonstrator are different. Several works have explored this idea. For

example, in [82, 83, 99, 100, 130, 132], action matching is addressed at a trajectory level. In these works, the demonstration is interpreted taking into account the different dynamics of the learner and the demonstrator. In [71, 93, 94], the same problem is addressed at a higher level of abstraction that considers the *effects* on objects.

Goal/Intention Inference

Understanding actions and inferring intentions generally requires a more explicit reasoning process than just a mirror-like mechanism [19]. As discussed in Section 2, by further abstracting the process of learning by imitation to *task level* it is possible to additionally include contextual cues [10, 64, 79]. At this level of abstraction the third issue identified in [151] becomes particularly relevant.

Identifying the goal driving a demonstration is a particularly complex inference process, indeed an ill-defined one. What to imitate depends on several physical, social and psychological factors (see Section 2.2). One possible way to answer this question relies on the concept of *imitation metrics*. These metrics evaluate "how good" imitation is. Imitation metrics were first explicitly introduced in [101] in order to quantify the quality of imitation, to guide learning and also to evaluate learned behavior. However, it is far from clear what "good imitation" is and, perhaps more important, how variable/well-defined the learned behavior can be. Some studies along this direction have characterized the quality of imitation in humans. In [111], subjects were asked to perform imitation tasks and quantitative results were obtained to assess the effect of rehearsal during observation and repetition of the task.

In any case, it is often not clear whether imitation concerns the motor intention or the underlying goal of that motor intention [17, 23, 79, 127]. In other words, it is often the case that the agent cannot unambiguously identify whether it should imitate the action, its outcome, or the reason driving the demonstrator to do it. In each of the following sections we we discuss in detail each of these three approaches to imitation learning. In particular, we refer to Section 6 for a more detailed discussion on the problem of inferring the goal behind a demonstration. In that section we survey several recent works in which a learner does infer the goal of the demonstrator and adopts this goal as its own [2, 10, 64, 80, 119].

The Role of Self-Observation

It is interesting to note that most of the necessary information about the robot's body and the world dynamics can be gathered by self-observation [36, 84]. Although slow in many situations, it often allows a greater adaptation to changing scenarios.

Many different techniques can be used by the robot to learn about its own body [41, 84, 108, 129, 144]. For example, several works adopt an initial phase of motor babbling [57, 76, 83, 84]. By performing random motions, a great amount of data becomes available, allowing the robot to infer useful relations about causes and consequences of actions. These relations can then be used to learn a body schema useful in different application scenarios. The specific methods used to learn body

models vary, and range from parametric methods [27, 61], neural network methods [76, 82, 83] to non-parametric regression [41, 84, 144] and graphical models [57, 135]. As for learning about the world dynamics, this is closely related to the concept of learning affordances. Repeated interaction with the world allows the robot to understand how the environment behaves under its actions [46, 94, 95, 122]. As seen in the previous section, the knowledge about the world dynamics and the capabilities of others strongly influences how actions are understood.

♢

So far in this chapter we presented insights from neurophysiology, psychology and robotics research on the problems involved in learning by demonstration. The next sections will provide an overview of methods that handle such problems. We divide those methods according to the different formalisms or sources of information used, namely (a) trajectory matching and generalization, where sample trajectories are the main source of information from which the learner generalizes; (b) object mediated imitation, where effects occurring on objects are the relevant features of a demonstration; and (c) imitation of inferred goals, where there is an explicit estimation of the demonstrator's goal/intention.

4 Imitating by Motor Resonance

This section presents several methods that learn by demonstration by first mapping state-action trajectories to the learner's own body and then generalizing them.

Following what is proposed in [83, 151], the imitation process consists of the steps enumerated below and illustrated in Fig. 3:

(i) The learner observes the demonstrator's movements;
(ii) A viewpoint transformation (VPT) is used to map a description in the demonstrator's frame *allo-image* to the imitator's frame *ego-image*;
(iii) Action recognition is used (if necessary) to abstract the observed motion; and
(iv) A sensory-motor map (SMM) is used to generate the motor commands that have the higher probability of generating the observed features.

In this section we survey several methods that adopt this general approach. In these methods not all steps enumerated above are explicitly dealt with, but are still implicitly ensured by considering different simplifying assumptions.

4.1 Visual Transformations

The same motor action can have very distinctive perceptual results if one considers different points-of-view. For example when a person gestures goodbye, she can see the back of her hand, while when someone else is doing the same she will see the palm of the other's hand. This poses a problem when mapping actions from the demonstrator to the learner, as already identified in [22] and depicted in Fig. 4.

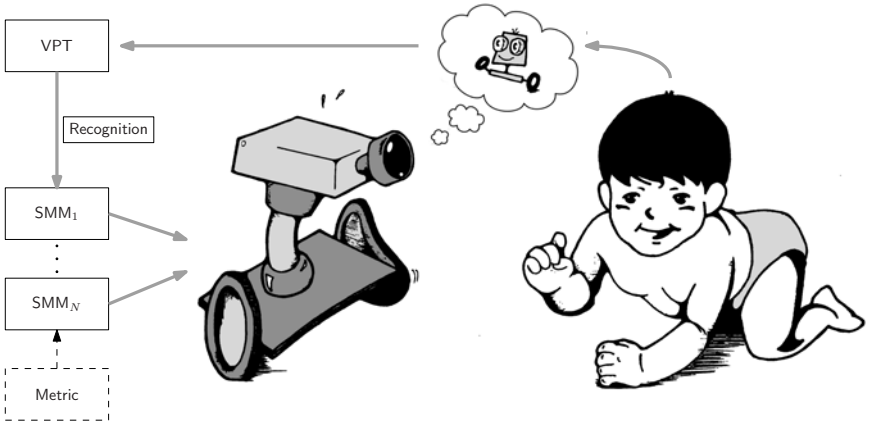


Fig. 3 Imitation architecture. Observed actions are first transformed to a ego frame of reference (VPT), where segmentation and recognition take place. After deciding on how to imitate, a correspondence between the two different bodies must be done by selecting the corresponding SMM. Finally, imitation is enacted.

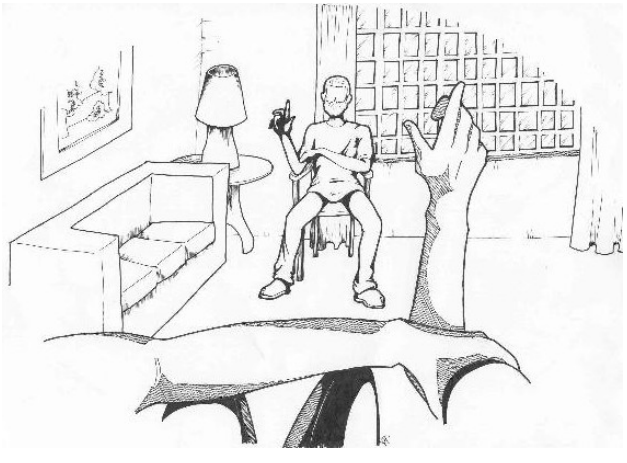


Fig. 4 Perceptual difference between the same gesture having an ego- or an allo- perspective.

The typical solutions for this problem is to perform a complete three-dimensional reconstruction. However, several works proposed alternative approaches that consider simplifications to such problem. In [8, 12, 52, 82, 83] several transformations are discussed, ranging from a simple image transformation (*e.g.,* mirroring the image) to a partial reconstruction assuming an affine camera and the full three dimensional reconstruction. These works also point out that such transformations can be seen as imitation metrics, because the depth information in some gestures can

indeed change the meaning of the action. Such transformations can also be done using neural networks [123].

4.2 Mimicking Behaviors and Automatic Imitation

Several works on imitation seek to transfer behaviors by a simple motor resonance process. The observation of perceptual consequences of simple motor actions can elicit an automatic mimicking behavior, and several initial works on imitation adopted this approach [34, 35, 53, 59]. In these works, the relation between changes in perceptual channels caused by a demonstrator are directly mapped into motor actions of the learner. Particularly in [34], the orientation of a mobile robot is controlled according to the observed position and orientation of a human head.

In these approaches the model about the own body is not explicit, the relations between action and perception is learned without concern about the true geometry and dynamics of the body.

Using the visual transformations introduced earlier it is possible to generate more complex behaviors. In [82, 83], the learner starts by acquiring a correspondence between its own perceptions and actions. The mimicking behavior results from mapping the perception of the demonstrator to its own using a view-point transformation, and then activating an inverse sensory-motor map. Different visual transformations result in different types of behavior.

Examples of other methods posing imitation within this visuo-somatic perspective include [7, 53]. An interesting approach is proposed in [18] where facial expression are learned by interacting with a person and creating a resonance of expressions.

4.3 Imitation through Motor Primitives

All previous approaches consider a simple perception-action loop in imitation. However, when considering a learner equipped with several motor primitives, it is possible to achieve more complex interactions. The "recognition" block in Fig. 3 represents the translation of observed trajectories in terms of such motor primitives. A motor sequence is perceived, maybe after some visual processing, and recognized as a specific motor primitive [86, 87]. This general approach can be used to perform human-machine interaction [20] or to learn how to sequence such primitives [23].

In [23] a string parsing mechanism is proposed to explain how apes are able to learn by imitation to process food. The string parsing mechanism is initialized with several sequences of primitives. Learning and generalization are performed by extracting regularities and sub-sequences. This approach can be seen as a grammatical inference process.

Other approaches use hidden Markov models to extract such regularities and filter behavior, usually in the context of tele-operation. The main goal of such approaches is to eliminate sluggish motion of the user and correct errors. We refer to [153] for an application of one such approach to the control of an Orbit Replaceable Unit. In this work, a robot observes an operator performing a task and builds a hidden Markov

model that describes that same task. In [63], a similar approach is used in assembly tasks. Such models can also be used to detect regularities in human motion [24].

Regularities of human motion can be represented in low-dimensions using principal component analysis [29], clustering [74] or other non-linear manifold learning techniques [65].

Some authors rely on completely separated methods to recognize and to generate motor primitives. Others combine both processes, thus exploring the self-modeling phase [37, 38, 66, 83]. As seen in Section 2.1, this process can be explained by the use of motor simulations of some kind. For example in both [38] and [151], action recognition is performed using the idea of coupled forward/backward models discussed in [152]. The former decouples each of the action primitives and is thus able to deal with a larger variety of tasks, while the latter is able to combine several primitives and deal with complex motor skills in a robust way. In [43, 44] dynamic neural networks are used to recognize actions goals taking into account task restrictions. A single neural network able to encode several behaviors was introduced in [137] and performs similar computations.

4.4 Learning of New Task Solutions

In some cases the learner has an explicit goal. However, it might be very difficult to plan how to reach such goal. This is specially important in complex environments or in situations involving highly redundant robots (*e.g.,* humanoid robots). One of the main motivations behind imitation learning is that it provides an easy way to program robots. Therefore, most approaches to imitation consider the learning of new actions. In such approaches two main trends have been adopted: one considers many demonstrations of the same task and tries to find invariants in the observed motions [17, 119]. The other uses only the observed trajectories as an initialization and then improves and generalizes further (*e.g.,* using reinforcement learning) [109].

In [112], imitation is used to speed up learning and several metrics were defined for evaluating the improvement in learning when using imitation. Imitation is used in [129] to learn dynamic motor primitives. As argued in this work, "*Movement primitives are parameterized policies that can achieve a complete movement behavior*". From this perspective, motor primitives can be seen as dynamical systems that generate different complex motions by changing a set of parameters. The authors also suggest the use of data from a demonstration to initialize such parameters. The parameters can then be optimized using, for example, policy gradient [109]. We point out that these methods consider classes of parameterized policies, namely the parameters of the dynamical system.

One of the few approaches taking uncertainty into account is proposed in [57]. The approach in this work starts by learning a Gaussian mixture model as a forward model, using self-observation [130]. The demonstration is taken as observation of a probabilistic process and the goal is to find a sequence of actions that maximizes the likelihood of such evidence. The work focuses on copying motions taking into account the dynamics of the robot and, as such, uses as observations the estimated state

trajectory and ignores the dynamic information. It achieves body correspondence by inferring the most probable trajectory using the imitator's body. Extra task restrictions can also be included. In [29] walking patterns are transferred from humans to robots after adapting it for different kinematics using low level representations.

Finally, several other methods are agnostic as to what is exactly the goal of the demonstrated task and aim only at learning the observed course of action. For example, in [31], learning from demonstration is formulated as a classification problem and solved using support-vector machines. These methods, in a sense, disregard the effects and social context of the demonstration, and focus only on replicating in each situation the demonstrated course of action (see Section 6 for a more detailed discussion on this). One disadvantage of this general approach is that it places excessive confidence on the demonstrator. Furthermore, the course of action learned is specific to the context and environment of the learner and, as such, is not generalizable to different environments.

5 Object Mediated Imitation

In the previous section, we have discussed imitation from a motor perspective. From this perspective, context is mainly provided by the body parts and corresponding motion. In this section we discuss a more abstract approach to imitation, where the context is enlarged to accommodate objects. In other words, the learner is now aware of the interaction with objects and, consequently, takes this information into account during learning. The most representative example of this type of behavior is *emulation* (see Fig. 2). In contrast with the motor resonance mechanisms discussed previously, which could perhaps be best described as mimicry, emulation focuses on copying (replicating) the results/effects of actions.

This abstraction from low-level control to higher-level representations of actions also facilitates reasoning about causality of actions, *i.e.,* how to induce specific changes to the environment. Consider the simple case of piling two objects. To learn this task, motor resonance alone does not suffice, as the learner must take into account which object can be placed on top of which depending on their sizes, shapes and other features. Another illustrative example is opening a door, where the shape of the handle provides meaningful information about how to perform the action. In the motor-resonance-based approaches, body correspondence addressed problems such as different number of degrees of freedom, or different kinematics and dynamics. In this section we discuss correspondence in terms of the usage that different objects have to different agents.

A biologically inspired concept related to the previous discussion is that of *affordances* [55]. Developed in the field of psychology, the theory of affordances states that the relation between an individual and the environment is strongly shaped by the individual's perceptual-motor skills. Back in the 70s, this theory established a new paradigm where action and perception are coupled at every level. Biological evidence of this type of coupling is now common in neuroscience [51] and

several experiments have shown the presence of affordance knowledge based on the perception of heaviness [141] or traversability [69].

Affordances have also been widely studied in robotics. In this section we discuss this concept of affordances in the context of imitation learning in robots, as well as the inclusion of object information and properties in the learning process. A thorough review on these topics can be found in [122], with special emphasis placed in affordance-based control.

We generally define affordances as mappings that relate *actions*, *objects* and *consequences* (effects). This very general concept can be modeled using different formalisms including dynamical systems [131], self-organizing maps [32], relational learning [58] and algebraic formulations [100].

However, independently of the selected representation or formalism, there are two core challenges to achieve affordance-based imitation: acquiring the model of affordances and exploiting this model. The latter depends heavily on the representation, but usually resorts to some type of action selection. For instance, if a dynamical system is used to encode a forward model, then the agent emulates by selecting the action that best matches the desired effect. It is worth mentioning that the approaches in this section are strongly dependent on a self-modeling phase as the concept of affordances is strictly a self-modeling idea.

The required data to infer the affordances model may be acquired either by *observation* or by *experimentation*. In the case of self-experimentation there is no body correspondence or visual transformation required, but such capability is important when interacting with other agents. When learning by observation such problem is immediately present. An advantage of *object mediated imitation* is that the match only occurs in the effects on object and so the specific body kinematics and action dynamics are not considers, thus simplifying several problems in imitation.

Affordances as Perception-Action Maps

A simple way of describing effects is to learn mappings from a set of predefined object features to changes in these features. This approach was used in a manipulation task to learn by experimentation the resulting motion directions as a function of the object shape and the poking direction [46]. Once the mapping has been learned, emulating an observed motion direction can be achieved by simply selecting the appropriate poking direction for the object. A similar approach was proposed in [71], where the imitation is also driven by the effects. In this case, the demonstrator's and imitator's actions are grouped according to the effect they produce in an object, irrespectively of their motor programs. Given an observed object and effect pair, the appropriate action (or sequence of actions) can then be easily retrieved. Another example is the use of invariant information [32, 48, 133]. In this case, the system learns invariant descriptions across several trials of the same action upon an object. Depending on the parameterization, the learned invariant descriptors may represent object characteristics or effects. Although this type of information has been usually applied in robot control, it is possible to use the invariants for emulation under the assumption that they are invariant to the viewpoint and that they capture the effects.

Grasping is a paradigmatic example of affordance knowledge that has been widely studied in the literature. Perception-action maps have appeared in several forms such as the Q-function of a reinforcement learning algorithm [155]; a pure regression map from object features to image points based labelled examples [125, 126] and self-experimentation [39, 92]; or as the correct position of a mobile robot to trigger a particular grasping policy [134].

Affordances as Dynamical Models

An alternative approach consists in modeling the dynamical system composed by the agent (demonstrator or imitator) and the environment. In [131], a hidden Markov model is used to encode the state of the agent and objects. In order to train the forward model, reflective markers were placed on the demonstrator and on the objects and tracked by a capture system. Viewpoint transformation then uses linear transformations between the demonstrator's and the imitator's body poses. Emulation is casted as a Bayesian decision problem over the Markov model, *i.e.,* selecting the maximum a posteriori action for each transition of the Markov chain. Interestingly, the model is able to modify the selected behavior with its own experience and refine the model previously learned solely by observation. Again, this is due to the fact that emphasis is placed on achieving the same effects instead of copying the action.

Dynamical systems have also been used for goal directed imitation in [43, 44]. The proposed architecture contains three interconnected layers corresponding to different brain areas responsible for the observed action, the action primitives and the goal. Each layer is implemented using a dynamic field that evolves with experience and its able to incorporate new representations using a correlation learning rule between adjacent neurons.

5.1 Bayesian Networks as Models for Affordances

Affordances can be seen as statistical relations between actions, objects and effects, modeled for example using Bayesian networks. One such approach was proposed in [94], in which the nodes in the network represent actions, object features or measured effects. As in standard Bayesian networks, the absence of a vertex between two nodes indicates conditional independence. Self-experimentation provides most of the data necessary to learn such relations. If a robot exerts its actions upon different objects, it can measure the effects of such actions. Even in the presence of noise the robot is able to infer that some actions have certain effects that depend on some of the object features. Also, the existence of irrelevant and redundant features is automatically detected.

Based on such prior experience, structure learning [60] can be used to distinguish all such relations. Once these dependencies are known, one can query the network to provide valuable information for several imitation behaviors. Table 1 summarizes the different input-output combinations. In particular, for emulation purposes, the ability to predict the effect conditioned on a set of available objects and the possible actions directly provides the robot with a way to select the appropriate action in a

Table 1 Affordances as relations between actions (A), objects (O) and effects (E) that can be used for different purposes: predict the outcome of an action, plan actions to achieve a goal or recognize objects or actions.

Inputs	Outputs	Function
(O,A)	E	Predict effect
(O,E)	A	Action recognition and planning
(A,E)	O	Object recognition and selection

single-step Bayesian decision problem. It is interesting to note that at this abstraction level the same mechanism is used for prediction and control, giving a mirror-like behavior.

This approach is halfway between learning specific maps and a full dynamical system description. If specific maps are learned, it is not easy to consider demonstrators with different dynamics nor to explicitly consider task restrictions. This approach is not as applicable as learning a full dynamical system description, because it is does not easily allow encoding long-term plans. It provides predictions for incremental state changes. For a more detailed discussion, we refer to [80].

5.2 Experiments

In this section we provide several experimental results obtained with a Bayesian network model for affordances. In particular, we describe both how the model of affordances can be learned by the robot and then used to attain affordance-based emulation behaviors.

For all experiments we used BALTAZAR [78], a robotic platform consisting of a humanoid torso with one anthropomorphic arm and hand and a binocular head (see Fig. 5). The robot is able to perform a set of different parameterized actions, namely $\mathscr{A} = \{a_1 = grasp(\lambda), a_2 = tap(\lambda), a_3 = touch(\lambda)\}$, where λ represents the height of the hand in the 3D workspace when reaching an object in the image. It also has implemented an object detector that extracts a set of features related to the object properties and the effects of the action.

Affordance Learning

We now describe the process by which the robot learned the affordance network used in the emulation behaviors. We recorded a total of 300 trials following the protocol depicted in Fig. 6. At each trial, the robot was presented with a random object of one of two possible shapes (round and square), four possible colors and three possible sizes (see Fig. 5 for an illustration of the objects used). BALTAZAR then randomly selects an action and moves its hand toward the object using pre-learned action primitives [84, 94]. When the reaching phase is completed, the robot then performs the selected action ($grasp(\lambda)$, $tap(\lambda)$ or $touch(\lambda)$) and finally returns

Fig. 5 Robot playground used in the experiments.

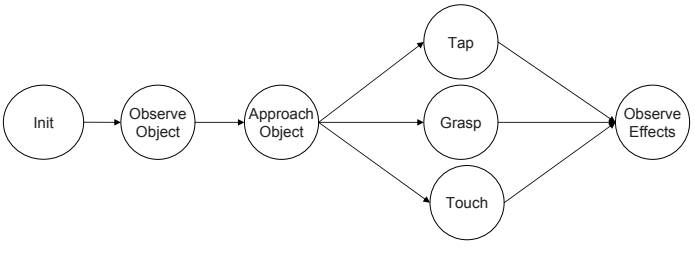


Fig. 6 Protocol used in the experiments. The object used in each trial is selected manually and the robot then randomly selects an action to interact with it. Object properties are recorded from the Init to the Approach states, when the hand is not occluding the object. The effects are recorded in the Observe state. Init moves the hand to a predefined position in open-loop.

the hand to the initial position. During the action, object features and effects are recorded.

Visual information is automatically clustered using the *X-means algorithm* [107]. The resulting classes constitute the input for the affordance learning algorithm. The features and their discretization are shown in Table 2. Summarizing, shape descriptors (*e.g.,* compactness and roundness) provided two different classes, size was discretized in 3 different classes and color in four. Based on this data, the robot adjusted the parameter λ for each action and then learned an affordance model as described above.

The learned model is shown in Fig. 7. The network was learned using Monte Carlo sampling with BDeu priors for the graph structure and a random network initialization. The dependencies basically state that color is irrelevant for the behavior of the objects under the available actions. In addition to this, a successful grasp requires the appropriate object size, while the velocity of the object depends on its

Table 2 Random variables in the network and possible values.

Symbol	Description	Values
A	Action	*grasp, tap, touch*
C	Color	$green_1$,$green_2$, *yellow, blue*
Sh	Shape	*ball, box*
S	Size	*small, medium, large*
OV	Object velocity	*small, medium, large*
HV	Hand velocity	*small, medium, large*
Di	Object-hand velocity	*small, medium, large*
Ct	Contact duration	*none, short, long*

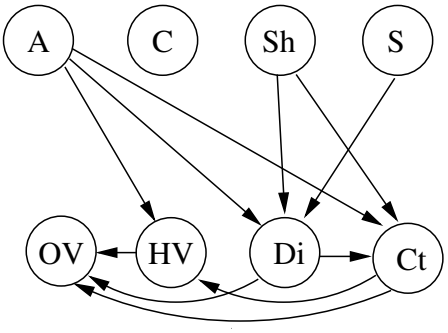


Fig. 7 Affordance network representing relations between actions, object features and the corresponding effects. Node labels are shown in Table 2.

shape and the action. We refer the reader to [94] for further details on the affordance learning.

Emulation

We now present the results obtained in several basic interaction games using the affordance network. The games proceed as follow. The robot observes a demonstrator performing an action on a given object. Then, given a specific imitation metric, it selects an action and an object to interact with so as to imitate (emulate) the demonstrator. Figure 8 depicts the demonstration, the different objects presented to the robot and the selected actions and objects for different metrics.

In the experiments we used two different demonstrations, a tap on a small ball (resulting in high velocity and medium hand-object distance) and a grasp on a small square (resulting in small velocity and small hand-object distance). Notice that contact information is not available when observing others.

The goal of the robot is to replicate the observed effects. The first situation (Fig. 8a) is trivial, as only tapping has a non-zero probability of producing high velocity. Hence, the emulation function selected a tap on the single object available.

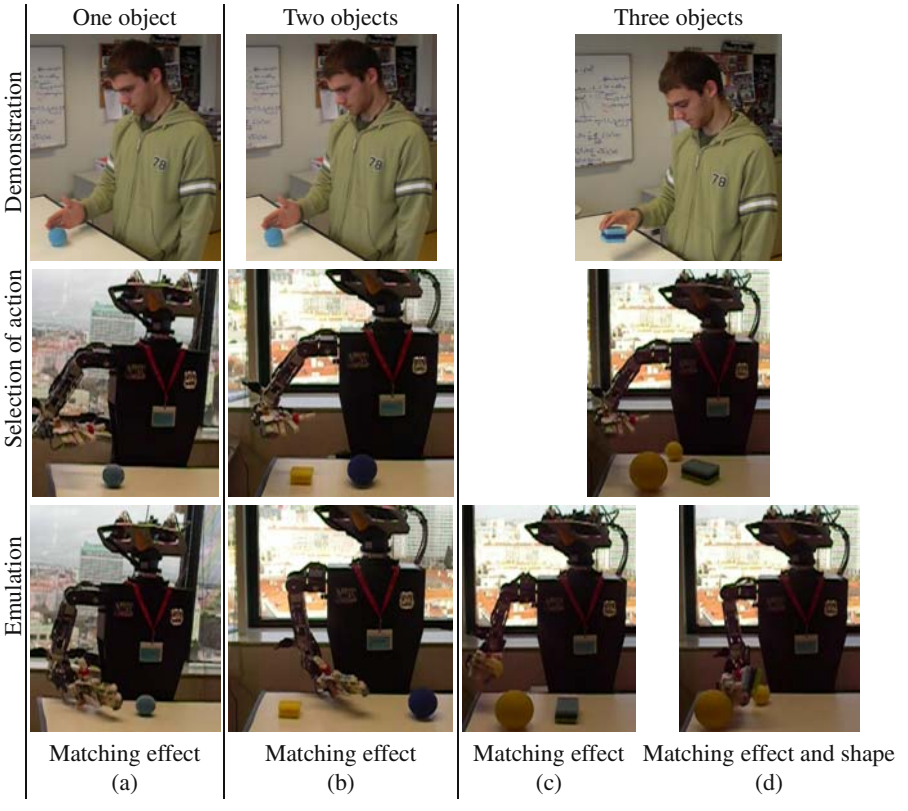


Fig. 8 Different emulation behaviors. Top row: Demonstration; Middle row: Set of potential objects; Bottom row: Emulation. Situations (a) through (d) represent: (a) emulation of observed action, (b) replication of observed effect, (c) replication of observed effect, and (d) replication of observed effect considering the shape of the object.

Table 3 Probability of achieving the desired effect for each action and the objects of Fig. 8b.

Obj \ Action	Grasp	Tap	Touch
Large blue ball	0.00	0.20	0.00
Small yellow box	0.00	0.06	0.00

In Fig. 8b the demonstrator performed the same action, but the robot now has to decide between two different objects. Table 3 shows the probabilities for the desired effects given the six possible combinations of actions and objects. The robot selected the one with highest probability and performed a tap on the ball.

Figures 8c and 8d illustrate how including the object features in the metric function produce different behaviors. After observing the grasp demonstration, the robot has to select among three objects: large yellow ball, small yellow ball and small blue

box. In the first case the objective was to obtain the same effects. The probability of grasping for each of the objects is 0.88, 0.92 and 0.52, respectively, and the robot grasped the small yellow ball even if the same object is also on the table (Fig. 8c). Notice that this is not a failure, since it maximizes the probability of a successful grasp which is the only requirement of the metric function.

We conclude by noting that other criteria can include more complex information, such as similarly shaped objects. When also taking this new criterion into account, the robot selected the blue box instead (see Fig. 8d).

6 Imitating by Inferring the Goal

So far in this chapter we have discussed learning by imitation at the motor level and at the effect level. The former focuses on replicating the exact movement observed, in a sense disregarding the effect on the environment and the social context in which the movement is executed. The latter addresses imitation at a higher level of abstraction, focusing on replicating the effects produced by the observed action in the environment, ignoring to some extent the exact motor trajectory executed. As seen in Section 2, knowledge about the world, the demonstrator and the learner's own body all influence the way a goal is inferred.

In this section we discuss imitation learning at a yet higher level of abstraction, approaching the concept of "imitation" according to the taxonomy in Fig. 2. Concretely, we discuss the fundamental process by which a learner can infer the *task* to be learned after observing the demonstration by another individual (*e.g.,* a human). We discuss several approaches from the literature that address the problem of inferring the goal of a demonstration at different levels. We then discuss in detail a recent approach to this problem that provides a close relation and potentially interesting insights into imitation in biological contexts.

6.1 Goal Inference from Demonstration

Inferring the goal behind a demonstration is, in general, a hard problem, as it requires some form of common background for the learner and the demonstrator. In social animals, this common background greatly depends on the social relation between the demonstrator and the learner. For example, social cues were found important in promoting imitation in infants [21, 91]. Several other studies address the general problem of understanding the process of inferring the goal/intention behind a demonstration [14, 67, 91]. Most such studies also address the related problem of understanding the process of perceiving *unfulfilled intentions*.

Translating this complex social learning mechanism into artificial systems usually requires the common background to be provided by the designer, who "imprints" in the system whatever of her own background knowledge it determines to be relevant for the particular environmental context of the system. As such, it is hardly surprising that different researchers address the problem of "goal inference" from perspectives as distinct as their own backgrounds and lines of work. For example, in [17] goal

inference is cast as an optimization problem. Motor theories of perception are used in [38] to build better imitation systems. These works essentially seek to determine which specific elements of the demonstration are relevant, seeking a *hard* answer to the fundamental problem of "What to imitate" discussed in Section 3.

Other recent works have adopted a fundamentally different approach, in which the learning agent chooses among a library of possible goals the one most likely to lead to the observed demonstration. For example, in [64] the problem of imitation is tackled within a planning approach. In this setting, the learner chooses between a pool of possible goals by assessing the optimality of the demonstration (viewed as a plan). Evaluative feedback from the demonstrator is also used to disambiguate between different possible goals.

One significant difficulty in inferring the goal behind a demonstration is that the same observed behavior can often be explained by several possible goals. Goal inference is, therefore, an ill-posed problem, and many approaches adopt a probabilistic setting to partly mitigate this situation [10, 119, 142]. For example, in [10], the authors address the problem of action understanding by children. To this purpose, they propose the use of a Bayesian model that matches observed inferences in children facing new tasks or environmental constraints. Similar ideas have been applied to robots in different works [80, 119, 142]. In [119], the goals of the robot are restricted to shortest-path problems while in [80, 142] general goals are considered. In [156], a maximum entropy approach is used to infer the goal in navigation tasks. The paper computes a distribution over "paths to the goal" that matches the observed empirical distributions but otherwise being as little "committed" as possible. Optimization is performed by a gradient-based approach. All these approaches handle the body correspondence problem by performing the recognition in terms of a self-world model.

In a sense, all the aforementioned approaches interpret the demonstration as providing "implicit" information about the goal of the demonstrator, a *soft* answer to the problem of "What to imitate". In other words, while the approaches in [17, 38] seek to single out a particular aspect of the demonstration to replicate, the latter assumes that the actual goal of the demonstration drives the action choice in the demonstration, but needs not be "contained" in it. This makes the latter approach more flexible to errors in the demonstration and non-exhaustive demonstrations. Another way of looking at the distinction between the two classes of approaches outlined above is by interpreting the latter as providing models and methods that allow the agent to extract a general *task description* from the demonstration, rather than a specific mapping from situations to actions that may replicate, to some extent, the observed behavior. This approach is closer to imitation in the biological sense, as defined in [25]. Finally, several recent works have proposed general models that contrast with those referred above in that they are able to generate *multiple* social-learning behaviors [79, 89].

In the remainder of this section we describe in detail the approach in [79]. Following the taxonomy in [25], our model takes into account several possible sources of information. Concretely, the sources of influence on our model's behavior are: beliefs about the world's possible states and the actions causing transitions between

them; a baseline preference for certain actions; a variable tendency to infer and share goals in observed behavior; and a variable tendency to act efficiently to reach the observed final states (or any other salient state).

6.2 Inverse Reinforcement Learning as Goal Inference

We now introduce the general formalism used to model the interaction of both the learning agent and the demonstrator with their environment. This approach shares many common concepts with those in [10, 119, 142], in that it infers a goal from the behaviors using bayesian inference to deal with noise and to disambiguate the set of possible goals.

Environment Model

At each time instant t, the environment can be described by its *state*, a random variable that takes values in a finite set of possible states (the state-space). The transitions between states are controlled to some extent by the actions of the agent (the demonstrator during the demonstration and the learner otherwise). In particular, at each time instant the agent (be it the learner or the demonstrator) chooses an action from its (finite) repertoire of action primitives and, depending on the action particular action chosen, the state evolves at time $t+1$ according to some transition probabilities $\mathbf{P}[X_{t+1} \mid X_t, A_t]$.

We assume that the learner has knowledge of its world, in the sense that it knows the set of possible states of the environment, its action repertoire and that of the demonstrator, and the world dynamics, *i.e.,* how both his and the demonstrator's actions affect the way the state changes (the transition probabilities). Note that we do not assume that this world knowledge is *correct*, in the sense that the agent may not know (or may know incorrectly) the transitions induced by certain actions. In any case, throughout this section we assume this knowledge as fixed – one can imagine the approach described herein eventually to take place after a period of self-modeling and learning about the world.[2] In this section, the modeled agent does not learn new actions, but instead learns how to combine known actions in new ways. In this sense, it is essentially distinct from the approach surveyed in Section 4.

Finally, in a first approach, we assume that the agent is able to recognize the actions performed by the demonstrator. In this section we do not discuss how this recognizer can be built, but refer that it can rely, for example, on the affordance models discussed in Section 5. Toward the end of the section we briefly discuss how the ability to recognize the demonstrator's actions affects the ability of the learner to recover the correct task to be learned (see also [80]).

In our adopted formalism, we "encode" a general task as a function r mapping states to real values that describes the "desirability" of each particular state. This function r can be seen as a *reward* for the learner and, once r is known, the problem

[2] To our knowledge, no work exists that explores knowledge acquisition simultaneously with learning by imitation, but we believe that such approach could yield interesting results.

falls back into the standard framework of Markov decision processes [115]. In fact, given the transition probabilities and the reward function r, it is possible to compute a function Q_r that, at each possible state, provides a "ranking" of all actions detailing how useful each particular action is in terms of the overall goal encoded in r. From this function $Q_r(x,a)$ it is possible to extract an optimal decision rule, henceforth denoted by π_r and referred as the *optimal policy for reward r*, that indicates the agent the best action(s) to choose at each state,

$$\pi_r(x) = \arg\max_a Q_r(x,a)$$

The computation of π_r, or equivalently Q_r, given r, is a standard problem and can be solved using any of several standard methods available in the literature [115].

Bayesian Recovery of the Task Description

We consider that the demonstration consists of a sequence $\mathscr{D}$ of state-action pairs

$$\mathscr{D} = \{(x_1,a_1),(x_2,a_2),\ldots,(x_n,a_n)\}.$$

Each pair (x_i,a_i) exemplifies to the learner the expected action (a_i) in each of the states visited during the demonstration (x_i). In the formalism just described, the goal inference problem lies in the estimation of the function r from the observed demonstration $\mathscr{D}$. Notice that this is closely related to the problem of *inverse reinforcement learning* as described in [1]. We adopt the method described in [89], which is a basic variation of the *Bayesian inverse reinforcement learning* (BIRL) algorithm in [116], but the same problem could be tackled using other IRL methods from the literature (see, for example, [102, 136]).

For a given reward function r, we define the *likelihood of a state-action pair*, (x,a), as

$$L_r(x,a) = \mathbf{P}\left[(x,a) \mid r\right] = \frac{e^{\eta Q_r(x,a)}}{\sum_b e^{\eta Q_r(x,b)}},$$

where η is a user-defined *confidence parameter* that we describe further ahead. The value $L_r(x,a)$ translates the plausibility of the choice of action a in state x when the underlying task is described by r. Therefore, the likelihood of a demonstration sequence $\mathscr{D}$ can easily be computed from the expression above. We use MCMC to estimate the distribution over the space of possible reward functions given the demonstration (as proposed in [116]) and choose the maximum *a posteriori*.

We note that it may happen that the transition model available is *inaccurate*. In this situation, the learner should still be able to perceive the demonstrated task, given that the "errors" in the model are not too severe. We also note that, in the process of estimating this maximum, the learner uses the knowledge concerning the action repertoire and world dynamics *of the demonstrator*. After the task description (the reward function r) is recovered, the learning agent then uses its own world model to compute the right policy for the recovered task in terms of its own world dynamics and action repertoire.

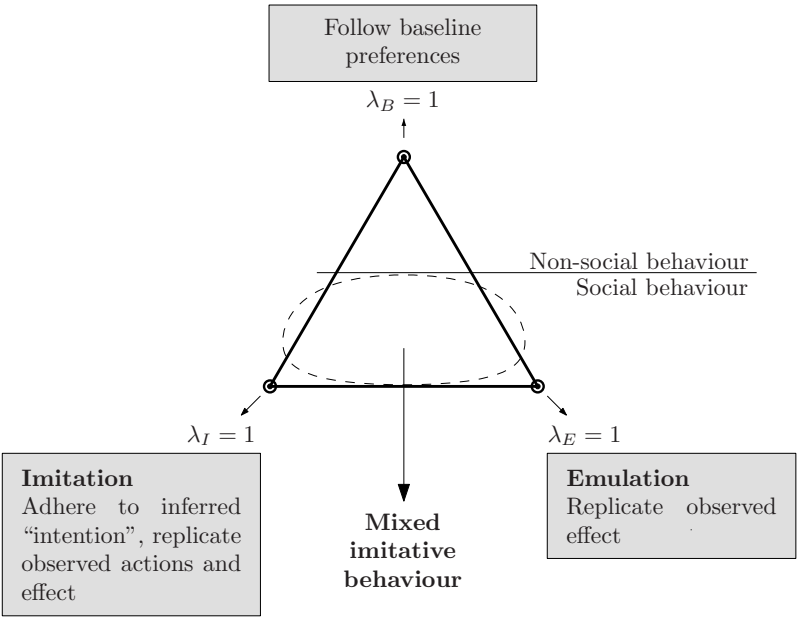


Fig. 9 Combination of several simple behaviors: Non-social behavior, emulation and imitation. The line separates the observed social vs non-social behavior, and does not correspond to the agent's reasoning (reproduced with permission from [79]).

A Model for Social Learning Mechanisms

Following the taxonomy in Fig. 2, we include in our model of social learning three sources of information to be used by the learner in determining the behavior to adopt [79]. The sources of influence on our model's behavior are baseline preferences for certain actions; a tendency to infer and share goals in observed behavior; and a tendency to act efficiently to reach rewarding states. Each of the three sources of information is quantified in terms of a utility functions Q_A, Q_I and Q_E, respectively. The learner will adhere to the decision-rule obtained by combining the three functions. In particular, the learner will adhere to the decision-rule associated with the function

$$Q^* = \lambda_A Q_A + \lambda_E Q_E + \lambda_I Q_I, \tag{1}$$

with $\lambda_A + \lambda_I + \lambda_E = 1$. By resorting to a convex combination as in Eq. 1, there is an implicit trade-off between the different sources of information (see also Fig. 9). It remains to discuss how Q_A, Q_I and Q_E are computed from the demonstration.

- The first source of information is the learner's *preference between actions*. This preference can be interpreted, for example, as representing a preference for "easier" actions than "harder" actions, in terms of the respective energetic costs. This preference corresponds to *natural inclinations* of the learner, and is independent

of the demonstration. The preference is translated in the corresponding utility function Q_A, whose values are pre-programmed into the agent.[3]

- The second source of information corresponds to the desire of the learner to replicate the *effects* observed in the demonstration. For example, the learner may wish to reproduce the change in the surroundings observed during the demonstration, or to replicate some particular transition experienced by the teacher. This can be translated in terms of a utility function Q_E by considering the reward function that assigns a positive value to the desired effect and then solving the obtained Markov decision process for the corresponding Q-function. The latter is taken as Q_E.
- The third and final source of information is related to the desire of the learner to pursue the same *goal* as the teacher. Given the demonstration, the learner uses the Bayesian approach outlined before, inferring the underlying intention of the teacher. Inferring this intention from the demonstration is thus achieved by a teleological argument [33]: the goal of the demonstrator is perceived as the one that more rationally explains its actions. Note that the goal cannot be reduced to the final effect only, since the means to reach this end effect may also be part of the demonstrator's goal. We denote the corresponding utility function by Q_I.

It is only to be expected that the use of different values for the parameters λ_A, λ_E and λ_I will lead to different behaviors from the learner. This is actually so, as illustrated by our experiments. We also emphasize that Q_E greatly depends on the world model of the *learner* while Q_I also depends on the world model of the *teacher*.[4]

6.3 Experiments

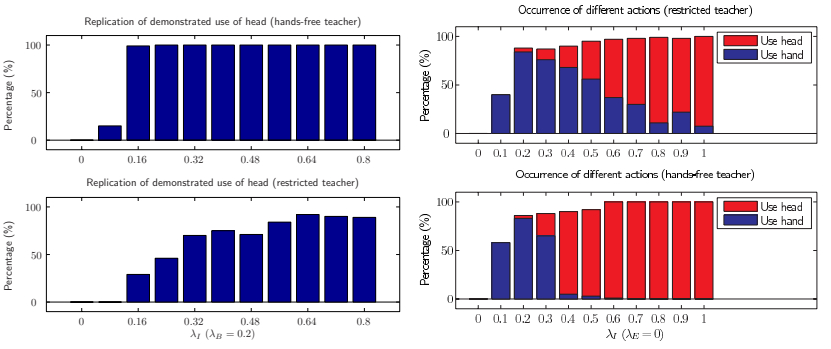
In this section we compare the simulation results obtained using our proposed model with those observed in a well-known biological experiment in children. We also illustrate the application of our imitation-learning framework in a task with a robot.

Modeling Imitation in Humans

In a simple experiment described in [90], several infants were presented with a demonstration in which an adult turned a light on by pressing it with the head. One week later, most infants replicated this peculiar behavior, instead of simply using their hand. Further insights were obtained from this experiment when, years later, a new dimension to the study was added by including task constraints [54]. In the new experiment, infants were faced with an adult turning the light on with the head but

[3] The exact values of Q_A translate, at each state, how much a given action is preferred to any other. The values are chosen so as to lie in the same range as the other utility functions, Q_I and Q_E.

[4] Clearly, the world model of the learner includes all necessary information relating the action repertoire for the learner and its ability to reproduce a particular effect. On the other hand, the world model of the teacher provides the only information relating the decision-rule of the teacher and its eventual underlying goal.



(a) (b)

Fig. 10 Percentages of replication of demonstrated action. (a) Percentage of runs in which the learner replicates the demonstrated use of head. Whenever the action was not performed with the head, it was performed with the hand. (b) Rates of occurrence of the different actions. When none of the two indicated actions is performed, no action is performed. In both plots, each bar corresponds to a trial of 2,000 independent runs.

having the hands restrained/occupied. The results showed that, in this new situation, children would display a more significant tendency to use their hands to turn the light on. The authors suggest that infants understand the goal and the restriction and so when the hands are occupied they emulate because they assume that the demonstrator did not follow the "obvious" solution because of the restrictions. Notice that, according to Fig. 2, using the head corresponds to *imitation* while using the hand corresponds to (goal) *emulation*.

We applied our model of social learning to an abstracted version of this experiment, evaluating the dependence of the behavior by the learner on the parameters λ_A, λ_E and λ_I in two distinct experiments. In the first experiment, we fixed the weight assigned to the baseline preferences (*i.e.,* we set $\lambda_A = 0.2$) and observed how the behavior changed as as λ_I goes from 0 to 1 (*i.e.,* as the learner increasingly adheres to the inferred goal of the demonstration). The results are depicted in Figure 10(a). Notice that, when faced with a restricted teacher, the learner switches to an "emulative" behavior much sooner, replicating the results in [54].

On a second experiment, we disregarded the observed effect (*i.e.,* we set $\lambda_E = 0$) and observed how the behavior of the learner changes as it assigns more importance to the demonstration and focuses less on its baseline preferences (*i.e.,* as λ_I goes from 0 to 1). The results are depicted in Figure 10(b). Notice that, in this test, we set λ_E to zero, which means that the agent is not explicitly considering the observed effect. However, when combining its own interests with the observed demonstration (that includes goals, actions and effects), the learner tends to *replicate the observed effect* and disregard the observed actions, thus displaying emulative behavior. This is particularly evident in the situation of a restricted teacher.

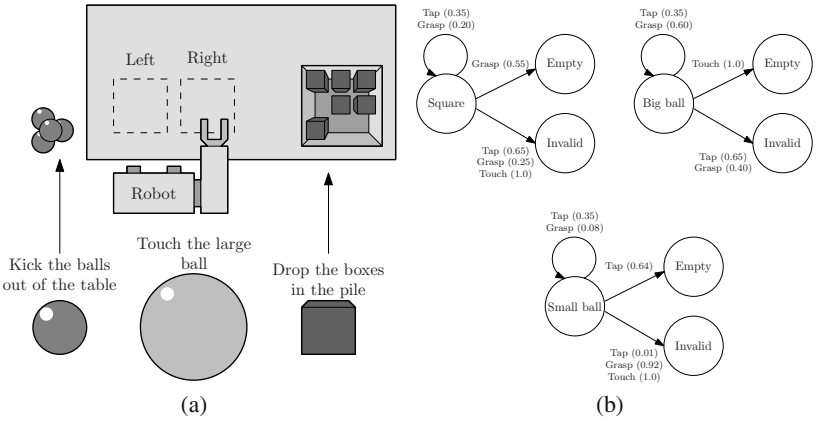


Fig. 11 Illustration of the recycling game. (a) The setup. (b) Transition diagrams describing the transitions for each slot/object.

We emphasize that the difference in behavior between the restricted and non-restricted teacher is due only to the *perceived difference on the ability of the teacher to interact with the environment*. We refer to [79] for further details.

Robot Learning by Imitation

We now present an application of our imitation learning model in a sequential task using BALTAZAR [78]. To test the imitation learning model in the robot we considered a simple recycling game, where the robot must separate different objects according to their shape (Figure 11). We set $\lambda_E = \lambda_A = 0$ and used only the imitation module to estimate the intention behind the demonstration. In front of the robot are two slots (Left and Right) where 3 types of objects can be placed: Large Balls, Small Balls and Boxes. The boxes should be dropped in a corresponding container and the small balls should be tapped out of the table. The large balls should be touched upon, since the robot is not able to efficiently manipulate them. Every time a large ball is touched, it is removed from the table by an external user. Therefore, the robot has available a total of 6 possible actions: Touch Left (TcL), Touch Right (ThR), Tap Left (TpL), Tap Right (TpR), Grasp Left (GrL) and Grasp Right (GrR).

For the description of the task at hand, we considered a state-space consisting of 17 possible states. Of these, 16 correspond to the possible combinations of objects in the two slots (including empty slots). The 17th state is an invalid state that accounts for the situations where the robot's actions do not succeed (for example, when the robot drops the ball in an invalid position in the middle of the table).

We first provided the robot with an error-free demonstration of the optimal behavior rule. As expected, the robot was successfully able to reconstruct the optimal policy. We also observed the learned behavior when the robot was provided with *two* different demonstrations, both optimal. The results are described in Table 4. Each

state is represented as a pair (S_1, S_2) where each S_i can take one of the values "Ball" (Big Ball), "ball" (Small Ball), "Box" (Box) or $\emptyset$ (empty). The second column of Table 4 then lists the observed actions for each state and the third column lists the learned policy. Notice that, as before, the robot was able to reconstruct an optimal policy, by choosing one of the demonstrated actions in those states where different actions were observed.

Table 4 Demonstration 1: Error free demonstration. Demonstration 2: Inaccurate and incomplete demonstration, where the boxed cells correspond to the states not demonstrated or in which the demonstration was inaccurate. Columns 3 and 5 present the learned policy for Demo 1 and 2, respectively.

State	Demo 1	Learned Pol.	Demo 2	Learned Pol.
($\emptyset$, Ball)	TcR	TcR	[-]	TcR
($\emptyset$, Box)	GrR	GrR	GrR	GrR
($\emptyset$, ball)	TpR	TpR	TpR	TpR
(Ball, $\emptyset$)	TcL	TcL	TcL	TcL
(Ball, Ball)	TcL,TcR	TcL,TcR	[GrR]	TcL
(Ball, Box)	TcL,GrR	GrR	TcL	TcL
(Ball, ball)	TcL	TcL	TcL	TcL
(Box, $\emptyset$)	GrL	GrL	GrL	GrL
(Box, Ball)	GrL,TcR	GrL	GrL	GrL
(Box, Box)	GrL,GrR	GrR	GrL	GrL
(Box, ball)	GrL	GrL	GrL	GrL
(ball, $\emptyset$)	TpL	TpL	TpL	TpL
(ball, ball)	TpL,TcR	TpL	TpL	TpL
(ball, Box)	TpL,GrR	GrR	TpL	TpL
(ball, ball)	TpL	TpL	TpL	TpL

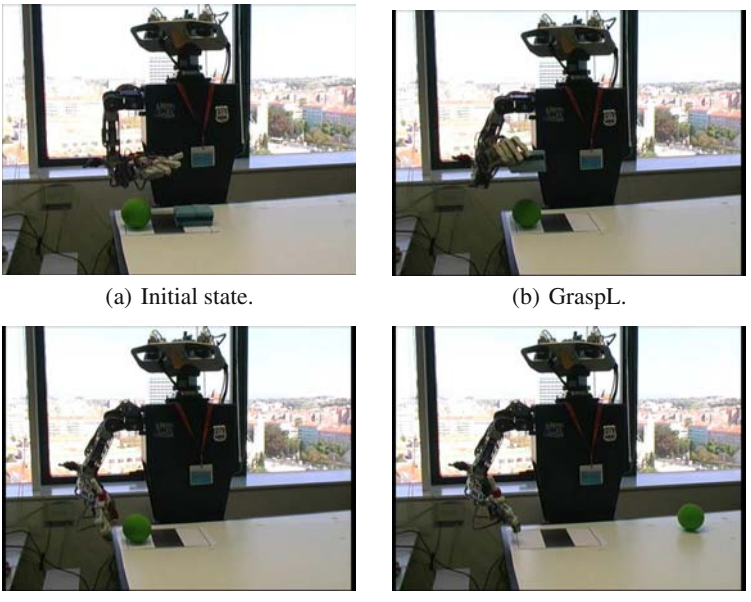
We then provided the robot with an *incomplete and inaccurate* demonstration. As seen in Table 4, the action at state ($\emptyset$, Ball) was never demonstrated and the action at state (Ball, Ball) was *wrong*. The last column of Table 4 shows the learned policy. Notice that in this particular case the robot was able to recover the *correct policy*, even with an incomplete and inaccurate demonstration.

In Figure 12 we illustrate the execution of the optimal learned policy for the initial state (Box, SBall).[5]

To assess the sensitivity of the imitation learning module to the action recognition errors, we tested the learning algorithm for different error recognition rates. For each error rate, we ran 100 trials. Each trial consists of 45 state-action pairs, corresponding to three optimal policies. The obtained results are depicted in Figure 13.

As expected, the error in the learned policy increases as the number of wrongly interpreted actions increases. Notice, however, that for small error rates ($\leq 15\%$) the robot is still able to recover the demonstrated policy with an error of only 1%. In

[5] For videos showing additional experiences see http://vislab.isr.ist.utl.pt/baltazar/demos/



(a) Initial state. (b) GraspL.

(c) TapR. (d) Final state.

Fig. 12 Execution of the learned policy in state (Box, SBall).

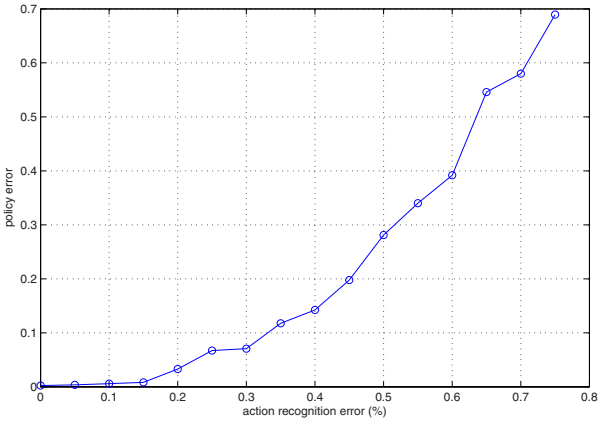


Fig. 13 Percentage of wrong actions in the learned policy as the action recognition errors increase.

particular, if we take into account the fact that the error rates of the action recognition method used by the robot are between 10% and 15%, the results in Figure 13 guarantee a high probability of accurately recovering the optimal policy.

We conclude by remarking that a more sophisticated model can be used in which observation noise is taken into account. This may allow more insensitivity to the noise, by including it explicit in the inference module that estimates the reward representing the goal of the demonstrator.

7 Other Imitation Settings

Imitation learning goes far beyond programming a single robot to perform a task, and has been used in many other settings.

For example, in [120], learning from demonstrated behavior somewhat resembles transfer learning, in which an agent observes demonstrations in different scenarios and uses this knowledge to recover a reward function that can be used in yet other scenarios. This problem is addressed as a max-margin structured learning problem to recover a reward function from a set of demonstrations. In order to simplify the problem and to leverage fast solution methods, the paper formulates the problem as a non-linear (non-differentiable) unconstrained optimization problem, that is tackled using subgradient techniques that rely on the solution structure of the (embedded) constraints.

In [45, 97], robots able to imitate have been used to interact with autistic children. On a related application, the Infanoid project [70, 72] deals with gesture imitation [71], interaction with people [73], and joint attention [98]. The robots in this project are used in human-robot interaction scenarios with particular emphasis on people with special needs. Although the results seem promising, and in short term people seem to react well, care must be taken in ensuring that the robots are used to promote socialization with other people, and not a stronger focus on the machine itself [121].

Some authors have also addressed imitation in multiagent scenarios, considering multiple demonstrators [132], multiple learners [30] and human-robot joint work [42]. In the presence of multiple demonstrators, these may be performing different tasks and the agent must actively select which one to follow. In [132], this observation led the authors to call their approach *active imitation*. Active learning approaches applied to imitation are very recent [81, 132]. Typically, the burden of selecting informative demonstrations has been completely on the side of the mentor. Active learning approaches endow the learner with the power to select which demonstrations the mentor should perform. Several criteria have been proposed: game theoretic approaches [132], entropy [81] and risk minimization [40].

Computational models of imitation have also been proposed to understand biology by synthesis. Examples include models of language and culture [3, 15], curiosity drives resulting in imitation behaviors [68], behavior switching in children and chimpanzees [79]. There have also been studies of imitation deficits relying on models of brain connections [110, 124]

We also note that there are other social learning mechanisms that fall outside the "imitation realm" in biological research. Often imitation is seen as a fundamental mental process for acquiring complex social skills but other mechanisms, although cognitively simpler, may have their own evolutionary advantages [89, 104, 105].

8 Discussion

In this chapter we presented an overview of imitation learning from two different perspectives. First, we discussed evidence coming from research in biology and neurophysiology and identified several cognitive processes required for imitation. We particularly emphasized two ideas that have a direct impact on imitation: 1) the action-perception coupling mechanisms involved, for instance, in the mirror system; and 2) the different social learning mechanisms found in infants and primates, not all of which can be classified as "true imitation". We also pointed out the importance of contextual cues to drive these mechanisms and to interpret the demonstration. As the bottom line, we stress that social learning happens at different levels of abstraction, from pure mimicry to more abstract cognitive processes.

Taking this evidence into account, we then reviewed imitation in artificial systems *i.e.,* methods to learn from a demonstration. As a result of the advances on this topic, there is currently a vast amount of work. Following the three main challenges identified in [151], we surveyed several methods that take advantage of the information provided by a demonstration in many different ways: as initial conditions for self-exploration methods (including planning), as exploration strategies, as data from which to infer world models, or as data to infer what the task is. These methods are being used with many different goals in mind, either to speed up robot programming, to develop more intuitive human-robot interfaces or to study cognitive and social skills of humans. In addition to this, we provide experimental results of increasing abstract imitation behaviors, from motor resonance to task learning.

An open question, and one we only addressed in an empirical way, is how all these methods are related or could be combined to achieve complex imitation behaviors. Indeed, different approaches usually tailor their formalisms to a particular domain of application. It is still not clear how different they are and if they can be used in several domains. If it becomes clear that they are indeed different, it would be interesting to understand how to switch between each mechanism, and eventually understand if there is a parallel in the brain.

Another important aspect that requires further research is related to perception. Although this is not specific to imitation, it plays a crucial role when interpreting the demonstration. Currently, robots are still unable to properly extract relevant information and perceive contextual restrictions from a general purpose demonstration. Due to this difficulty, having a robot companion that learns by imitation is still beyond our technological and scientific knowledge.

Nevertheless, most of these problems can be somewhat reduced when robot programming is conducted by skilled people that can handle more intrusive sensory modalities. In the chapter, we analyzed more in detail an alternative path to imitation which relies on previously learned models for the robot and the environment that help the understanding of the demonstration.

Using prior knowledge may simplify the interpretation of the demonstration, but requires the acquisition of good motor, perceptual and task descriptions. Most approaches consider predefined feature spaces for each of these entities. When considering object-related tasks, this problem is even more important than when

addressing pure motor tasks. A given world state may be described in terms of object locations, object-object relations, robot-object relations, among many others, but it is not easy to automatically extract, or choose, among the correct representations.

Finally, a recent trend in imitation learning tries to learn task abstractions from demonstrations. The rationale is that, once the robot has understood a task in an abstract manner, it can easily reason about the contextual cues that drive imitation behaviors, include them in future plans and, as a result, generalize better to other situations. In our experimental results, we showed how to combine multiple task descriptions to switch between different social learning behaviors through a biologically-inspired computational imitation model. Also, having such a representation opens the door to more general cognitive imitation architectures for robots.

Future applications of imitation will handle human-robot collaboration in cooperative settings (with several robots or people) and active strategies for interaction with the demonstrator.

We conclude this review by stating our belief that imitation and learning by demonstration will become one of the capabilities that future fully autonomous robots will extensively use, both to acquire new skills and to adapt to new situations in an efficient manner. The path to this objective is still full of exciting research challenges and fascinating links to the way we, humans, develop and learn.

Acknowledgement

This work was (partially) supported by: the Portuguese Fundação para a Ciência e a Tecnologia, the Information and Communications Technologies Institute (under the Carnegie Mellon-Portugal Program), the Programa Operacional Sociedade de Conhecimento (POS_C) that includes FEDER funds, and the project PTDC/EEA-ACR/70174/2006), and by the EU Projects (IST-004370) RobotCub and (EU-FP6-NEST-5010) Contact.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st International Conference on Machine Learning (ICML 2004), pp. 1–8 (2004)
2. Alissandrakis, A., Nehaniv, C.L., Dautenhahn, K.: Imitating with alice: Learning to imitate corresponding actions across dissimilar embodiments. IEEE Transactions on Systems, Man, & Cybernetics, Part A: Systems and Humans 32(4), 482–496 (2002)
3. Alissandrakis, A., Nehaniv, C.L., Dautenhahn, K.: Towards robot cultures? - learning to imitate in a robotic arm test-bed with dissimilarly embodied agents. Interaction Studies 5(1), 3–44 (2004)
4. Alissandrakis, A., Nehaniv, C.L., Dautenhahn, K.: Action, state and effect metrics for robot imitation. In: 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2006), Hatfield, United Kingdom, pp. 232–237 (2006)
5. Arbib, M.A., Billard, A., Iacoboni, M., Oztop, E.: Synthetic brain imaging: grasping, mirror neurons and imitation. Neural Networks 13, 975–997 (2000)

6. Argall, B., Chernova, S., Veloso, M.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5), 469–483 (2009)
7. Asada, M., Ogino, M., Matsuyama, S., Ooga, J.: Imitation learning based on visuo-somatic mapping. In: Marcelo, O.K., Ang, H. (eds.) 9th Int. Symp. Exp. Robot., vol. 21, pp. 269–278. Springer, Berlin (2006)
8. Asada, M., Yoshikawa, Y., Hosoda, K.: Learning by observation without three-dimensional reconstruction. In: Intelligent Autonomous Systems (IAS-6), pp. 555–560 (2000)
9. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: 14th International Conference on Machine Learning, pp. 12–20. Morgan Kaufmann, San Francisco (1997)
10. Baker, C.L., Tenenbaum, J.B., Saxe, R.R.: Bayesian models of human action understanding. In: Advances in Neural Information Processing Systems, vol. 18 (2006)
11. Bakker, P., Kuniyoshi, Y.: Robot see, robot do: An overview of robot imitation. In: Fogarty, T.C. (ed.) AISB-WS 1996. LNCS, vol. 1143, pp. 3–11. Springer, Heidelberg (1996)
12. Bandera, J.P., Marfil, R., Molina-Tanco, L., Rodríguez, J.A., Bandera, A., Sandoval, F.: Robot Learning by Active Imitation, pp. 519–544. ARS Publ. (2007)
13. Bekkering, H., Wohlschläger, A., Gattis, M.: Imitation of gestures in children is goal-directed. Quarterly J. Experimental Psychology 53A, 153–164 (2000)
14. Bellagamba, F., Tomasello, M.: Re-enacting intended acts: Comparing 12- and 18-month-olds. Infant Behavior and Development 22(2), 277–282 (1999)
15. Billard, A.: Imitation: A means to enhance learning of a synthetic proto-language in an autonomous robot. In: Imitation in Animals and Artifacs, pp. 281–311. MIT Press, Cambridge (1999)
16. Billard, A., Calinon, S., Dillman, R., Schaal, S.: Robot Programming by Demonstration. In: Handbook of Robotics, ch. 59. Springer, Heidelberg (2007)
17. Billard, A., Epars, Y., Calinon, S., Cheng, G., Schaal, S.: Discovering optimal imitation strategies. Robotics and Autonomous Systems 47(2-3) (2004)
18. Boucenna, S., Gaussier, P., Andry, P.: What should be taught first: the emotional expression or the face? In: 8th International conference on Epigenetic Robotics, Brighton, UK (2008)
19. Brass, M., Schmitt, R.M., Spengler, S., Gergely, G.: Investigating action understanding: Inferential processes versus action simulation. Current Biology 17(24), 2117–2121 (2007)
20. Breazeal, C.: Imitation as social exchange between humans and robots. In: AISB Symp. Imitation in Animals and Artifacts, pp. 96–104 (1999)
21. Brugger, A., Lariviere, L.A., Mumme, D.L., Bushnell, E.W.: Doing the right thing: Infants' selection of actions to imitate from observed event sequences. Child Development 78(3), 806–824 (2007)
22. Bruner, J.: Nature and use of immaturity. American Psychologist 27, 687–708 (1972)
23. Byrne, R.W.: Imitation without intentionality using string parsing to copy the organization of behaviour. Animal Cognition 2, 63–72 (1999)
24. Calinon, S., Billard, A.: Learning of gestures by imitation in a humanoid robot. In: Dautenhahn, K., Nehaniv, C.L. (eds.) Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions, pp. 153–177. Cambridge University Press, Cambridge (2007)
25. Call, J., Carpenter, M.: Three sources of information in social learning. In: Imitation in animals and artifacts. MIT Press, Cambridge (2002)

26. Cantin-Martinez, R., Lopes, M., Melo, F.: Inverse reinforcement learning with noisy observations. Tech. rep., Institute for Systems and Robotics, Lisbon, Portugal (2009)
27. Cantin-Martinez, R., Lopes, M., Montesano, L.: Active body schema learning. Tech. rep., Institute for Systems and Robotics, Lisbon, Portugal (2009)
28. Carpenter, M., Call, J., Tomasello, M.: Twelve- and 18-month-olds copy actions in terms of goals. Developmental Science 1(8), F13–F20 (2005)
29. Chalodhorn, R., Rao, R.P.N.: Learning to imitate human actions through Eigenposes. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 357–381. Springer, Heidelberg (2010)
30. Chernova, S., Veloso, M.: Teaching collaborative multi-robot tasks through demonstration. In: IEEE-RAS International Conference on Humanoid Robots, Korea (2008)
31. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. J. Artificial Intelligence Research 34, 1–25 (2009)
32. Cos-Aguilera, I., Cañamero, L., Hayes, G.: Using a SOFM to learn object affordances. In: Workshop of Physical Agents (WAF), Girona, Spain (2004)
33. Csibra, G., Gergely, G.: "Obsessed with goals": Functions and mechanisms of teleological interpretation of actions in humans. Acta Psychologica 124, 60–78 (2007)
34. Demiris, J., Hayes, G.: Imitative learning mechanisms in robots and humans. In: European Workshop on Learning Robots, pp. 9–16 (1996)
35. Demiris, J., Rougeaux, S., Hayes, G.M., Berthouze, L., Kuniyoshi, Y.: Deferred imitation of human head movements by an active stereo vision head. In: 6th IEEE Int. Workshop on Robot Human Communication, pp. 88–93 (1997)
36. Demiris, Y., Dearden, A.: From motor babbling to hierarchical learning by imitation: a robot developmental pathway. In: EPIROB 2005, Japan, pp. 31–37 (2005)
37. Demiris, Y., Hayes, G.: Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In: Dautenhahn, K., Nehaniv, C. (eds.) Imitation in Animals and Artifacts. MIT Press, Cambridge (2002)
38. Demiris, Y., Khadhouri, B.: Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems 54, 361–369 (2006)
39. Detry, R., Baseski, E., Popovi, M., Touati, Y., Kruger, N., Kroemer, O., Peters, J., Piater, J.: Learning object-specific grasp affordance densities. In: IEEE 8th International Conference on Development and Learning (2009)
40. Doshi, F., Pineau, J., Roy, N.: Reinforcement learning with limited reinforcement: using bayes risk for active learning in pomdps. In: Proceedings of the 25th international conference on Machine learning (ICML 2008), pp. 256–263 (2008)
41. D'Souza, A., Vijayakumar, S., Schaal, S.: Learning inverse kinematics. In: International Conference on Intelligent Robots and Systems, Hawaii, USA, pp. 298–303 (2001)
42. Erlhagen, W., Mukovskiy, A., Chersi, F., Bicho, E.: On the development of intention understanding for joint action tasks. In: 6th IEEE International Conference on Development and Learning (ICDL 2007), London, UK (2007)
43. Erlhagen, W., Mukovsky, A., Bicho, E.: A dynamic model for action understanding and goal-directed imitation. Brain Research 1083, 174–188 (2006)
44. Erlhagen, W., Mukovsky, A., Bicho, E., Panin, G., Kiss, C., Knoll, A., van Schie, H., Bekkering, H.: Goal-directed imitation in robots: a bio-inspired approach to action understanding and skill learning. Robotics and Autonomous Systems 54(5), 353–360 (2006)
45. Field, T., Field, T., Sanders, C., Nadel, J.: Children with autism become more social after repeated imitation sessions. Autism, 317–324 (2001)

46. Fitzpatrick, P., Metta, G., Natale, L., Rao, S., Sandini., G.: Learning about objects through action: Initial steps towards artificial cognition. In: IEEE International Conference on Robotics and Automation, Taipei, Taiwan (2003)
47. Fogassi, L., Gallese, V., Buccino, G., Craighero, L., Fadiga, L., Rizzolatti, G.: Cortical mechanism for the visual guidance of hand grasping movements in the monkey: A reversible inactivation study. Brain 124(3), 571–586 (2001)
48. Fritz, G., Paletta, L., Breithaupt, R., Rome, E., Dorffner, G.: Learning predictive features in affordance based robotic perception systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China (2006)
49. Furse, E.: A model of imitation learning of algorithms from worked examples. Cybernetics and Systems 32, 121–154 (2001)
50. Galantucci, B., Fowler, C.A., Turvey, M.T.: The motor theory of speech perception reviewed. Psychonomic Bulletin & Review 13(3), 361–377 (2006)
51. Gallese, V., Fadiga, L., Fogassi, L., Rizolaatti, G.: Action recognition in the premotor cortex. Brain 119, 593–609 (1996)
52. Gardner, M.: Imitation and egocentric perspective transformation. In: Virtual Poster associated with Perspectives on Imitation Conference (2002)
53. Gaussier, P., Moga, S., Quoy, J.B.,, M.: From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. Applied Artificial Intelligence: An Internation Journal 1(7) (1997)
54. Gergely, G., Bekkering, H., Király, I.: Rational imitation in preverbal infants. Nature 415, 755 (2002)
55. Gibson, J.J.: The Ecological Approach to Visual Perception. Houghton Mifflin, Boston (1979)
56. Gopnik, A.: Cells that read minds? what the myth of mirror neurons gets wrong about the human brain. Slate: Brains: A special issue on neuroscience and neuroculture (2007)
57. Grimes, D.B., Rashid, D.R., Rao, R.P.: Learning nonparametric models for probabilistic imitation. In: Advances in NIPS (2007)
58. Hart, S., Grupen, R., Jensen, D.: A relational representation for procedural task knowledge. In: Proceedings of the 2005 American Association for Artificial Intelligence (AAAI) Conference, Pittsburgh, USA (2005)
59. Hayes, G.M., Demiris, J.: A robot controller using learning by imitation. In: Int. Symp. Intelligent Robotic Systems, pp. 198–204 (1994)
60. Heckerman, D., Geiger, D., Chickering, M.: Learning bayesian networks: the combination of knowledge and statistical data. Machine Learning (1995)
61. Hersch, M., Sauser, E., Billard, A.: Online learning of the body schema. International Journal of Humanoid Robotics 5(2), 161–181 (2008)
62. Horner, V., Whiten, A.: Causal knowledge and imitation/emulation switching in chimpanzees (Pan troglodytes) and children (Homo sapiens). Animal Cognition 8, 164–181 (2005)
63. Hovland, G., Sikka, P., McCarragher, B.: Skill acquisition from human demonstration using a hidden markov model. In: IEEE International Conference on Robotics and Automation, Minneapolis, MN, pp. 2706–2711 (1996)
64. Jansen, B., Belpaeme, T.: A model for inferring the intention in imitation tasks. In: 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2006), Hatfield, UK (2006)
65. Jenkins, O., Bodenheimer, R., Peters, R.: Manipulation manifolds: Explorations into uncovering manifolds in sensory-motor spaces. In: International Conference on Development and Learning (ICDL 2006), Bloomington, IN, USA (2006)

66. Jenkins, O.C., Matarić, M.J., Weber, S.: Primitive-based movement classification for humanoid imitation. In: IEEE International Conference on Humanoid Robots, Humanoids 2000 (2000)
67. Johnson, S., Booth, A., O'Hearn, K.: Inferring the goals of a nonhuman agent. Cognitive Development 16(1), 637–656 (2001)
68. Kaplan, F., Oudeyer, P.Y.: The progress drive hypothesis: an interpretation of early imitation. In: Models and Mechanims of Imitation and Social Learning: Behavioural, Social and Communication Dimensions, pp. 361–377. Cambridge University Press, Cambridge (2007)
69. Konczak, J., Meeuwsen, H., Cress, M.: Changing affordances in stair climbing: The perception of maximum climbability in young and older adults. Journal of Experimental Psychology: Human Perception & Performance 19, 691–697 (1992)
70. Kozima, H.: Infanoid: An experimental tool for developmental psycho-robotics. In: International Workshop on Developmental Study, Tokyo, Japan (2000)
71. Kozima, H., Nakagawa, C., Yano, H.: Emergence of imitation mediated by objects. In: 2nd Int. Workshop on Epigenetic Robotics (2002)
72. Kozima, H., Nakagawa, C., Yano, H.: Attention coupling as a prerequisite for social interaction. In: IEEE International Workshop on Robot and Human Interactive Communication, Millbrae, USA (2003)
73. Kozima, H., Yano, H.: Designing a robot for contingency-detection game. In: International Workshop on Robotic and Virtual Agents in Autism Therapy, Hertfordshire, England (2001)
74. Kulić, D., Nakamura, Y.: Incremental Learning of Full Body Motion Primitives. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 383–406. Springer, Heidelberg (2010)
75. Kuniyoshi, Y., Inaba, M., Inoue, H.: Learning by watching: Extracting reusable task knowledge from visual observation of human performance. IEEE Trans. on Robotics and Automation 10(6), 799–822 (1994)
76. Kuniyoshi, Y., Yorozu, Y., Inaba, M., Inoue, H.: From visuo-motor self learning to early imitation-a neural architecture for humanoid learning. In: IEEE Int. Conf. Robotics and Automation, vol. 3, pp. 3132–3139 (2003)
77. Liberman, A.M., Mattingly, I.G.: The motor theory of speech perception revised. Cognition 21, 1–36 (1985)
78. Lopes, M., Beira, R., Praça, M., Santos-Victor, J.: An anthropomorphic robot torso for imitation: design and experiments. In: International Conference on Intelligent Robots and Systems, Sendai, Japan (2004)
79. Lopes, M., Melo, F., Kenward, B., Santos-Victor, J.: A computational model of social-learning mechanisms. Adaptive Behavior (to be published)
80. Lopes, M., Melo, F.S., Montesano, L.: Affordance-based imitation learning in robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, USA, pp. 1015–1021 (2007)
81. Lopes, M., Melo, F.S., Montesano, L.: Active learning for reward estimation in inverse reinforcement learning. In: European Conference on Machine Learning (ECML/PKDD), Bled, Slovenia (2009)
82. Lopes, M., Santos-Victor, J.: Visual Transformations in Gesture Imitation: What you see is what you do. In: IEEE Int. Conf. Robotics and Automation (2003)
83. Lopes, M., Santos-Victor, J.: Visual learning by imitation with motor representations. IEEE Trans. Systems, Man, and Cybernetics - Part B: Cybernetics 35(3) (2005)

84. Lopes, M., Santos-Victor, J.: A developmental roadmap for learning by imitation in robots. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 37(2) (2007)
85. Lyons, D.E., Young, A.G., Keil, F.C.: The hidden structure of over imitation. Proceedings of the National Academy of Sciences 104(50), 19751–19756 (2005)
86. Maistros, G., Marom, Y., Hayes, G.: Perception-action coupling via imitation and attention. In: AAAI Fall Symp. Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems (2001)
87. Matarić, M.J.: Sensory-motor primitives as a basis for learning by imitation: linking perception to action and biology to robotics. In: Imitation in Animals and Artifacts. MIT Press, Cambridge (2002)
88. McGuigan, N., Whiten, A., Flynn, E., Horner, V.: Imitation of causally opaque versus causally transparent tool use by 3- and 5-year-old children. Cognitive Development 22, 353–364 (2007)
89. Melo, F., Lopes, M., Santos-Victor, J., Ribeiro, M.I.: A unified framework for imitation-like behaviors. In: 4th International Symposium on Imitation in Animals and Artifacts, Newcastle, UK (2007)
90. Meltzoff, A.N.: Infant imitation after a 1-week delay: Long-term memory for novel acts and multiple stimuli. Developmental Psychology 24(4), 470–476 (1988)
91. Meltzoff, A.N.: Understanding the intentions of others: Re-enactment of intended acts by 18-month-old children. Developmental Psychology 31(5), 838–850 (1995)
92. Montesano, L., Lopes, M.: Learning grasping affordances from local visual descriptors. In: IEEE 8th International Conference on Development and Learning, China (2009)
93. Montesano, L., Lopes, M., Bernardino, A., Santos-Victor, J.: Affordances, development and imitation. In: IEEE - International Conference on Development and Learning, London, UK (2007)
94. Montesano, L., Lopes, M., Bernardino, A., Santos-Victor, J.: Learning object affordances: From sensory–motor coordination to imitation. IEEE Transactions on Robotics 24(1), 15–26 (2008)
95. Montesano, L., Lopes, M., Bernardino, A., Santos-Victor, J.: Learning object affordances: From sensory–motor coordination to imitation. IEEE Transactions on Robotics 28(1) (2008)
96. Murata, A., Fadiga, L., Fogassi, L., Gallese, V., Raos, V., Rizzolatti, G.: Object representation in the ventral premotor cortex (area f5) of the monkey. Journal of Neurophysiology 78(4), 2226–2230 (1997)
97. Nadel, J.: Imitation and imitation recognition: functional use in preverbal infants and nonverbal children with autism. In: The imitative mind. Cambridge University Press, Cambridge (2002)
98. Nagai, Y., Asada, M., Hosoda, K.: A developmental approach accelerates learning of joint attention. In: International Conference on Development and Learning (2002)
99. Nakaoka, S., Nakazawa, A., Yokoi, K., Hirukawa, H., Ikeuchi, K.: Generating whole body motions for a biped humanoid robot from captured human dances. In: IEEE International Conference on Robotics and Automation, Taipei, Taiwan (2003)
100. Nehaniv, C., Dautenhahn, K.: Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In: European Workshop on Learning Robots (1998)
101. Nehaniv, C.L., Dautenhahn, K.: Like me? - measures of correspondence and imitation. Cybernetics and Systems 32, 11–51 (2001)

102. Neu, G., Szepesvári, C.: Apprenticeship learning using inverse reinforcement learning and gradient methods. In: Uncertainty in Artificial Intelligence (UAI), pp. 295–302 (2007)
103. Nielsen, M.: Copying actions and copying outcomes: Social learning through the second year. Developmental Psychology 42(3), 555–565 (2006)
104. Noble, J., Franks, D.W.: Social learning mechanisms compared in a simple environment. In: Artificial Life VIII: Proceedings of the Eighth International Conference on the Simulation and Synthesis of Living Systems, pp. 379–385. MIT Press, Cambridge (2002)
105. Oudeyer, P.Y.: The Social Formation of Acoustic Codes with "Something Simpler". In: Dautenham, K., Nehaniv, C. (eds.) Second International Workshop on Imitation in Animals and Artefacts, AISB 2003, Aberystwyth, Wales (2003)
106. Oztop, E., Kawato, M., Arbib, M.: Mirror neurons and imitation: A computationally guided review. Neural Networks 19(3), 254–271 (2006)
107. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: International Conference on Machine Learning, San Francisco, CA, USA (2000)
108. Peters, J., Schaal, S.: Reinforcement learning of motor skills with policy gradients. Neural Networks, 682–697 (2008)
109. Peters, J., Vijayakumar, S., Schaal, S.: Natural Actor-Critic. In: Proc. 16th European Conf. Machine Learning, pp. 280–291 (2005)
110. Petreska, B., Billard, A.: A Neurocomputational Model of an Imitation Deficit following Brain Lesion. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 770–779. Springer, Heidelberg (2006)
111. Pomplun, M., Matarić, M.J.: Evaluation metrics and results of human arm movement imitation. In: IEEE-RAS Int. Conf. Humanoid Robotics (2000)
112. Price, B.: Accelerating reinforcement learning with imitation. Ph.D. thesis, University of British Columbia (2003)
113. Price, B., Boutilier, C.: Implicit imitation in multiagent reinforcement learning. In: Proc. ICML, pp. 325–334 (1999)
114. Price, B., Boutilier, C.: Accelerating reinforcement learning through implicit imitation. J. Artificial Intelligence Research 19, 569–629 (2003)
115. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Chichester (1994)
116. Ramachandran, D., Amir, E.: Bayesian inverse reinforcement learning. In: 20th Int. Joint Conf. Artificial Intelligence, India (2007)
117. Ramachandran, V.: Mirror neurons and imitation learning as the driving force behind the great leap forward in human evolution. Edge 69 (2000)
118. Range, F., Viranyi, Z., Huber, L.: Selective imitation in domestic dogs. Current Biology 17(10), 868–872 (2007)
119. Rao, R., Shon, A., Meltzoff, A.: A Bayesian model of imitation in infants and robots. In: Imitation and social learning in robots, humans, and animals. Cambridge University Press, Cambridge (2007)
120. Ratliff, N., Bagnell, J., Zinkevich, M.: Maximum margin planning. In: Proc. 23rd Int. Conf. Machine Learning, pp. 729–736 (2006)
121. Robins, B., Dautenhahn, K., Dubowski, J.: Robots as isolators or mediators for children with autism? a cautionary tale. In: AISB 2005: Social Intelligence and Interaction in Animals, Robots and Agents, Hatfield, UK, pp. 12–15 (2005)

122. Sahin, E., Cakmak, M., Dogar, M., Ugur, E., Ucoluk, G.: To afford or not to afford: A new formalization of affordances towards affordance-based robot control. Adaptive Behavior 15(5), 447–472 (2007)
123. Sauser, E., Billard, A.: View sensitive cells as a neural basis for the representation of others in a self-centered frame of reference. In: 3rd Int. Symp. Imitation in Animals & Artifacts (2005)
124. Sauser, E., Billard, A.: Parallel and Distributed Neural Models of the Ideomotor Principle: An Investigation of Imitative Cortical Pathways. Neural Networks 19(3), 285–298 (2006)
125. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. In: International Journal of Robotics Research, IJRR (2008)
126. Saxena, A., Wong, L., Ng, A.Y.: Learning grasp strategies with partial shape information. In: AAAI (2008)
127. Schaal, S.: Is imitation learning the route to humanoid robots. Trends in Cognitive Sciences 3(6), 233–242 (1999)
128. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Phil. Trans. of the Royal Society of London: Series B, Biological Sciences 358(1431) (2003)
129. Schaal, S., Peters, J., Nakanishi, J., Ijspeert, A.: Learning movement primitives. In: International Symposium on Robotics Research, ISRR 2003 (2003)
130. Shon, A., Grochow, K., Rao, R.: Robotic imitation from human motion capture using gaussian processes. In: IEEE/RAS International Conference on Humanoid Robots, Humanoids (2005)
131. Shon, A.P., Joshua, S.J., Rao, R.P.N.: Towards a real-time bayesian imitation system for a humanoid robot. In: IEEE - International Conference on Robotics and Automation, ICRA (2007)
132. Shon, A.P., Verma, D., Rao, R.P.N.: Active imitation learning. In: AAAI (2007)
133. Stoytchev, A.: Behavior-grounded representation of tool affordances. In: International Conference on Robotics and Automation, Barcelona, Spain (2005)
134. Stulp, F., Fedrizzi, A., Beetz, M.: Learning and performing place-based mobile manipulation. In: IEEE 8th International Conference on Development and Learning (2009)
135. Sturm, J., Plagemann, C., Burgard, W.: Adaptive body scheme models for robust robotic manipulation. In: RSS - Robotics Science and Systems IV, Zurich, Switzerland (2008)
136. Syed, U., Schapire, R., Bowling, M.: Apprenticeship learning using linear programming. In: Proceedings of the 25th International Conference on Machine Learning (ICML 2008), pp. 1032–1039 (2008)
137. Tani, J., Ito, M., Sugita, Y.: Self-organization of distributedly represented multiple behavior schemata in a mirror system: Reviews of robot experiments using rnnpb. Neural Networks 17(8/9), 1273–1289 (2004)
138. Tennie, C., Call, J., Tomasello, M.: Push or pull: Imitation vs. emulation in great apes and human children. Ethology 112(12), 1159–1169 (2006)
139. Thurau, C., Paczian, T., Sagerer, G.: Bayesian imitation learning in game characters. Int. J. Intelligent Systems Technologies and Applications 2(2/3) (2007)
140. Tomasello, M., Kruger, A.C., Ratner, H.H.: Cultural learning. Behavioral and Brain Sciences 16(3), 495–511 (1993)
141. Turvey, M., Shockley, K., Carello, C.: Affordance, proper function, and the physical basis of perceived heaviness. Cognition 73 (1999)
142. Verma, D., Rao, R.: Goal-based imitation as probabilistic inference over graphical models. In: Advances in NIPS, vol. 18 (2006)

143. Visalberghi, E., Fragaszy, D.: "Do monkeys ape?": ten years after. In: Imitation in Animals and Artifacts. MIT Press, Cambridge (2002)
144. Wang, J.M., Fleet, D.J., Hertzmann, A.: Gaussian process dynamical models for human motion. Trans. PAM, 283–298 (2008)
145. Want, S., Harris, P.: Learning from other people's mistakes: Causal understanding in learning to use a tool. Child Development 72(2), 431–443 (2001)
146. Want, S.C., Harris, P.L.: How do children ape? Applying concepts from the study of non-human primates to the development study of "imitation" in children. Developmental Science 5(1), 1–13 (2002)
147. Whiten, A., Custance, D., Gomez, J.C., Teixidor, P., Bard, K.A.: Imitative learning of artificial fruit processing in children (Homo sapiens) and chimpanzees (Pan troglodytes). Journal of Comparative Psychology 110, 3–14 (1996)
148. Whiten, A., Horner, V., Litchfield, C.A., Marshall-Pescini, S.: How do apes ape? Learning & Behavior 32(1), 36–52 (2004)
149. Williams, T.G., Rowland, J.J., Lee, M.H.: Teaching from examples in assembly and manipulation of snack food ingredients by robot. In: 2001 IEEE/RSJ, International Conference on Intelligent Robots and Systems, pp. 2300–2305 (2001)
150. Williamson, R.A., Markman, E.M.: Precision of imitation as a function of preschoolers' understanding of the goal of the demonstration. Developmental Psychology 42(4), 723–731 (2006)
151. Wolpert, D.M., Doya, K., Kawato, M.: A unifying computational framework for motor control and social interaction. Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 358(1431), 593–602 (2003)
152. Wolpert, D.M., Kawato, M.: Multiple paired forward and inverse models for motor control. Neural Networks 11(7-8), 1317–1329 (1998)
153. Yang, J., Xu, Y., Chen, C.: Hidden markov model approach to skill learning and its application to telerobotics. IEEE Transations on Robotics and Automation 10(5), 621–631 (1994)
154. Zentall, T.R.: Imitation in animals: Evidence, function, and mechanisms. Cybernetics and Systems 32(1), 53–96 (2001)
155. Zhang, J., Rössler, B.: Self-valuing learning and generalization with application in visually guided grasping of complex objects. Robotics and Autonomous Systems 47, 117–127 (2004)
156. Ziebart, B., Maas, A., Bagnell, J., Dey, A.: Maximum entropy inverse reinforcement learning. In: Proc. 23rd AAAI Conf. Artificial Intelligence, pp. 1433–1438 (2008)

Learning to Imitate Human Actions through Eigenposes

Rawichote Chalodhorn and Rajesh P.N. Rao

Programming a humanoid robot to perform an action that takes into account the robot's complex dynamics is a challenging problem. Traditional approaches typically require highly accurate prior knowledge of the robot's dynamics and the environment in order to devise complex control algorithms for generating a stable dynamic motion. Training using human motion capture (mocap) data is an intuitive and flexible approach to programming a robot but direct usage of kinematic data from mocap usually results in dynamically unstable motion. Furthermore, optimization using mocap data in the high-dimensional full-body joint-space of a humanoid is typically intractable. In this chapter, we purposes a new model-free approach to tractable imitation-based learning in humanoids by using *eigenposes*.

The proposed framework is depicted in Fig. 1. A motion capture system transforms the Cartesian positions of markers attached to the human body to joint angles based on kinematic relationships between the human and robot bodies. Then, linear PCA is used to create eigenpose data, which are representation of whole-body posture information in a compact low-dimensional subspace. Optimization of whole-body robot dynamics to match human motion is performed in the low dimensional subspace by using eigenposes. In particular, sensory feedback data are recorded from the robot during motion and a causal relationship between eigenpose actions and the expected sensory feedback is learned. This learned sensory-motor mapping allows humanoid motion dynamics to be optimized. An inverse mapping that maps optimized eigenpose data from the low-dimensional subspace back to the original joint space is then used to generate motion on the robot. We present several results

Rawichote Chalodhorn
Neural Systems Laboratory, Department of Computer Science and Engineering,
University of Washington, Seattle, WA 98195-2350 U.S.A.
e-mail: choppy@cs.washington.edu

Rajesh P.N. Rao
Neural Systems Laboratory, Department of Computer Science and Engineering,
University of Washington, Seattle, WA 98195-2350 U.S.A.
e-mail: rao@cs.washington.edu

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 357–381.
springerlink.com

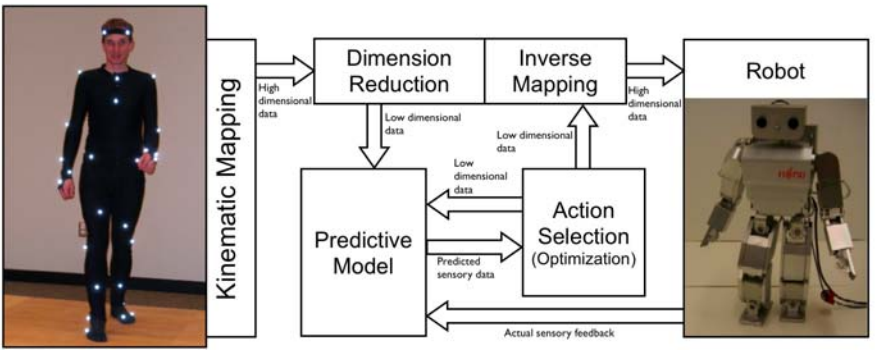


Fig. 1 A framework for learning human behavior by imitation through sensory-motor mapping in low dimensional subspaces.

demonstrating that the proposed approach allows a humanoid robot to learn to walk based solely on human motion capture without the need for a detailed physics-based model of the robot.

1 Related Work

Imitation is an important learning mechanism in many biological systems including humans [16]. In humans, a wide range of behaviors, from styles of social interaction to tool use, are passed from one generation to another through imitative learning. Unlike trial-and-error-based learning methods such as reinforcement learning (RL) [18], imitation-based learning is fast: given a demonstration of a desired behavior, the learning agent only has to search for the optimal solution within a small search space. The potential for rapid behavior acquisition through demonstration has made imitation learning an increasingly attractive alternative to manually programming robots. It is straightforward to recover kinematic information from human motion using, for example, motion capture, but imitating the motion with stable robot dynamics is a much harder problem. Stable imitation requires deriving appropriate action commands that matches the robot's dynamics and the dynamic interaction between the robot and its environment. Sensory feedback data must also be taken into account for achieving stable imitation.

The idea of using imitation to train robots has been explored by a number of researchers. Demiris and Hayes [5] demonstrated imitative learning using a wheeled mobile robot that learned to solve a maze problem by imitating another homologous robot. Billard [2] showed that imitation is a mechanism that allows the robot imitator to share a similar set of proprio- and exteroceptions with teacher. Ijspeert et al. [8] designed a nonlinear dynamical system to imitate trajectories of joints and end-effectors of a human teacher; the robot learned and performed tennis swing motions by imitation. The mimesis theory of [9, 4] is based on action acquisition and action

symbol generation but does not address dynamics compensation for real-time biped locomotion.

Traditional model-based approaches based on zero-moment point (ZMP) [20, 17] or the inverted pendulum model [11, 10] require a highly accurate model of robot dynamics and the environment in order to achieve a stable walking gait. Learning-based approaches such as RL are more flexible and can adapt to environmental change but such methods are typically not directly applicable to humanoid robots due to the "curse of dimensionality" problem engendered by the high dimensionality of the full-body joint space of the robot. Morimoto et al. [15] demonstrated that stepping and walking policies could be improved by using RL and kernel dimension reduction (KDR): stepping and walking controllers are provided, and the learning system improves the performance of these controllers. The framework proposed in this chapter does not assume a specific type of nonlinear dynamical system or a specific gait as in [15] but is designed for learning general human motion from demonstrations. It can be used for learning different gaits for different tasks without redesign the algorithm.

Gaussian Process Dynamical Model (GPDM) [21] is a dimensionality reduction method for modeling high-dimensional sequential data. A temporal sequence of human walking data was modeled and reproduced without prior information. The resulted walking gait was reproduced kinematically without involving interactions with the environment. In contrast, the motion learning framework proposed in this chapter learns a dynamic model of interaction between the robot and its environment by learning a causal relationship between low-dimensional posture commands and sensory feedback.

2 3-D Eigenposes

Nonlinear dimensionality reduction algorithms had previously been applied to representation of human posture [3, 7]. Tatani and Nakamura [19] explored using a low-dimensional subspace to kinematically reproduce human motion on a humanoid robot via non-linear principal components analysis (NLPCA) [13]. However, these methods have some parameters that have to be well-tuned. Properties of the resulting low-dimensional subspaces used in these algorithms have not been well studied. Principal components analysis (PCA) is a linear dimensionality reduction technique whose properties have been well studied. We utilize PCA for the motion learning framework in this chapter.

2.1 Eigenposes as Low-Dimensional Representation of Postures

Particular classes of motion such as walking, sidestepping, or reaching for an object are intrinsically low-dimensional. We apply linear PCA to parameterize the low-dimensional motion subspace $\mathbb{X}$. Vectors in the high-dimensional joint angle space are mapped to the low-dimensional space by multiplication with the transformation matrix $\mathbf{C}$. The rows of $\mathbf{C}$ consist of the eigenvectors, computed via

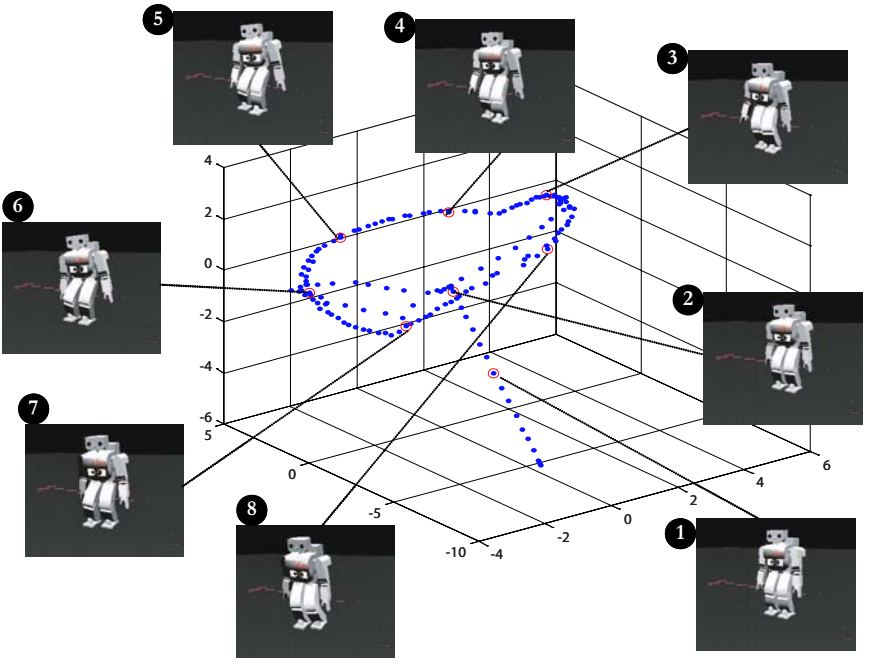


Fig. 2 Posture subspace and example poses from a hand coded walking gait. A three-dimensional space produced by PCA represents the posture of the Fujitsu HOAP2 robot. Blue points along a loop represent different robot postures during a single walk cycle. The first two labeled postures are intermediate postures between an initial stable standing pose and a point along the periodic gait loop represented by postures three through eight.

eigenvalue decomposition of the motion covariance matrix. Eigenvalue decomposition produces transformed vectors whose components are uncorrelated and ordered according to the magnitude of their variance. These transformed vectors shall be referred as *eigenposes*.

For example, let Θ be the 20×1 vector of joint angles (the high-dimensional space) and $\mathbf{x}$ be the 3×1 vector of eigenpose in a 3D subspace. We can calculate $\mathbf{x}$ by first calculating $\mathbf{p} = \mathbf{C}\Theta$, where $\mathbf{p}$ is a 20×1 vector of all principal component coefficients of Θ and $\mathbf{C}$ is the 20×20 PCA transformation matrix. We then pick the first three elements of $\mathbf{p}$ (corresponding to the first three principal components) to be $\mathbf{x}$. The inverse mapping $\widetilde{\Theta}$, which is an approximation to Θ, can be computed by $\widetilde{\Theta} = \mathbf{C}^{\mathbf{T}}\tilde{\mathbf{p}}$ when the first three components of a full-rank-vector $\tilde{\mathbf{p}}$ are the elements of $\mathbf{x}$ and the rest of the elements are zero.

Examples of the low dimensional representation of the joint angle space of a HOAP-2 robot executing a walking gait (in the absence of gravity) are shown in Fig. 2. The robot images in the figure were produced by inverse PCA mapping. The figure demonstrates that the temporal sequence of motion data is still preserved in the low-dimensional subspace representation of the motion.

2.2 Action Subspace Embedding

PCA reduces the redundancy of posture data in high dimensional joint space. We use the reduced dimensional subspace $\mathbb{X}$ for constraining the postures of a motion pattern.

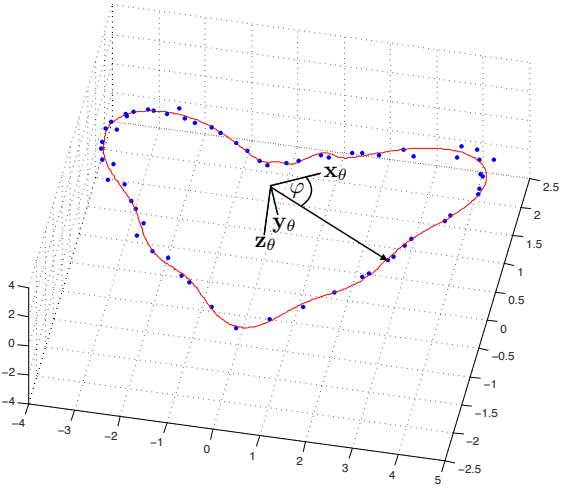


Fig. 3 Embedded action subspace of a humanoid walking gait. Training data points in the reduced posture space (shown in blue-dots) are converted to cylindrical coordinates relative to the coordinate frame $\mathbf{x}_\theta, \mathbf{y}_\theta, \mathbf{z}_\theta$. The points are then represented as a function of the phase angle φ, which forms an embedded action subspace (shown as a red solid-line curve).

A periodic movement such as walking can be represented by a closed-curve pattern $\mathbb{X}$. As an example, the periodic part of the data in Fig. 2 was manually segmented and is shown as the blue dots pattern in Fig. 3. In the general case, we consider a non-linear manifold representing the action space $\mathbf{A} \subseteq \mathbb{X}$. Non-linear parameterization of the action space allows further reduction in dimensionality. In particular, a one-dimensional representation of the original motion in the three-dimensional subspace is obtained by converting each point to its representation in a cylindrical coordinate frame. This is done by establishing a coordinate frame with three basis directions $\mathbf{x}_\theta, \mathbf{y}_\theta, \mathbf{z}_\theta$. The zero point of the coordinate frame is the empirical mean of the data points in the reduced space. The data are re-centered around this new zero point and the resulting data is labeled $\hat{\mathbf{x}}^i$.

Then, the principal axis of rotation $\mathbf{z}_\theta$ is computed as:

$$\mathbf{z}_\theta = \frac{\Sigma_i(\hat{\mathbf{x}}^i \times \hat{\mathbf{x}}^{i+1})}{\|\Sigma_i(\hat{\mathbf{x}}^i \times \hat{\mathbf{x}}^{i+1})\|} \tag{1}$$

Next, $\mathbf{x}_\theta$ is chosen to align with the maximal variance of $\mathbf{x}^i$ in a plane orthogonal to $\mathbf{z}_\theta$. Finally, $\mathbf{y}_\theta$ is specified as orthogonal to $\mathbf{x}_\theta$ and $\mathbf{z}_\theta$. The final embedded training

data is obtained by cylindrical conversion to (φ, r, h) where r is the radial distance, h is the height above the $\mathbf{x}_\theta - \mathbf{y}_\theta$, and φ is the angle in $\mathbf{x}_\theta - \mathbf{y}_\theta$ plane measured counter-clockwise from $\mathbf{x}_\theta$.

Given the loop topology of the latent training points, one can parameterize r and h as a function of φ. The embedded action space is represented by a learned approximation of the function:

$$[r, h] = g(\varphi) \tag{2}$$

where $0 \leq \varphi \leq 2\pi$. This function is approximated using a radial basis function (RBF) network. Note that the angle φ can be interpreted as the motion phase angle because it indicates how far the current posture is from the beginning of the motion cycle, which in our case is a walking gait. The first-order time derivative of φ tells us the speed of movement.

2.3 Action Subspace Scaling

The high-dimensional joint angle data are normalized before PCA. The data for each joint dimension are originally in different scales of values, but after normalization, they are scaled to the same range. When the normalized data are multiplied by a scalar value, the results are similar postures but with a different magnitude, allowing posture scaling. Note that posture scaling yields reasonable results only when the motion data set contains one specific type of motion.

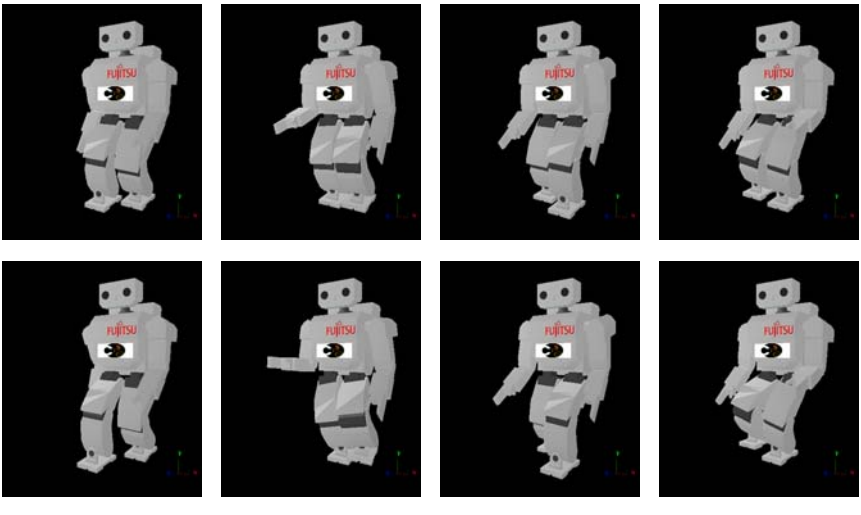


Fig. 4 Motion scaling of a walking gait. The first row shows four different postures of a walking gait. The second row shows coherent postures of the first row when a multiplying factor $f = 2.0$ is applied to the low-dimensional representation of this walking gait.

Scaling up and down motion patterns in the low-dimensional subspace produces similar motion patterns but with differences in the magnitude of motion. This means posture scaling ability is preserved after PCA is applied. If $\mathbf{A}$ represents a walking gait, multiplying $\mathbf{A}$ by a factor $f > 1$ will result in a similar walking gait but with larger steps. Multiplying $\mathbf{A}$ by a factor $f < 1$ results in a walking gait with a smaller step size. Fig. 4 shows an example of posture scaling of a walking motion.

It should be noted that action subspace scaling only produces similarity in kinematic postures. The result of scaling may not be dynamically stable, especially when the scaling factor $f > 1$. To achieve stable motion, the new motion has to be gradually learned as will be described in Section 5.1. We use action subspace scaling for $f < 1$ to initialize the learning process.

3 Learning to Predict Sensory Consequences of Actions

A key component of the proposed framework is learning to predict future sensory inputs based on current actions. This learned predictive model is used for optimal action selection. The goal is to predict, at time step t, the future sensory state of the robot, denoted by s_{t+1}. In general, the state space $\mathbf{S} = \Theta \times \mathbf{P}$ is the Cartesian product of the high-dimensional joint space Θ and the space of other percepts $\mathbf{P}$. Other percepts include, for example, measurements from the torso accelerometer or gyroscope, foot pressure sensors, and information from camera images. The goal is to learn a function $F : \mathbf{S} \times \mathbf{A} \mapsto \mathbf{S}$ that maps the current state and action to the next state. In this framework, F is assumed to be deterministic.

Often the perceptual state s_t by itself is not sufficient for predicting future states. In such cases, one may learn a higher order mapping based on a history of perceptual states and actions, as given by an n-th order Markovian function:

$$s_{t+1} = F(s_t, s_{t-1}, ..., s_{t-n-1}, a_t, a_{t-1}, ..., a_{t-n-1}) \tag{3}$$

For this chapter, unless stated otherwise, we use a second-order $(n = 2)$ time-delay radial basis function (RBF) network for learning the predictive model. The state

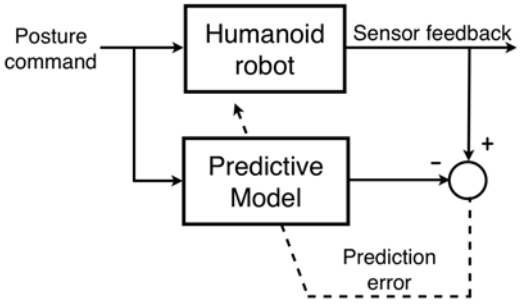


Fig. 5 Sensory signals (e.g., gyroscope signals) at the next time step are predicted based on the current posture command (eigenpose) as well as sensory signals and posture commands from previous time steps. The predictor is learned by comparing the predicted sensory signals to the actual sensor readings from the robot.

vector is taken to be the three-dimensional gyroscope signal ($s_t \equiv \omega_t$). As discussed in the previous section, an action is represented by a phase angle, radius, and height in latent posture space ($a_t \equiv \chi_t \in \mathbb{X}$). A schematic diagram of the learning method is shown in Fig. 5.

4 Predictive Model Motion Optimization

The algorithm presented in this section combines optimization and sensory prediction (see previous section) to select an optimal action plan for a humanoid robot. Fig. 6 illustrates this optimization process.

One may express the desired sensory states that the robot should attain during a particular class of actions using an objective function $\Gamma(\mathbf{s})$. The algorithm then selects actions $a_t^*, \ldots, a_T^*$ such that the predicted future states $s_t, \ldots, s_T$ will be optimal with respect to $\Gamma(\mathbf{s})$:

$$a_t^* = \arg\min_{a_t} \Gamma(F(s_t, \ldots, s_{t-n-1}, a_t, \ldots, a_{t-n-1})). \tag{4}$$

In our work, the objective function Γ measures torso stability as defined by the following function of gyroscope signals:

$$\Gamma(\omega) = \lambda_x \omega_x^2 + \lambda_y \omega_y^2 + \lambda_z \omega_z^2, \tag{5}$$

where $\omega_x, \omega_y, \omega_z$ refer to gyroscope signals in the $\mathbf{x}, \mathbf{y}, \mathbf{z}$ axes respectively. The constants $\lambda_x, \lambda_y, \lambda_z$ allow one to weight rotation in each axis differently. Assuming that

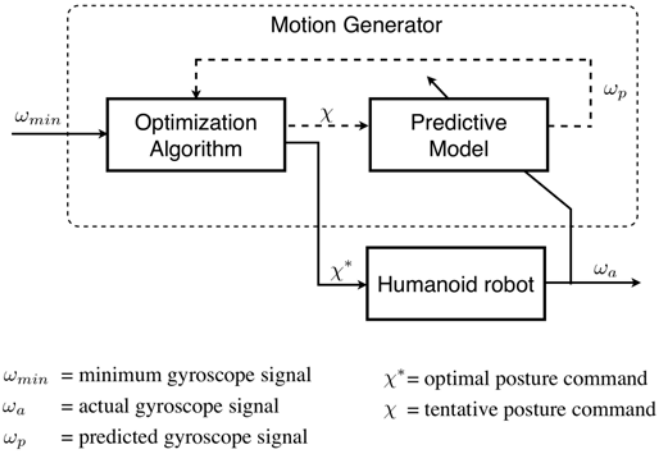


Fig. 6 Model predictive motion generator for optimizing motion stability. At time t, the optimization algorithm generates tentative actions or posture commands ($a_t \equiv \chi \in \mathbb{X}$). The predictive model predicts values of subsequent gyroscope signals ω_p. The optimization algorithm then selects the optimal posture command χ^* based on ω_p. The optimal posture command χ^* is executed by a robot/simulator. The resulting gyroscope signals are recorded for retraining the predictive model.

the starting posture is statically stable, one may simply minimize overall rotation of the robot body during motion in order to maintain balance by minimizing the sum of squares of the gyroscope signals. The objective function (5) thus provides a measure of stability of the posture during motion.

For the second-order predictive function F, the optimization problem becomes one of searching for the optimal stable action at time t given by:

$$\chi_t^* = \arg\min_{\chi_t \in S} \Gamma(F(\omega_t, \omega_{t-1}, \chi_t, \chi_{t-1})) \tag{6}$$

$$S = \left\{ \left[\varphi_s \ r_s \ h_s \right]^T | \varphi, r, h \text{ defined in Section 2.2} \right\} \tag{7}$$

For efficient optimization, the search space is restricted to a local region in the action subspace:

$$\varphi_{t-1} < \varphi_s \leq \varphi_{t-1} + \varepsilon_\varphi \tag{8}$$

$$r_a - \varepsilon_r \leq r_s \leq r_a + \varepsilon_r \tag{9}$$

$$h_a - \varepsilon_h \leq h_s \leq h_a + \varepsilon_h \tag{10}$$

$$0 < \varepsilon_\varphi < 2\pi \tag{11}$$

$$[r_a, h_a] = g(\varphi_s) \tag{12}$$

In the above, the phase-motion-command search-range φ_s begins after the position of the phase motion command φ_{t-1} at the previous time step, the range for the radius search r_s begins from a point in the action subspace embedding $\mathbf{A}$ defined by (12) in both positive and negative directions from r_a along $\mathbf{r}$ for the distance $\varepsilon_r \geq 0$, and the search range for h_s is defined in the same manner as r_s according to h_a and ε_h. In the experiments, the parameters $\varepsilon_\varphi, \varepsilon_r$,and ε_h were chosen to ensure efficiency while at the same time allowing a reasonable range for searching for stable postures.

Note that the search process exploits the availability of a human demonstrator by using the demonstrated action, as captured by (12), to constrain the search for stable robot actions. This imitation-based approach contrasts with more traditional approaches based on trial-and-error reinforcement learning or complex physics-based models.

Additionally, since selected actions will only truly be optimal if the sensory predictor is accurate, the prediction model is periodically re-trained based on the posture commands generated by the optimization algorithm and the sensory feedback obtained from executing these commands.

The entire motion optimization and action selection process can be summarized as follows:

1. Use PCA to obtain eigenpose data from the human-demonstrated joint angle data.
2. Apply action subspace embedding for parameterization of the periodic motion pattern.

3. Start the learning process by inverse mapping the eigenpose actions back to the original joint space and executing the corresponding sequence of servo motor commands in a simulator or a real robot.
4. Use the sensory measurements and motor commands from the previous step to update the sensory-motor predictor as described in Section 3. In the present work, the state vector comprised of three channels of the gyroscope signal and the action variables were φ, r, and h in the low-dimensional subspace.
5. Use the learned model to estimate a sequence of actions according to the model predictive controller scheme described above (Fig.6).
6. Execute the computed actions and record sensory (gyroscope) feedback.
7. Repeat steps 4 through 6 until satisfactory motion is obtained.

4.1 One-Dimensional Motion Optimization

Our first experiment involved simulation in the Webots dynamic environment [14]. The goal was to increase the stability of a hand-coded walking gait (shown in Fig. 2) by using the motion optimization technique. The experiment also demonstrates the utility of action subspace embedding and the physical meaning of the parameter φ. Since this experiment involves one-dimensional optimization along φ, the parameters ε_r and ε_h in (9) and (10) are set to zero. Thus, (6) becomes:

$$\varphi_t^* = \arg\min_{\varphi_t} \Gamma(F(\omega_t, \omega_{t-1}, \varphi_t, \varphi_{t-1})). \tag{13}$$

We refer to this process as *motion-phase optimization* because only the parameter φ is optimized – the values of r and h are implicitly optimized through (2). The three channels of the gyroscope signal are regarded as *state* and the motion phase φ is regarded as the *action*. This *state-action* data is used for training the

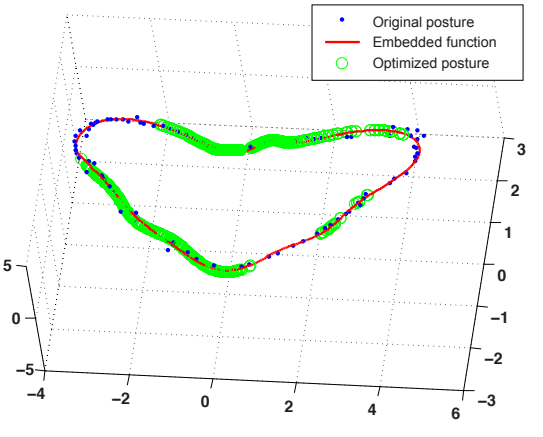


Fig. 7 Motion-phase optimization.

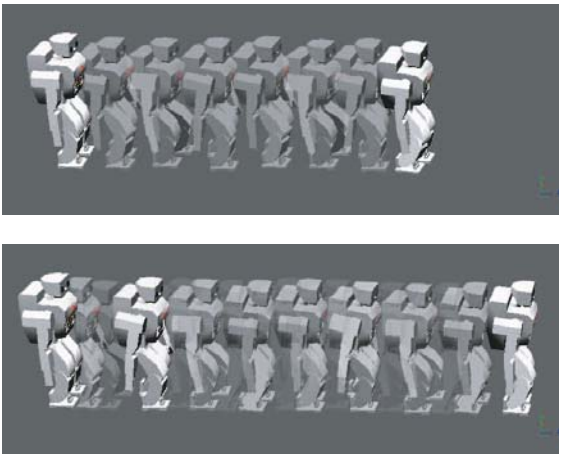


Fig. 8 Comparison of initial and optimized walking gait. The composite image at the top depicts the robot performing the initial walking gait for 2 seconds. Motion-phase optimization results in a faster walking gait as shown in the bottom image.

time-delay RBF network (depicted in Fig.5) to predict the gyroscope signals at the next time step. The optimization algorithm uses the learned predictor to obtain a new optimized action plan, which is then executed in the simulator.

The optimization result after three iterations of learning is shown in Fig.7. As seen in the figure, motion-phase optimization is essentially a line-search and the result of the optimization remains on the constraint pattern. Thus, no new posture is derived from this optimization. However, the phase of the motion is altered in such a way that the selected actions minimize gyroscope signal oscillation. Fig. 8 shows that the optimized walking gait is significantly faster than the original gait. The walking speed of the optimized walking gait could be increased to three times the original gait in further optimization. Thus, we conclude that φ is controlling the timing of motion.

4.2 Three-Dimensional Motion Optimization

The second experiment focused on three-dimensional optimization of an initial walking gait based on equations (6)-(12). Since the optimization process is performed in the three-dimensional space of ϕ, $\mathbf{r}$ and $\mathbf{h}$ in cylindrical coordinates, novel postures resulting from optimized actions that do not lie on the constraint pattern can be expected.

The optimized walking gait in the low dimensional subspace shown in Fig. 9 was obtained after three iterations of sensory-motor prediction learning. An improved dynamically balanced walking gait was achieved. The new trajectory has a shape similar to the initial one but has a larger magnitude and is shifted. After remapping this trajectory back to the high dimensional space, the optimized motion pattern was

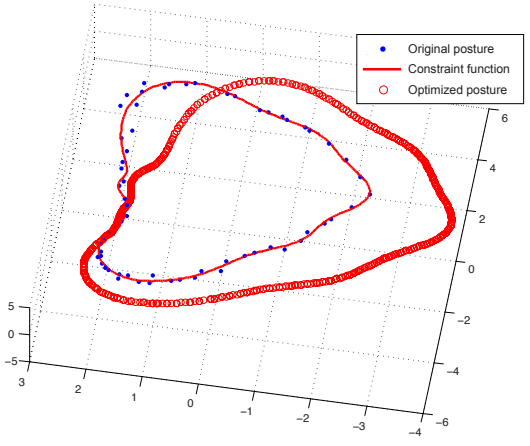


Fig. 9 Three-dimensional optimization results for a walking motion pattern based on an action subspace embedding in a low-dimensional subspace.

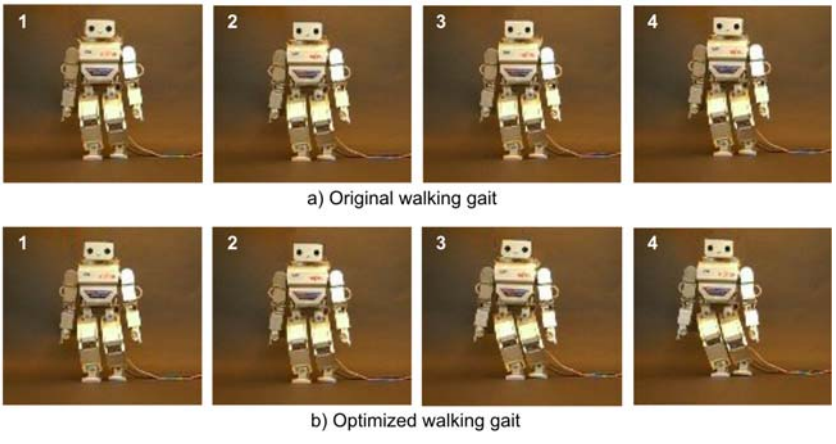


Fig. 10 Walking gait on a Fujitsu HOAP2 robot after three-dimensional optimization.

tested with the simulator and the real robot. The gyroscope readings from the new walking pattern are shown in Fig.11. The RMS values for the optimized walking gait along the **x**, **y** and **z** axes are 0.0521, 0.0501 and 0.0533 respectively, while the values for the original walking gait were 0.3236, 04509 and 0.3795. The RMS values for the optimized gait are thus significantly less than the original walking gait, indicating significant improvement in the dynamic stability of the robot. The robot walks with a larger step size but slower walking speed than the original walking gait.

The original and optimized gaits are shown in Fig. 10. The optimized walking gait has a different balance strategy compared to the original walking gait. In the

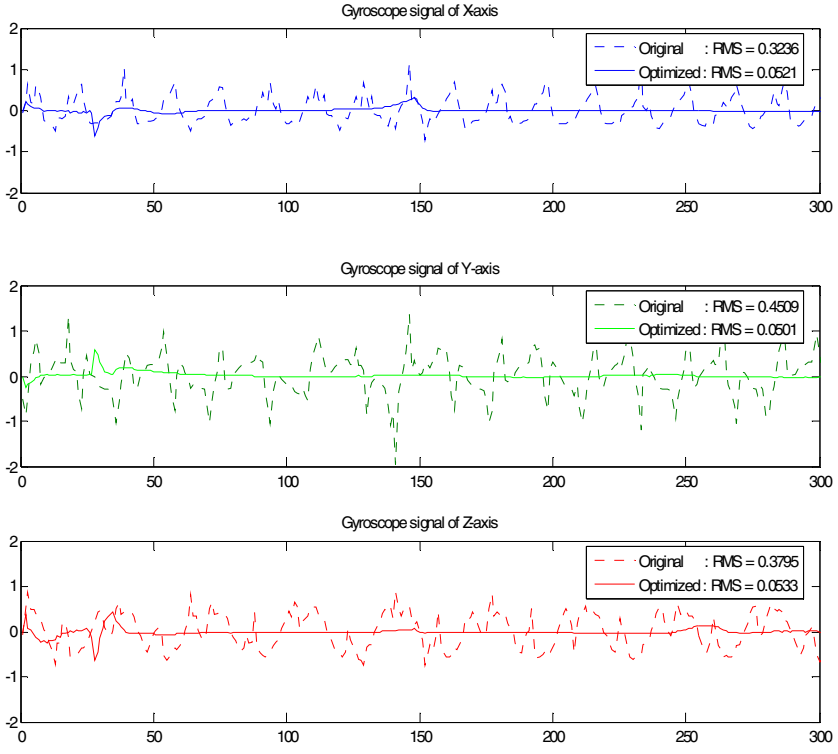


Fig. 11 Comparison of gyroscope signals before and after optimization. The plots from top to bottom show the gyroscope signals for the axes **x**, **y** and **z** recorded during the original and optimized walking motion. Notice that the root mean squared (RMS) values are significantly reduced for the optimized motion.

original gait, the robot quickly swings the whole body on the side of the support leg while it moves the other leg forward. For the optimized gait, the robot leans on the side of the support leg, bends the torso back in the opposite direction while it moves the other leg forward slowly. With the optimized gait, the robot also keeps its torso vertically straight throughout the motion. Fig.11 confirms that the algorithm was able to optimize the motion in such a way that the gyroscopic signals for the optimized motion are almost flat.

5 Learning Human Motion through Imitation

To be able to imitate a human motion, one must first solve the correspondence problem, which in our case is the problem of kinematic mapping of whole-body postures between a human demonstrator and a humanoid robot (we use a Fujitsu HOAP-2 robot). The human subject and the robot share similar humanoid appearances, but

their kinematic structure (skeletons) are dissimilar. The correspondence problem is solved by searching for a set of joint angles for the robot that generates the best matching pose with respect to the human demonstration. A Vicon optical system running at 120Hz and a set of 41 reflective markers was used for recording human motion. Initially, the markers are attached to the human subject and the 3-D positions of the markers are recorded for each pose during motion. The recorded marker positions provide a set of Cartesian points in the 3D capture volume for each pose. To obtain the robot's poses, the marker positions are used as positional constraints on the robot's skeleton and a set of joint angles is obtained using the standard numerical inverse kinematics (IK) solver in the Vicon motion capture system.

As depicted in Fig. 12, in order to generate robot joint angles, the human subject's skeleton is simply replaced by a robot skeleton of the same dimension. For example, the shoulders were replaced with three distinct 1-dimensional rotating joints rather than a single 3-dimensional human ball joint. The IK routine then directly generates the desired joint angles on the robot skeleton for each pose. One limitation of this technique is that there may be particular kinds of motion for which the robot's joints cannot approximate the human pose. This implies that the human demonstrator should try to avoid certain types of motion that the robot cannot imitate. For example, using toes in a walking gait should be avoided. In the case of arm movement, since the learner robot is a HOAP-2 robot having only four degrees of freedom (DoFs) in each arm, demonstration of actions that require six DoFs should be avoided. For the present work, since we only considered human motion that the robot has the potential to achieve, the above method proved to be a very efficient way of generating large sets of human motion data for robotic imitation.

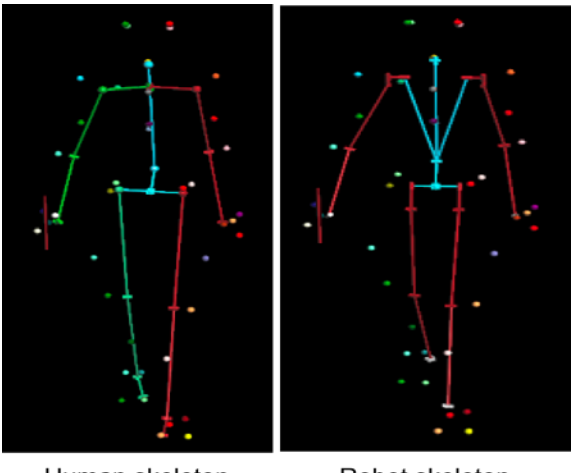


Fig. 12 Human skeleton (left) and robot skeleton (right) for kinematic mapping.

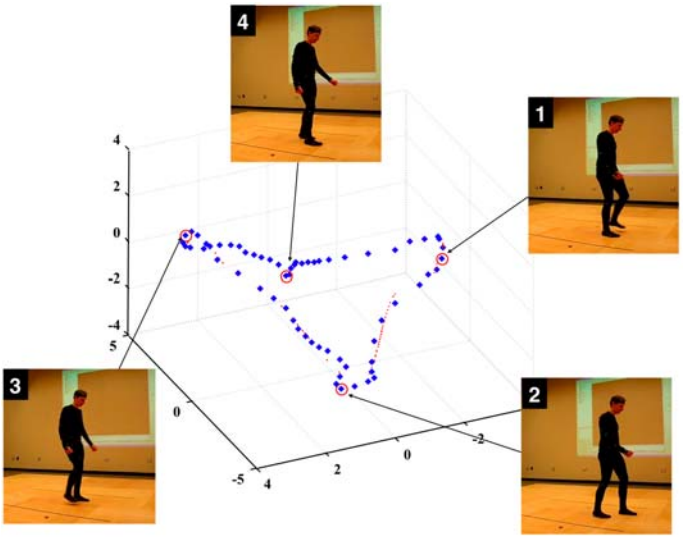


Fig. 13 Posture subspace and example poses obtained from human motion capture. Linear PCA was applied to joint angle data from a human kinematic configuration obtained via motion capture as described in Section 5. Blue diamonds represent different human postures during a single walking cycle. Red circles mark various example poses as shown in the numbered images.

5.1 Optimization of Motion Capture Data

Our first experiment with human motion capture data focused on making robot learn how to walk from a human demonstration of walking. An example of the low-dimensional representation of joint angle data from human walking are shown in Fig. 13. Note that the data pattern from human mocap data in Fig. 13 is more irregular than the data from the hand-coded walking gait in Fig. 2. Optimization based on human mocap data is difficult because the initial gait is unstable. We therefore used a motion scaling strategy in a low-dimensional subspace as described in Section 2.3. When the initial walking pattern in the low-dimensional subspace is scaled down, it produces smaller movements of the humanoid robot, resulting in smaller changes in dynamics during motion. The initial walking pattern is scaled down until a dynamically stable motion is found, after which the learning process is started. The motion optimization method in Section 4 is applied to the scaled-down pattern until its dynamic performance reaches an optimal level. The trajectory of the optimized result is gradually scaled up toward the target motion pattern. In this experiment, a scaling factor of 0.3 applied to the original motion pattern was found to be stable enough to start the learning process. The final optimization result is shown as a trajectory of red circles in Fig. 14. It corresponds to about 80% of the full scale motion from mocap data.

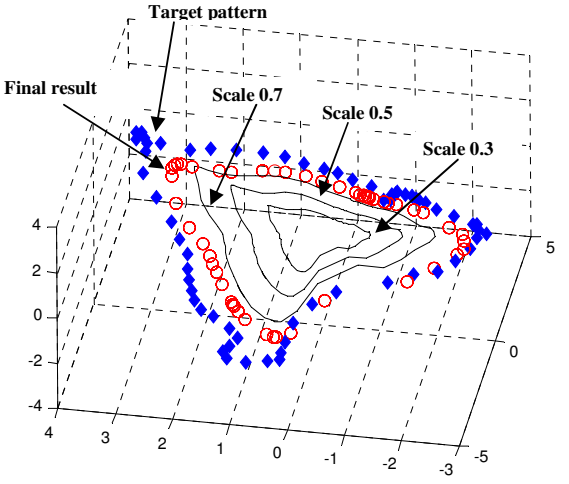


Fig. 14 Motion pattern scaling and optimization of human motion capture data. The target motion pattern is scaled down until it can produce a stable motion which is used to start the motion optimization process.

For the results in Fig. 14, five learning iterations with scale values 0.3, 0.5 and 0.7 were performed, and for the final result, ten iterations were performed for the scale 0.8. Note that the optimization time depends on the parameters $\varepsilon_\varphi, \varepsilon_r$ and ε_h. The parameter ε_φ must be defined such that value of φ_s is greater than the maximum difference in motion-phase-angle in the original mocap data. This will ensure that the optimization algorithm can search for a pose in a range that the original movement achieved. The longer the range of φ_s, the better the exploration but greater the optimization time. For r and h, the same parameter setup as φ can be applied. The value of ε_r and ε_h were set to 0.5 for all of the optimizations. The objective function in (5) has three tuning parameters, which are λ_x, λ_y and λ_z. At the beginning, we set the values of these parameters to 1. From observation of the first learning iteration, the parameters may be tuned, after which the values are maintained for the rest of learning iterations. In this chapter, λ_x and λ_z were set to 1.0. λ_y, which corresponds to the vertical direction, was set to 2.0 to allow the algorithm to compensate for the unexpected turns seen during the first learning iteration.

The simulation and experimental results are shown in Fig. 15. The learning process is performed in the simulator [14] and the resulting motion is tested on the real robot to minimize damage to the robot during the learning process. We observed that as expected, the walking gait on the real robot is not as stable as the results in the simulator because of differences in the frictional forces modeled in the simulator and the actual forces on the floor. We believe that performing further learning directly on the real robot (where permissible) could rectify this problem and improve performance.

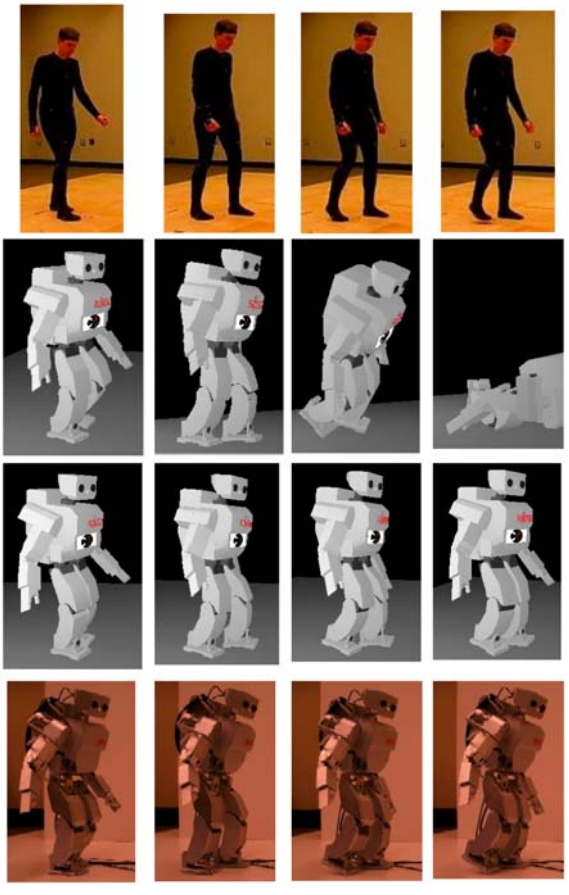


Fig. 15 Learning to walk through imitation. The first row shows a human subject demonstrating a walking gait in a motion capture system. The second row shows simulation results for this motion before optimization. The third row shows simulation results after optimization. The last row shows results obtained on the real robot.

Note that the learned motion is indeed dynamic and not quasi-static motion because there are only two postures in the walking gait that can be considered statically stable, namely, the two postures in the walking cycle where the two feet of the robot contact the ground. The rest of the postures in the walking gait do not need to be statically stable to maintain balance.

6 Lossless Motion Optimization

In the previous sections, the eigenposes that have been used for motion learning were three-dimensional. 3-D data are convenient for visualization and for

developing motion optimization algorithms based on analytical geometry. Periodic motion patterns such as hand coded walking and human mocap walking can be successfully learned using 3-D eigenposes but for some motion patterns, using only three dimensions cannot preserve the significant characteristics of the original motion. In this section, we extend our technique to larger-dimensional eigenposes. In particular, we show how the phase-motion optimization concept in Section 4.1 can be implemented with a new cylindrical coordinate transformation technique for large dimensional subspaces. We demonstrate how the algorithm can be used in a HOAP-2 humanoid robot to learn a sidestepping motion from a human demonstrator using a motion capture system.

6.1 Human Motion Capture of Sidestepping Motion

A motion capture session of a human demonstrator performing a sidestepping motion (to the right) as shown in Fig.16 was used as the target motion to be imitated. The motion sequence can be divided into four major steps starting from a standing posture. First, the right leg is lifted off the ground. Second, the right leg lands on the ground. Third, the left leg lifts off the ground. Fourth, the left leg swings in toward the right leg. For purposes of later discussion, we define the sidestep motion as being comprised of four phases: *right-lift, right-landing, left-lift*, and *left-landing*.

The kinematic mapping process described in Section 5 resulted in 20 dimensions of joint angle data, which were transformed into orthogonal principal axes using PCA as described in Section 2.1.

Fig.17 plots the accuracy of data reconstruction as a function of the number of principal components of the mocap data. When only the first three principal components are used, less than 80% of accuracy is achieved in reconstructing the original joint angle data. Accuracy increases gradually until 100% accuracy is obtained

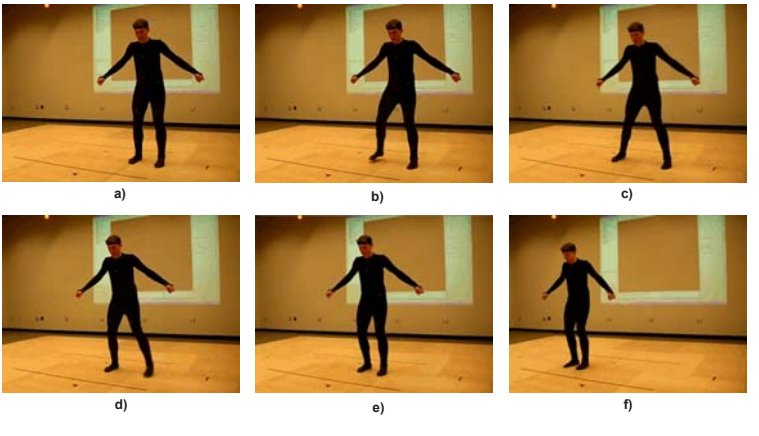


Fig. 16 Motion capture of sidestepping motion. Six samples of a rightward sidestepping motion sequence are shown in a) through f). Note that f) is a standing posture after one cycle of sidestepping. Each sidestepping cycle takes about 1 second.

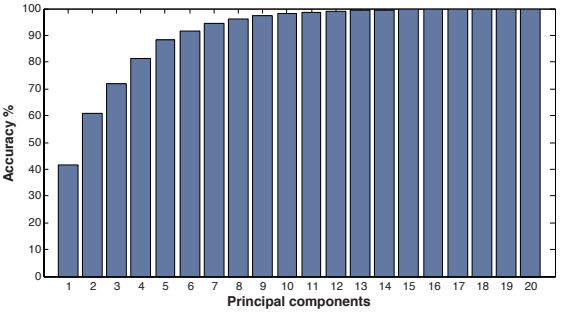


Fig. 17 Reconstruction accuracy as a function of the number of principal components of the sidestepping motion data from Figure 16.

when all 20 dimensions of eigenpose data are used. In this section, we illustrate our technique using 20-dimensional eigenposes for learning.

6.2 Large-Dimensional Cylindrical Coordinate System Transformation

The motion-phase optimization described above was performed in a cylindrical coordinate system. Transformation of data from a 3-D Cartesian coordinate system to a 3-D cylindrical coordinate is straightforward. However, that is not the case for transformation of data that have more than three dimensions. In this section, we suggest an extension of the cylindrical coordinate transformation idea to higher dimensions.

When $n = 3$, transformation from a Cartesian space $\mathbb{X}$ to a cylindrical coordinate system Φ is given by the mapping:

$$f(x,y,z) \rightarrow f(\varphi,r,h) \tag{14}$$

where

$$\varphi = \arctan(\frac{y}{x}), \tag{15}$$

$$r = \sqrt{x^2 + y^2}, \tag{16}$$

and

$$h = z. \tag{17}$$

When $n > 3$, an n-dimensional function may be written as:

$$f(d_1, d_2, d_3, \ldots, d_n) \tag{18}$$

where the d_i, $i = 1, \ldots, n$ $(n > 3)$, represent variables along orthogonal axes in $\mathbb{R}^n$.

We can express the function in (18) as:

$$f(x, y, z_1, \ldots, z_{n-2}) \tag{19}$$

The key idea is to represent the function f using a set of multiple cylindrical coordinate frames. Suppose, for example, that f is a 5-dimensional function. Then, f can be expressed in the form of (19) as:

$$f(x, y, z_1, z_2, z_3). \tag{20}$$

We use a piecewise mapping of f to cylindrical coordinates as follows:

$$\begin{array}{ll} f(x,y,z_1) & f(\varphi, r, h_1) \\ f(x,y,z_2) \Rightarrow & f(\varphi, r, h_2) \\ f(x,y,z_3) & f(\varphi, r, h_3) \end{array} \tag{21}$$

where φ and r are defined by Equations (15) and (16). Similarly, h_1, h_2 and h_3 are defined as in Equation (17).

Thus, the n-orthogonal dimensions of f are mapped to multiple cylindrical coordinate systems as:

$$f(x, y, z_1, \ldots, z_{n-2}) \rightarrow f(\varphi, r, h_1, \ldots, h_{n-2}). \tag{22}$$

For the 20-dimensional eigenpose data for sidestepping, 18 cylindrical coordinate frames are used by the above method.

6.3 Motion-Phase Optimization of Hyperdimensional Eigenposes

Trying to perform optimization on all of the orthogonal components of high-dimensional eigenposes may be intractactable, due to the curse of dimensionality problem. We therefore extend the one-dimemsional motion optimization idea from Section 4.1 to the higher dimensional case. For 3-D data, the action subspace embedding (described in section 2.2) is a single parameter function of motion-phase angle φ that produces values for the radius r and the height h of a periodic motion pattern in a cylindrical coordinate system. For n-dimensional eigenpose data, the action subspace embedding is a single parameter function of motion-phase angle φ that produces values for $r, h_1, h_2, \ldots, h_{n-2}$ of a periodic motion pattern. In particular, for the sidestepping motion pattern, the action subspace embedding is given by:

$$[r, h_1, h_2, \ldots, h_{18}] = g(\varphi). \tag{23}$$

The motion-phase optimization procedure in (13) can be directly applied to (23).

Fig. 18 shows the result of optimization (after five learning episodes) in the 3-D coordinate frame defined by the first three principal axes. From the figure, it can be seen that the optimized eigenposes are points on the original motion pattern, but distributed differently from the original pattern. This is because the motion-phase

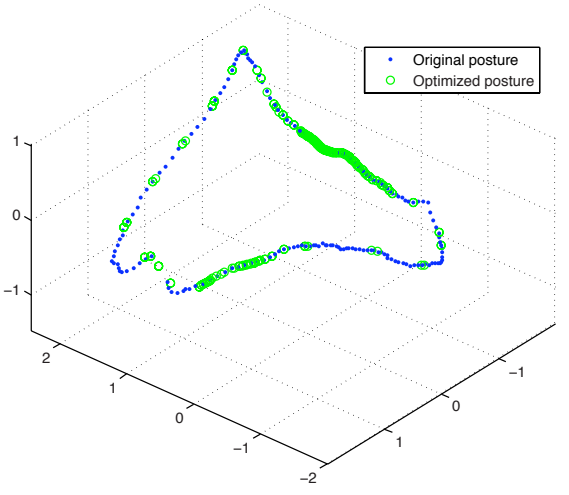


Fig. 18 Result of motion-phase angle optimization for sidestepping eigenpose data.

optimization is a one-dimensional optimization of the parameter φ in (23). The optimized eigenposes are thus strictly constrained to be within the original set of postures. The differences in distribution between the original pattern and the optimized pattern means that timing of postures during the motion have been altered. Notice in the figure that some gaps in the original pattern have been closed by the optimized postures. There are two reasons for this phenomena. First, the action subspace embedding is modeled as a closed-curve. Second, based on sensory feedback during learning episodes, the optimization algorithm found that it can achieve lower gyroscopic signal oscillation by choosing postures in the gap in the motion trajectory. As a result, the movement is smoother. Another attempt by the algorithm to obtain smoother movement can be noticed in the lower-left corner of the trajectory: the algorithm decided to plan the trajectory across the irregular corner of the original trajectory.

Fig. 19 show the simulation results for sidestepping motion. Column a) shows the original sidestepping motion of a human demonstrator. Column b) shows HOAP-2 robot performing the sidestep motion sequence at motion scale 0.5 without optimization in a dynamics simulator. Column c) shows the sidestep motion after five learning episodes at motion scale 0.5. The first and the last rows are the standing postures at the beginning and end of the motion sequence, respectively. The second row is the right-lift phase. The third row is the right-landing phase. The fourth row is the left-lift phase, and the fifth row is the left-landing phase. In column b), the right foot of the robot was bouncing at the right-landing phase, causing the robot to be unable to lift its left foot up in the subsequent lift phase. As a result, the robot dragged its left foot along the ground during the left-landing period. This made the whole body of the robot turn, as can be observed in the last two rows of column b).

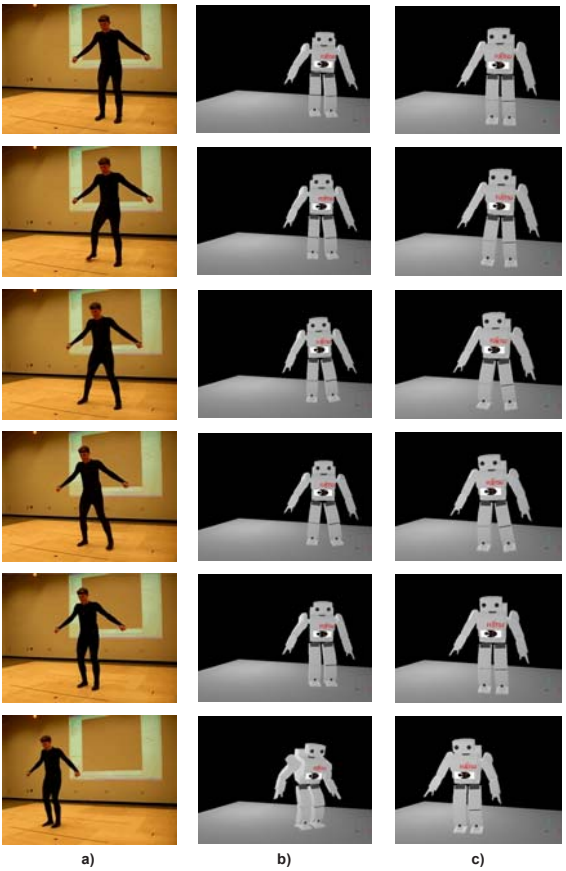


Fig. 19 Simulation results for sidestepping. Column a) shows original sidestepping motion sequence by the human demonstrator. Column b) shows the sidestep motion sequence on a HOAP-2 robot in a dynamics simulator without optimization at the motion scale 0.5. Column c) shows the sidestep motion after five learning episodes at motion scale 0.5.

In column c), the robot could perform the sidestep motion without the undesirable turn of the body.

While all of the key postures in column c) look very similar to the human postures in column a), timing of the movements are significantly different. The right-landing and left-landing phases of the optimized motion in column c) are relatively slower than the original human motion. These can also be observed in Fig. 18: there are two regions of the motion pattern with high density of optimized postures. These corespond to the slow landing phases. The slow landing phases also prevented the robot from dragging its left foot on the ground. As a result, the undesirable turn of the body was avoided and the sidestep motion was successfully learned.

7 Conclusion

This chapter proposed a framework that allows a humanoid robot to learn bipedal locomotion by imitation of human motion. Whole-body postures are represented using eigenposes computed from PCA. These eigenposes are used for learning a predictive sensory-motor model that allows a humanoid robot learn to walk by imitation of a human gait. Taken together, our results demonstrate that the physics of a complex dynamical system can be learned and manipulated in a low-dimensional subspace. Using a low-dimensional subspace greatly reduces computational complexity and facilitates the learning process. Since all of the joints are always constrained to encode postures near the ones to be imitated, the low-dimensional subspace reduces the occurrence of unmeaningful or potentially harmful actions (such as self-intersection) in the learning process.

The action subspace embedding in cylindrical coordinates not only further reduces dimensionality and complexity, but also provides meaningful variables in the low-dimensional subspace such as the *motion-phase-angle* φ and radius r. Optimization of the motion-phase-angle was shown to be equivalent to optimizing posture timing during the motion, while the radius r reflects magnitude of the motion, as determined by the first two principal components of the motion pattern. The physical meaning of the parameter h is yet to be clearly interpreted.

The human imitation-based learning framework described in this chapter demonstrates how a humanoid robot can learn basic human actions such as walking. These basic actions could be used a building blocks for learning more complex behaviors using approaches such as reinforcement learning. To learn actions other than walking and sidestepping, the objective function in (5) could be modified to accommodate different sensory variables. The robustness of the learned models to noise could be improved using a probabilistic approach as described in [6].

The proposed framework functions as an off-line motion planner rather than an on-line feedback controller. Thus, it cannot be applied directly to the problem of navigation on uneven terrain. One way of adding robustness to an off-line motion planner is to use a motion stabilizer [12], which is a combination of simple force/torque and gyroscope-based feedback controllers. We also investigating the possibility of a real-time feedback controller based on learning an inverse model of the predictor (3). Also under investigation are methods for learning non-periodic human motion as well as motion parameterization using eigenposes.

Acknowledgements. This work was supported by National Science Foundation (NSF) grant 0622252, the Office of Naval Research (ONR) Cognitive Science program, and a Packard Fellowship to RPNR.

References

1. Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, Taipei, Taiwan. IEEE, Los Alamitos (2003)
2. Billard, A.: Imitation: a means to enhance learning of a synthetic protolanguage in autonomous robots, pp. 281–310 (2002)

3. Chalodhorn, R., MacDorman, K.F., Asada, M.: An algorithm that recognizes and reproduces distinct types of humanoid motion based on periodically-constrained nonlinear pca. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 370–380. Springer, Heidelberg (2005)
4. Dana Kulic Dongheui Lee, C.O., Nakamura, Y.: Incremental learning of full body motion primitives for humanoid robots. In: IEEE International Conference on Humanoid Robots, pp. 326–332 (2008)
5. Demiris, J., Hayes, G.: A robot controller using learning by imitation. In: Proceedings of the 2nd International Symposium on Intelligent Robotic Systems, Grenoble, France (1994)
6. Grimes, D.B., Chalodhorn, R., Rao, R.P.N.: Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In: Sukhatme, G.S., Schaal, S., Burgard, W., Fox, D. (eds.) Robotics: Science and Systems. The MIT Press, Cambridge (2006)
7. Grochow, K., Martin, S.L., Hertzmann, A., Popovic, Z.: Style-based inverse kinematics. ACM Trans. Graph. 23(3), 522–531 (2004)
8. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Trajectory formation for imitation with nonlinear dynamical systems. In: Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 752–757 (2001)
9. Inamura, T., Toshima, I., Nakamura, Y.: Acquisition and embodiment of motion elements in closed mimesis loop. In: ICRA, pp. 1539–1544. IEEE, Los Alamitos (2002)
10. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: ICRA [1], pp. 1620–1626
11. Kajita, S., Matsumoto, O., Saigo, M.: Real-time 3d walking pattern generation for a biped robot with telescopic legs. In: ICRA, pp. 2299–2306. IEEE, Los Alamitos (2001)
12. Kajita, S., Nagasaki, T., Kaneko, K., Yokoi, K., Tanie, K.: A running controller of humanoid biped hrp-2lr. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, Barcelona, Spain, April 18-22, pp. 616–622 (2005)
13. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. Journal of the American Institute of Chemical Engineers 37(2), 233–243 (1991)
14. Michel, O.: Webots: Symbiosis between virtual and real mobile robots. In: Heudin, J.-C. (ed.) VW 1998. LNCS (LNAI), vol. 1434, pp. 254–263. Springer, Heidelberg (1998)
15. Morimoto, J., Hyon, S.H., Atkeson, C.G., Cheng, G.: Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction. In: 2008 IEEE International Conference on Robotics and Automation, ICRA 2008, Pasadena, California, USA, May 19-23, pp. 2711–2716 (2008)
16. Rao, R.P.N., Shon, A.P., Meltzoff, A.N.: A Bayesian model of imitation in infants and robots. In: Nehaniv, C.L., Dautenhahn, K. (eds.) Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions. Cambridge University Press, UK (2007)
17. Sobotka, M., Wollherr, D., Buss, M.: A jacobian method for online modification of precalculated gait trajectories. In: Proceedings of the 6th International Conference on Climbing and Walking Robots, Catania, Italy, pp. 435–442 (2003)
18. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

19. Tatani, K., Nakamura, Y.: Dimensionality reduction and reproduction with hierarchical nlpca neural networks-extracting common space of multiple humanoid motion patterns. In: ICRA [1], pp. 1927–1932
20. Vukobratović, M., Yu, S.: On the stability of anthropomorphic systems. Mathematical Biosciences 15, 1–37 (1972)
21. Wang, J., Fleet, D., Hertzmann, A.: Gaussian process dynamical models. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) Advances in Neural Information Processing Systems, vol. 18, pp. 1441–1448. MIT Press, Cambridge (2006)

Incremental Learning of Full Body Motion Primitives*

Dana Kulić and Yoshihiko Nakamura

Abstract. This paper describes an approach for autonomous and incremental learning of motion pattern primitives by observation of human motion. Human motion patterns are abstracted into a dynamic stochastic model, which can be used for both subsequent motion recognition and generation. As new motion patterns are observed, they are incrementally grouped together using local clustering based on their relative distance in the model space. The clustering algorithm forms a tree structure, with specialized motions at the tree leaves, and generalized motions closer to the root. The generated tree structure will depend on the type of training data provided, so that the most specialized motions will be those for which the most training has been received. A complete system for online acquisition and visualization of motion primitives from continuous observation of human motion will also be described, allowing interactive training.

1 Introduction

Learning from observation is an attractive paradigm for humanoid robots, as it would allow robots to learn how to accomplish tasks by simply observing a human teacher,

Dana Kulić
Electrical and Computer Engineering Department, University of Waterloo,
200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1
e-mail: dkulic@ece.uwaterloo.ca

Yoshihiko Nakamura
Department of Mechano-Informatics, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku,
113-8656 Tokyo, Japan
e-mail: nakamura@ynl.t.u-tokyo.ac.jp

* This chapter is based on work by the authors first reported in the International Journal of Robotics Research [39] and the International Symposium on Robot and Human Interactive Communication (RO-MAN) 2009 [32].

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 383–406.
springerlink.com

rather than through programming or trajectory planning. It is an especially attractive approach for humanoids robots, as these robots have complex, multi-degree of freedom systems. By learning from observation, the humanoid robot can take advantage of the similar structure between itself and the human. In addition, such a programming interface is suitable for non expert demonstrators, allowing robots to acquire new skills through observation of everyday people in their natural environments.

Learning from observation has received considerable attention in the literature [7, 58, 31, 46]. However, many of the approaches proposed thus far consider the case where the motion primitives to be learned are specified in advance by the designer. The motion primitives to be learned are segmented and clustered a-priori, and then an off-line, one-shot learning process is applied. Following learning, the robot is able to execute the learned motions, but no further learning or improvement takes place. However, a robot that cohabits with humans in their environment should be able to learn incrementally, over a lifetime of observations, as well as accumulate knowledge and improve performance over time. The robot should be capable of observing, segmenting and classifying demonstrated actions on-line during co-location and interaction with the (human) teacher. If the robot is able to extract motion primitives during on-line interaction with the teacher, it may then also be able to employ other learning modalities, such as teacher feedback [53, 45, 12], improvement through practice [14, 28, 56], or a combination of these approaches [3]. In this type of interactive and online learning framework, the number of motion primitives may not be known in advance and may be continuously increasing as the teacher demonstrates new tasks, and must be determined autonomously by the robot, as it is observing the motions. In addition, as the number of observed motions and learned motion primitives grows, the robot must be able to organize the acquired knowledge in an efficient and easily searchable manner.

In order to extract motion primitives during on-line observation, several key issues must be addressed by the learning system: automated motion segmentation, recognition of previously learned motions, automatic clustering and learning of new motions, and the organization of the learned data into a storage system which allows for easy data retrieval. In this paper, our focus is on the last three items: motion learning and recognition, clustering and organization.

1.1 Related Work

Robot skill learning from observation is a longstanding area of research [40, 41, 44, 10]. Breazeal and Scasellati [7] and Schaal et al. [58] provide an overview on recent research on motion learning by imitation. Lopes et al. [46] provide a recent review focusing also on the human cognitive processes on which much of the research is based. As noted by Breazeal and Scasellati, the majority of algorithms discussed in the literature assume that the motions to be learned are segmented a-priori, and that the model training takes place off-line. Several different techniques have been applied to modeling the motion primitives, including dynamical models [51], polynomial functions [55], neural networks [54] and stochastic models [6, 18]. Imitation

learning problems have also been formulated as inverse reinforcement learning problems [52, 1], where the goal is to learn the optimal cost function given a set of expert demonstrations [48].

ining what to imitate based on the variance in each signal type across demonstrations.

Hidden Markov Models have been a popular technique for human motion modeling, and have been used in a variety of applications, including skill transfer [65, 10], robot assisted surgery [30, 13], sign language and gesture modeling [59, 4, 17], motion prediction [16, 2] and the mimesis model of symbol emergence for motion primitives [19, 20, 21, 49]. A common paradigm is *Programming by Demonstration* (PbD) [10, 9, 4]. While PbD describes a general framework including online acquisition, for the majority of the approaches proposed thus far, the number of motions is specified a-priori, the motions are clustered and trained off-line, and then a static model is used during the recognition phase.

When using HMMs to represent the motion primitive, there is also an issue of how a motion primitive should be represented. The best representation may differ with the type of task being demonstrated, for example, for dance movements, joint angles may be the best representation, whereas for assembly tasks, object positions and Cartesian hand coordinates may be more important. The representation to be used is typically selected by the designer a-priori. Lee and Nakamura [43] develop an approach for converting between joint angle and Cartesian representations of motion primitives. Billard et al. [6] propose an approach for autonomously determining what to imitate based on the variance in each signal type across demonstrations.

Jenkins and Mataric [25] describe a system for extracting behaviors from motion capture data. In their algorithm, joint angle data is embedded in a lower dimensional space using the spatio-temporal Isomap (ST-Isomap) algorithm [26]. Once the data has been reduced, it is clustered into groupings using the ”sweep-and-prune” technique. Once a model of the primitive behaviors is formed, a higher level re-processing of the data is performed to discover meta-behaviors, i.e., probabilistic transition probabilities between the behaviors. Similarly to primitive behaviors, meta-level behaviors are derived by extracting meta-level feature groups using ST-Isomap. Chalodhorn and Rao [8] also apply dimensionality reduction using Principal Components Analysis (PCA) to learn the kinematics and dynamics as well as the relationship to low-level sensory information of the actions to be learned. Tatani and Nakamura [62] have used non-linear PCA to generate the reduced dimensionality motion space. However, these algorithms cannot operate incrementally, as the entire range of motions is required to form the lower-dimensional space embedding.

Takano and Nakamura [61] develop a system for automated segmentation, recognition and generation of human motions based on Hidden Markov Models. In their approach, a set of HMMs is trained incrementally, based on automatically segmented data. Each new motion sequence is added to the HMM which has the highest likelihood of generating the motion. The selected HMM is then trained for competitive learning with the new data. The number of HMMs is fixed and determines the level of abstraction. A choice of smaller number results in a higher level of abstraction.

However, no mechanism is proposed for the emergence of a hierarchy among different levels of abstraction.

Ogata et al. [54] are one of the few works that consider the case of long term, incremental learning. In their work, a neural network is used to learn a navigation task during cooperative task execution with a human partner. The authors introduce a new training method for the recursive neural network, which avoids the problem of memory corruption during new training data acquisition. However, in this case, the robot learns only one task, and no hierarchical organization of knowledge takes place.

Another important issue during model learning is the selection of the model size and structure. Dixon, Dolan and Khosla [11] describe a system for partially automating the programming of an industrial robot manipulator by facilitating the definition of trajectory waypoints. Based on previous programming examples, the proposed system predicts where the user may move the end-effector and automatically moves the robot to that point. Continuous density HMMs are used to model previously observed user programming inputs. Relative position and orientation of the end effector is used as the observed data. Both the structure and the parameters of the HMM are trained on-line. Initially, each example is constructed as a front-to-back model, assigning each waypoint to one state. Next, similar nodes are iteratively merged, resulting in the final topology. To facilitate real-time operation, a one-shot training procedure is used to estimate the model parameters.

Ekvall, Aarno and Kragic [13] describe a system for robotic assistance during microsurgery, where adaptive virtual fixtures are used to guide the surgeons hands. For each task, an HMM is trained by using SVM to determine the appropriate number of states (corresponding to straight lines in the trajectory) and to estimate the probability density function for each state of the HMM. The HMM is then used to recognize the current state (line direction), and apply the appropriate fixture to constrain the manipulator's motion.

Billard et al. [6] use HMM models for motion recognition and generation of humanoid motions. The Bayesian Information Criterion (BIC) is used to select the optimal number of states for the HMM. In the experiments, three different tasks are demonstrated using the HOAP-2 humanoid robot. The robot is trained using kinesthetic training. However, all the exemplar motion patterns are acquired and grouped before the training begins, and the number of motions to be learned is specified a priori.

The use of the Akaike criterion [37] has also been proposed, however, both the AIC and BIC criteria are based on a tradeoff between model performance at characterizing the observations, and the number of parameters, and both require a time-consuming search of the model space to find the best matching model.

In the approach proposed herein, a variable structure Hidden Markov Model based representation is used to abstract motion patterns as they are perceived. Individual motion patterns are then clustered in an incremental fashion, based on intra model distances. The resulting clusters are then used to form a group model, which can be used for both motion recognition and motion generation. The model size is adjusted automatically on-line, based on the accuracy requirements in the given

region of the knowledge space. A new algorithm for sequentially training the stochastic models is used [39], to allow fast, on-line training, and take advantage of existing knowledge stored in the lower-accuracy model. Section 2 describes the stochastic encoding of each motion primitive and model training algorithm. The incremental behavior learning and hierarchy formation algorithm is described in Section 3. An interactive learning environment for collecting and learning motions is described in Section 4. Experimental results on a motion capture data set are described in Section 5; concluding remarks are presented in Section 6.

2 Factorial Hidden Markov Models

A Hidden Markov Model (HMM) is a type of stochastic model which abstracts time series data as a first order stochastic dynamic process. The dynamics of the process are modeled by a hidden discrete state variable, which varies according to a stochastic state transition model $A[N,N]$, where N is the number of states in the model (i.e., the number of unique states the hidden state variable can assume). Each state value is associated with a continuous output distribution model $B[N,K]$, where K is the number of outputs. Typically, for continuous data, a Gaussian or a mixture of Gaussians output observation model is used. HMMs are commonly used for encoding and abstracting noisy time series data, such as speech [57] and human motion patterns [6, 60], because they are able to encode data with both temporal and spatial variability. In the imitation learning domain, HMMs are also attractive since they are a generative model, so the same model can be used both for motion recognition and for motion generation. Efficient algorithms have been developed for model training (the Baum-Welch algorithm), pattern recognition (the forward algorithm) and hidden state sequence estimation (the Viterbi algorithm) [57]. The Baum-Welch algorithm is an Expectation-Maximization (EM) algorithm. In the Expectation step, given a set of model parameters and a data sequence, the posterior probabilities over the hidden states are calculated. Then, in the Maximization step, a new set of model parameters are calculated which maximize the log likelihood of the observations.

Once trained, the HMM can also be used to generate a representative output sequence by sampling the state transition model to generate a state sequence, and then sampling the output distribution model of the current state at each time step to generate the output time series sequence. A schematic of an HMM is shown in Figure 1.

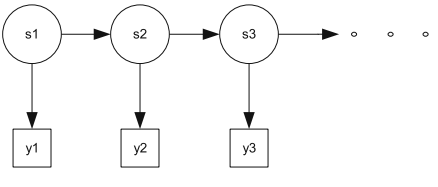


Fig. 1 Hidden Markov Model. In the figure, s refers to the hidden state, while y refers to the observation.

A Factorial Hidden Markov Model (FHMM), first proposed by Ghahramani and Jordan [15] is a generalization of the HMM model, where there may be multiple dynamic processes interacting to generate a single output. In an FHMM, multiple independent dynamic chains contribute to the observed output. Each dynamic chain m is represented by its own state transition model $A_m[N_m, N_m]$ and output model $B_m[N_m, K]$, where M is the number of dynamic chains, N_m is the number of states in dynamic chain m, and K is the number of outputs. At each time step, the outputs from all the dynamic chains are combined, and output through an expectation function to produce the observed output. The expectation function is a multivariate Gaussian function with the chain output as the means, and a covariance matrix representing the signal noise. FHMMs have been used to model speech signals from multiple speakers [5], and the dynamics of a robot and changing environment for simultaneous localization and mapping [47]. FHMMs have also been used for modeling the shape and timing of motion primitives during handwriting [63]. For human motion representation, FHMMs have been found to improve discrimination ability between similar motions, and can provide more accurate modeling of the motion primitives for motion generation [38]. Figure 2 shows a schematic of an FHMM with two dynamic chains.

An FHMM can be trained by a straightforward extension of the Baum-Welch algorithm [15]. However, this (exact) algorithm has a time complexity of $O(TMN^{M+1})$ where T is the length of the data sequence. As the time complexity increases exponentially with the number of chains, the E-step of the EM algorithm becomes intractable for a large number of chains. Faster, approximate algorithms, for which the time complexity is quadratic as a function of the number of chains, have been developed for FHMM training, which implement an approximate rather than the exact E step. These are based on variational methods [15], or generalized backfitting [22].

With either the variational [15], or the generalized-backfitting algorithm [22], the chains are trained simultaneously, significantly increasing the training time required. Similarly, due to the more complex structure of the FHMM model, the recognition algorithm (the forward procedure) is more complex and time consuming as compared to a single chain HMM. Therefore, it would be preferrable to use a compact representation (single HMM chain) when patterns are dissimilar and easy to

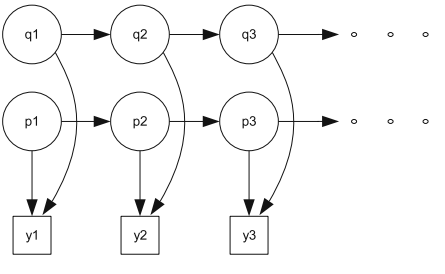


Fig. 2 Factorial Hidden Markov Model. In the figure, p and q are two hidden variables evolving independently, but combining to generate the observation y.

distinguish, and a more detailed representation (multiple chains) during generation and when motions are very similar to each other. For this approach, a modified training algorithm is developed [36, 39], training the chains sequentially instead of simultaneously. With the proposed approach, the single chain (simple HMM) is trained first, and subsequent chains are added as needed. Each time a new chain is added, the new chain is trained on the error between the training data and the output generated by the existing chains.

Once the FHMM is trained, pattern recognition can be implemented by an adaptation of the forward-backward algorithm [15], and pattern generation can be obtained by first sampling the state sequence and then sampling from the output distribution of each state to compute the generated output.

2.1 Human Motion Pattern Representation Using FHMMs

HMMs have been widely used to model human motion data for both recognition and motion generation [6, 60]. In our approach, each motion primitive is initially encoded as a simple Hidden Markov Model (HMM) $\lambda_p(\pi, A, B)$, where π is the vector of initial state probabilities (the likelihood that the model begins in state i), A_{NxN} is the NxN state transition matrix, and B is the output observation model. N represents the number of states in the model. For representing non-periodic human motion primitives, a front-to-back HMM is used, such that the state sequence must begin with the first state, i.e.,

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{1}$$

In addition, the state transition matrix is constrained such that no backwards state transitions are allowed, and state transitions can occur only between consecutive nodes, i.e.,

$$a_{ij} = \begin{cases} a_{ij}, & j = i or j = i+1 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The output observation vector is composed of continuous observations. Either joint angles, or cartesian positions of key body locations (such as the wrist, elbow, head, knee, etc.) can be used as the observation vector. The output observation model B is a finite mixture of Gaussians of the form

$$b_j(\mathbf{O}) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\mathbf{O}, \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \tag{3}$$

where $b_j(\mathbf{O})$ is the mixture of Gaussians representation of the observation vector $\mathbf{O}$ being modeled for state j, c_{jm} is the mixture weight for mixture m in state j, $\mathcal{N}$ is the Gaussian probability density with mean vector $\boldsymbol{\mu}_{jm}$ and covariance matrix $\boldsymbol{\Sigma}_{jm}$. The number of Gaussian mixtures is denoted by M. To reduce the number of training parameters, the covariance matrix $\boldsymbol{\Sigma}_{jm}$ is constrained to be diagonal.

However, when using the same HMM structure for both recognition and generation, there is an inherent tradeoff when selecting the HMM model size, (i.e. the number of states). An HMM model with a low number of states will be better at generalizing across data with high variability, and better at correctly recognizing new data when the motions being recognized are dissimilar. In general, an HMM with a low number of states (5 - 20) provides excellent recognition performance [6, 60] in such a scenario. When automatically selecting the number of states based on Bayesian [6] or Akaike [37] information criteria, which consider only recognition performance, under 10 states are usually selected for typical human motion patterns such as walking, kicking, punching, etc. On the other hand, a model with a low number of states will not be able to faithfully reproduce the observed motion during generation, a criterion which is not considered by the information criteria. Conversely, a large state model will be better at reproducing the observed motion, but will be prone to over-fitting. Factorial HMMs, which use a distributed rather than a single multinomial state representation, provide a more efficient approach for combining good generalization for recognition purposes with sufficient detail for better generation. Therefore, rather than selecting a fixed model size, in our approach the model size is adapted by adding chains to the initial HMM model to form FHMMs when the recognition and generation requirements increase.

3 Incremental Behavior Learning and Hierarchy Formation

The aim of the proposed approach is to allow the robot to learn motion primitives through continuous observation of human motion, by extracting motion segments (the motion primitives) that repeatedly occur in the demonstration. Motion primitive candidates are first extracted from the continuous data stream through automatic segmentation [24, 35]. Each time a new motion primitive candidate is observed, the system must determine if the observed motion is a known motion, or a new motion to be learned. In addition, over the lifetime of the robot, as the number of observed motions becomes large, the robot must have an effective way of storing the acquired knowledge for easy retrieval and organization. In the proposed approach, a hierarchical tree structure is incrementally formed representing the motions learned by the robot. Each node in the tree represents a motion primitive (i.e., a learned behavior primitive), learned from repeated observation of that motion segment. Each node contains the data segments the system judges to be exemplar observations of that behavior (the group of similar observations), and a group model synthesizing the observations into a motion primitive model. The group model can be used to recognize a similar motion, and also to generate the corresponding motion for the robot. Rather than using a fixed size model for the motions, the model accuracy is adjusted based on the recognition requirements in the given region of the knowledge database. If there are many motions similar to the modeled motion, a better (higher number of chains in the FHMM) model will be applied.

The algorithm initially begins with one behavior group (the root node). Each time a motion is observed from the teacher, it is encoded into a Hidden Markov model.

The encoded motion is then compared to existing behavior groups via a tree search algorithm, using the symmetric model distance measure [57, 38], and placed into the closest group. Each time a group is modified, a hierarchical agglomerative clustering algorithm [23] is performed within the exemplars of the group. If a cluster with sufficiently similar data is found, a child group is formed with this data subset. The time series data of the motion examples forming the child group is then used to generate a single group model, which is subsequently used for both behavior recognition and generation. Therefore the algorithm incrementally learns and organizes the motion primitive space, based on the robot's lifetime observations. The algorithm pseudocode is shown in Figure 4, while a schematic of the incremental memory structure formation is shown in Fig. 3. Figure 4 shows the detailed algorithm sequence that occurs each time a new motion is observed, while Fig. 3 shows the evolution of the tree structure in a 2 dimensional representation of the motion space as more and more motions are observed.

This algorithm allows the robot to incrementally learn and classify behaviors observed during continuous observation of a human demonstrator. The generation of a hierarchical structure of the learned behaviors allows for easier retrieval, and the automatic generation of the relationships between behaviors based on their similarity and inheritance. In addition, the robot's knowledge is organized based on the type

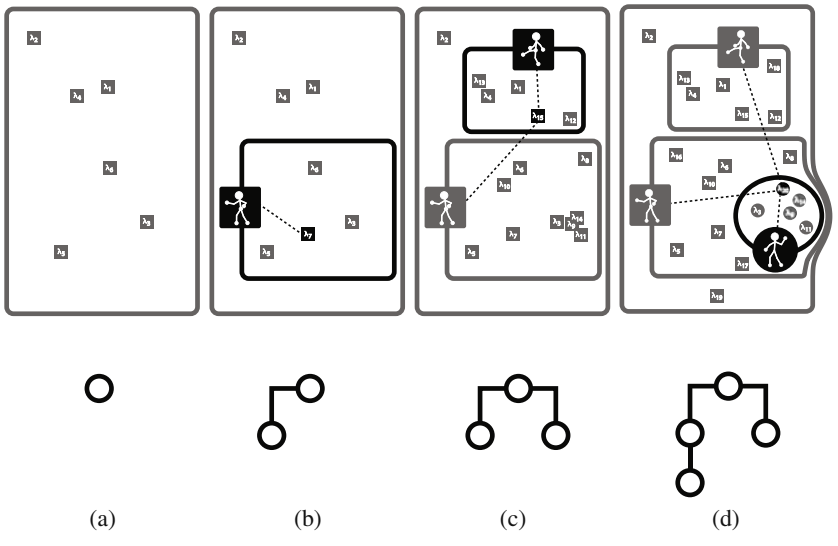


Fig. 3 Schematic Illustration of the Clustering Algorithm. The top part of each figure shows a 2 dimensional visualization of the motion space, while the bottom part shows the corresponding tree structure. (a) initial state, when only one group is present; (b) a child group forms when enough similar examples are observed; (c) new observations are located into the closest group based on the distance between the new observation and the group model; (d) a higher order model is used in dense areas of the motion space.

1: **procedure** INCREMENTALCLUSTER
2: **Step1** Encode observation sequence O_i into an HMM λ_i
3: **Step2** Search the behavior tree for the closest group λ_{Gj} to the current observation model λ_i, based on the inter-model distance
4: **Step3** Place λ_i into the closest group G_c
5: **Step4** Perform clustering on all the exemplar motions within G_c
6: **Step5** If a sufficiently similar subgroup of motions is found, form a new group G_n, as a child of G_c, containing the observation sequences of the subgroup
7: **Step6** Using the observations sequences of the new subgroup, form the group model λ_{Gn}
8: **end procedure**

Fig. 4 Clustering Algorithm Pseudocode.

of training received, so that the robot's knowledge will be most specialized in those areas of the behavior space where the most data has been observed.

3.1 Step 1: Observation Sequence Encoding

Each newly acquired observation sequence is encoded into a Hidden Markov Model. In order to train the model, the HMM parameters, such as the number of states and the number of gaussian mixtures must be selected. We use a variable size model, where the number of dynamics chains in an FHMM model are increased based on the density of the motion exemplars in the relevant region of the motion space. With this approach, each motion is initially represented by a simple, single-chain, front-to-back HMM, with 6 - 8 hidden states, and using a single Gaussian as the observation model. If a better model is required, additional chain(s) are added as described in Section 3.4 below. below.

3.2 Steps 2 and 3: Intra-model Distance Calculation

Once the newly observed behavior is encoded into a model, it is compared to existing groups (if any). Here, the distance between two models can be computed as the difference in likelihood that an observation sequence generated by one model could have also been generated by the second model [57]. This distance measure represents a Kullback-Leibler distance between the two models:

$$D(\lambda_1,\lambda_2) = \frac{1}{T}[logP(O^{(2)};\lambda_1) - logP(O^{(2)};\lambda_2)] \tag{4}$$

where λ_1, λ_2 are two HMM models, $O^{(2)}$ is an observation sequence *generated* by λ_2 and T is the length of the observation sequence. Since this measure is not symmetric, the average of the two intra HMM distances is used to form a symmetric measure:

$$D_s = \frac{D(\lambda_1,\lambda_2) + D(\lambda_2,\lambda_1)}{2} \tag{5}$$

The formulation of the distance based on the model probability means that this measure can similarly be applied to Factorial HMM models, by using the modified forward procedure [15] to calculate the log likelihood, as well as used to compare FHMM and HMM models [37]. The modified forward procedure [15] makes use of the independence of the underlying Markov chains to sum over the transition matrices of each chain, simplifying the computation.

The distance measure quantifies the level of difficulty in discriminating between two models λ_1, λ_2. By encoding more information about each pattern, FHMMs can improve the ability to discriminate between motion patterns, which can be especially useful when there are many similar motion patterns. Using the more detailed FHMM models increases the intra-model distances, as depicted conceptually in Figure 5. In addition, if the FHMM and HMM models of the same motion remain sufficiently similar, FHMM models may be combined with HMM models, by using FHMM models only in dense areas of the motion model space where better discriminative ability is required (shown in Figure 6).

The repository of known groups is organized in a tree structure, so that the new observation sequence does not need to be compared to all known behaviors. The comparison procedure is implemented as a tree search. At each node of the tree, the

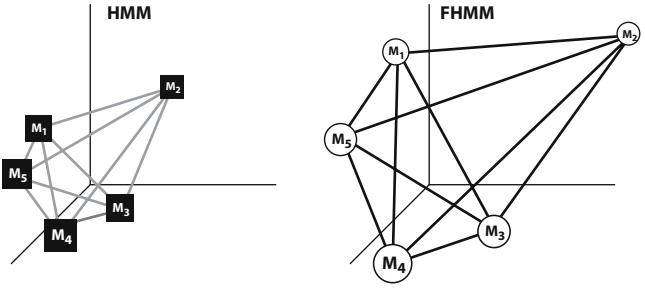


Fig. 5 Schematic comparing an HMM model vector space and an FHMM model vector space. The axes represent principal directions in the vector space (note the space does not necessarily need to be 3 dimensional).

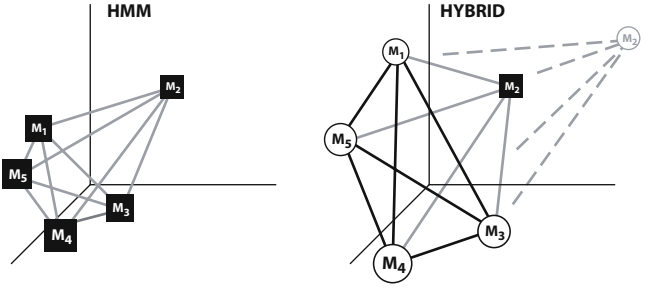


Fig. 6 Schematic comparing an HMM model vector space and a hybrid HMM-FHMM model vector space.

new observation sequence is compared to the leaves of that node. If the distance between the new observation sequence and one of the child nodes is sufficiently small, the search recurses to the most similar child node, otherwise, the new observation sequence is added to the current node.

$$D_{thresh} = K_{maxGD} D^G_{max} \tag{6}$$

D_{thresh} is the distance threshold at which a new observation sequence is considered for inclusion to a node, K_{maxGD} is the multiplication factor applied and D^G_{max} is the maximum intra observation distance for the given node. If the distance between the new observation and the cluster is larger than D_{thresh}, this cluster will not be considered as a possible match for the new observation sequence. If there are multiple candidate clusters, the new sequence is placed in the closest cluster. If there are no candidates, the new sequence is placed in the parent cluster. In the case of a new motion pattern which is completely dissimilar to any existing motion patterns, the motion pattern will be placed into the root node.

The maximum intra observation distance for the placement node D^G_{max} is also the criterion used to decide the level of model complexity required for the motion sequence. If the new motion is most similar to a node which D^G_{max} falls below a certain threshold, the FHMM model is generated by adding additional chain(s) to the current representation, to increase the discriminative ability of the model.

Once the best matching cluster is selected for the newly observed sequence, the distance between the new observation and all the other observations in the cluster is calculated and added to the distance matrix stored for that node. This matrix is used for new cluster formation, as described in the next section.

3.3 Steps 4 and 5: Clustering and New Group Formation

When a new observation sequence is added to a cluster, a clustering procedure is invoked within the modified group, to determine if a subgroup may be formed. A subgroup consists of a group of motion primitives which are more similar to each other than the level of similarity found in the group. The complete link hierarchical clustering algorithm is used to generate the data clusters within a group [23]. Clusters can then be formed based on two criteria: number of leaves in the cluster, and the maximum proximity measure of the potential cluster. Herein, both a minimum number of elements and a maximum distance measure are used. The maximum distance measure is based on the average of the inter motion distances in the cluster:

$$D_{cutoff} = K_{cutoff} \mu \tag{7}$$

where D_{cutoff} is the distance cutoff value (i.e., only clusters where the maximum distance is less than this value will be formed), and μ is the average distance between observations.

3.4 Step 6: New Behavior Instantiation

If a new subgroup is generated in Step 5, a new group model is trained using the raw observation sequences from all the group elements. The structure of the new group model (i.e., HMM vs. FHMM) is determined based on the maximum intra observation distance for group, D^G_{max}. If motions are becoming increasingly similar to each other, so that better discrimination ability is required, additional chains are added to the existing model, one at a time, based on a simple threshold evaluation. A training algorithm allowing incremental training of each chain has also been developed [39]. In the experiments described below, the threshold value was determined manually, however, the threshold could also be determined automatically based on the inter-distance distribution within the group, or the decision could be based by testing for the presence of a multimodal distribution within the group. The generated model is subsequently used by the robot to generate behaviors. The group model replaces the individual observations in the parent node.

If one of the group elements allocated to the new cluster is already a group model, the generated motion sequence based on that model is used for the training. In this case, a modified form of the re-estimation formulas for multiple observation sequences [57] is used. The algorithm is modified by over-weighting the group models, in order to account for the fact that there are multiple observation sequences stored in the generated model, and therefore more weight should be given to the group model, as compared to the individual observation sequences.

3.5 Deterministic Motion Generation

Once a cluster node has been formed, the group model for the node constitutes the abstraction of the motion primitive. This model is then used to generate a motion trajectory for the robot to execute.

When the generated motion sequence is to be used for robot motion commands, we do not want to introduce the noise characteristics abstracted by the stochastic model. We therefore use a greedy policy to estimate the optimum state sequence. First, for each chain m, the starting state q^m_0 is selected by choosing the highest value from the initial state probability distribution. At each state, the state duration is calculated based the state transition matrix,

$$\bar{d}^m_i = \frac{1}{1 - a^m_{ii}} \tag{8}$$

Following $\bar{d}^m_i$ samples in state i, the next state is selected by greedy policy from the state transition matrix, excluding the $1 - a^m_{ii}$ probability. If the model type is front-to-back, the algorithm iterates until the final state is reached, otherwise the state sequence is generated for the specified number of time steps. Once the state sequence has been generated for each chain, the output sequence is calculated by summing the contribution from each chain at each time step, based on that chain's current state value. This algorithm selects the state sequence by a local greedy

policy over the state transition algorithm. Alternatively, if a motion most similar to a recently observed motion is required, the optimal state sequence could be generated by using the Viterbi algorithm, as described in Lee and Nakamura [42].

After the trajectory is generated, some low-pass filtering or smoothing is required as a post-processing step to eliminate the artifacts caused by discrete state switching and generate a smooth trajectory for use as a command input.

4 Interactive Motion Learning Environment

During the demonstration process, (as well as following learning), it is important for the human demonstrator to have visibility into the material learned so far, so that the teacher can tailor further demonstrations to the current knowledge of the learner. An interactive training system has been developed for this purpose. In the developed system, at any time during the demonstration, as well as off-line, the human demonstrator is able to view animations of the motion primitives acquired thus far, an overview of the current tree structure of the database, and a visualization of the current version of the motion primitive graph. During the acquisition process itself, the demonstrator views an animation of the motion currently being executed, and is also notified when the system has recognized the current motion primitive as one of the known motions.

In the implemented system, motion capture is used to acquire the motions of the demonstrator. The system receives the marker data from the motion capture system directly. The marker data is first converted to joint angle data via on-line inverse kinematics [50, 64], based on the kinematic model of the animation character. The animation character, in terms of the body shape and the number of degrees of freedom, can be specified by the user based on the type of training task. For training tasks where motions are being acquired for human motion analysis, the model can be selected based on the DoF requirements of the task being demonstrated, or, in the case of motions being acquired for later re-targeting to the humanoid robot, the

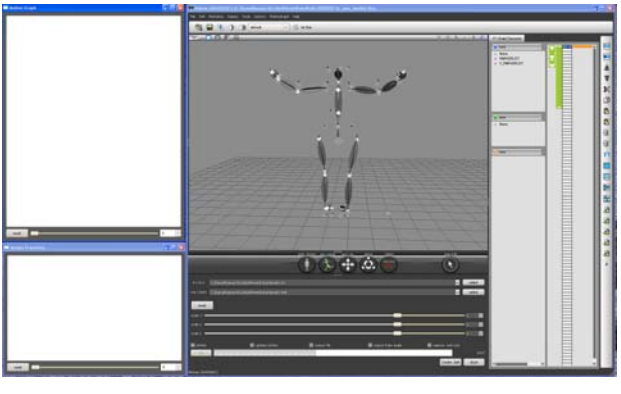


Fig. 7 Screen shot of the visualization software during motion acquisition.

model can be specified based on the kinematic structure of the robot. The current motion of the demonstrator is displayed on the animated character simultaneously with motion execution. This function is useful to allow the demonstrator to verify that the inverse kinematics is working properly and that motions are being retargeted to the model joint angle data correctly, without introducing representational singularities. Figure 7 shows a screen shot of the visualization system during motion acquisition.

The time series joint angle data is first segmented into motion primitive candidates, using an automated segmentation approach [24, 35]. The segmentation approach used tries to find segment points which result in segments of minimum variance. This is done via the Kohlmorgen and Lemm algorithm [29], where a Hidden Markov Model is used to represent the incoming data sequence, where each model state represents the probability density estimate over a window of the data. Based on the assumption that data belonging to the same motion primitive will have the same underlying distribution, the segmentation is implemented by finding the optimum state sequence over the developed model.

Each extractive motion primitive candidate is then passed to the incremental clustering algorithm, as described in Section 3. Each time a new motion primitive is abstracted, a node is added to the tree structure representing the current status of the knowledge database. The tree structure can be viewed at any time within the visualization system; the formation of the first node during training is shown in Figure 8.

Each time a previously learned motion primitive is recognized in the current demonstration, the visualization system notifies the demonstrator that the performed motion is already known to the system. This is indicated (see Figure 9) by changing the color of the identified node in the tree and motion primitive graph.

Once the demonstration is over, the user can load all the learned motions (behaviors), and play them back on the animated character to verify that motion primitives have been abstracted correctly.

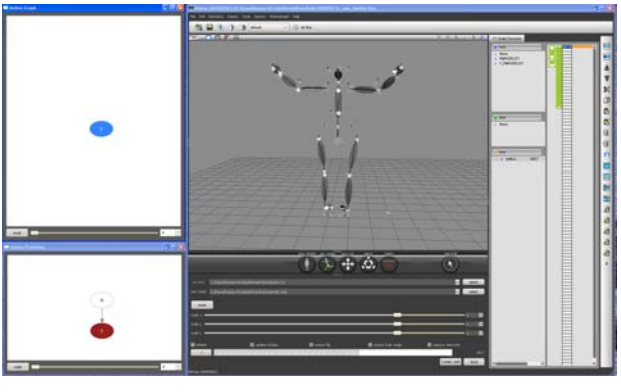


Fig. 8 Screen shot of the visualization software during node formation.

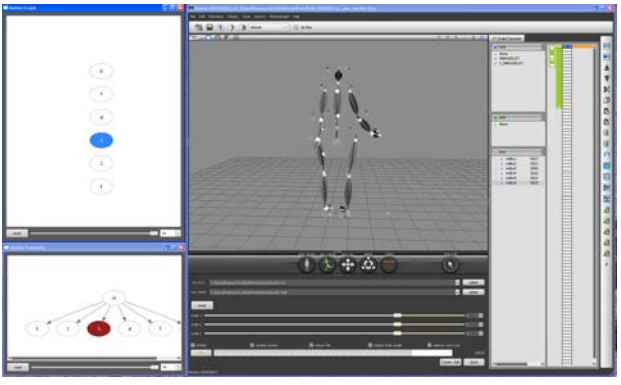


Fig. 9 Screen shot of the visualization software during motion recognition.

5 Experiments

The system is demonstrated with two sets of experiments. In the first set, a manually segmented data set is used to illustrate the behavior of the clustering algorithm. In the second set of experiments, the interactive learning system is demonstrated. Combined with a low-level controller for ensuring postural stability, the approach described here has also been used to implement behaviors on a physical humanoid robot [33].

5.1 Incremental Clustering Experiments

The experiments described below have been performed on a data set containing a series of 9 different human movement observation sequences obtained through a motion capture system [27]. The data set contains joint angle data for a 20 degree of freedom humanoid model from multiple observations of walking (WA - 28 observations), cheering (CH - 15 observations), dancing (DA - 7 observations), kicking (KI - 19 observations), punching (PU - 14 observations), sumo leg raise motion (SL - 13 observations), squatting (SQ - 13 observations), throwing (TH - 13 observations) and bowing (BO - 15 observations). Figure 10 shows an example of a walking motion from the dataset.

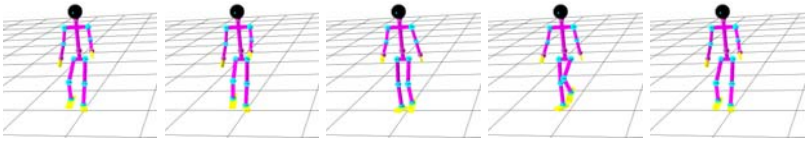


Fig. 10 Sample Walking Motion.

The motion sequences are presented to the algorithm in random order. Motion sequences are presented one at a time, simulating on-line, sequential acquisition. After each motion is presented, the algorithm is executed, performing incremental clustering. In each simulation performed, the algorithm correctly segments the behaviors such that the resulting leaf nodes represent the grouping that would be obtained with an off-line method. Out of 100 simulation runs performed at each of the parameter settings, there was no cases of misclassification at the leaf nodes, showing that the final clustering is robust to presentation order. Experiments using only single-chain HMMs, as well as experiments using adaptable models were performed. In tests where single-chain HMMs were used, the appropriate model size was selected via the Akaike criterion. Sample clustering results for the single-chain HMMs are shown in Figs. 11, 12 and 13. Note that the actual order of node formation will vary depending on the motion presentation order.

The algorithm parameter K_{cutoff} (the parameter which controls when a new cluster is formed from an existing cluster) determines the resultant tree structure. For high cutoff values (see Figure 11), the resulting tree structure is flat, and fairly insensitive to presentation order, consistent with the off-line clustering result. In about 9% of cases, the 'dance' group fails to form, since this group contains the least examples. At the high cutoff value, the punch and throw motions are too similar to subcluster, resulting in a single hybrid generated motion (indicated as GPT in Figure 11). The generated motion resulting from that subcluster is shown in Figure 14. As can be seen in the figure, the motion is an averaging of the two motions.

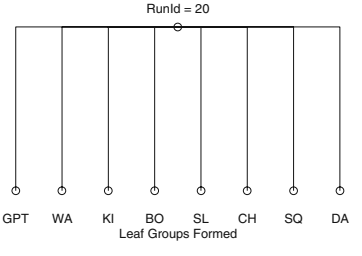


Fig. 11 Sample Segmentation Result, $K_{cutoff} = 1.2$.

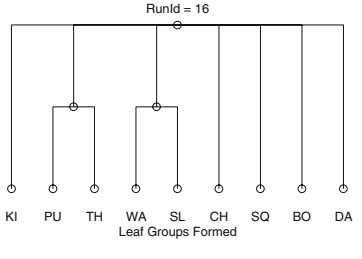


Fig. 12 Sample Segmentation Result, $K_{cutoff} = 0.9$.

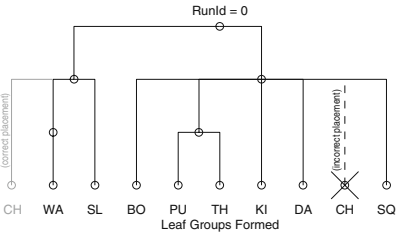


Fig. 13 Sample Segmentation Result, $K_{cuttoff} = 0.9$.

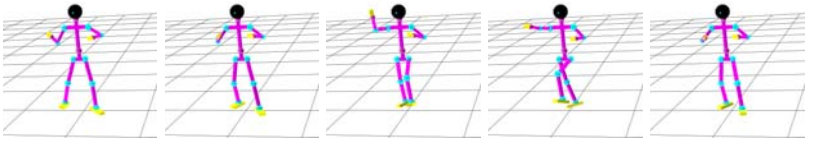


Fig. 14 Generated Hybrid Punch/Throw Motion.

When low values of K_{cutoff} are used, nodes are quicker to form, and the resulting tree structure becomes more dependant on presentation order. The similarity level at which nodes will form is highly dependent on presentation order. Figures 12 and 13 show two examples of different tree structures formed, from two simulation runs. Note that the identified leaf nodes remain the same. In addition, using the lower cutoff value makes it easier to subdivide the similar throw and punch motions. Even though the cutoff level was the same for both experiments, the similarity level of the nodes formed differed, based on the presentation order. The result in Figure 12 is consistent with global clustering, while in the result shown in Figure 13, one node is incorrectly assigned. This type of error is due to the local nature of the algorithm, i.e. clustering is being performed when only a part of the data set is available. Therefore, there is a tradeoff when selecting the K_{cutoff} value between facilitating quick node formation and differentiation and introducing misclassifications in the hierarchy tree. However, since the leaf nodes are identified correctly regardless of the K_{cutoff} value, a slower rate tree correction algorithm can be periodically applied, to reposition leaf nodes to the correct branch as more data becomes available [34].

When single-chain HMM models are used, at high levels of K_{cutoff}, the similar motions of punch and throw cannot be distinguished. However, if both HMM and FHMM models are used, such that FHMM models are inserted into a dense region of the motion space, as described in section III.B, better discrimination ability can be achieved, given the same number of training examples. A sample result of the clustering performance using the adaptable models is shown in Fig. 15. The adaptable model can distinguish between similar motions TH and PU, whereas those motions cannot be distinguished when only single chain HMM models are used (see Figure 11).

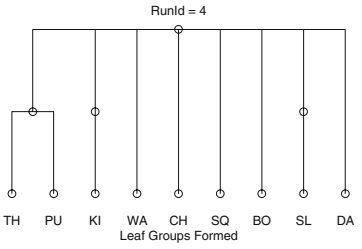


Fig. 15 Sample Segmentation Results when using adaptable models.

5.2 Experiments with the Interactive Training System

The combined segmentation and clustering system was tested on-line with the visualization system using the motion capture studio. The human demonstrator is outfitted with 34 reflective markers located on various parts of the body, and the marker [x,y,z] position is captured and computed by the motion capture system online, at a sample rate of 5ms. During the demonstration, the demonstrator can view the current status of the learning system via a large screen located in the motion capture studio. The marker positioning and experimental setup are shown in Figure 16. The demonstrator performs single and both arm raise, bow and squat motions. The demonstrator performs approximately 6 repetitions of each motion type.

The marker data is then passed to the combined motion extraction and visualization system, which performs the on-line inverse kinematics to generate joint angle data, displays an animation character performing the demonstrator motions, and simultaneously passes the data to the segmentation, clustering and motion primitive extraction module for processing. In previous work, a simplified inverse kinematics model was used (20DoF), however, here we use a more realistic 40 DoF model,

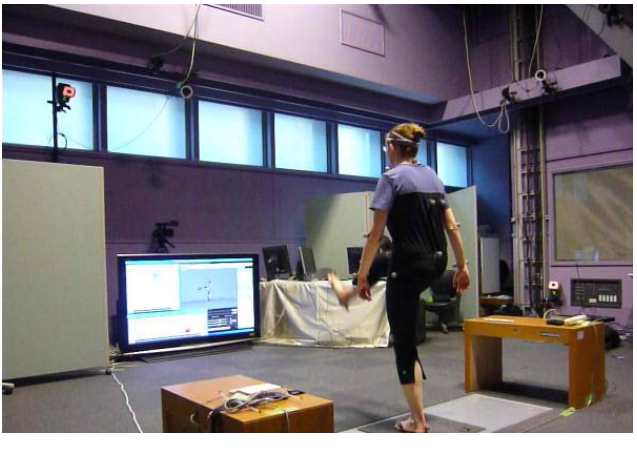


Fig. 16 Marker Setup used for the motion capture experiments.

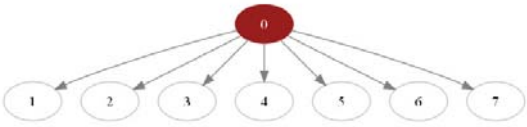


Fig. 17 Final tree structure following 2.5 minutes of data. Node 1 represents the both arms raise primitive, Node 2 represents the both arms lower primitive, Node 3 represents left arm motions, Node 4 represents right arm motions, Node 5 represents Squat Raise, Node 6 represents Squat Lower, and Node 7 represents the Bend down motion.

which is better able to capture the full range of human motion. Screen shots illustrating the visualization system appearance during motion acquisitions are shown in Figures 7, 8 and 9.

Following the completion of the demonstration (approximately 2.5 minutes of data), the system has learned 7 of the 10 motions demonstrated. The resulting tree structure is shown in Figure 17. The single arm motions (both the raise and the lower) are not yet differentiated, but are instead grouped together in nodes 3 and 4. Further examples of the bend raise and single arm motions result in subsequent differentiation and abstraction of these motions as well.

The developed system is able to autonomously extract common motion primitives from on-line demonstration consisting of a continuous sequence of behaviors, as well as visualize the extracted motions to the demonstrator, allowing the demonstrator insight into the learning process, and the ability to interactively adjust further training.

6 Conclusions

This paper develops a novel approach towards on-line, long term incremental learning and hierarchical organization of whole body motion primitives. The learned motions are aggregates of the observed motions, which have been autonomously clustered during observation. The appropriate level of accuracy required for each motion model is determined based on the similarity of the motions to be distinguished, such that a larger model is only used in dense regions of the knowledge base.

The clustered motions are organized into a hierarchical tree structure, where nodes closer to the root represent broad motion descriptors, and leaf nodes represent more specific motion patterns. The tree structure and level of specialization will be based on the history of motions observed by the robot. The resulting knowledge structure is easily searchable for recognition tasks, and can also be utilized to generate the learned robot motions.

An interactive training system is also developed for performing on-line training and allowing the demonstrator to visualize the motions learned thus far, as well as the structure of the learned database of motions.

Acknowledgements. The authors gratefully acknowledge the assistance of Akihiko Murai and Wataru Takano with the data set collection, and Hirotaka Imagawa with the visualization system implementation. This work was supported by the Japanese Society for the Promotion of Science Category S Grant-in-Aid for Scientific Research 20220001.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the International Conference on Machine Learning (2004)
2. Bennewitz, M., Burgard, W., Cielniak, G., Thrun, S.: Learning motion patterns of people for compliant robot motion. International Journal of Robotics Research 24(1), 31–48 (2005)
3. Bentivegna, D.C., Atkeson, C.G., Cheng, G.: Learning similar tasks from observation and practice. In: Proceedings of the International Conference on Intelligent Robots and Systems, pp. 2677–2683 (2006)
4. Bernardin, K., Ogawara, K., Ikeuchi, K., Dillmann, R.: A sensor fusion approach for recognizing continuous human grasping sequences using hidden markov models. IEEE Transactions on Robotics 21(1), 47–57 (2005)
5. Betkowska, A., Shinoda, K., Furui, S.: Fhmm for robust speech recognition in home environment. In: Symposium on Large Scale Knowledge Resources, pp. 129–132 (2006)
6. Billard, A., Calinon, S., Guenter, F.: Discriminative and adaptive imitation in uni-manual and bi-manual tasks. Robotics and Autonomous Systems 54, 370–384 (2006)
7. Breazeal, C., Scassellati, B.: Robots that imitate humans. Trends in Cognitive Sciences 6(11), 481–487 (2002)
8. Chalodhorn, R., Rao, R.P.N.: Learning to imitate human actions through eigenposes. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 357–381. Springer, Heidelberg (2010)
9. Dillmann, R.: Teaching and learning of robot tasks via observation of human performance. Journal of Robotics and Autonomous Systems 47, 109–116 (2004)
10. Dillmann, R., Rogalla, O., Ehrenmann, M., Zollner, R., Bordegoni, M.: Learning robot behaviour and skills based on human demonstration and advice: The machine learning paradigm. In: Proceedings of the International Symposium on Robotics Research, pp. 229–238 (1999)
11. Dixon, K.R., Dolan, J.M., Khosla, P.K.: Predictive robot programming: Theoretical and experimental analysis. International Journal of Robotics Research 23(9), 955–973 (2004)
12. Dominey, P.F., Metta, G., Nori, F., Natale, L.: Anticipation and initiative in human-humanoid interaction. In: Proceedings of the IEEE International Conference on Humanoid Robots, pp. 693–699 (2008)
13. Ekvall, S., Aarno, D., Kragic, D.: Online task recognition and real-time adaptive assistance for computer-aided machine control. IEEE Transactions on Robotics 22(5), 1029–1033 (2006)
14. Erlhagen, W., Mukovskiy, A., Bicho, E., Panin, G., Kiss, C., Knoll, A., van Schie, H., Bekkering, H.: Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning. Robotics and Autonomous Systems 54, 353–360 (2006)
15. Ghahramani, Z., Jordan, M.I.: Factorial hidden markov models. Machine Learning 29, 245–273 (1997)
16. Ho, M.A.T., Yamada, Y., Umetani, Y.: An adaptive visual attentive tracker for human communicational behaviors using hmm-based td learning with new state distinction cpapbility. IEEE Transactions on Robotics 21(3), 497–504 (2005)

17. Iba, S., Paredis, C.J.J., Khosla, P.K.: Interactive multi-modal robot programming. International Journal of Robotics Research 24(1), 83–104 (2005)
18. Inamura, T., Toshima, I., Tanie, H., Nakamura, Y.: Embodied symbol emergence based on mimesis theory. The International Journal of Robotics Research 23(4-5), 363–377 (2004)
19. Inamura, T., Nakamura, Y., Ezaki, H., Toshima, I.: Imitation and primitive symbol acquisition of humanoids by the integrated mimesis loop. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4208–4213 (2001)
20. Inamura, T., Toshima, I., Nakamura, Y.: Acquisition and embodiment of motion elements in closed mimesis loop. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1539–1544 (2002)
21. Inamura, T., Toshima, I., Nakamura, Y.: Acquisition and embodiment of motion elements in closed mimesis loop. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1032–1037 (2002)
22. Jacobs, R.A., Jiang, W., Tanner, M.A.: Factorial hidden markov models and the generalized backfitting algorithm. Neural Computation 14, 2415–2437 (2002)
23. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
24. Janus, B., Nakamura, Y.: Unsupervised probabilistic segmentation of motion data for mimesis modeling. In: Proceedings of the IEEE International Conference on Advanced Robotics, pp. 411–417 (2005)
25. Jenkins, O.C., Matarić, M.: Performance-derived behavior vocabularies: Data-driven acquisition of skills from motion. International Journal of Humanoid Robotics 1(2), 237–288 (2004)
26. Jenkins, O.C., Matarić, M.: A spatio-temporal extension to isomap nonlinear dimension reduction. In: Proceedings of the International Conference on Machine Learning, pp. 441–448 (2004)
27. Kadone, H., Nakamura, Y.: Symbolic memory for humanoid robots using hierarchical bifurcations of attractors in nonmonotonic neural networks. In: Proceedings of the International Conference on Intelligent Robots and Systems, pp. 2900–2905 (2005)
28. Keysers, C., Perrett, D.I.: Demystifying social cognition: a hebbian perspective. Trends in Cognitive Sciences 8(11), 501–507 (2004)
29. Kohlmorgen, J., Lemm, S.: A dynamic hmm for on-line segmentation of sequential data. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) NIPS 2001: Advances in Neural Information Processing Systems, vol. 14, pp. 793–800 (2002)
30. Kragic, D., Marayong, P., Li, M., Okamura, A.M., Hager, G.D.: Human-machine collaborative systems for microsurgical applications. International Journal of Robotics Research 24(9), 731–742 (2005)
31. Krueger, V., Kragic, D., Ude, A., Geib, C.: The meaning of action: A review on action recognition and mapping. Advanced Robotics 21(13), 1473–1501 (2007)
32. Kulić, D., Imagawa, H., Nakamura, Y.: Online acquisition and visualization of motion primitives for humanoid robots. In: Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (to appear, 2009)
33. Kulić, D., Lee, D., Ott, Ch., Nakamura, Y.: Incremental learning of full body motion primitives for humanoid robots. In: Proceedings of the IEEE International Conference on Humanoid Robots, pp. 326–332 (2008)
34. Kulić, D., Nakamura, Y.: Incremental learning and memory consolidation of whole body motion patterns. In: Proceedings of the International Conference on Epigenetic Robotics, pp. 61–68 (2008)

35. Kulić, D., Nakamura, Y.: Scaffolding on-line segmentation of full body human motion patterns. In: Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems, pp. 2860–2866 (2008)
36. Kulić, D., Takano, W., Nakamura, Y.: Incremental learning of full body motions via adaptive factorial hidden markov models. In: Proceedings of the International Conference on Epigenetic Robotics, pp. 69–76 (2007)
37. Kulić, D., Takano, W., Nakamura, Y.: Incremental on-line hierarchical clustering of whole body motion patterns. In: Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication, pp. 1016–1021 (2007)
38. Kulić, D., Takano, W., Nakamura, Y.: Representability of human motions by factorial hidden markov models. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, pp. 2388–2393 (2007)
39. Kulić, D., Takano, W., Nakamura, Y.: Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. International Journal of Robotics Research 27(7), 761–784 (2008)
40. Kuniyoshi, Y., Inaba, M., Inoue, H.: Teaching by showing: Generating robot programs by visual observation of human performance. In: Proceedings of the International Symposium on Industrial Robots, pp. 119–126 (1989)
41. Kuniyoshi, Y., Inoue, H.: Learning by watching: Extracting reusable task knowledge from visual observation of human performance. IEEE Transactions on Robotics and Automation 10(6), 799–822 (1994)
42. Lee, D., Nakamura, Y.: Mimesis from partial observations. In: Proceedings of the International Conference on Intelligent Robots and Systems, pp. 1911–1916 (2005)
43. Lee, D., Nakamura, Y.: Mimesis scheme using a monocular vision system on a humanoid robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2162–2167 (2007)
44. Liu, S., Asada, H.: Transferring manipulative skills to robots: representation and acquisition of tool manipulative skills using a process dynamics model. Journal of Dynamic Systems, Measurement and Control 114(2), 220–228 (1992)
45. Lockerd, A., Breazeal, C.: Tutelage and socially guided robot learning. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3475–3480 (2004)
46. Lopes, M., Melo, F., Montesano, L., Santos-Victor, J.: Abstraction Levels for Robotic Imitation: Overview and Computational Approaches. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 313–355. Springer, Heidelberg (2010)
47. Murphy, K.P.: Bayesian map learning in dynamic environments. In: Neural Information Processing Systems (1999)
48. Bagnell, J.A., Ratliff, N.D., Silver, D.: Learning to search: Functional gradient techniques for imitation learning. Autonomous Robots 27(1), 25–53 (2009)
49. Nakamura, Y., Inamura, T., Tanie, H.: A stochastic model of embodied symbol emergence. In: Proceedings of the International Symposium of Robotics Research (2003)
50. Nakamura, Y., Takano, W., Yamane, K.: Mimetic communication theory for humanoid robots interacting with humans. In: Proceedings of the International Symposium of Robotics Research (2005)
51. Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., Kawato, M.: Learning from demonstration adn adaptation of biped locomotion. Robotics and Autonomous Systems 47, 79–91 (2004)
52. Ng, A.Y., Russell, S.: Algorithms for inverse reinforcement learning. In: Proceedings of the International Conference on Machine Learning (2000)

53. Nicolescu, M.N., Matarić, M.J.: Task learning through imitation and human-robot interaction. In: Dautenhahn, K., Nehaniv, C. (eds.) Imitation and social learning in robots, humans and animals: behavioral, social and communicative dimensions. Cambridge University Press, Cambridge (2005)
54. Ogata, T., Sugano, S., Tani, J.: Open-end human-robot interaction from the dynamical systems perspective: mutual adaptation and incremental learning. Advanced Robotics 19, 651–670 (2005)
55. Okada, M., Tatani, K., Nakamura, Y.: Polynomial design of the nonlinear dynamics for the brain-like information processing of whole body motion. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1410–1415 (2002)
56. Peters, J., Schaal, S.: Reinforcement learning of motor skills with policy gradients. Neural Networks 21(4), 682–697 (2008)
57. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
58. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philosophical Transactions of the Royal Society of London B: Biological Sciences 358, 537–547 (2003)
59. Startner, T., Pentland, A.: Visual recognition of american sign language using hidden markov models. In: Proceedings of the International Conference on Automatic Face and Gesture Recognition, pp. 189–194 (1995)
60. Takano, W.: Stochastic segmentation, proto-symbol coding and clustering of motion patterns and their application to signifiant communication between man and humanoid robot. PhD thesis, University of Tokyo (2006)
61. Takano, W., Nakamura, Y.: Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. In: Proceedings of the IEEE International Conference on Humanoid Robots, pp. 425–431 (2006)
62. Tatani, K., Nakamura, Y.: Dimensionality reduction and reproduction with hierarchical nlpca neural networks. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1927–1932 (2003)
63. Williams, B.H., Toussaint, M., Storkey, A.: Modelling motion primitives and their timing in biologically executed movements. Advances in Neural Information Processing Systems 20, 1609–1616 (2008)
64. Yamane, K., Nakamura, Y.: Natural motion animation through constraining and deconstraining at will. IEEE Transactions on Visualization and Computer Graphics 9(3), 352–360 (2003)
65. Yang, J., Xu, Y., Chen, C.S.: Human action learning via hidden markov model. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 27(1), 34–44 (1997)

Can We Learn Finite State Machine Robot Controllers from Interactive Demonstration?

Daniel H. Grollman and Odest Chadwicke Jenkins

Abstract. There is currently a gap between the types of robot control polices that can be learnt from interactive demonstration and those manually programmed by skilled coders. Using regression-based robot tutelage, robots can be taught previously unknown tasks whose underlying mappings between perceived world states and actions are many-to-one. However, in a finite state machine controller, multiple subtasks may be applicable for a given perception, leading to perceptual aliasing. The underlying mapping may then be one-to-many. To learn such policies, users are typically required to provide additional information, such as a decomposition of the overall task into subtasks, and possibly indications of where transitions occur. We examine the limits of a regression-based approach for learning an FSM controller from demonstration of a basic robot soccer goal-scoring task. While the individual subtasks can be learnt, full FSM learning from this data will require techniques for model selection and data segmentation. We discuss one possibility for addressing these issues and how it may be combined with existing work.

1 Introduction

We investigate regression-based interactive **Robot Learning from Demonstration** (RLfD) for use in instantiating a soccer-style goal-scoring **Finite State Machine** (FSM) controller on a robot dog and conclude that overlaps between subtasks make the underlying regression problem ill-posed. In order to fully learn the FSM controller, additional information such as the segmentation of data into subtasks must be given. Without this information, we are limited to learning many-to-one (unimap) policies, resulting in a divide between policies that can be taught, and those that can be programmed, as visualized in Figure 1. It may, however, be possible to derive the

Daniel H. Grollman and Odest Chadwicke Jenkins
Brown University, 115 Waterman Street, Providence RI 02912-1910
e-mail: {dang,cjenkins}@cs.brown.edu

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 407–430.
springerlink.com

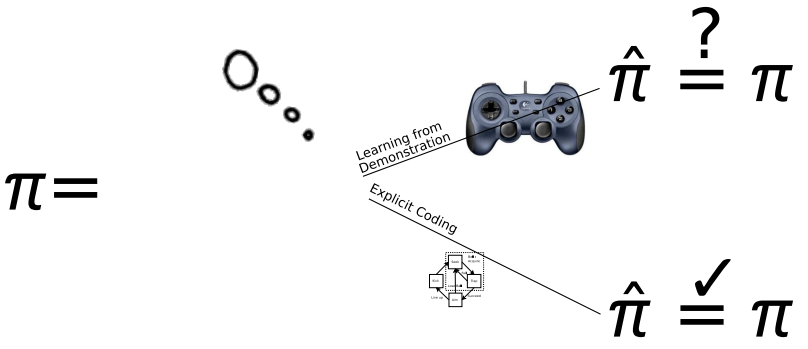


Fig. 1 An illustration of the gap between robot control policies that are currently learnable from interactive, regression based learning from demonstration, and those manually programmed by skilled coders. Particularly, finite state machine controllers may result in perceptual aliasing and cause learned behaviors to differ from those desired. Hand coded controllers can deal with this aliasing and perform correctly.

needed subtask information from the demonstration data itself, by performing both model selection (choosing the number of subtasks) and data segmentation (the allocation of data to subtasks) at the same time. Policies for these discovered subtasks and transitions between them may then be learnt with existing methods, providing for full FSM learning from collected perception-actuation tuples.

More broadly, our aim is to perform Human-Robot Policy Transfer (HRPT), the realization of decision-making control policies for robots representative of a human user's intended robot behavior. As robots and the tasks they perform become more complicated, it is possible that traditional programming techniques will be overwhelmed. Learning from Demonstration (LfD, sometimes called Programming by Demonstration or Learning by Imitation) is a technique that may enable non-programmers to imbue robots with new capabilities throughout its lifetime [31].

Formally, LfD for HRPT is the estimation of a policy $\pi(\hat{\mathbf{s}}) \rightarrow \mathbf{a}$, a mapping from perceived states to actions, from demonstrations of a desired task. Data ($\mathbf{D} = \{\hat{\mathbf{s}}_i, \mathbf{a}_i\}$) is limited to pairs of experienced perception and demonstrated actions, from which learning forms an approximated policy, $\hat{\pi}$. In the absence of a demonstrator, this learned policy can be used to control the robot.

Within RLfD, there are multiple possible approaches for actually inferring $\hat{\pi}$ from $\mathbf{D}$. One method is to assume a known parametric model for the task's control policy. Learning is then the estimation of parameters for this model from the human demonstration. Approximations in the model may be dealt with by learning additional higher-level task-specific parameters [6].

An alternative technique is to use demonstrated trajectories as constraints on an underlying reward signal that the user is assumed to be attempting to maximize, in a form of *Inverse Reinforcement Learning*. Several techniques can then be used to find a policy that maximizes this discovered reward function [35]. By modeling the

reward function directly, this approach can enable robots to perform better than their demonstrators, as in acrobatic helicopter flight [4].

In interactive learning for discrete action spaces, Confidence-Based Autonomy [12] uses classification to segment the robot's perception space into areas where different actions are applicable. This is a form of *Direct Policy Approximation*, where the policy itself is learned without intermediate steps [46]. This specific approach has been extended to consider situations where the correct action is ambiguous, but the possibilities are equivalent [11].

Here we will use the same interactive robot learning technique, or tutelage, where the user trains the robot, immediately observes the results of learning, and can incrementally provide new demonstrations to improve task performance [25]. This paradigm differs from that of batch training, where all data is gathered before learning occurs, which itself takes place offline over extended periods of time. A major advantage of tutelage is its ability to generate targeted data. That is, the demonstrator only needs to provide demonstrations of the portions of the task that the learner fails to execute properly, which become evident as the robot behaves. The user does not, therefore, need to *anticipate* what portions of the task will be difficult to learn.

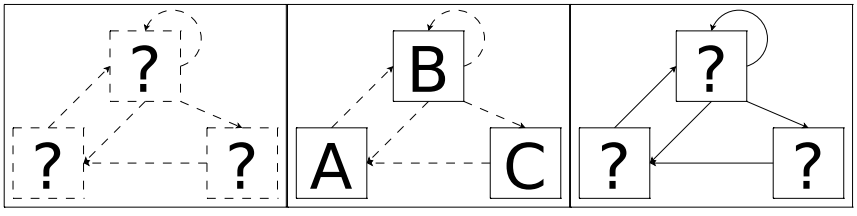
Our use of learning, to approximate the control policy, is in contrast to those which learn a model (dynamic or kinematic) of the robot itself, which can then be used for decision making [36]. We pursue learning for decision making as one difficulty of HRPT is that the true robot policy desired by the user is latent in their mind and may be difficult to express through traditional computer programming. It may further be challenging to formulate an appropriate reward function or control basis for the task itself. LfD offers a compelling alternative for HRPT, allowing users to implicitly instantiate policies through examples of task performance [18, 24].

This chapter is organized as such: This section introduced the robot tutelage paradigm, and Section 2 discusses learning tasks composed of multiple subtasks. Section 3 presents the architecture we use, Dogged Learning, and explains how learning takes place. In Section 4 we present one regression-based learning approach and show experiments in using it to learn our scoring task in Section 5. We examine the issue of perceptual aliasing in Section 6 and present a workaround for use with standard regression, along with our thoughts on a possible new approach to simultaneous model selection, data segmentation, and subtask learning. We conclude in Section 7 with an answer to our titular question.

2 Subtasks

We consider tasks formed from multiple subtasks, such as a finite state machine, where each machine state is an individual subtask. Learning a full FSM from demonstration can be divided into three aspects:

1. Learn the number of FSM states (subtasks).
2. Learn their individual policies.
3. Learn the transitions between them.



(a) Full FSM learning **(b)** Transition learning **(c)** Policy learning

Fig. 2 a) The three steps in learning an FSM. Learn the subtasks (boxes), their policies (letters), and the transitions between them (lines). Dashed lines and question marks indicate unknown quantities. Previous work has learned transitions given policies (b), or policies given transitions (c).

An optional additional step is to improve the performance of the learned controller beyond that of the demonstrator. Figure 2a represents full FSM learning.

If a set of subtask policies is given, approaches exist for learning the transitions between them. One approach is to examine the subtask activity over time, and estimate perceptual features that indicate when certain subtasks should start and end [38]. Likewise, [7] takes skills as given and learns tasks composed of subtasks from observation, learning where each subtask can be appropriately performed. We illustrate this portion of FSM learning in Figure 2b, where the subtasks themselves are known in advance, and it is only the execution order that must be learned.

An alternative to transitioning between subtasks is to combine, or *fuse* them [39]. There, outputs from each subtask are weighted by subtask applicability and combined. Alternatively, subtasks can be ordered and constrained to operate in each other's null spaces [41].

If, instead of the subtasks, the transitions between them are given, multiple subtasks can be learned at once. We illustrate this approach in Figure 2c. For example, vocal cues from the user identifying subtask transitions can be used to segment data according to a pre-determined hierarchy [56]. The segmented data can then be used to learn each subtask [45]. Layered Learning [50] goes further, and utilizes the relationships between subtasks to ease learning of the individual policies.

More similar to our work with subtasks, *Skill Chaining* [30] seeks to discover the necessary subtasks for performing a given task. That work operates in the reinforcement learning domain, and discovers a chain of subtasks, each of which has as a goal to reach the start of the next skill in the chain. They, however, use intrinsic motivation (saliency) to drive task goals, instead of user demonstration.

Combining both LfD and RL are approaches that leverage the human demonstration as a form of exploration [47]. That is, in large state-action spaces with sparse rewards, standard RL techniques may fail to converge, or require long time periods. Human demonstration can highlight areas of high reward, guiding the algorithms.

High dimensional spaces can also be dealt with using dimensionality reduction techniques. For example, after projecting human demonstrations into a latent task space, [20] estimates sequences of actions using belief propagation. There they perform learning at multiple levels, both on the actions to perform, and also to learn a dynamics model of the robot. Dynamics learning can also be coupled with RL to learn perceptually driven motion primitives [28].

Returning to combining subtasks, information about task structure can also be inferred from the order of subtask execution. For example, [40] use the order in which subtasks are utilized to build rough topological maps of the execution environment. They, however, take the task execution as given.

Here we attempt to use direct policy approximation to learn tasks, which may be formed of multiple subtasks, without providing subtask policies, manually segmenting the data, or providing a hierarchical decomposition. We seek to perform this learning in the realm of robot tutelage, which puts certain constraints on our learning algorithms.

3 Tutelage Based Robot Learning

There are several desirable qualities in a learning algorithm for robot tutelage:

1. Interactive speed: The algorithm must be able to produce a new approximate policy as training data is generated (inference), and control the robot in realtime (prediction).
2. Scalability: The algorithm should be able to handle large data sets, on the order of the lifetime of the robot.
3. Noise: The algorithm should learn in the presence of noise in perception, actuation, and demonstration.
4. Unknown mappings: There is no reason to assume that the mapping from perception to actuation is of a known form, that is, linear or parametric.

These aspects of an algorithm are interrelated. For instance, an algorithm may initially run at interactive speeds, but slow down as it scales to larger datasets. In contrast, we desire an algorithm that continues to be interactive even as the data size grows. We thus focus our consideration on incremental, sparse algorithms. Incremental in the sense that they update the approximation $\hat{\pi}$ as new data arrives, instead of recomputing it anew, and sparse in that they do not require that all previous data be kept for future consideration.

We note that the speed of a learning algorithm depends not only on its time and space complexity, but on the underlying hardware as well. That is, batch algorithms, that process all data after each new datapoint arrives and thus require that all data be stored, can be interactive, if the underlying computational and memory devices are fast enough. However, we argue that in the limit, as robots operate over longer lifetimes, the amount of data generated will overwhelm any batch algorithm with finite storage and computational power. For fixed-lifetime robots, this may not be an issue. Likewise, advances in computational and memory hardware may alleviate this problem to a certain degree.

In terms of noise, we acknowledge that a robot's sensors and actuators are inherently noisy. Thus, the control policy and its learned approximation must be robust to motors that do not do what is commanded, and world states that appear different over time. Additionally, the human demonstrator is a source of noise as well. That is, while we assume the demonstrator is *attempting* to perform the task optimally, we do not assume that the resulting commanded controls are noise free. The learning system must operate in the presence of this noise, and attempt to learn what the demonstrator means to do.

Lastly, we do not wish to assume a known model for the mapping itself, as we desire robots that can learn unknown tasks over their entire lifetime. We will thus eschew linear and parametric models in favor of nonlinear and nonparametric ones. However, by avoiding known models, we must contend even more so with noise, as it will become harder to separate signal from noise. We thus rely on a preponderance of data to enable successful learning, and put an emphasis on interpolation between observed data rather than extrapolation beyond the limits of what has been seen. Also note that the techniques we use are not without their assumptions as well. As we will see, even assuming that the mapping is many-to-one puts limits on the types of policies that can be learned.

3.1 Interactivity

As mentioned, interactive learning is a requirement for robot tutelage. That is, for robot tutelage to occur, the human demonstrator must interact with the learned autonomy in realtime, observing the approximated policy and providing additional demonstration as needed. One issue that must be addressed is the method by which the demonstrations themselves are observed. For instance, if a video feed of a human performing the task is used, there must be a system for recognizing poses and actions and determining a corresponding robot pose or action [27]. Additionally, if the robot and human have different viewpoints, their perspectives may differ and need to be matched to resolve ambiguities [9, 53].

We avoid both of these issues by using a teleoperative interface, where a user controls the robot to perform the desired task whilst observing a representation of the robot's perception. Using this sort of scenario effectively combines the accessibility of teleoperative interfaces with the autonomous behavior of explicitly programmed robots. While this setup does require some training of the user before they can demonstrate tasks, many potential users are already familiar with our interfaces from current consumer products such as remote-control toys and video games. In addition, the learning itself does not rely on the teleoperative interface. Thus, as progress is made on the above issues, new interfaces can be utilized.

By using a teleoperative interface, providing more demonstration data involves taking over physical control of the robot. Interacting with the learned autonomy in this manner can be seen as a form of sliding autonomy [14]. In particular, the autonomy of the robot is shut off entirely during demonstration, as the user makes all of the actuation decisions. An alternative would be to allow partial autonomy on the part of the robot.

When providing additional demonstration data, the user can benefit from feedback as to what the robot has learned [52]. By revealing the internal state of the learner, for instance, a teacher can more accurately understand what errors are being made [13]. Alternatively, the learner can report a confidence score, indicating how sure it is that it has learned the policy correctly, for a particular state, or overall. Such confidence measures can be used to prompt the user for more demonstration, in a form of active learning [17]. Note that this is not true active learning, as the query points can not be generated by the learning system. Instead, it can only ask for correct actuations for perceptions that naturally arise during task performance.

We can further use the idea of confidence to combine active learning and sliding autonomy by performing *Mixed-Initiative Control* (MIC) [5]. That is, we enable both the user and the autonomous policy to take and give control of the physical robot based on their confidence in their policy. When the autonomous system's confidence is low, it gives control to the user and prompts for more demonstration (active learning). Conversely, the user can take control away from the autonomous system (sliding autonomy) when desired. Likewise, the autonomous system can take control, and the user can give control. If both the user and the autonomy want control of the robot, arbitration between the control signals is necessary.

3.2 Dogged Learning

To perform robot tutelage, we use the Dogged Learning (DL) architecture, introduced in [21]. Figure 3 shows an overview of the system, which is designed to be agnostic: Applicable to a variety of robot platforms, demonstrator interfaces, and learning algorithms. Platforms are taken as sources of perception data, and sinks for actuation commands. Beyond that, the specific details of the platform's construction and low level operation are hidden.

Briefly, using data from the true state of the world ($\mathbf{s}$), the platform's sensors and perceptual processes are responsible for forming a state estimate ($\hat{\mathbf{s}}$). The generated perception is displayed to the demonstrator, whose function is to provide examples from the desired control policy mapping, or $\pi(\hat{\mathbf{s}}) \rightarrow \mathbf{a}$. The platform is assumed to transform $\mathbf{a}$ into appropriate actuator commands and bring about a change in the world, in tandem with the environment. The paired perception-action tuples resulting from demonstration are used to train the learning system, which forms an

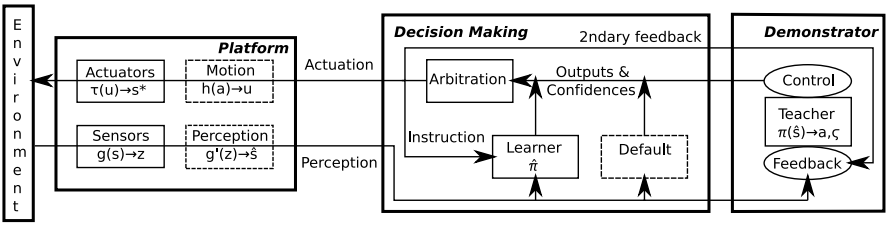


Fig. 3 The Dogged Learning architecture. It abstracts out the details of the robot platform, the user interface, and the learning algorithm.

approximation of the policy, $\hat{\pi}$. As mentioned, the learning system always maintains a current estimate, and updates it as new data arrives. It can thus be thought of as performing the mapping $(\hat{\pi}_t, \hat{\mathbf{s}}_t, \mathbf{a}_t) \rightarrow \hat{\pi}_{t+1}$. If this update is performed incrementally, data can be discarded after each update, save for that needed by $\hat{\pi}$ itself.

Note that the definition of a platform does not require any particular physical embodiment. It can, for instance, be a simulated robot, or some other purely software agent. Additionally, it can be multiple physical robots as well, whose perception and action spaces are combined.

The demonstrator itself, like the platform, is defined abstractly. It can be any system that provides task-correct actuation, given a perception. In many cases, this could be a human user, who is seeking to train a robot to perform some new task. However, it could also be a hand-coded control system, for testing purposes, or another robot, for multi-robot task-transfer.

Once an approximate policy is learned, it can be used to control the robot platform in place of the demonstrator. However, as mentioned, we must arbitrate between the two controllers if they both attempt to control the platform at the same time. We thus require that both the demonstrator and learner return confidence values (ς) in addition to $\mathbf{a}$. Using these confidences, we give control to the more confident controller. Additionally, we achieve active learning by providing a default controller, that stops the robot and requests more demonstration, if the learner is acting and its confidence is below a threshold.

Information internal to the learner, such as the current confidence value, is presented back to the user as a form of secondary feedback. This data is in addition to the perceptual information of the platform itself and can help guide the teaching process. Additionally, because the learning system, platform, and demonstration system are disjoined, and all interaction between them mediated by the DL framework, the actual modules need not be co-located. One possibility is then that demonstration control can take place distally from the robot itself, perhaps over the internet. By enabling remote users to control robots, we may be able to gather more data, and learn more robust behaviors.

4 Regression-Based Learning

Regression is the estimation of a mapping $f(x) = y$ from known input-output pairs $\{x_i, y_i\}, i \in 1:N$. Often x and y are continuous variables, as classification techniques may be more appropriate if they are discrete. Regression techniques can be divided into two groups: Parametric approaches, which take as given the form of the target mapping, and nonparametric ones, which do not.

A common parametric technique is to fit a known polynomial form to the data, such that $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_d x^d$ up to the order of the desired polynomial, d. If $d = 1$, linear regression is achieved. The fit can often be performed by minimizing the squared error between predicted outputs and known outputs, which is equivalent to assuming that the observed data comes from the model

$$y_i = f(x) + \mathcal{N}(0, \sigma^2)$$

where observed outputs are distributed in a Gaussian fashion around the true output. Such an approach is termed *Least Squares* regression. More generally, approaches that assume that observed outputs are unimodally distributed perform estimation of a many-to-one mapping, which we term a unimap. This distinction serves to identify the class of maps that we will encounter later, the multimaps, where an individual input may have more than one appropriate output.

Nonparametric approaches, on the other hand, do not take as given a form for the mapping. That is not to say they do not have parameters, instead nonparametric methods can be thought of as those where the data itself are the parameters, or the parameterization grows with the data. One example would be if the degree of the polynomial model above were to grow with the data, such that $d = N$. Nonparametric models thus grow in complexity with the data, but also run the risk of overfitting.

Given that our data, $\mathbf{D}$, is continuous and of unknown form, we consider nonparametric regression. Of course, as the parameterization can grow with the data, an algorithm may require unlimited data storage. We must then focus on approaches that explicitly limit the growth of the parameterization, or *sparse* nonparametric regressors, that do not require all previous data to make a prediction. Even within this subset of techniques, there are many possible methods for learning $\hat{\pi}$ in an interactive, scalable, robust approach.

We initially consider Locally Weighted Projection Regression (LWPR) [54]. LWPR is a local approximator, in that it fits the overall mapping by dividing the input space into multiple, overlapping regions called receptive fields (RF). Each RF performs a local projection and linear regression on the data in that region, and predictions from multiple RFs can be combined to form the total system's output. LWPR is sparse in that only the sufficient statistics for the RFs need to be kept, so that once a datapoint has been incorporated, it can be discarded. Incorporation of new data (inference) is incremental, through the use of partial least squares regression in each RF, and an incremental update to the area of the RFs themselves. LWPR has the added benefit of explicitly considering that there may be unnecessary dimensions in the data, and seeks to project them out.

Other possible regression algorithms that have been used for learning robot control include K-Nearest Neighbors [49], Neural Nets [51], and Gaussian Mixture Regression [10]. Herein we will consider the global approximator Sparse Online Gaussian Processes (SOGP) [15] for illustrative means, although other algorithms mentioned above obtain similar results. In particular, all will exhibit the multimap error due to perceptual aliasing that we will see below.

4.1 SOGP

Gaussian Process Regression (GPR) is a popular statistical nonparametric regression technique, a review of which can be found in [32]. Briefly, a Gaussian Process is a Gaussian distribution over functions defined by a mean and covariance function, $\mathcal{N}(f, \boldsymbol{\Sigma})$. Starting with a mean zero prior, and given a set of data $(\mathbf{X}, \mathbf{y}) = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, we first define a kernel function which represents similarity between two points in

input space. A popular kernel that we use in our work is the squared exponential, or Radial Basis Function (RBF):

$$\texttt{RBF}(\mathbf{x},\mathbf{x}';\sigma_k^2) = \exp\left(-0.5 * \frac{||\mathbf{x}-\mathbf{x}'||^2}{\sigma_k^2}\right)$$

σ_k^2 is termed the kernel width, and controls how strongly nearby points interact.

A posterior distribution over functions is then defined where

$$f(\mathbf{x}') = \mathbf{k}_{\mathbf{x}'}^\top \mathbf{C}^{-1}\alpha$$

$$\Sigma(\mathbf{x}') = k^* - \mathbf{k}_{\mathbf{x}'}^\top \mathbf{C}^{-1}\mathbf{k}_{\mathbf{x}'}$$

where $\mathbf{k}_{\mathbf{x}'}$ is shorthand for $[k_1,k_2...k_N], k_i = \texttt{RBF}(\mathbf{x}',\mathbf{x}_i;\sigma_k^2)$, or the kernel distance between the query point and all previously seen input points $\mathbf{X} = \{\mathbf{x}_i\}_i^N$. α is the output vector, which in this case equals the known data's outputs, $\mathbf{y}$. $\mathbf{C}$ is the covariance matrix of the data, where $\mathbf{C}_{ij} = \texttt{RBF}(\mathbf{x}_i,\mathbf{x}_j) + \delta(i == j)\sigma_0^2$. This second part represents observation noise (modeled as a Gaussian with mean 0 and variance σ_0^2). Without it we would have just the Gram matrix ($\mathbf{Q}$), or all-pairs kernel distance.

As $\mathbf{C}$ occupies $O(N^2)$ space and requires $O(N^3)$ time to invert, GPR is not directly suitable for our learning scenario. Using the partitioned inverse equations, $\mathbf{C}^{-1}$ can be computed directly and incrementally as data arrives, removing the inversion step. The space requirements must be dealt with separately using one of a variety of approximation techniques [43].

One sparsification technique is to divide the input space into regions, each of which is approximated by a separate "local" GP [37]. By limiting the area of expertise of each GP, the overall covariance matrix is made block-diagonal, and can be stored in a smaller space. This approach to approximation recognizes that the effects that two datapoints that are far away from each other have on each other is nearly zero, and can often be ignored. A major issue that must be addressed when using these techniques is determining the number and location of the local GPs. Ad-hoc approaches such as tiling the input space or using a threshold distance may result in the creating of unnecessary local models [48].

Alternative techniques that retain the global nature of the GP approximation do so by replacing the full posterior distribution based on N points with an approximation based on fewer, $\beta < N$. These fewer points are called the basis set, or the basis vectors (BV). This reduction limits the size of the Gram matrix to β^2, which can be tuned for desirable properties, such as speed of computation or percentage of system memory used, for each particular implementation. The approximating distribution itself is chosen optimally to minimize the KL-divergence with the true distribution.

The Sparse Online Gaussian Process (SOGP) algorithm proposed by [16] performs this minimization incrementally, which makes it very suitable for our needs. In essence, when the $\beta + 1$ point arrives, it is initially included as a basis vector and then all points are assigned a score corresponding to the residual error between the distribution based on all points and the distribution based on all points except this

Algorithm 1. Sparse Online Gaussian Processes

Inference	**Prediction**
Require: Training pair $(\mathbf{x}, y)$	**Require:** Query point $(\mathbf{x})$
Basis Vectors (BV), model $(\alpha, \mathbf{C}^{-1}, \mathbf{Q})$	Basis Vectors (BV), model $(\alpha, \mathbf{C}^{-1})$
GP parameters $(\theta = \{\sigma_k^2, \sigma_0^2\})$	GP parameters $(\theta = \{\sigma_k^2, \sigma_0^2\})$
capacity (β),$\lvert\text{BV}\rvert < \beta$	capacity (β),$\lvert\text{BV}\rvert < \beta$
Ensure: Updated model and BV, $\lvert\text{BV}\rvert < \beta$	**Ensure:** predicted output $\hat{y}$, stddev σ^2
	for $b = 1 : \lvert\text{BV}\rvert$ **do**
add $\mathbf{x}$ to BV and update $\alpha, \mathbf{C}^{-1}\mathbf{Q}$	$k_b = \text{RBF}(\text{BV}_b, \mathbf{x}'; \sigma_k^2)$
if $\lvert\text{BV}\rvert > \beta$ **then**	$k^* = \text{RBF}(\mathbf{x}', \mathbf{x}'; \sigma_k^2)$
for $b = 1 : \lvert\text{BV}\rvert$ **do**	$\hat{y} = \mathbf{k}^\top \alpha$
$\varepsilon_b = \alpha_b / \mathbf{C}^{-1}_{b,b}$	$\sigma^2 = \sigma_0^2 + k^* - \mathbf{k}^\top \mathbf{C}^{-1} \mathbf{k}$
Delete j from BV, $j = \text{argmin}_j \varepsilon_j$	

one. The point with the lowest score is selected for removal. An overview of the algorithm can be seen in Algorithm 1. The parameters $(\sigma_0^2, \sigma_k^2, \beta)$ can be chosen in multiple fashions, but we choose $\beta = 300$ to ensure realtime performance on our system, and set $\sigma_0^2 = \sigma_k^2 = 0.1$ using a separate test data set.

Note that in switching from the full GP to an approximation, we must make a distinction between the output vector α and the data's outputs $\mathbf{y}$. Similarly, $\mathbf{C}$ now no longer tracks $\mathbf{Q} + I\sigma_0^2$ and the two must be stored separately. By doing so, information from points not currently in the BV (because they had been deleted) can still be used to approximate the distribution. This information is not found in $\mathbf{y}$ or $\mathbf{Q}$, which depend only on the basis vectors. While now two matrices are now stored, they are both of size β^2, so total memory usage is still $O(\beta^2)$.

It should be noted that all the discussion of GPs in this section is with respect to scalar outputs. We apply these techniques to vector outputs by providing a interdependence matrix, or, as we do here, assuming independence between the outputs. This assumption, while almost always false, often provides good results, and has done so in our case. New techniques may enable us to improve these results further by learning the dependencies between outputs [8].

5 Experiments

We seek to learn a RoboCup swarm-team style goal scorer policy on a robot dog, such as shown in Figure 4. This task has four distinct stages: The robot first approaches the ball, then uses its head and neck to trap the ball close to the body, then turns with the ball towards the goal, and finally checks that the goal is lined up correctly and kicks. To minimize noise and enable repeatable data generation, we used a hand-coded controller to demonstrate this task to the learning system and used SOGP to learn from interactive demonstration using the DL architecture. However, initial experiments in learning this task were unsuccessful, and we instead considered each of the stages in turn, in an attempt to 'build up to' the complete task.

(a) Seek **(b)** Trap **(c)** Aim **(d)** Kick

Fig. 4 Desired Goal-scoring behavior.

5.1 Platform

Our platform, pictured in Figure 5, is a commercially available robot platform, the Sony AIBO robot dog. We have equipped it with a rudimentary vision system, consisting of color segmentation and blob detection. That is, all perceived colors are binned into one of six categories (black, orange, blue, yellow, green, white) and each color is treated as a blob. The x and y locations (in image coordinates) and blob size (pixel count) of each color serve as input to our learning system. In addition, we take as input the motor pose of the four motors in the head (tilt, pan, neck and mouth) and two of the tail (pan and tilt), for a total of 24 dimensions of noisily observed perception (§). Note that we do not use any of the touch or distance sensors on the robot as inputs.

Our platform also has a basic walk gait generation system. Previous work [22, 29] has learned this walk gait directly on the motors using both parametric and nonparametric regression, so here we take it as given. Taking in a desired walk speed in the lateral and perpendicular directions, as well as a turning rate, it generates leg motor positions for the robot. These 3 parameters, along with new pose information

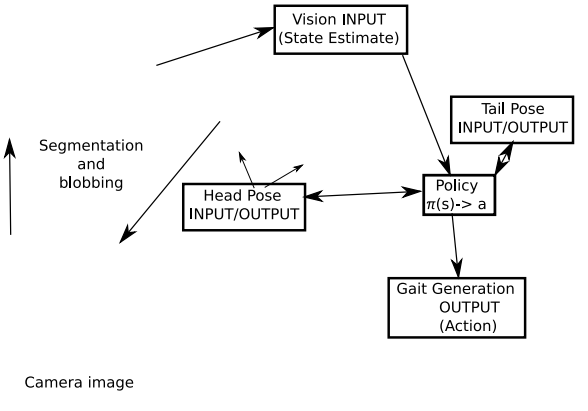


Fig. 5 Our robot platform, a Sony AIBO equipped with rudimentary vision (color segmentation and blob detection) and walk-gait generation. Control policy estimation involves approximating the observed policy mapping from state estimate INPUTs to action OUTPUTs.

Table 1 Platform perceptual and action spaces.

PERCEPTION	ACTUATION
24D: 6 Colors × 3D (image X, Y, and size) + 4 head motors (pan, neck tilt, chin tilt, mouth) + 2 tail motors (pan, tilt)	10D: Walk Parameters (X, Y, and α) + 4 head motors + 2 tail motors + kick (discrete)

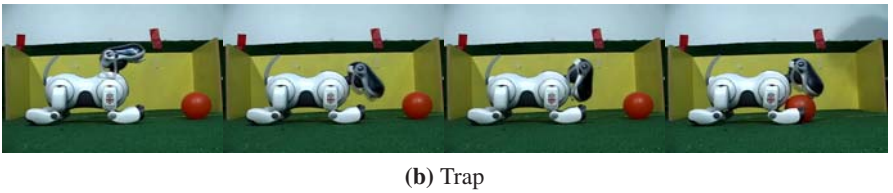
for the head and tail motors, plus a kick potential, form the 10 dimensional action output space (**a**). When the kick potential rises above 0.5, a prerecorded kicking motion is performed. A summary of all inputs and outputs is shown in Table 1. All dimensions are normalized to lie in the range [-1,1].

5.2 Learned Behaviors

We show here learning of the first two stages of the goal-scoring task presented in Figure 4. The first, the seek stage (Figure 6a) involves locating and approaching the ball. We initially use a hand-coded controller that causes the robot to rotate until it sees the ball, and then walk towards the ball, rotating to keep it centered as needed. As the robot approaches the ball, its head dips down to keep the ball in view, and the forward speed reduces until the robot comes to a stop. Using interactive teaching, a human user toggles the activity of this controller, alternating between providing demonstration data to the learner, and observing the learned policy in control of the robot. During interaction, the robot and ball were randomly reset whenever an approach was complete. Interaction ended and was deemed successful after the robot performed 10 approaches without requiring further demonstrations.

The trap stage, shown in Figure 6b, was taught in a similar manner. However, as it is not a mobile task, the ball was alternatingly placed in and removed from the trap location by the user. In addition to the hand coded controllers, both behaviors

(a) Seek



(b) Trap

Fig. 6 The learned trap and seek subtasks of the goal-scoring behavior.

Table 2 MSE of SOGP and LWPR on the trap and seek stages.

	SOGP	LWPR
Seek	0.0184 ± 0.0127	0.1978 ± 0.0502
Trap	0.0316 ± 0.0165	0.0425 ± 0.0159

were taught using human teleoperation as demonstration as well. In this scenario, the human demonstrator used a two joystick control pad to provide demonstration, which was used for learning. Again, interactive tutelage was used, and deemed a success after 10 autonomous executions of the task.

For quantitative results, we collected a further 5 examples of each stage using the hand-coded controller, and calculated the mean squared error of the code-taught learned controller on the perception inputs with respect to the hand-coded controller. The results are shown in Table 2, compared with those when using LWPR. Results are shown averaged over all 5 test sets with one standard deviation.

6 Perceptual Aliasing from Subtask Switching

Despite our ability to learn the first two component stages of our goal-scoring behavior, we are unable to learn their composition. We believe this to be an issue of *perceptual aliasing*, where perceptions that are similar require different actions. We consider three common sources of perceptual aliasing:

1. Equivalent Actions: There are multiple actions that accomplish the same effect, and user variance causes them all to appear in the collected data.
2. Hidden State: Information not visible to the robot is used by the demonstrator to select the correct action to perform.
3. Changed objective: The goal of the behavior being performed has changed, meaning that what was appropriate for a given perception is no longer so.

These three sources of aliasing are related, and can be seen as variations of one another. For example, user generation of equivalent actions can be seen as relying on hidden state, the preference for a particular action on the part of the user. Over many users, or even over time in the same user, this preference may differ.

Likewise a change in objective can be viewed as moving the robot to a different area of state space, but one that just looks the same to the robot. That is, if the true state space of the robot included information about the objective of the task, the robot would occupy different portions of the state space when the objective changed. As the robot cannot observe this objective information, it is hidden state.

A hidden state framework may thus be suitable for addressing all types of perceptual aliasing, as both user preferences and subtask goal can be thought of as unseen state. The POMDP model [42] is such a framework, where perceived states are mapped into beliefs over the true state space and those beliefs are used for action selection. However, POMDP approaches often take as given knowledge of the true state space, and only assume incomplete perception of it. They are thus inappropriate

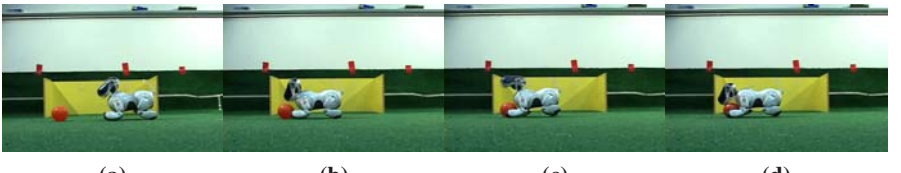
for when the underlying state space is truly unknown. This situation occurs in the first and third types of perceptual aliasing discussed above, as the number of equivalent actions or subtasks is unknown.

Another method that may be generally applicable to perceptual aliasing and does not require full knowledge of the state space is to consider a history of past perceptions and actuations, instead of just the current ones. Knowledge of where the robot just was, and what it did may help resolve aliased states. Using history, [33] learns the true state space over time in a POMDP model. It does this by considering the differences in reward achieved per state-action pair over time, and splits states when it determines that history matters. However, only a finite amount of history is considered at any given time. For truly ambiguous, equivalent actions, it may be that no amount of history will serve to uniquely identify the appropriate one. More generally, for unknown tasks, we do not know how much history will be necessary.

6.1 Analysis

We here analyze the combination of the first two stages of the goal scoring task and see how it results in perceptual aliasing. Specifically, it is perceptual aliasing of the third kind, when a change in objective leads to multiple actions being recorded for the same perception. We can perform learning then by either treating it as an issue of hidden state, and making that state observable, or reworking the overall task to avoid ambiguous situations.

The first half of the goal scoring task shown in Figure 4 is the combination of the previously learned subtasks, and involves approaching and acquiring control of the ball. We call this the ball-acquire (AQ) task and it consists of the robot first locating and walking to the ball, and then trapping it for further manipulation, as shown in Figure 7. Using a hand-coded controller, after 10 minutes of interactive training the learned autonomy was unable to successfully acquire control of the ball. As both of the constituent stages were successfully learned in the previous section, we posit that it is the *overlap* between these two tasks occurring at the transition point in Figure 7c that causes issues. This belief is further borne out by the observed behavior of the learned autonomy. Instead of waiting until the ball was in the correct position to trap, the robot instead performs portions of the trap randomly, whilst seeking the



(a) **(b)** **(c)** **(d)**

Fig. 7 The Ball-Acquire (AQ) task. The input state at the transition point in (c) is ambiguous, as the ball is not visible. The robot can either be looking for the ball (a) or trapping it (d). A context variable, indicating which subtask is being performed, removes the confusion.

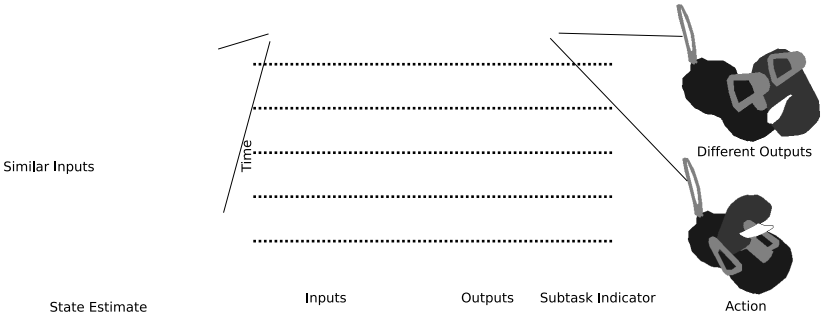


Fig. 8 Data from the seek/trap transitions of the AQ task. Raw inputs and outputs, with extraneous dimensions removed, are shown in the middle, as is the true subtask indicator (light is seek, dark is trap). Six subtask transitions are shown. One datapoint on either side of the first transition is highlighted and shown graphically. State estimates (head pose and perceived ball location) are on the left, and actions (commanded head pose [walk velocities are zero]), on the right.

ball. Thus we can infer that certain perceptual states, which occur during seeking, are also associated with trapping, and that the learning system cannot distinguish what action to perform.

To more closely examine this hypothesis, we focus on data from the transition between seeking and trapping in the AQ policy, where we expect to find a large amount of overlap. Figure 8 (center) shows raw data from around this transition, with extraneous dimensions such as non-ball color blobs and tail position removed. We highlight one transition and examine more closely the data on either side, when the controller has switched from performing the seeking subtask to the trapping subtask. On the left we show the perception inputs, with the head and ball positions. Note that both states have very similar inputs. On the right we show the corresponding outputs. When seeking, the controller keeps the head down, and the walk parameters are zero. When trapping, the walk parameters are still zero, but instead the head is raised (to initiate the trap).

In our unimap regressor, there is an implicit assumption that data that is similar in input space is similar in output space as well (continuity, or smoothness). This assumption is formalized by the squared exponential, or radial basis function, repeated here for reference

$$\mathrm{RBF}(\mathbf{x},\mathbf{x}';\sigma_k^2) = \exp\left(-0.5 * \frac{||\mathbf{x}-\mathbf{x}'||^2}{\sigma_k^2}\right)$$

which computes the similarity between two datapoints, and weighs their outputs for prediction. We see that for the two points on either side of the transition, the RBF measure is 0.9978. SOGP then assumes that the outputs would be similarly similar, which is not the case. Instead, the outputs have a measure of 0.1295 between them. Because the two inputs are so similar, SOGP (and other standard regression

algorithms) assume that the observed outputs are noise-corrupted versions of the true output, and approximate them by their average, resulting in incorrect behavior.

6.1.1 Subtasks as Hidden State

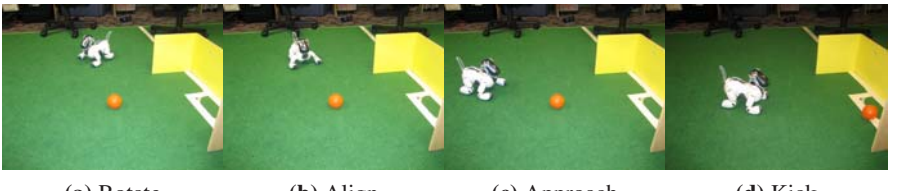
Let us, as discussed above, frame subtask switching as an issue of hidden state. As there are only two subtasks, a single binary indicator variable will serve to differentiate them. With this variable included in the perceptual space of the robot, the similarity between the input points becomes 0.4866. Thus, this indicator serves to effectively 'move' the two subtasks apart in perception space. As the inputs are now less similar, their associated outputs are not combined as heavily, and standard regression can learn to differentiate them.

Using this indicator variable, we are able to learn the AQ task successfully from a hand-coded controller. The indicator variable in this instance came from the internal state of the code itself. If human demonstration were used, there would need to be a method to get this information from the user during demonstration. Note that doing so would require the user to first analyze the task to determine subtasks, and then provide the indicator variable in addition to commanded actuations.

Using this information is, unfortunately, a stop-gap measure. In general, we are interested in learning various unknown tasks from demonstration. Because we do not know the tasks or their structure beforehand, we cannot a priori determine the number of needed subtasks. Further, requiring the user to perform both analysis and labeling during demonstration may unnecessarily limit our user base.

6.1.2 Unambiguous Goal Scoring

With some creativity, the goal scoring behavior can be reformulated to remove perceptual aliasing. The new policy, shown in Figure 9, has the robot first circle the ball until it is lined up with the goal, then approach it and kick. As the ball and the goal are maintained in view at all times, there is no ambiguity as to what should be done. Using both a hand-coded controller and human teleoperative demonstration, this policy was learned successfully. Again, the robot and ball were repositioned



(a) Rotate **(b)** Align **(c)** Approach **(d)** Kick

Fig. 9 An alternate goal scorer policy where the robot maintains sight of the ball and goal. Actions are thus uniquely determined by the perception, and standard unimap regression can learn the task.

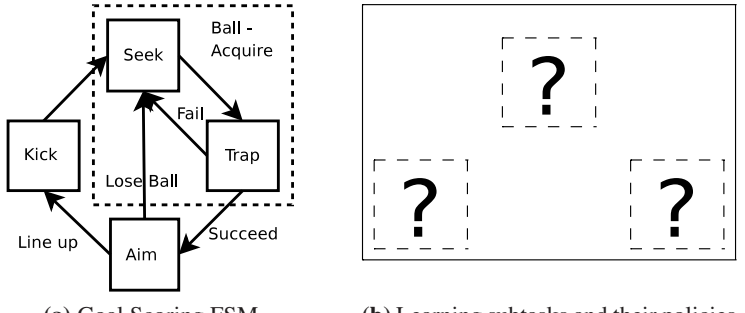
randomly after each goal, or when the system entered an unstable state (ball behind the goal or stuck). Interaction was terminated after 10 autonomous scores.

This approach to scoring, however, is less desirable than our original formulation. For one, it is slower, and also requires more open field to operate in. To learn our original policy, we must be able to learn π in the presence of perceptual aliasing. However, as we have seen, standard regression techniques are unable to learn such mappings. We require an approach that can automatically derive subtasks from the data, and learn each subtask policy individually.

6.2 Finite State Machines from Demonstration

We claim that learning general FSMs from interactive demonstration is an open problem for RLfD. Looking again at our original goal scoring policy in Figure 4, we reshow it as a finite state machine in Figure 10a, where the AQ task is a subgraph within this overall behavior. From our previous discussion we see that using regression to learn directly from data consisting of examples from two or more subtasks may result in perceptual aliasing and improper autonomous behavior. In terms of the underlying mapping, we believe it is multimap scenarios caused by perceptual aliasing due to a change in objectives that limits the applicability of standard regression for direct policy approximation.

Recall from Section 2 that approaches exist for learning to transition between known subtasks to perform a desired behavior. In terms of goal scoring this scenario would correspond to knowing in advance the subtasks of seek, trap, kick and aim. Likewise, there are techniques for learning subtasks given the decomposition. For our goal-scoring task, this approach would be equivalent to knowing that there are four subtasks, and the shape of the transitions between them. One method for learning in this scenario would be to include the subtask indicator, as we did above and in [22]. This variable effectively tells us the number of subtasks (by the different values it takes on), and the transitions between them (by when it changes).



(a) Goal Scoring FSM **(b)** Learning subtasks and their policies

Fig. 10 The goal-scoring policy as a finite state machine. Current LfD systems can learn transitions between known policies, we propose a technique for learning the number of subtasks and their individual policies.

Both of the above approaches learn FSMs, but depend on specific knowledge in addition to the demonstration data, and use that information to resolve perceptual aliasing. If the possible subtasks are known, multiple actions for a given perception can be recognized as coming from different subtasks, and therefore will not cause confusion during learning. Likewise, if the transitions are known, the multiple actions can be assigned to different subtasks, and then used to learn different policies.

What we seek is an approach that derives the assignment of data to subtasks, the subtask policies, and the transitions between them from the data itself. In particular, as the number of subtasks is not known a priori, it must too be deduced. As learning transitions between known subtasks is already feasible, we focus on the first two of these inferences, as illustrated in Figure 10b.

One approach to learning the number of subtasks is to take an upper limit on that number, and allow some of the learnt subtasks to be null. This is the approach used, for example, by [40], and when using the Bayesian Information Criterion (BIC) to fit finite mixture models [23]. However, for long-life robots, who learn multiple tasks over their existence, this may not be a valid approach, as the total number of subtasks may grow without bound. In addition, we argue that for such a robot it may be more appropriate to say something about the *rate* at which subtasks are generated, rather than their number.

We thus propose to use multimap regression to learn from demonstration data with multiple overlapping subtasks. This approach, described below, would take in unannotated perception-actuation data, and produce an assignment of each datapoint to a subtask, the total number of which is not known a priori. Using this derived annotation, individual subtask policies can be learned using standard LfD approaches such as SOGP regression discussed above. Further, the annotations may provide the necessary transition information for use in inferring the overall FSM topology.

6.2.1 Multimap Regression

Multiply valued mappings, or multimaps, refer to mappings $f(x) \rightarrow y_1, y_2 \ldots y_n$ which are one-to-many. For a given input, x, there may be multiple possible "correct" outputs. This scenario is exactly what occurs during perceptual aliasing, and leads to ill-posed problems for standard regression.

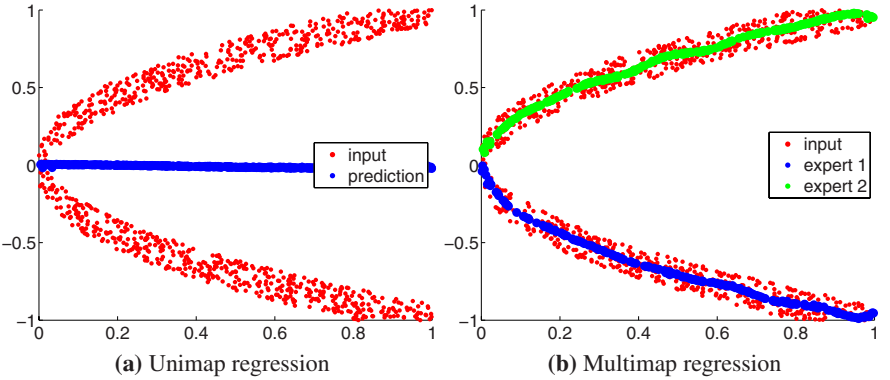
Probabilistically, the shift from unimaps to multimaps can be thought of in terms of the distribution of observed actions given a perception, $P(\mathbf{a}|\hat{\mathbf{s}})$. In unimap regression, this distribution is taken to be unimodal (Gaussian in SOGP). Thus, all observed demonstrations for a given input are averaged to create the predicted action. In multimaps, the distribution is instead multimodal, with one mode corresponding to each of the possible subtasks that may be applicable at $\hat{\mathbf{s}}$.

As an illustrative multimap example consider the squareroot mapping, $\sqrt{x}$ shown in Figure 11. Here, the same input $x = 4$, can lead to two different outputs, denoted as $\sqrt{4} \rightarrow \{2, -2\}$. Standard regression techniques applied to data from this mapping average the possible outputs, as seen in Figure 11a. What we seek is a multimap regression algorithm which can detect the underlying multimodality of the output distribution such as in Figure 11b.

We advocate an approach to multimap regression that models a multimap as a collection of unimaps, where each subtask (mode of the distribution) is a separate unimap. Multimap regression then becomes an issue of model selection (determining the number of unimaps), which corresponds to identifying the number of subtasks, and individual standard regression for each one, which corresponds to policy learning for each subtask. Further, as we do not know the number of subtasks in advance, we must be able to create new subtasks as needed.

A mixture of experts approach [26] is thus appropriate, where each expert is a unimap subtask. As we have already used SOGP to learn individual subtasks, a mixture of SOGP experts would use a separate SOGP expert to learn each subtask, and then transition between them to perform the overall behavior. The infinite mixture of Gaussian experts model [34, 44] provides a method for automatically inferring the number of GP experts needed to approximate a sample multimap. Further, by using a prior over the segmentation of data, they introduce bias into the rate at which subtasks are formed. The approach presented is, however, batch, and thus requires all data to be collected in advance of processing. A similar, incremental approach [55] was used to learn the multimap in Figure 11b. Further sparsification, by using SOGP experts for example, may enable the application of this technique to realtime robot tutelage.

Basically, for each datapoint, the infinite mixture of GP experts model computes the likelihood of that datapoint being generated by every existing GP expert, as well as a new, previously unseen one. The actual assignment used is drawn from this distribution, and multiple possibilities are tracked to approximate the full distribution over assignments of data to experts via Monte-Carlo integration. The result of the algorithms (both batch and incremental) is a partition of the data, or set of indicator variables that indicates which expert gave rise to each datapoint. Because new, empty, experts are always considered, the number of experts which have data associated with them can be as great as the number of datapoints. The individual experts themselves are standard GP regressors, and can be used to generate control signals for the robot.



(a) Unimap regression **(b)** Multimap regression

Fig. 11 The square root, a multimap. Unimap regression averages the two outputs, while multimap regression learns a separate unimap for each branch.

These techniques, however, only infer the number of subtasks and the policy associated with each one, but not the transitions between them. There is then the issue of selecting which should be used during robot execution, when more than one is applicable. Randomly selecting a subtask for execution is likely to result in improper robot behavior as there is no consistency in time. That is, the robot may rapidly oscillate between subtasks, performing nither. For temporal continuity, [19] learn an infinite hidden Markov model that "sticks," or favors self-transitions. It may be possible to merge infinite mixtures of experts, sparse nonparametric regression, and sticky HMMs into a system that determines the number of experts, assigns data to them and learns their policies, and infers the necessary transition matrix between them for task performance, all at interactive speeds.

7 Conclusion

Interactive robot learning from demonstration is a promising approach to enable autonomous robot control policies to be instantiated without programming. One method for doing so is to use regression to directly approximate the policy. However, in our experiments in learning a robot soccer goal-scorer we have observed that finite state machine controllers may result in ill-posed regression problems, or multimap scenarios. Specifically, perceptual aliasing can occur when multiple subtasks with different objectives overlap in perception space. In light of the FSM formulation of our policy, we cast tutelage as FSM inference and view learning as estimating the number of subtasks, their policies, and the transitions between them. Current state-of-the-art techniques exist for inferring individual subtasks, the transitions between them given subtasks, or multiple subtasks simultaneously given the transitions. Estimating the number of subtasks can be viewed as a problem of multimap regression, and infinite mixtures of expert models may be bought to bear.

While current techniques can perform multimap regression, in order to fully apply them to the robot tutelage scenario for FSMs, we require incremental, sparse versions that can discover subtasks in realtime. Further, we need to incorporate some method of inferring the transitions between the subtasks, so that the correct action, of the multiple applicable ones, is executed. With these added capabilities, it may be possible to infer a complete FSM directly from unsegmented demonstration data and use that FSM to control the robot immediately after demonstration ceases.

The title of this chapter is a question: Can we learn FSM robot controllers from interactive demonstration? We have only examined regression based approaches, but within that domain we optimistically conclude "No, not yet."

References

1. Snelson, E., Ghahramani, Z.: Sparse Gaussian Processes using Pseudo-inputs. In: NIPS, Vancouver, CAN, December 2006, pp. 1257–1264 (2006)
2. Dasgupta, S., Hsu, D., Monteleoni, C.: A general agnostic active learning algorithm. In: NIPS, Vancouver, CAN, December 2007, pp. 353–360 (2007)

3. Calinon, S., Billard, A.: A Probabilistic Programming by Demonstration Framework Handling Constraints in Joint Space and Task Space. In: IROS, Nice, France, September 2008, pp. 367–372 (2008)
4. Abbeel, P., Coates, A., Quigley, M., Ng, A.Y.: An application of reinforcement learning to aerobatic helicopter flight. In: Neural Information Processing Systems [1], pp. 1–8
5. Adams, J.A., Rani, P., Sarkar, N.: Mixed initiative interaction and robotic systems. Technical Report WS-04-10, Vanderbilt University (2004)
6. Atkeson, C., Schaal, S.: Robot learning from demonstration. In: International Conference on Machine Learning, Nashville, TN, July 1997, pp. 12–20 (1997)
7. Bentivegna, D.C., Atkeson, C.G., Cheng, G.: Learning tasks from observation and practice. Robotics and Autonomous Systems 47(2-3), 163–169 (2004)
8. Bonilla, E.V., Chai, K.M.A., Williams, C.K.I.: Multi-task Gaussian process prediction. In: Neural Information Processing Systems [2], pp. 153–160
9. Breazeal, C., Berlin, M., Brooks, A.G., Gray, J., Thomaz, A.L.: Using perspective taking to learn from ambiguous demonstrations. Robotics and Autonomous Systems 54(5), 385–393 (2006)
10. Calinon, S., Billard, A.: A probabilistic programming by demonstration framework handling constraints in joint space and task space. In: International Conference on Intelligent Robots and Systems [3], pp. 367–372
11. Chernova, S., Veloso, M.: Learning equivalent action choices from demonstration. In: International Conference on Intelligent Robots and Systems [3], pp. 1216–1221
12. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. Journal of Artifical Intelligence Research 34(1), 1–25 (2009)
13. Colombetti, M., Dorigo, M.: Training agents to perform sequential behavior. Adaptive Behavior 2(3), 247–275 (1994)
14. Crandall, J.W., Goodrich, M.A.: Experiments in adjustable autonomy. In: International Conference on Systems, Man, and Cybernetics, Tuscan, AZ, October 2001, pp. 1624–1629 (2001)
15. Csató, L.: Gaussian Processes - Iterative Sparse Approximations. PhD thesis, Aston University (March 2002)
16. Csató, L., Opper, M.: Sparse Online Gaussian Processes. Neural Computation 14(3), 641–669 (2002)
17. Dasgupta, S., Hsu, D., Monteleoni, C.: A general agnostic active learning algorithm. In: Neural Information Processing Systems [2], pp. 353–360
18. Dillmann, R., Rogalla, O., Ehrenmann, M., Zöllner, R.D., Bordegoni, M.: Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm. In: International Symposium of Robotics Research, Snowbird, Utah, USA (October 1999)
19. Fox, E.B., Sudderth, E.B., Jordan, M.I., Willsky, A.S.: An HDP-HMM for systems with state persistence. In: International Conference on Machine Learning, Helsinki, Finland, July 2008, pp. 312–319 (2008)
20. Grimes, D.B., Chalodhorn, R., Rao, R.P.N.: Dynamic imitation in a humanoid robot through nonparametric probabilistic inference. In: Robotics: Science and Systems, Philidelphia, PA (August 2006)
21. Grollman, D.H., Jenkins, O.C.: Dogged learning for robots. In: International Conference on Robotics and Automation, Rome, Italy, April 2007, pp. 2483–2488 (2007)
22. Grollman, D.H., Jenkins, O.C.: Learning robot soccer skills from demonstration. In: International Conference on Development and Learning, London, UK, July 2007, pp. 276–281 (2007)

23. Grollman, D.H., Jenkins, O.C., Wood, F.: Discovering natural kinds of robot sensory experiences in unstructured environments. Journal of Field Robotics 23(11-12), 1077–1089 (2006)
24. Hayes, G., Demiris, J.: A robot controller using learning by imitation. In: International Symposium on Intelligent Robotic Systems, Grenoble, France (July 1994)
25. Inamura, T., Inaba, M., Inoue, H.: Acquisition of probabilistic behavior decision model based on the interactive teaching method. In: International Conference on Advanced Robotics, Tokyo, Japan, October 1999, pp. 523–528 (1999)
26. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation 3(1), 79–87 (1991)
27. Jenkins, O.C., Gonzalez, G., Loper, M.M.: Interactive human pose and action recognition using dynamical motion primitives. International Journal of Humanoid Robotics 4(2), 365–385 (2007)
28. Kober, J., Mohler, B., Peters, J.: Learning perceptual coupling for motor primitives. In: International Conference on Intelligent Robots and Systems [3], pp. 834–839
29. Kohl, N., Stone, P.: Machine learning for fast quadrupedal locomotion. In: National Conference on Artificial Intelligence, pp. 611–616 (2004)
30. Konidaris, G.D., Barto, A.G.: Skill discovery in continuous reinforcement learning domains using skill chaining. Technical Report UM-CS-2008-24, University of Massachusetts Amherst (2008)
31. Lopes, M., Melo, F., Montesano, L., Santos-Victor, J.: Abstraction Levels for Robotic Imitation: Overview and Computational Approaches. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 313–355. Springer, Heidelberg (2010)
32. Mackay, D.J.C.: Introduction to Gaussian Processes. In: Mackay, D.J.C. (ed.) Neural Networks and Machine Learning. Springer, Heidelberg (1998)
33. Mccallum, A.K.: Reinforcement learning with selective perception and hidden state. PhD thesis, The University of Rochester (May 1996)
34. Meeds, E., Osindero, S.: An alternative infinite mixture of Gaussian process experts. In: Neural Information Processing Systems, Vancouver, CAN, December 2005, pp. 883–890 (2005)
35. Ng, A.Y., Russell, S.: Algorithms for inverse reinforcement learning. In: International Conference on Machine Learning, Stanford, CA, June 2000, pp. 663–670 (2000)
36. Nguyen-Tuong, D., Peters, J., Seeger, M.: Computed torque control with nonparametric regression models. In: Proceedings of the 2008 American Control Conference, ACC 2008 (2008)
37. Nguyen-Tuong, D., Seeger, M., Peters, J.: Real-time Local GP Model Learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 193–207. Springer, Heidelberg (2010)
38. Nicolescu, M., Matarić, M.J.: Natural methods for robot task learning: Instructive demonstration, generalization and practice. In: International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, AUS, July 2003, pp. 241–248 (2003)
39. Nicolescu, M.N., Jenkins, O.C., Stanhope, A.: Fusing robot behaviors for human-level tasks. In: International Conference on Development and Learning, London, UK, July 2007, pp. 76–81 (2007)
40. Nolfi, S., Tani, J.: Extracting regularities in space and time through a cascade of prediction networks: The case of a mobile robot navigating in a structured environment. Connection Science 11(2), 129–152 (1999)

41. Platt Jr., R., Fagg, A.H., Grupen, R.A.: Manipulation gaits: sequences of grasp control tasks. In: International Conference on Robotics and Automation, April - May 2004, pp. 801–806 (2004)
42. Porta, J.M., Vlassis, N., Spaan, M.T.J., Poupart, P.: Point-based value iteration for continuous pomdps. Journal of Machine Learning Research 7(11), 2329–2367 (2006)
43. Quiñonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate gaussian process regression. Journal of Machine Learning Research 6(12), 1939–1959 (2005)
44. Rasmussen, C., Ghahramani, Z.: Infinite mixtures of Gaussian process experts. In: Neural Information Processing Systems, Vancouver, CAN, December 2001, pp. 881–888 (2001)
45. Rybski, P.E., Yoon, K., Stolarz, J., Veloso, M.: Interactive robot task training through dialog and demonstration. In: International Conference on Human-Robot Interaction, Arlington, VA, March 2007, pp. 255–262 (2007)
46. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philosophical Transaction of the Royal Society of London 358(1431), 537–547 (2003)
47. Smart, W.D., Kaelbling, L.P.: Effective reinforcement learning for mobile robots. In: International Conference on Robotics and Automation, Washington, D.C., May 2002, pp. 3404–3410 (2002)
48. Snelson, E., Ghahramani, Z.: Sparse gaussian processes using pseudo-inputs. In: Neural Information Processing Systems [1], pp. 1257–1264
49. Stone, P., Veloso, M.: Beating a defender in robotic soccer: Memory-based learning of a continuous function. In: Neural Information Processing Systems, Vancouver, CAN, December 1996, pp. 896–902 (1996)
50. Stone, P., Veloso, M.M.: Layered learning. In: European Conference on Machine Learning, Barcelona, Catalonia, Spain, May 2000, pp. 369–381 (2000)
51. Tani, J., Nolfi, S.: Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems. Neural Networks 12(7-8), 1131–1141 (1999)
52. Thomaz, A.L., Breazeal, C.: Transparency and socially guided machine learning. In: International Conference on Development and Learning, Bloomington, IN, May 2006, pp. 3475–3480 (2006)
53. Trafton, J.G., Schultz, A.C., Bugajska, M., Mintz, F.: Perspective-taking with robots: experiments and models. In: International Symposium on Robot & Human Interaction, Nashville, TN, August 2005, pp. 580–584 (2005)
54. Vijayakumar, S., D'Souza, A., Schaal, S.: Incremental online learning in high dimensions. Neural Computation 17(12), 2602–2634 (2005)
55. Wood, F., Grollman, D.H., Heller, K.A., Jenkins, O.C., Black, M.: Incremental Nonparametric Bayesian Regression. Technical Report CS-08-07, Brown University Department of Computer Science (2008)
56. Wu, X., Kofman, J.: Human-inspired robot task learning from human teaching. In: International Conference on Robotics and Automation, Pasadena, CA, May 2008, pp. 3334–3339 (2008)

Mobile Robot Motion Control from Demonstration and Corrective Feedback

Brenna D. Argall, Brett Browning, and Manuela M. Veloso

Abstract. Robust motion control algorithms are fundamental to the successful, autonomous operation of mobile robots. Motion control is known to be a difficult problem, and is often dictated by a *policy*, or state-action mapping. In this chapter, we present an approach for the refinement of mobile robot motion control policies, that incorporates *corrective* feedback from a human teacher. The target application domain of this work is the low-level motion control of a mobile robot. Within such domains, the rapid sampling rate and continuous action space of policies are both key challenges to providing policy corrections. To address these challenges, we contribute *advice-operators* as a corrective feedback form suitable for providing *continuous*-valued corrections, and *Focused Feedback For Mobile Robot Policies (F3MRP)* as a framework suitable for providing feedback on policies sampled at a high frequency. Under our approach, policies refined through teacher feedback are initially derived using *Learning from Demonstration (LfD)* techniques, which generalize a policy from example task executions by a teacher. We apply our techniques within the *Advice-Operator Policy Improvement (A-OPI)* algorithm, validated on a Segway RMP robot within a motion control domain. A-OPI refines LfD policies by correcting policy performance via advice-operators and F3MRP. Within our validation domain, policy performance is found to improve with corrective teacher feedback, and moreover to be similar or superior to that of policies provided with more teacher demonstrations.

Brenna D. Argall
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: bargall@ri.cmu.edu

Brett Browning
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: brettb@cs.cmu.edu

Manuela M. Veloso
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: veloso@cs.cmu.edu

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 431–450.
springerlink.com

1 Introduction

Whether an exploration rover in space or recreational robot for the home, successful autonomous mobile robot operation requires an algorithm for motion control. A control *policy* provides such an algorithm, by mapping an observation of the world to an action available on the robot. This mapping is fundamental to many robotics applications, yet in general is complex to develop.

Even with a carefully crafted policy, a robot often will not behave as the developer expects or intends in all areas of the execution space. One way to address behavior shortcomings is to update a policy based on execution experience, which can increase policy robustness and overall performance. For example, such an update may expand the state-space areas in which the policy is valid, or increase the likelihood of successful task completion.

This chapter contributes our approach for refining mobile robot policies with human feedback. The feedback consists of policy corrections, which are provided based on human teacher observations of policy executions by the robot. The target domain of this work is low-level motion control on a mobile robot. Challenges to providing corrective feedback within this domain include the continuous state-action space of the policy, and the rapid rate at which the policy is sampled. We introduce techniques to address both of these challenges.

1.1 Mobile Robot Motion Control

A motion control policy defines a mapping from world state to robot action. Motion control policies are able to represent actions at a variety of control levels:

Low-level actions: Low-level actions directly control the movement mechanisms of the robot. These actions are in general continuous-valued and of short time duration, and a low-level motion policy is sampled at a high frequency. An example low-level action is a command for wheel speed that updates at $50Hz$.

High-level actions: High-level actions encode a more abstract action representation, which is then translated through other means to affect the movement mechanisms of the robot; for example, through another controller. These actions are in general discrete-valued and of longer time duration, and their associated control policies are sampled at a low frequency. An example high-level action is to approach and pick up an object, executing over tens of seconds.

The focus of this chapter is on low-level motion control policies. The continuous action-space and high sampling rate of low-level control are both key considerations during policy development and refinement.

The state-action mapping represented by a motion control policy is typically complex to develop. One reason for this complexity is that the target observation-action mapping is unknown. What *is* known is the desired robot motion *behavior*, which must somehow be represented through an unknown observation-action mapping. A second reason for this complexity are the complications of motion policy

execution in real world environments. The world is observed through sensors that are typically noisy; models of world dynamics are only an approximation to the true dynamics; and actions are motions executed with real hardware, which depends on physical considerations like calibration accuracy and inevitably executes with some level of imprecision.

The development of control policies therefore generally requires a significant measure of effort and expertise, and frequently demands extensive prior knowledge and parameter tuning. The required prior knowledge ranges from details of the robot and its movement mechanisms, to details of the execution domain and how to implement a given control algorithm. Any successful application typically has the algorithm highly tuned for operation with a particular robot in a specific domain. Furthermore, existing approaches are often applicable only to simple tasks due to computation or task representation constraints.

Traditional approaches to robot control model the domain dynamics and derive policies using those mathematical models [35]. While theoretically well-founded, these approaches typically depend heavily upon the accuracy of the model, which can require considerable expertise to develop and becomes increasingly difficult to define as robot become more complex. Other approaches, such as *Reinforcement Learning (RL)* [37], guide policy learning by providing reward feedback about the desirability of visiting particular states. To define a function to provide these rewards, however, is known to be a difficult problem that also requires considerable expertise to address. Furthermore, building the policy necessitates gathering information by visiting states to receive rewards, which is non-trivial for a mobile robot executing physical actions.

The experience of personally developing numerous motion behaviors by hand for this robot [5], and subsequent desire for more straightforward policy development techniques, was a strong motivating factor in this work. Similar frustrations have been observed in other roboticists, further underlining the value of approaches that ease the policy development process. Another, more hypothetical, motivating factor is that as familiarity with robots within general society becomes more prevalent, it is expected that future robot operators will include those who are *not* robotics experts. We anticipate a future requirement for policy development approaches that not only ease the development process for experts, but are accessible to non-experts as well.

1.2 Learning from Demonstration

Learning from Demonstration (LfD) is one policy development technique with the potential for both application to non-trivial tasks and straightforward use by robotics-experts and non-experts alike [4, 11]. Under the LfD paradigm, a teacher first demonstrates a desired behavior to the robot, producing an example state-action trace. The robot then generalizes from these examples to derive a policy, thus learning a state-action mapping.

1.2.1 Support for Demonstration Learning

LfD has many attractive points for both learner and teacher. To develop a policy within a LfD paradigm typically does not require expert knowledge of the domain dynamics, which removes the performance dependence on model accuracy. The relaxation of the expert knowledge requirement also opens policy development to those who are not robotics-experts, satisfying a need that we expect to increase as robots become more commonly available. Furthermore, demonstration has the attractive feature of being an intuitive medium for communication from humans, who already use demonstration to teach other humans.

More concretely, the application of LfD to motion control has many advantages:

Implicit behavior to mapping translation. By demonstrating a desired motion behavior, and recording the encountered states and actions, the translation of a behavior into a representative state-action mapping is immediate and implicit.

Robustness under real world uncertainty. The uncertainty of the real world means that multiple demonstrations of the same behavior will not execute identically. Generalization over demonstration examples thus produces a policy that does not depend on a strictly deterministic world, and therefore should execute more robustly under real world uncertainty.

Focused policies. Demonstration has the practical feature of focusing the dataset of examples to areas of the state-action space actually encountered during behavior execution. This is particularly useful in continuous action space domains, with an infinite number of state-action combinations.

The LfD approach to obtaining a policy is in contrast to other techniques in which a policy is learned from *experience*, for example building a policy based on data acquired through exploration, as in RL. Furthermore a policy derived under LfD is necessarily defined only in those states encountered, and for those actions taken, during the example executions.

1.2.2 Formalism

Our approach to policy development derives an initial policy from teacher demonstrations. Within this chapter, we formally define the world to consist of states S and actions A, with the mapping between states by way of actions being governed by the probabilistic transition function $T(s'|s,a) : S \times A \times S \rightarrow [0,1]$. We assume that state is not fully observable, and instead the learner has access to observed state Z, through a mapping $S \rightarrow Z$. A teacher demonstration $d \in D$ is represented as n pairs of observations and actions, such that $d = \{(\mathbf{z}_i, \mathbf{a}_i)\} \in D,\ \mathbf{z}_i \in Z,\ \mathbf{a}_i \in A,\ i = 0 \cdots n$. Within the typical LfD paradigm, the set D of these demonstrations is provided to the learner. A policy $\pi : Z \rightarrow A$, that selects actions based on an observation of the current world state, or *query point*, is then derived from the dataset D.

1.2.3 Related Work

LfD has found success on a variety of robot platforms and applications [4, 11]. Since demonstration for real robots involves executing actions in physical environments, differences in *embodiment* between the learner and teacher become of crucial importance. The challenges that arise from these differences, where teacher demonstrations may not map directly to the learner due to differences in sensing or motion, are broadly referred to as *correspondence issues* within the LfD literature [12, 26]. Key design decisions therefore include the choice of teacher controlling, and platform executing, the demonstration, as well as how the demonstration is recorded.

A variety of approaches exist for executing and recording teacher demonstrations. At one extreme lies *teleoperation*, where the passive robot platform records from its own sensors while under direct teacher control [9, 13]. This approach is very effective at reducing teacher-learner correspondence issues, but does require actively controlling the robot during the task, which might not be manageable for example if controlling a high-DoF humanoid for low-level motion control. Another approach has the robot learner actively *mimic* the teacher during the demonstration executions, again while recording from its own sensors [18, 29]. This has the advantage of not requiring the teacher to actively control the robot, but does require that the learner be able to identify and track the teacher; furthermore, the observations made by the *teacher* during the execution are not directly recorded.

Other demonstration techniques do not employ the actual learner platform during the demonstration. One such technique has the *teacher* wear sensors during task execution with her *own* body [14, 21, 23]. This requires specialized sensors and introduces another level of teacher-learner correspondence, but does not require that the learner platform be actively operated or able to track the teacher during task execution. Lastly, at the opposite extreme to teleoperation, sensors *external* to the teacher's body may record his execution of the task with his own body [8, 10]. This has the lowest requirements in terms of specialized sensors or actively operating the robot, but is the most likely to encounter correspondence issues when transferring the recorded teacher demonstrations to the learner platform.

Once the dataset is recorded, a policy must be derived from it. A variety of policy derivation techniques are employed within LfD, the majority of which fall into three categories. The first category directly *approximates the function mapping* states to actions $f() : Z \rightarrow A$, using regression or classification techniques. Successful LfD implementations of this policy derivation approach include tasks with humanoids [10, 14, 20], Sony AIBOs [15, 19] and a variety of other platforms [25]. The second category learns a *state-action transition model* $T(s'|s,a)$ from the demonstration data, pairs this with a reward function $R(s)$ (either hand-engineered or learned) and derives a policy using RL techniques. Successful LfD applications under this approach span tasks with autonomous helicopters [1, 9, 27] to small quadriped robots [22, 32], humanoids [24] and robotic arms [8]. The third category learns task *plans* from the demonstration set, including small wheeled robot [29] and companion robot [33] applications. Note that each of these techniques are employed for the

derivation of policies with high-level actions, while only the first two are used to derive policies with low-level actions.

Our work takes the policy derivation approach of directly approximating the underlying function mapping states to actions. Since our action space is continuous, regression techniques are employed. During the empirical validation of our techniques, teleoperation is the approach that will be used for gathering demonstrations.

1.3 Policy Refinement within LfD

LfD is inherently linked to the information provided in the demonstration dataset. As a result, learner performance is heavily limited by the quality of this information. Though LfD has enabled successful policy development for a variety of robot platforms and applications, this approach is not without its limitations.

1.3.1 Potential Dataset Limitations

One common cause for poor learner performance is dataset sparsity, or the existence of state space areas in which no demonstration has been provided. Dataset sparsity is a trade off to focusing the dataset to areas visited during task execution, since the learner is provided with an indication of which action to take only in those states visited during demonstration. In all but the most simple domains the teacher will be unable to demonstrate from every state, and so there will be areas of the state space absent from the demonstration set. Note however that dataset sparsity may be overcome to a certain extent by the generalization ability of the policy derivation technique.

A second cause is poor quality of the dataset examples. Poor quality examples can result from the demonstration abilities of the teacher, who may in fact provide suboptimal or ambiguous demonstrations. Poor quality examples also can results from poor correspondence between the teacher and learner, who may differ in sensing or motion capabilities.

To summarize, common sources of LfD limitations include:

1. Uncovered areas of the state space, absent from the demonstration dataset.
2. Suboptimal or ambiguous teacher demonstrations.
3. Poor translation from teacher to learner, due to correspondence issues.

One way to address dataset limitations is to extend LfD by having the robot update its policy based on execution experience.

1.3.2 Related Work

One popular approach for dealing with poor or ambiguous teacher demonstrations is to provide more demonstration data in response to execution experience with the policy. There are approaches that acquire new demonstrations by enabling the learner to evaluate its confidence in selecting a particular action, based on the confidence of the underlying classification algorithm. For example, the robot can

indicate to the teacher its certainty in performing various elements of the task [19], or request additional demonstration in states that are either very different from previously demonstrated states or for which a single action cannot be selected with certainty [16]. Other approaches rely on teacher observation of the policy performance alone, for example to provide a robot with new demonstrations through kinesthetic teaching that moves passive joints through desired position trajectories [14].

Another approach is to pair LfD with RL techniques, which is particularly relevant for implementations that already derive their policies using RL. The goal of RL is to maximize cumulative reward over time, and typically each state s is associated with a value according to the function $V(s)$ (or associating state-action pair s,a with a Q-value according to the $Q(s,a)$) [37]. By updating the state values $V(s)$ with rewards received during execution [36], a policy derived under LfD also updates. We note that these are rewards seen during *learner* execution, and not during demonstration. To visit and evaluate new states not seen in the demonstration set, an exploration policy may be employed [30, 34], though we note that in general taking exploratory steps on a real robot can be inefficient and even dangerous. Finally, execution experience may also update a learned transition function $T(s'|s,a)$ [2].

2 Corrective Feedback for Policy Refinement

Our approach to the improvement of LfD policies through experience is to provide corrections on policy executions. Corrective feedback is provided by a human teacher, in response to policy executions by a robot learner. In particular, feedback corrects state-action mappings produced during a student execution to generate new examples for the LfD policy. We begin by motivating and discussing corrective feedback for the improvement of LfD policies (Sec. 2.1), followed by a detailing of the contributed techniques that enable a human teacher to provide continuous-valued corrections (Sec. 2.2) to policies sampled at a high frequency (Sec. 2.3). We then present an algorithm that employs our corrective feedback techniques (Sec. 2.4).

2.1 Policy Corrections

To address potential LfD limitations, the approach of correcting poor policy predictions we argue is particularly direct. While overall performance evaluations or state rewards can provide an indication of the quality of a policy prediction, they do not provide any guidance on what might have been a more suitable *alternate prediction*. Providing a correction on poor predictions therefore provides more focused and detailed policy improvement information. Furthermore, while more demonstrations can populate sparse areas of the state space or demonstrate a corrected behavior, they require *state re-visitation*, which can be impractical within real world domains.

Policy correction has seen limited attention within LfD however. The selection of a policy correction in general is sufficiently complex to preclude it being provided with a simple function. Approaches that do correct policy predictions therefore provide corrections through human teachers, which our techniques do as well.

Furthermore, since the correction involves selecting a preferred state or action, within the existing literature corrections are only provided within action spaces that are *discrete* and with actions of significant time duration, and therefore sampled with *low* frequency. For example, the correct action from a discrete set is provided by a human teacher to update a high-level action classifier [15], and the structure of a hierarchical Neural Network of robot behaviors [29].

Approaches that correct policies within *continuous* action-spaces sampled at *high* frequency are absent from the existing LfD literature. These considerations have prompted our development of a corrective feedback form that *is* appropriate for continuous-valued action domains (Sec. 2.2). We furthermore develop a feedback framework (Sec. 2.3) that is suitable for domains with rapidly sampled policies.

2.2 Advice-Operators

To address the challenge of providing continuous-valued corrections, we introduce *advice-operators* [3] as a language through which a human teacher provides policy corrections to a robot student.

Concretely defined, an advice-operator is a mathematical computation performed on an observation input or action output. Given a policy execution by the learner, an operator is indicated by the teacher and applied to a state-action pair recorded during the execution. Key characteristics of advice-operators are that they:

1. Perform mathematical computations on datapoints.
2. Are defined commonly between the student and advisor.
3. May be applied to observations or actions.

Figure 1 presents a diagram of data synthesis from student executions and teacher feedback (bottom, shaded area); for comparison, LfD data from teacher executions is also shown (top). To illustrate with an example, consider a simple operator that modifies translational acceleration by a static amount δ. Suppose the teacher indicates this operator for application over 15 data points from the learner execution.

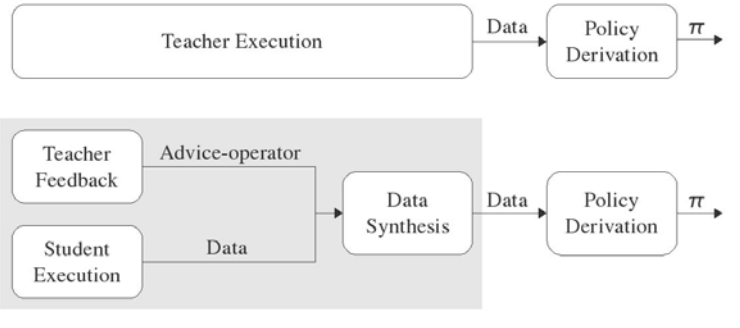


Fig. 1 Generating demonstration data under classical LfD (top) and advice-operators (bottom).

The translational speed a^0 of executed point 0 then updates to $\hat{a}^0 \leftarrow a^0 + \delta$, point 1 to $\hat{a}^1 \leftarrow a^1 + \delta$, and so forth until point 14 updates to $\hat{a}^{14} \leftarrow a^{14} + \delta$.

Policy correction under advice-operators does *not* rely on teacher demonstration to indicate a corrected behavior. The advice-operator approach thus includes the following strengths:

No need to recreate state. This is especially useful if the world states where corrective demonstration is needed are dangerous (e.g., lead to a collision), or difficult to access (e.g., in the middle of a motion trajectory).

Not limited by the demonstrator. Corrections are not limited to the execution abilities of the demonstration teacher, who may be suboptimal.

Unconstrained by correspondence. Corrections are not constrained by physical differences between the teacher and learner.

Possible when demonstration is not. Further demonstration may in fact be impossible (e.g., teleoperation over a 40 minute Earth-Mars communications lag).

We thus contribute a formulation for corrective feedback, as a predefined list of mathematical functions. Advice-operators enable the translation of a statically-defined high-level correction into a continuous-valued, execution-dependent, low-level correction. Moreover, when combined with our techniques for *providing* feedback (Sec. 2.3), a single piece of advice corrects multiple execution points. The selection of a *single advice-operator* thus translates into *multiple continuous-valued corrections*, and therefore is suitable for modifying low-level motion control policies sampled at high frequency.

2.3 Focused Feedback for Mobile Robot Policies

To address the challenge of providing feedback to policies sampled at a rapid rate, we introduce *Focused Feedback For Mobile Robot Policies (F3MRP)* [6] as a framework through which portions of a policy execution are selected to receive feedback. The target application domain for F3MRP is *mobile* robot motion control.

At the core of the F3MRP framework is a visual presentation of the 2-D path physically taken by the mobile robot on the ground.[1] For experiments with a simulated robot, the 2-D path is represented in real-time as the robot executes. For experiments with a real robot, the 2-D path is played back, at true speed, after the learner execution completes, to mitigate inaccuracies due to network lag.

The visual path presentation is a key component of the interface for the identification of those portions of the learner execution that require correction. Through this interface, the teacher selects segments of the 2-D ground path that correspond to those portions of the execution during which the policy performed poorly. An

[1] The F3MRP framework was designed specifically for mobile robot applications. To apply the framework to non-mobile robots would require an alternative to the 2-D ground path, to serve as the visualization component of the interface for segment selection.

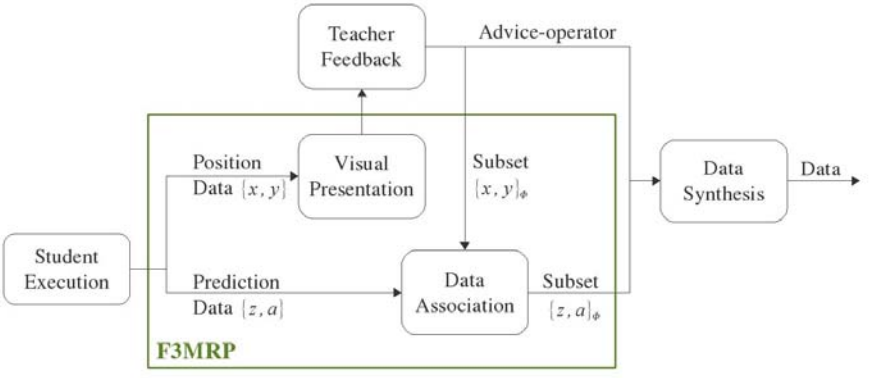


Fig. 2 Path visualization, subset selection and data association under the F3MRP framework.

overview of the F3MRP interface is shown in Figure 2, which expands the shaded area in Figure 1 with the details of segment selection.

Corrective feedback provided under the F3MRP framework must associate closely with the underlying learner execution, since the feedback corrects specific execution points. By contrast, consider an overall performance measure, that need only associate with the execution as a whole and thus links to the data at a fairly coarse scale. To accomplish close feedback-execution association under F3MRP, the teacher selects problem segments of the graphically displayed ground path. Segment sizes are determined dynamically by the teacher, and may range from a single point to all points in the trajectory.

The F3MRP framework then associates the selected segment of the *position trace*, i.e. the ground path, with the corresponding segment of the *prediction trace*, i.e. the state-action sequence, recorded during the learner execution. This process is the tool through which the human flags state-action pairs for modification: by selecting segments of the displayed ground path, which the framework then associates with the state-action trace of the policy.

2.4 Algorithm Advice-Operator Policy Improvement

The *Advice-Operator Policy Improvement (A-OPI)* algorithm [3] refines a motion control policy, initially derived from LfD, by providing corrections through advice-operators and the F3MRP framework. The algorithm operates in two phases. During the *demonstration phase*, a set of teacher demonstrations is provided to the learner. This demonstration set D consists of example executions of the target behavior, during which state-action pairs are recorded. From this set the learner generalizes an initial policy. During the *feedback phase*, the learner executes with this initial policy. Feedback on the learner execution is offered by a human teacher, and is used by the learner to update its policy. The learner then executes with the updated policy, and the *execute-feedback-update* cycle continues to the satisfaction of the teacher.

Algorithm 1. *Advice-Operator Policy Improvement*

1: Given **D**
2: *initialize* $\pi \leftarrow$ `policyDerivation`(D)
3: **while** *practicing* **do**
4: *initialize* $d \leftarrow \{\}$, $tr \leftarrow \{\}$
5: **repeat**
6: *predict* $\mathbf{a}^t \leftarrow \pi(\mathbf{z}^t)$
7: *execute* $\mathbf{a}^t$
8: *record* $d \leftarrow d \cup (\mathbf{z}^t, \mathbf{a}^t)$, $tr \leftarrow tr \cup (x^t, y^t, \theta^t)$
9: **until** done
10: *advise* $\{ op, \Phi \} \leftarrow$ `teacherFeedback`(tr)
11: **for all** $\varphi \in \Phi$, $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \in d$ **do**
12: **if** op is observation-modifying **then**
13: *modify* $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \leftarrow (op(\mathbf{z}^\varphi), \mathbf{a}^\varphi)$
14: **else** {op is action-modifying}
15: *modify* $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \leftarrow (\mathbf{z}^\varphi, op(\mathbf{a}^\varphi))$
16: **end if**
17: *update* $D \leftarrow D \cup (\mathbf{z}^\varphi, \mathbf{a}^\varphi)$
18: **end for**
19: *rederive* $\pi \leftarrow$ `policyDerivation`(D)
20: **end while**
21: **return** π

Algorithm 1 presents pseudo-code for the A-OPI algorithm. To begin, an initial policy π is derived from the set of teacher demonstrations (line 2). A single practice run (lines 3-20) consists of a single execution-feedback-update cycle.

During the learner execution portion of a practice run (lines 5-9), the learner executes the task. At each timestep the learner observes the world, and predicts action $\mathbf{a}^t$ according to policy π (line 6). This action is executed and recorded in the *prediction* trace d, along with observation $\mathbf{z}^t$ (line 8). The information recorded in the trace d will be incorporated into the policy update. The global position x^t, y^t and heading θ^t of the mobile robot is additionally recorded, into the *position* trace tr. Information recorded in tr will be used by the F3MRP framework, when visually depicting the path taken by the robot on the ground during execution.

During the teacher feedback portion of the practice phase, the teacher first indicates, through the F3MRP interface, a segment Φ of the learner execution trajectory requiring improvement. The teacher further indicates an advice-operator op, selected from a finite list, to correct the execution within this segment (line 10).

The teacher feedback is then applied across all points recorded in d and within the indicated subset Φ (lines 11-18). For each point $(\mathbf{z}^\varphi, \mathbf{a}^\varphi) \in d$, $\varphi \in \Phi$, the algorithm modifies either its observation (line 13) or action (line 15), depending on the type of the indicated advice-operator. The modified datapoints are added to the demonstration set D (line 17), and the policy is rederived (line 19).

3 Empirical Validation of A-OPI

This section presents an empirical validation of our corrective feedback approach. We validate the A-OPI algorithm that employs our corrective feedback techniques - that is, advice-operators and the F3MRP framework - on a Segway Robot Mobility Platform (RMP) [28] performing a spatial positioning task. Policy modifications due to corrective feedback are shown to improve policy performance, which furthermore is found to be similar or superior to the more typical approach of providing more teacher demonstrations.

3.1 Experimental Setup

Empirical validation of the A-OPI algorithm is performed through a spatial positioning task with a Segway RMP. This section presents the task and domain, followed by policy development and evaluation; further empirical details may be found in [3].

3.1.1 Task and Domain

The Segway RMP is a two-wheeled dynamically-balancing differential drive robot, which may only drive forward or turn and cannot go sideways. The robot accepts wheel speed commands, but does not allow access to its balancing control mechanisms. We therefore treat Segway RMP control as a black box, since we do not know the specific gains or system parameter values. The inverted pendulum dynamics of the robot present an additional element of uncertainty for low level motion control.

The spatial positioning task consists of attaining a 2-D planar target position (x_g, y_g) with a heading θ_g (Fig. 3). For this task smoothly coupled rotational and translational speeds are preferred, in contrast to turning on spot to θ_g after attaining (x_g, y_g). To mathematically define for this specific robot platform the desired motion trajectories for our task is thus non-trivial, encouraging the use of alternate control approaches such as A-OPI. That the task is straightforward for a human to evaluate and correct further supports A-OPI as a candidate approach. While the task was chosen for its suitability to validate A-OPI, to our knowledge this work also constitutes the first implementation of such a motion task on a real Segway RMP platform.

Fig. 3 Segway RMP performing the spatial positioning task (approximate ground path in yellow).

To gather demonstration examples, a human teleoperates the platform as the robot records from its own sensors. Teleoperation minimizes correspondence issues between demonstrator and learner, and is reasonable to perform for this task and robot platform. The robot observes its global position and heading through wheel encoders sampled at $30Hz$.

To derive a policy from the demonstration examples, the function mapping states to actions is directly approximated via regression techniques. We employ a form of Locally Weighted Learning [7]. Worthwhile to note however is that A-OPI is not restricted to a particular regression technique, and *any* are appropriate for use within the algorithm. Given observation $\mathbf{z}^t$, action $\mathbf{a}^t$ is predicted through an averaging of the actions in D, weighted by a kernelized distance between their associated datapoint observations and the current observation $\mathbf{z}^t$. Thus,

$$\mathbf{a}^t = \sum_{(\mathbf{z}_i,\mathbf{a}_i)\in D} \phi\left(\mathbf{z}^t,\mathbf{z}_i\right)\cdot\mathbf{a}_i \quad , \quad \phi\left(\mathbf{z}^t,\mathbf{z}_i\right) = \frac{e^{(\mathbf{z}_i-\mathbf{z}^t)^T\,\Sigma^{-1}(\mathbf{z}_i-\mathbf{z}^t)}}{\sum_{\mathbf{z}_j\in D} e^{\left(\mathbf{z}_j-\mathbf{z}^t\right)^T\,\Sigma^{-1}\left(\mathbf{z}_j-\mathbf{z}^t\right)}} \tag{1}$$

where the weights $\phi\left(\mathbf{z}^t,:\right)$ are normalized over i. In this work the distance computation is Euclidean, the kernel is Gaussian and Σ^{-1} is a constant diagonal matrix that scales each observation dimension and embeds the bandwidth of the Gaussian kernel. All parameters are tuned through Leave-One-Out-Cross-Validation (LOOCV), minimizing the least squared error of the regression prediction on the set D.

The observations for this task are 3-dimensional, and are feature computations involving the global and target position and heading: (i) squared Euclidean distance to the target position, (ii) angle between the target position and current robot heading and (iii) angle between the current and target robot headings. The actions are 2-dimensional: (i) translational speed and (ii) rotational speed. The motion control operators developed for this domain adjust observation inputs (Tbl. 1, Operator 0), single action dimensions by non-static amounts (Operators 1-6) or multiple action dimensions by non-static amounts (Operators 7-8). The amount of the non-static adjustments are determined as a function of the executed values of the observations and actions.

Table 1 Advice-operators for the spatial positioning task.

	Operator	Parameter
0	Reset goal, recompute observation	
1	No turning	
2	Start turning	[cw ccw]
3	Smooth rotational speed	[dec inc]
4	No translation	
5	Smooth translational speed	[dec inc]
6	Translational [ac/de]celeration	[dec inc]
7	Turn tightness	[less more]
8	Stop all motion	

Key: $(c)cw$=$(counter)clockwise$, (dec/inc)=$(de/in)crease$

3.1.2 Policy Development and Evaluation

The set D is seeded with demonstrations recorded as the teacher teleoperates the robot learner (9 demonstrations, totaling 900 datapoints). Policy improvement proceeds as follows. A random goal is selected (without replacement) from a practice set consisting of (x_g, y_g, θ_g) goals, drawn uniformly within the bounds of the demonstration dataset. The robot executes with its current policy to attain this goal. The advisor observes this execution, and optionally offers policy improvement information. The policy is re-derived, and drawing a new goal initiates another practice run.

Three policies are developed using distinct techniques, differing in *what* is offered as policy improvement information. A total of four policies are therefore developed within this domain:

1. *Baseline Policy (Base)*: Derived from the initial demonstration set.
2. *Feedback Policy (FB)*: Provided with policy corrections, via advice-operators.
3. *Feedback-Hybrid Policy (FB-H)*: Initially provided with more teacher demonstrations; later provided with policy corrections via advice-operators.
4. *More-Demonstration Policy (M-Demo)*: Provided with more teacher demonstrations.

The final three are referred to collectively as the *improvement policies*. Note that in the case of policy *M-Demo*, a practice run consists of a single execute-*demonstrate*-update cycle.

Policies are evaluated for accuracy and success, on an independent test set of (x_g, y_g, θ_g) goals. Here *accuracy* is defined as Euclidean distance between the final robot and goal positions $e_{x,y}$, and the final robot and goal headings e_θ. *Success* is defined generously as $e_{x,y} < 1.0\ m$ and $e_\theta < \frac{\pi}{2}\ rad$. Practice runs were halted once performance on the test set no longer improved (number of practice runs: 60 *FB*, 59 *FB-H* and 51 *M-Demo*).

3.2 Results

Policy performance was found to improve with corrective feedback, in both execution success and accuracy [3]. A-OPI additionally enabled similar or superior performance when compared to a policy derived from more teacher demonstrations. Furthermore, by concentrating new data exclusively to the areas visited by the robot and needing improvement, A-OPI produced noticeably smaller datasets.

3.2.1 Success and Accuracy Improvement

Table 2 presents the percent execution success of each policy on the independent test set. When compared to policy *Base*, all policy improvement approaches display an increase in success. Both of the feedback policies additionally achieve higher success than policy *M-Demo*.

Table 2 Execution Percent Success.

Baseline	Feedback	Feedback-Hybrid	More-Demonstration
32	88	**92**	80

Figure 4 plots, for each policy, the average position and heading error on the test set goals. For positional error, all improvement policies displayed similar performance, which was a dramatic improvement over policy *Base*. For heading, policy *FB* reduced more error than policy *FB-H*, with both showing marked improvements over policy *Base*. By contrast, policy *M-Demo* displayed *no* improvement in heading error over policy *Base*. That heading error was in general more difficult to improve than positional error is consistent with our prior experience with this robot platform, which is highly sensitive to the accumulation of rotational dead reckoning error.

The iterative nature of policy development under A-OPI produces many intermediate policies, a sampling of which were also evaluated on the test set. Figure 5 shows the average position and heading error of the intermediate policies on the test set goals, to mark the progress of each policy improvement technique.

Superior heading performance was consistently produced by corrective feedback, with policy *FB* attaining lower heading error than policy *M-Demo* throughout policy improvement. By contrast, initially greater improvement in positional error is seen with more demonstration and thus with policy *M-Demo*. While corrective feedback reduces positional error more slowly, policy *FB* does however eventually converge to the level attained through more demonstration.

Policy *FB-H* initially displays the superior reduction in positional error, and inferior reduction in heading error, of policy *M-Demo*. This performance is followed by substantial reductions in heading error, akin to policy *FB*. These results reflect to the development technique of policy *FB-H*. The policy was initially seeded with

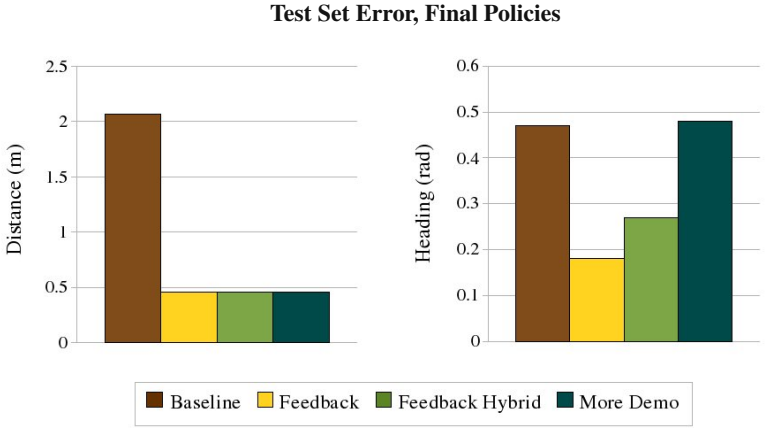


Fig. 4 Average test set error on target position (left) and heading (right), with the *final* policies.

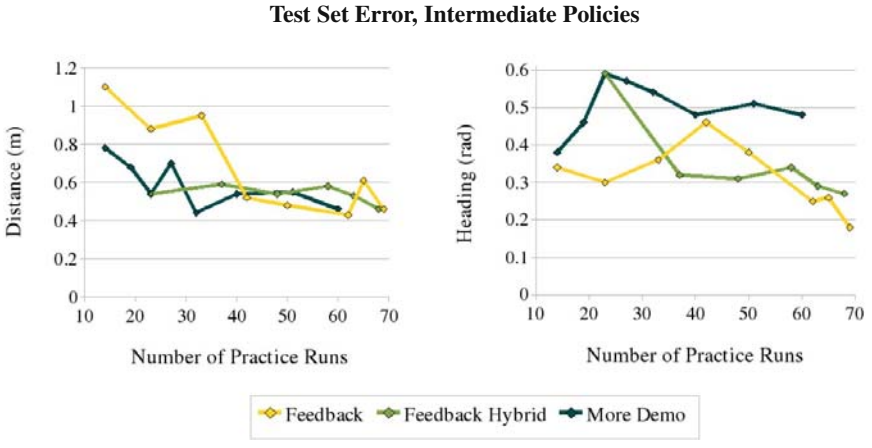


Fig. 5 Average test set error on target position (left) and heading (right), with *intermediate* policies.

an intermediate version of policy *M-Demo* (resulting after 23 practice runs), and following the seeding was offered exclusively corrective feedback.

3.2.2 More Focused Datasets

How many datapoints were added with each practice run varied greatly depending on whether the execution received corrective feedback or more demonstrations (Fig. 6). The reason is that, in contrast to teleoperation, only subsets of a corrected execution were added to the dataset; in particular, only those execution points which actually received corrections. States visited during good performance portions of the student execution were *not* redundantly added to the dataset. In this manner, the final policy performances shown in Figure 4 were achieved with much *smaller* datasets

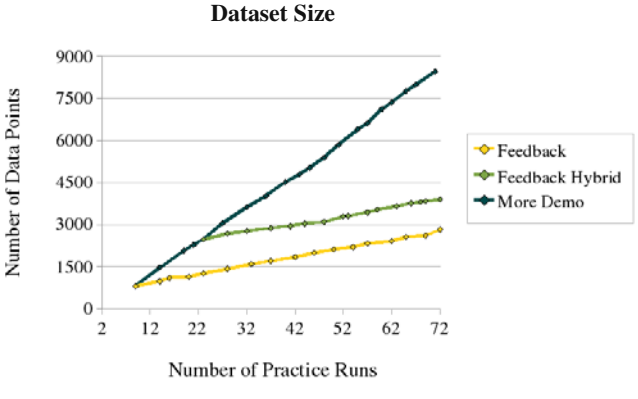


Fig. 6 Growth in dataset size with practice runs.

for both feedback policies, in comparison to policy *M-Demo*. Note that the results of Figure 5 are plotted against the number of *practice runs* contributing to the dataset, and not the number of *datapoints* in the set.

4 Conclusion

To define an algorithm for motion control on a mobile robot is a difficult and challenging task, which we address by continuing to adapt and refine a control policy based on execution experience. Our approach to motion control demonstrates a behavior to the robot, and addresses potential limitations in the resultant dataset through execution experience. In particular, policy refinement is achieved through corrective feedback provided by a human teacher.

There are two key challenges to providing feedback within low-level motion control domains. The first is the continuity of the action space: continuous-valued actions require continuous-valued corrections, and thus selection from an infinite set. The second is the sampling rate of the policy: a rapid sampling rate means that multiple execution points are responsible for a particular behavior being corrected. We have developed techniques to address each of these challenges.

The first technique, named *advice-operators*, is a language through which a human teacher provides corrections to a robot student. Advice-operators perform mathematical computations on continuous-valued datapoints. To provide a correction, the teacher selects from a finite list of advice-operators. The robot learner applies the operator to an execution datapoint, modifying its value and producing a continuous-valued correction.

The second technique, named *Focused Feedback For Mobile Robot Policies (F3MRP)*, is a framework through which a human teacher provides feedback on mobile robot motion control executions. Through the F3MRP interface, the teacher selects segments of the execution to receive feedback, which simplifies the challenge of providing feedback to policies sampled at a high frequency. A crucial element of the interface is the visual presentation of the ground path taken by the robot during execution; the framework thus targets *mobile* robots in particular.

By pairing these two techniques, the selection of a *single* advice-operator and application segment therefore provides *continuous*-valued corrections on *multiple* execution points. In this manner, our approach enables correction-giving that is reasonable and effective for a human to provide, even within a continuous-valued, rapidly sampled, domain.

We have validated these techniques through our *Advice-Operator Policy Improvement (A-OPI)* algorithm, which employs both advice-operators and the F3MRP framework. A-OPI was implemented on a Segway RMP robot, performing a spatial positioning task. Within this domain, corrective feedback was found to improve policy performance, and to enable similar or superior performance when compared to a policy derived from more teacher demonstrations. Furthermore, by concentrating new data exclusively to the areas visited by the robot and in need of improvement, corrective feedback also produced noticeably smaller datasets, and without a

sacrifice in policy performance, suggesting the datasets to be more focused and contain less redundant data.

Acknowledgements. The research is partly sponsored by the Boeing Corporation under Grant No. CMU-BA-GTA-1, BBNT Solutions under subcontract No. 950008572, via prime Air Force contract No. SA-8650-06-C-7606, the United States Department of the Interior under Grant No. NBCH-1040007 and the Qatar Foundation for Education, Science and Community Development. The views and conclusions contained in this document are solely those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

References

1. Abbeel, P., Coates, A., Quigley, M., Ng, A.Y.: An application of reinforcement learning to aerobatic helicopter flight. In: Proceedings of Advances in Neural Information Proccessing, NIPS 2007 (2007)
2. Abbeel, P., Ng, A.Y.: Exploration and apprenticeship learning in reinforcement learning. In: Proceedings of the 22nd International Conference on Machine Learning, ICML 2005 (2005)
3. Argall, B., Browning, B., Veloso, M.: Learning robot motion control with demonstration and advice-operators. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008 (2008)
4. Argall, B., Chernova, S., Browning, B., Veloso, M.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57(5), 469–483 (2009)
5. Argall, B., Gu, Y., Browning, B., Veloso, M.: The first segway soccer experience: Towards peer-to-peer human-robot teams. In: First Annual Conference on Human-Robot Interactions, HRI 2006 (2006)
6. Argall, B.D.: Learning Mobile Robot Motion Control from Demonstration and Corrective Feedback. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Technical report CMU-RI-TR-09-13 (2009)
7. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning. Artificial Intelligence Review 11, 11–73 (1997)
8. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Proceedings of the Fourteenth International Conference on Machine Learning, ICML 1997 (1997)
9. Bagnell, J.A., Schneider, J.G.: Autonomous helicopter control using reinforcement learning policy search methods. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2001 (2001)
10. Bentivegna, D.C.: Learning from Observation Using Primitives. PhD thesis, College of Computing, Georgia Institute of Technology, Atlanta, GA (2004)
11. Billard, A., Callinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. In: Siciliano, B., Khatib, O. (eds.) Handbook of Robotics, ch. 59. Springer, New York (2008)
12. Breazeal, C., Scassellati, B.: Robots that imitate humans. Trends in Cognitive Sciences 6(11), 481–487 (2002)
13. Browning, B., Xu, L., Veloso, M.: Skill acquisition and use for a dynamically-balancing soccer robot. In: Proceedings of 19th National Conference on Artificial Intelligence, AAAI 2004 (2004)

14. Calinon, S., Billard, A.: Incremental learning of gestures by imitation in a humanoid robot. In: Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interactions, HRI 2007 (2007)
15. Chernova, S., Veloso, M.: Learning equivalent action choices from demonstration. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2008 (2008)
16. Chernova, S., Veloso, M.: Multi-thresholded approach to demonstration selection for interactive robot learning. In: Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction, HRI 2008 (2008)
17. Dautenhahn, K., Nehaniv, C.L. (eds.): Imitation in animals and artifacts. MIT Press, Cambridge (2002)
18. Demiris, Y., Hayes, G.: Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In: Dautenhahn and Nehaniv [17], ch. 13
19. Grollman, D.H., Jenkins, O.C.: Dogged learning for robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2007 (2007)
20. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning rhythmic movements by demonstration using nonlinear oscillators. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2002 (2002)
21. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2002 (2002)
22. Kolter, J.Z., Abbeel, P., Ng, A.Y.: Hierarchical apprenticeship learning with application to quadruped locomotion. In: Proceedings of Advances in Neural Information Proccessing, NIPS 2008 (2008)
23. Kulic, D., Nakamura, Y.: Incremental learning of full body motion primitives. In: Peters, Sigaud [31], pp. 381–404
24. Lopes, M., Melo, F., Montesano, L., Santos-Victor, J.: Abstraction levels for robotic imitation: Overview and computational approaches. In: Peters, Sigaud [31], pp. 313–355
25. Matarić, M.J.: Sensory-motor primitives as a basis for learning by imitation: Linking perception to action and biology to robotics. In: Dautenhahn, Nehaniv [17], ch. 15
26. Nehaniv, C.L., Dautenhahn, K.: The correspondence problem. In: Dautenhahn, Nehaniv [17], ch. 2
27. Ng, A.Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., Liang, E.: Inverted autonomous helicopter flight via reinforcement learning. In: International Symposium on Experimental Robotics (2004)
28. Nguyen, H.G., Morrell, J., Mullens, K., Burmeister, A., Miles, S., Thomas, K., Gage, D.W.: Segway robotic mobility platform. In: SPIE Mobile Robots XVII (2004)
29. Nicolescu, M.N., Matarić, M.J.: Methods for robot task learning: Demonstrations, generalization and practice. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2003 (2003)
30. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing 71(7-9), 1180–1190 (2008)
31. Peters, J., Sigaud, O. (eds.): From Motor Learning to Interaction Learning in Robots. SCI, vol. 264. Springer, Heidelberg (2010)
32. Ratliff, N., Bradley, D., Bagnell, J.A., Chestnutt, J.: Boosting structured prediction for imitation learning. In: Proceedings of Advances in Neural Information Processing Systems, NIPS 2007 (2007)
33. Rybski, P.E., Yoon, K., Stolarz, J., Veloso, M.M.: Interactive robot task training through dialog and demonstration. In: Proceedings of the 2nd ACM/IEEE International Conference on Human-Robot Interactions, HRI 2007 (2007)

34. Smart, W.D.: Making Reinforcement Learning Work on Real Robots. PhD thesis, Department of Computer Science, Brown University, Providence, RI (2002)
35. Stefani, R., Shahian, B., Savant, C., Hostetter, G.: Design of Feedback Control Systems. Oxford University Press, Oxford (2001)
36. Stolle, M., Atkeson, C.G.: Knowledge transfer using local features. In: Proceedings of IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, ADPRL 2007 (2007)
37. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)

Learning Continuous Grasp Affordances by Sensorimotor Exploration

R. Detry, E. Başeski, M. Popović, Y. Touati, N. Krüger, O. Kroemer, J. Peters, and J. Piater

Abstract. We develop means of learning and representing object grasp affordances probabilistically. By *grasp affordance*, we refer to an entity that is able to assess whether a given relative object-gripper configuration will yield a stable grasp. These affordances are represented with *grasp densities*, continuous probability density functions defined on the space of 3D positions and orientations. Grasp densities are registered with a visual model of the object they characterize. They are exploited by aligning them to a target object using visual pose estimation. Grasp densities are refined through experience: A robot "plays" with an object by executing grasps drawn randomly for the object's grasp density. The robot then uses the outcomes of these grasps to build a richer density through an importance sampling mechanism. Initial grasp densities, called *hypothesis* densities, are bootstrapped from grasps collected using a motion capture system, or from grasps generated from the visual model of the object. Refined densities, called *empirical* densities, represent affordances that have been confirmed through physical experience. The applicability of our method is demonstrated by producing empirical densities for two object with a real robot and its 3-finger hand. Hypothesis densities are created from visual cues and human demonstration.

1 Introduction

Grasping previously unknown objects is a fundamental skill of autonomous agents. Human grasping skills improve with growing experience with certain

R. Detry and J. Piater
University of Liège, Belgium
e-mail: Renaud.Detry@ULg.ac.be

E. Başeski, M. Popović, Y. Touati, and N. Krüger
University of Southern Denmark

O. Kroemer and J. Peters
MPI for Biological Cybernetics, Tübingen, Germany

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 451–465.
springerlink.com

objects. In this chapter, we describe a mechanism that allows a robot to learn grasp affordances [11] of objects described by learned visual models. Our first aim is to organize and memorize, independently of grasp information sources, the whole knowledge that an agent has about the grasping of an object, in order to facilitate reasoning on grasping solutions and their likelihood of success. A *grasp affordance* corresponds to the the different ways to place a hand or a gripper near an object so that closing the gripper will produce a stable grip. We represent the affordance of an object for a certain grasp type through a continuous probability density function defined on the 6D gripper pose space $SE(3)$, within an object-relative reference frame. The computational encoding is nonparametric: A density is represented by a large number of weighted samples called *particles*. The probabilistic density in a region of space is given by the local density of the particles in that region. The underlying continuous density function is accessed through kernel density estimation [28].

The second contribution of this chapter is a framework that allows an agent to learn initial affordances from various grasp cues, and enrich its grasping knowledge through experience. Affordances are initially constructed from human demonstration, or from a model-based method [1]. The grasp data produced by these *grasp sources* is used to build continuous *grasp hypothesis densities* (Section 5). These densities are registered with 3D visual object model learned beforehand [8], which allows a robotic agent to execute *samples* from a grasp hypothesis density under arbitrary object poses, by using the visual model to estimate the 3D pose of the object.

The success rate of grasp samples depends on the source that is used to produce initial grasp data. However, no existing method can claim to be perfect. For example, data collected from human demonstration will suffer from the physical and mechanical difference between a human hand and a robotic gripper. In the case of grasps computed from a 3D model, results will be impeded by errors in the model, such as missing parts or imprecise geometry. In all cases, only a fraction of the hypothesis density samples will succeed; it thus seems necessary to also learn from experience. To this end, we use samples from grasp hypothesis densities that lead to a successful grasp to learn *grasp empirical densities*, i.e. grasps that have been confirmed through experience.

A unified representation of grasp affordances can potentially lead to many different applications. For instance, a grasp planner could combine a grasp density with hardware physical capabilities (robot reachability) and external constraints (obstacles) in order to select the grasp that has the largest chance of success within the subset of achievable grasps. Another possibility is the use of continuous grasp success likelihoods to infer robustness requirements on the execution particular grasp: if a grasp is centered on a narrow peak, pose estimation and servoing should be performed with more caution than when the grasp is placed in a wide region of high success likelihood.

2 Related Work

Object grasps can emerge in many different ways. One can for instance learn 2D image patches that predict stable grasps. For example, Saxena et al. [27] have trained a classifier on a set of 2D images that were hand-labeled with good grasping points. Good grasping points are then identified in several views of an object and triangulated to infer their 3D position.

Grasping solutions can also emerge from the geometric properties of an object, typically obtained from a 3D object model. The most popular 3D model for grasping is probably the 3D mesh [15, 22], obtained e.g. from CAD or superquadrics fitting [2]. However, grasping has also successfully been achieved using models consisting of 3D surface patches [26], 3D edge segments [1], or 3D points [13]. When combined with an object pose estimation technique, the previous methods allow a robot to execute a grasp on a specific object. This involves object pose estimation, computation of a grasp on the aligned model, then servoing to the object and performing the grasp [15].

In learning a continuous grasp affordance, one has a choice between learning success probabilities or learning success-conditional grasp densities. Denoting by O a random variable encoding grasp outcomes (success or failure), and by G a random variable encoding grasp poses, this translates to learning $\mathbf{p}(O|G)$ or learning $\mathbf{p}(G|O)$. The former allows one to directly compute a probability of success. The latter allows for grasp sampling, while still providing direct means of computing *relative* success probabilities – e.g. grasp a is twice as likely to succeed as grasp b. We note that one can theoretically be computed from the other using Bayes' rule. However, depending on the means of function representation, this process may prove either too costly or too noisy to be computationally feasible.

This chapter develops a method for learning success-conditional grasp densities, closely related in spirit to the work of de Granville et al. [5]. In their work, affordances correspond to object-relative hand approach orientations, although an extension where object-relative positions are also modeled is under way [4]. The aim of the authors is to build compact sets of canonical grasp approaches from human demonstration; they mean to compress a large number of examples provided by a human teacher into a small number of clusters. An affordance is expressed through a density represented as a mixture of position-orientation kernels; machine learning techniques are used to compute mixture and kernel parameters that best fit the data. This is quite different from our approach, where a density is represented with a much larger number of simpler kernels. Conceptually, using a larger number of kernels allows us to use significantly simpler learning methods (down to mere resampling of input data, see Section 5.1). Also, the representation of a grasp cluster through a single position-orientation kernel requires the assumption that hand position and orientation are independent within the cluster, which is generally not true. Representing a cluster with many

particles can intrinsically capture more of the position-orientation correlation (see Section 6, and in particular Fig. 6). The affordance densities presented by de Granville et al. correspond to the hypothesis densities developed in this chapter.

Instead of considering success-conditional grasp probabilities, Montesano et al. [23] formalize grasp affordances as success probabilities $\mathbf{p}(O|I)$, where I is a local image patch. A robot thus learns a mapping from 2D image patches to grasp success probabilities, where a grasp is parametrized by its 2D gripper position. From a procedural viewpoint, the method of Montesano et al. differs from ours in its explicit exploitation of failed grasps, whereas in our work, empirical densities are learned from successful grasps only. We note that, in a probabilistic sense, our learning method does take failed grasps into account, through the absence of learning data in regions where grasps were sampled and failed. However, we agree that making active use of failed trials may increase robustness, and we intend to evaluate this option in future work. Another promising avenue lies in active learning of grasp options, as demonstrated by Kroemer et al. [16].

Learning grasp affordances from experience was also demonstrated by Stoytchev [29, 30]. In his work, a robot discovers successful grasps through random exploratory actions on a given object. When subsequently confronted with the same object, the robot is able to generate a grasp that should present a high likelihood of success.

In this chapter, learning may be bootstrapped with grasp data provided by a motion capture system, a process that constitutes a simple form of imitation learning. For a broader discussion of imitation learning, we refer the reader to two dedicated chapters within this collection [20, 19].

The system developed in this chapter is build on a set of existing methods which are described in Section 3. The visual object model to which affordances are attached is the part-based model of Detry et al. [8] (Section 3.3). An object is modeled with a hierarchy of increasingly expressive object parts called *features*. The single top feature of a hierarchy represents the whole object. Features at the bottom of the hierarchy represent short 3D edge segments for which evidence is collected from stereo imagery via the Early-Cognitive-Vision (ECV) system of Krüger et al. [17, 25] (Section 3.1). In the following, we refer to these edge segments as *ECV descriptors*. The hierarchical model grounds its visual evidence in ECV reconstructions: a model is learned from segmented ECV descriptors, and the model can be used to recover the pose of the object within an ECV representation of a cluttered scene.

The mathematical representation of grasp densities and their association to hierarchical object models is discussed in Section 4. In Section 5, we demonstrate the learning and refining of grasp densities from two grasp sources. The first source is imitation of human grasps. The second source uses a model-based algorithm which extracts grasping cues from an ECV reconstruction (Section 3.2).

(a) ECV descriptors **(b)** Accumulated reconstructions

Fig. 1 ECV reconstruction. Each ECV descriptor is rendered with a small plane patch. Patch normals are not part of ECV descriptors; they are arbitrarily defined for the purpose of 3D rendering.

3 Methods

This section briefly describes the methods that are brought together for modeling the visual percepts of an object, and for bootstrapping hypothesis densities from visual cues. These sophisticated methods have proved essential for a robust execution of grasps on arbitrary objects in arbitrary poses.

3.1 Early Cognitive Vision

ECV descriptors [17, 25] represent short edge segments in 3D space, each ECV descriptor corresponding to a circular image patch with a 7-pixel diameter. To create an ECV reconstruction, pixel patches are extracted along image contours, within images captured with a calibrated stereo camera. The ECV descriptors are then computed with stereopsis across image pairs; each descriptor is thus defined by a 3D position and 3D edge orientation. Descriptors may be tagged with color information, extracted from their corresponding 2D patches (Fig. 1a).

ECV reconstructions can further be improved by manipulating objects with a robot arm, and *accumulating* visual information across several views through structure-from-motion techniques [12]. Assuming that the motion adequately spans the object pose space, a complete 3D-edge reconstruction of the object can be generated, eliminating self-occlusion issues [14] (see Fig. 1b).

3.2 Grasp Reflex From Co-planar ECV Descriptors

Pairs of ECV descriptors that are on the same plane and which have color information such that two similar colors are pointing towards each other can be used to define grasps. Grasp position is defined by the location of one of the descriptors. Grasp orientation is calculated from the normal of the plane linking the two descriptors, and the orientation of the descriptor at which the

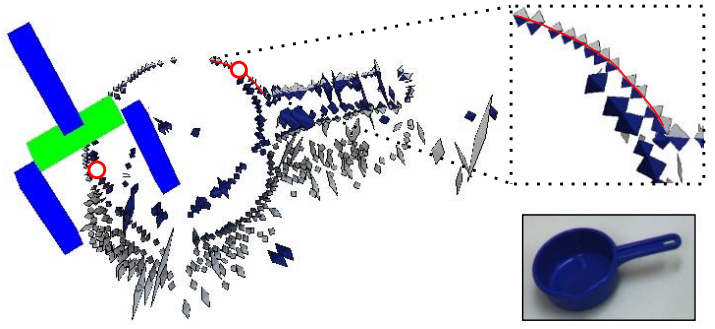


Fig. 2 Grasp reflex based on visual data. Each ECV descriptor is rendered with a small plane patch. Patch normals are not part of ECV descriptors; they are arbitrarily defined for the purpose of 3D rendering.

grasp is located [14] (see Fig. 2). The grasps generated by this method will be referred to as *reflexes*. Since each pair of co-planar descriptors generates multiple reflexes, a large number of these are available.

3.3 Feature Hierarchies for 3D Visual Object Representation

As explained in Section 3.1, an ECV reconstruction models a scene or an object with low-level descriptors. This section outlines a higher-level 3D object model [8] that grounds its visual evidence in ECV representations.

An object is modeled with a hierarchy of increasingly expressive object parts called *features*. Features at the bottom of the hierarchy (*primitive* features) represent ECV descriptors. Higher-level features (*meta*-features) represent geometric configurations of more elementary features. The single top feature of a hierarchy represents the object.

Unlike many part-based models, a hierarchy consists of features that may have several *instances* in a scene. To illustrate this, let us consider a part-based model of a bike, in which we assume a representation of wheels. Traditional part-based models [10, 3] would work by creating two wheel parts – one for each wheel. Our hierarchy however uses a single *generic* wheel feature; it stores the information on the existence of *two* wheels *within* the wheel feature. Likewise, a primitive feature represents a *generic* ECV descriptor, e.g. any descriptor that has a red-like color. While an object like the basket of Fig. 1b produces hundreds of red ECV descriptors, a hierarchy representing the basket will, in its simplest form, contain a single red-like primitive feature; it will encode internally that this feature has many instances within a basket object.

A hierarchy is implemented in a Markov tree. Features correspond to hidden nodes of the network; when a model is associated to a scene (during

learning or detection), the pose distribution of feature i in the scene is represented through a random variable X_i. Random variables are thus defined over the pose space, which exactly corresponds to the Special Euclidean group $SE(3) = \mathbb{R}^3 \times SO(3)$. The random variable X_i associated to feature i effectively links that feature to its instances: X_i represents as one probability density function the pose distribution of all the instances of feature i, therefore avoiding specific model-to-scene correspondences.

The geometric relationship between two neighboring features i and j is encoded in a compatibility potential $\psi_{ij}(X_i, X_j)$. A compatibility potential represents the pose distribution of all the instances of the child feature in a reference frame defined by the parent feature; potentials are thus also defined on $SE(3)$.

The only observable features are primitive features, which receive evidence from the ECV system. Each primitive feature i is linked to an observed variable Y_i; the statistical dependency between a hidden variable X_i and its observed variable Y_i is encoded in an observation potential $\phi_i(X_i)$, which represents the pose distribution of ECV descriptors that have a color similar to the color of primitive feature i.

Density functions (random variables, compatibility potentials, observation potentials) are represented nonparametrically: a density is represented by a set of particles [8].

3.4 Pose Estimation

The hierarchical model presented above can be used to estimate the pose of a known object in a cluttered scene. Estimating the pose of an object amounts to deriving a posterior pose density for the top feature of its hierarchy, which involves two operations [8]:

1. Extract ECV descriptors, and transform them into observation potentials.
2. Propagate evidence through the graph using an applicable inference algorithm.

Each observation potential $\phi_i(X_i)$ is built from a subset of the early-vision observations. The subset that serves to build the potential $\phi_i(X_i)$ is the subset of ECV descriptors that have a color that is close enough to the color associated to primitive feature i.

Evidence is propagated through the hierarchy using a belief propagation (BP) algorithm [24, 31]. BP works by exchanging *messages* between neighboring nodes. Each message carries the belief that the sending node has about the pose of the receiving node. In other words, a message allows the sending feature to probabilistically vote for all the poses of the receiving feature that are consistent with its own pose – consistency being defined by the compatibility potential through which the message flows. Through message passing, BP propagates collected evidence from primitive features to the top of the hierarchy; each feature probabilistically votes for all possible object configurations consistent with its pose density. A consensus emerges among the

available evidence, leading to one or more consistent scene interpretations. The pose likelihood for the whole object is eventually read out of the top feature; if the object is present twice in a scene, the top feature density should present two major modes. The global belief about the object pose may also be propagated from the top node down the hierarchy, reinforcing globally consistent evidence and permitting the inference of occluded features.

Algorithms that build hierarchies from accumulated ECV reconstructions are discussed in prior work [7].

4 Representing Grasp Densities

This section is focused on the probabilistic representation of grasp affordances. By *grasp affordance*, we refer to the different ways to place a hand or a gripper near an object so that closing the gripper will produce a stable grip. The grasps we consider are parametrized by a 6D gripper pose composed of a 3D position and a 3D orientation.

From a mathematical point of view, grasp densities are identical to the visual potentials of Section 3.3. They can thus be encoded with the same nonparametric representation. Density evaluation is performed by assigning a kernel function to each particle supporting the density, and summing the evaluation of all kernels. Sampling from a distribution is performed by sampling from the kernel of a particle ℓ selected from $\mathbf{p}(\ell = i) \propto w^i$, where w^i is the weight of particle i.

Grasp densities are defined on the Special Euclidean group $SE(3) = \mathbb{R}^3 \times SO(3)$, where $SO(3)$ is the Special Orthogonal group (the group of 3D rotations). We use a kernel that factorizes into two functions defined on $\mathbb{R}^3$ and $SO(3)$. Denoting the separation of an $SE(3)$ point x into a translation λ and a rotation θ by

$$x = (\lambda, \theta), \quad \mu = (\mu_t, \mu_r), \quad \sigma = (\sigma_t, \sigma_r),$$

we define our kernel with

$$\mathbf{K}(x; \mu, \sigma) = \mathbf{N}(\lambda; \mu_t, \sigma_t)\, \mathbf{\Theta}(\theta; \mu_r, \sigma_r) \tag{1}$$

where μ is the kernel mean point, σ is the kernel bandwidth, $\mathbf{N}(\cdot)$ is a trivariate isotropic Gaussian kernel, and $\mathbf{\Theta}(\cdot)$ is an orientation kernel defined on $SO(3)$. Denoting by θ' and μ_r' the quaternion representations of θ and μ_r [18], we define the orientation kernel with the Dimroth-Watson distribution [21]

$$\mathbf{\Theta}(\theta; \mu_r, \sigma_r) = \mathbf{W}(\theta'; \mu_r', \sigma_r) = C_w(\sigma_r) e^{\sigma_r (\mu_r'^{\top} \theta')^2} \tag{2}$$

where $C_w(\sigma_r)$ is a normalizing factor. This kernel corresponds to a Gaussian-like distribution on $SO(3)$. The Dimroth-Watson distribution inherently handles the double cover of $SO(3)$ by quaternions [5].

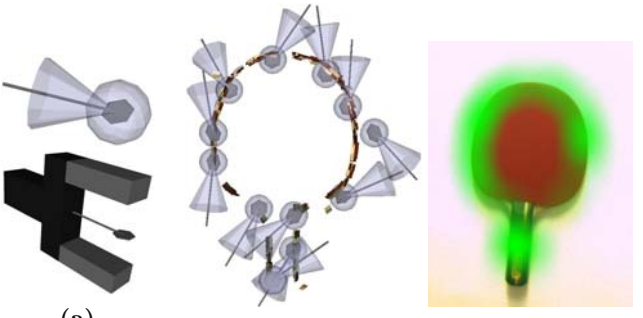


Fig. 3 Grasp density representation. The top image of Fig. (a) illustrates a particle from a nonparametric grasp density, and its associated kernel widths: the translucent sphere shows one position standard deviation, the cone shows the variance in orientation. The bottom image illustrates how the schematic rendering used in the top image relates to a physical gripper. Fig. (b) shows a 3D rendering of the kernels supporting a grasp density for a table-tennis paddle (for clarity, only 12 kernels are rendered). Fig. (c) indicates with a green mask of varying opacity the values of the location component of the same grasp density along the plane of the paddle (orientations were ignored to produce this last illustration).

The bandwidth σ associated to a density should ideally be selected jointly in $\mathbb{R}^3$ and $SO(3)$. However, this is difficult to do. Instead, we set the orientation bandwidth σ_r to a constant allowing about 10° of deviation; the location bandwidth σ_t is then selected using a k-nearest neighbor technique [28].

The expressiveness of a single $SE(3)$ kernel (1) is rather limited: location and orientation components are both isotropic, and within a kernel, location and orientation are modeled independently. Nonparametric methods account for the simplicity of individual kernels by employing a large number of them: a grasp density will typically be supported by a thousand particles. Fig. 3a shows an intuitive rendering of an $SE(3)$ kernel from a grasp density. Fig. 3b and Fig. 3c illustrate continuous densities.

Grasp densities are defined in the same reference frame as visual features. Once visual features have been aligned to an object pose (Section 3.4), the object grasp density can be similarly aligned, and one can readily draw grasps from the density and execute them onto the object. A deeper integration of the visual model with grasp densities has been covered in prior work [6].

5 Learning Grasp Densities

This section explains how hypothesis densities are learned from source data (Section 5.1), and how empirical densities are learned from experience (Section 5.2).

5.1 Hypothesis Densities from Examples

Initial grasp knowledge, acquired for instance from imitation or reflex, is structured as a set of grasps parametrized by a 6D pose. Given the non-parametric representation, building a density from a set of grasps is straightforward – grasps can directly be used as particles representing the density. We typically limit the number of particles in a density to a thousand; if the number of grasps in a set is larger than 1000, the density is *resampled*: kernels are associated the particles, and 1000 samples are drawn and used as a representation replacement.

5.2 Empirical Densities through Familiarization

As the name suggests, hypothesis densities do not pretend to reflect the true properties of an object. Their main defect is that they may strongly suggest grasps that might not be applicable at all, for instance because of gripper discrepancies in imitation-based hypotheses. A second, more subtle issue is that the grasp data used to learn hypothesis densities will generally be afflicted with a source-dependent spatial bias. A very good example can be made from the reflex computation of Section 3.2. Reflexes are computed from ECV visual descriptors. Therefore, parts of an object that have a denser visual resolution will yield more reflexes, incidentally biasing the corresponding region of the hypothesis density to a higher value. The next paragraph explains how grasping experience can be used to compute new densities (*empirical* densities) that better reflect gripper-object properties.

Empirical densities are leaned from the execution of *samples* from a hypothesis density, intuitively allowing the agent to familiarize itself with the object by discarding wrong hypotheses and refining good ones. Familiarization thus essentially consists in autonomously learning an *empirical* density from the outcomes of sample executions. A simple way to proceed is to build an empirical density directly from successful grasp samples. However, this approach would inevitably propagate the spatial bias mentioned above to empirical densities. Instead, we use importance sampling [9] to properly weight grasp outcomes, allowing us to draw samples from the physical grasp affordance of an object. The weight associated to a grasp sample x is computed as $\mathbf{a}(x) \,/\, \mathbf{q}(x)$, where $\mathbf{a}(x)$ is 1 if the execution of x has succeeded, 0 else, and $\mathbf{q}(x)$ corresponds to the value of the continuous hypothesis density at x. A set of these weighted samples directly forms a grasp empirical density that faithfully and uniformly reflects intrinsic gripper-object properties.

5.3 Updating Empirical Densities in Long-Term Interaction

In long-term interaction, a robot is constantly producing new evidence which should ideally be used to continuously enhance empirical densities. The

methodology presented above can easily be adapted for basic long-term learning. Essentially, the solution stems from observing that the IS learning of empirical densities may in fact use any arbitrary function as hypothesis density. In an initial learning cycle, the hypothesis density is computed from grasp cues. Let us call this initial hypothesis density the *bootstrap* density. In the next learning cycle, the empirical density from the first cycle may be linearly combined with the bootstrap density to form a new hypothesis density that represents a trade-off between exploration of new possibilities and safe reproduction of known experience. Once enough samples from the new hypothesis density have been experienced, the empirical density can be replaced by an updated representation. In long-term interaction, hypothesis densities are thus successively computed as weighted sums of the current empirical density and the bootstrap density. Giving a high weight to the empirical density triggers top-down learning, i.e. refining globally known affordance. Conversely, focusing on the bootstrap density corresponds to bottom-up learning, i.e. integrating new low-level evidence into the model.

6 Results

This section illustrates hypothesis densities learned from imitation and reflexes, and empirical densities are learned by grasping objects with a 3-finger Barrett hand. Densities are built for two objects: the table-tennis paddle of Fig. 3, and a toy plastic jug (Fig. 5). The experimental scenario is described below.

For each object, the experiment starts with a visual hierarchical model, and a set of grasps. For the paddle, grasps are generated with the method described in Section 3.2. Initial data for the jug was collected through human

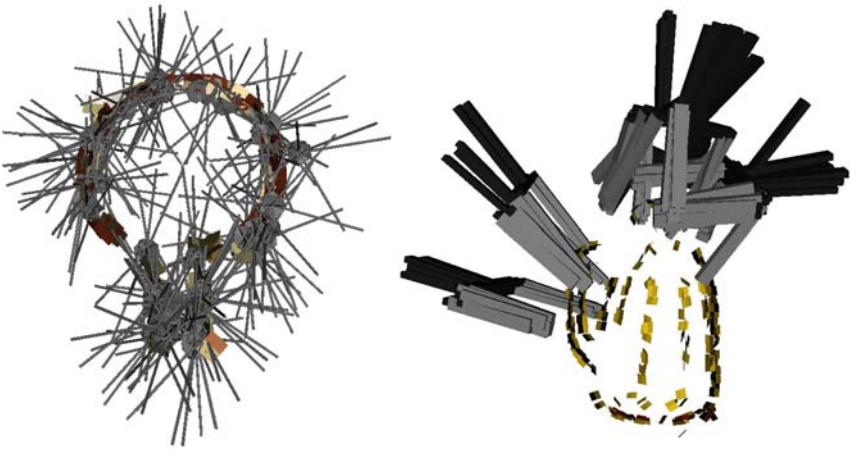


Fig. 4 Particles supporting grasp hypothesis densities.

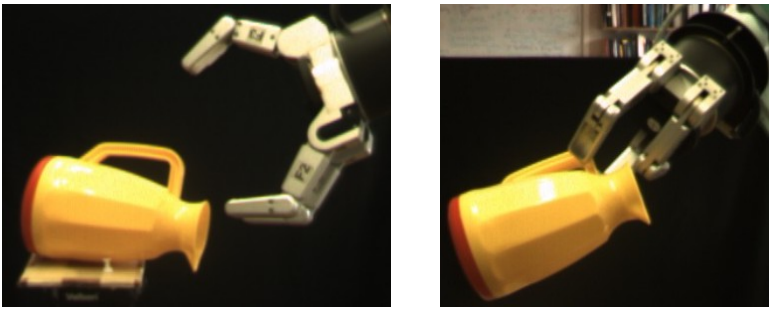


Fig. 5 Barrett hand grasping the toy jug.

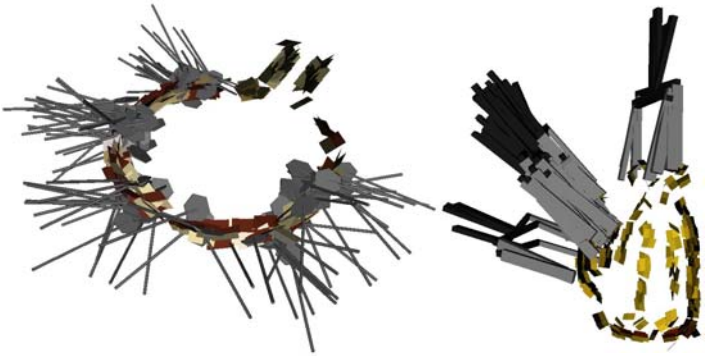


Fig. 6 Samples drawn from grasp empirical densities.

demonstration, using a motion capture system. From these data, a hypothesis density is built for each object. The particles supporting the hypothesis densities are rendered in Fig. 4.

In order to refine affordance knowledge, feedback on the execution of hypothesis density samples is needed. Grasps are executed with a Barrett hand mounted on an industrial arm. As illustrated in Fig. 5, the hand preshape is a parallel-fingers, opposing-thumb configuration. The reference pose of the hand is set for a pinch grasp, with the tool center point located in-between the tips of the fingers – similar to the reference pose illustrated in Fig. 3a. A grasp is considered successful if the robot is able to firmly lift up the object, success being asserted by raising the robotic hand while applying a constant, inward force to the fingers, and checking whether at least one finger is not fully closed. Sets of 100 and 25 successful grasps were collected for the paddle and the jug respectively. This information was then used to build a grasp empirical density, following the procedure described in Section 5.2. Samples from the resulting empirical densities are shown in Fig. 6. For the paddle, the main evolution from hypothesis to empirical density is the removal of a large number of grasps for which the gripper wrist collides with the paddle body. Grasps presenting a steep approach relative to the plane of the paddle were

also discarded, thereby preventing fingers from colliding with the object during hand servoing. None of the pinch-grasps at the paddle handle succeeded, hence their absence from the empirical density.

While grasping the top of the jug is easy for a human hand, it proved to be very difficult for the Barrett hand with parallel fingers and opposing thumb. Consequently, a large portion of the topside grasps suggested by the hypothesis density are not represented in the empirical density. The most reliable grasps approach the handle of the jug from above; these grasps are strongly supported in the empirical density.

The left image of Fig. 6 clearly illustrates the correlation between grasp positions and orientations: moving along the edge of the paddle, grasp approaches are most often roughly perpendicular to the local edge tangent. The nonparametric density representation successfully captures this correlation.

7 Conclusion and Future Work

We presented a framework for representing and learning object grasp affordances probabilistically. The affordance representation is nonparametric: an affordance is recorded in a continuous probability density function supported by a set of particles.

Grasp densities are initially learned from visual cues or imitation, leading to grasp hypothesis densities. Using the visual model for pose estimation, an agent is able to execute *samples* from a hypothesis density under arbitrary object poses. Observing the outcomes of these grasps allows the agent to learn from experience: an importance sampling algorithm is used to infer faithful object grasp properties from successful grasp samples. The resulting *grasp empirical densities* eventually allow for more robust grasping.

Importance Sampling is a batch learning method, that requires the execution of a large number of grasps before an empirical density can be built. Learning empirical densities *on-line* would be very convenient, which is a path we plan to explore next.

We currently learn visual and grasp models independently. Yet, a part-based representation offers an elegant way to *locally* encode visuomotor descriptions. One of our goals is to learn visual parts that share the same grasp properties across different objects. This way, a grasp *feature* would be directly and exclusively connected to the visual evidence that predicts its applicability, allowing for its generalization across objects.

Acknowledgments

This work was supported by the Belgian National Fund for Scientific Research (FNRS) and the EU Cognitive Systems project PACO-PLUS (IST-FP6-IP-027657). We thank Volker Krüger and Dennis Herzog for their support during the recording of the human demonstration data.

References

1. Aarno, D., Sommerfeld, J., Kragic, D., Pugeault, N., Kalkan, S., Wörgötter, F., Kraft, D., Krüger, N.: Early reactive grasping with second order 3D feature relations. In: The IEEE International Conference on Advanced Robotics (2007)
2. Biegelbauer, G., Vincze, M.: Efficient 3D object detection by fitting superquadrics to range image data for robot's object manipulation. In: IEEE International Conference on Robotics and Automation (2007)
3. Bouchard, G., Triggs, B.: Hierarchical part-based visual object categorization. Computer Vision and Pattern Recognition 1, 710–715 (2005)
4. de Granville, C., Fagg, A.H.: Learning grasp affordances through human demonstration. Submitted to the Journal of Autonomous Robots (2009)
5. de Granville, C., Southerland, J., Fagg, A.H.: Learning grasp affordances through human demonstration. In: Proceedings of the International Conference on Development and Learning, ICDL 2006 (2006)
6. Detry, R., Başeski, E., Krüger, N., Popović, M., Touati, Y., Kroemer, O., Peters, J., Piater, J.: Learning object-specific grasp affordance densities. In: International Conference on Development and Learning (2009)
7. Detry, R., Piater, J.H.: Hierarchical integration of local 3D features for probabilistic pose recovery. In: Robot Manipulation: Sensing and Adapting to the Real World (Workshop at Robotics, Science and Systems) (2007)
8. Detry, R., Pugeault, N., Piater, J.: A probabilistic framework for 3D visual object representation. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (2009)
9. Doucet, A., de Freitas, N., Gordon, N.: Sequential Monte Carlo Methods in Practice. Springer, Heidelberg (2001)
10. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient matching of pictorial structures. In: Conference on Computer Vision and Pattern Recognition (CVPR 2000), p. 2066 (2000)
11. Gibson, J.J.: The Ecological Approach to Visual Perception. Lawrence Erlbaum Associates, Mahwah (1979)
12. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
13. Huebner, K., Ruthotto, S., Kragic, D.: Minimum volume bounding box decomposition for shape approximation in robot grasping. In: IEEE International Conference on Robotics and Automation, 2008. ICRA (2008)
14. Kraft, D., Pugeault, N., Başeski, E., Popović, M., Kragic, D., Kalkan, S., Wörgötter, F., Krüger, N.: Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes. Special Issue on "Cognitive Humanoid Robots" of the International Journal of Humanoid Robotics (2008)
15. Kragic, D., Miller, A.T., Allen, P.K.: Real-time tracking meets online grasp planning. In: Proceedings of the 2001 IEEE International Conference on Robotics and Automation, pp. 2460–2465 (2001)
16. Kroemer, O., Detry, R., Piater, J., Peters, J.: Active learning using mean shift optimization for robot grasping. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2009)
17. Krüger, N., Lappe, M., Wörgötter, F.: Biologically Motivated Multi-modal Processing of Visual Primitives. The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour 1(5), 417–428 (2004)

18. Kuffner, J.: Effective sampling and distance metrics for 3D rigid body path planning. In: Proc. 2004 IEEE Int'l Conf. on Robotics and Automation (ICRA 2004), May 2004. IEEE, Los Alamitos (2004)
19. Lallee, S., Yoshida, E., Mallet, A., Nori, F., Natale, L., Metta, G., Warneken, F., Dominey, P.F.: Human-robot cooperation based on interaction learning. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 491–536. Springer, Heidelberg (2010)
20. Lopes, M., Melo, F., Montesano, L., Santos-Victor, J.: Abstraction Levels for Robotic Imitation: Overview and computational approaches. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 313–355. Springer, Heidelberg (2010)
21. Mardia, K.V., Jupp, P.E.: Directional Statistics. Wiley Series in Probability and Statistics. Wiley, Chichester (1999)
22. Miller, A.T., Knoop, S., Christensen, H., Allen, P.K.: Automatic grasp planning using shape primitives. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2003, vol. 2, pp. 1824–1829 (2003)
23. Montesano, L., Lopes, M.: Learning grasping affordances from local visual descriptors. In: International Conference on Development and Learning (2009)
24. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco (1988)
25. Pugeault, N.: Early Cognitive Vision: Feedback Mechanisms for the Disambiguation of Early Visual Representation. Vdm Verlag Dr. Müller (2008)
26. Richtsfeld, M., Vincze, M.: Robotic grasping based on laser range and stereo data. In: International Conference on Robotics and Automation (2009)
27. Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic Grasping of Novel Objects using Vision. The International Journal of Robotics Research 27(2), 157 (2008)
28. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman & Hall/CRC, Boca Raton (1986)
29. Stoytchev, A.: Toward learning the binding affordances of objects: A behavior-grounded approach. In: Proceedings of AAAI Symposium on Developmental Robotics, Stanford University, March 21-23, pp. 17–22 (2005)
30. Stoytchev, A.: Learning the affordances of tools using a behavior-grounded approach. In: Rome, E., Hertzberg, J., Dorffner, G. (eds.) Towards Affordance-Based Robot Control. LNCS (LNAI), vol. 4760, pp. 140–158. Springer, Heidelberg (2008)
31. Sudderth, E.B., Ihler, A.T., Freeman, W.T., Willsky, A.S.: Nonparametric belief propagation. In: Computer Vision and Pattern Recognition. IEEE Computer Society, Los Alamitos (2003)

Multimodal Language Acquisition Based on Motor Learning and Interaction

Jonas Hörnstein, Lisa Gustavsson, José Santos-Victor, and Francisco Lacerda

Abstract. This work presents a developmental and ecological approach to language acquisition in robots, which has its roots in the interaction between infants and their caregivers. We show that the signal directed to infants by their caregivers include several hints that can facilitate the language acquisition and reduce the need for preprogrammed linguistic structure. Moreover, infants also produce sounds, which enables for richer types of interactions such as imitation games, and for the use of motor learning. By using a humanoid robot with embodied models of the infant's ears, eyes, vocal tract, and memory functions, we can mimic the adult-infant interaction and take advantage of the inherent structure in the signal. Two experiments are shown, where the robot learn a number of word-object associations and the articulatory target positions for a number of vowels.

1 Introduction

Language acquisition is a complex process that involves several tasks, such as producing speech sounds, learning how to group different sounds into a consistent and manageable number of classes or speech units, grounding speech, and recognizing the speech sounds when uttered by other persons. Despite tremendous research effort in the area of automatic speech recognition (ASR), machines are still far from human language capabilities in terms of robustness and flexibility. Unfortunately, this may not be a result of insufficient processing power or insufficient speech samples in the training database, but rather a fundamental flaw in the architecture of current models [42]. Children are able to acquire impressive language skills from very little speech exposure. This poverty of the stimulus has lead researchers to

Jonas Hörnstein and José Santos-Victor
Institute for System and Robotics (ISR), Instituto Superior Técnico, Lisbon, Portugal
e-mail: {jhornstein,jasv}@isr.ist.utl.pt

Lisa Gustavsson and Francisco Lacerda
Department of Linguistics, Stockholm University, Stockholm, Sweden
e-mail: {lisag,frasse}@ling.su.se

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 467–489.
springerlink.com

believe that children do not learn language automatically from mere exposure, as is mostly the case in current ASR-systems. Nativists argue that language is essentially preprogrammed in the human brain and that acquisition mainly consists of setting the parameters for the specific target language. On the other hand, supporters of the ecological and emergent perspective believe that the language acquisition is guided mainly by biological constraints and through interactions, and that it does not depend on any preprogrammed linguistic knowledge.

While the level of preprogrammed versus learned language structure is still a source of debate, this work aims at two intertwined goals: (i) to investigate how much of linguistic structure that can be derived directly from the speech signal directed to infants by (ii) designing, building and testing computational models for language acquisition in a humanoid robot. While this is a very broad and challanging scope, our ambition at this stage is mainly to provide an architecture that can be implemented in a robotic platformed in order to further study these topics, and to provide a few concrete examples where robot is able to learn word-object relations and articulatory target positions for a number of vowels. In line with the ecological and emergent approach we have therefore avoided to implement preprogrammed linguistic knowledge, such as phonemes, in the robotic architecture. Instead we make use of embodiment and interactions in order to find the structure using general methods such as pattern matching and hierarchical clustering techniques.

The interaction between an adult caregiver and an infant is very different from the interaction between two adults. Speech directed to infants is highly structured and characterized by what seems like physically motivated tricks to maintain the communicative connection to the infant, actions that at the same time also may enhance linguistically relevant important aspects of the signal. Also, whereas communication between adults is usually about exchanging information, speech directed to infants is more of a referential nature. The adult refers to objects, people and events in the world surrounding the infant [33]. Because of this, the sound sequences the infant hears are very likely to co-occur with actual objects or events in the infant's visual field. The expanded intonation contours, the repetitive structure of Infant Directed Speech (IDS) and the modulation of the sentence intensity are likely to play an important role in helping the infant establishing an implicit and plausible word-object link. This kind of structuring might very well be one of the first steps in speech processing, a coarse segmenting of the continuous signal in chunks that stand out because of some recurrent pattern the infant learns to recognize. Infants are very sensitive to the characteristic qualities of this typical IDS style, and a number of studies indicate that infants use this kind of information to find implicit structure in acoustic signals [27] [11] [32] [6] [47]. Some evidence on the usefulness of these co-ocurring events can also be found in robotics. In the CELL [46], Cross-channel Early Lexical Learning, an architecture for processing multisensory data is developed and implemented in a robot called Toco the Toucan. The robot is able to acquire words from untranscribed acoustic and video input and represent them in terms of associations between acoustic and visual sensory experience. Compared to conventional ASR systems that maps speech signal to human specified labels, this is an important step towards creating more ecological models. However, important

shortcuts are still taken, such as the use of a predefined phoneme-model where a set of 40 phonemes are used and the transition probabilities are trained off-line on large scale database. In [52], no external database is used. Instead the transition probabilities are trained online only taking into account utterances that have been presented to the system at the specific instance in time. While this make the model more plausible from a cognitive perspective, infants may not rely on linguistic concepts as phonemes at all during these early stages of language development. In this work we have instead chosen a more direct approach and map the auditory impression of the word as a whole to the object. Underlying concepts like phonemes instead are seen as emergent consequences imposed by increasing representation needs [45] [35].

In this work we have chosen to represent those underlying structures, i.e. pseudo-phonemes, in the form of target position in motor space, rather than as auditory goals. The rationale for this can be found in the motor theory of speech perception [37], which hypothesizes that we recognize speech sound by first mapping the sound to our motor capabilities. This statement is also supported by more recent work in neuroscience that demonstrates an increased activity in the tongue muscles when listening to words that requires large tongue movements [9]. This leads to believe that the motor area in the brain is involved not only in the task of production, but also in that of recognition. Earlier works including neurophysiologic studies of monkeys have shown a similar relationship between visual stimulation and the activation of premotor neurons [14]. Those neurons, usually referred to as mirror neurons, fire both when executing a motor command and when being presented with an action that involves the same motor command. To learn the audio-motor maps and finding key target positions, interactions again play an important role in the form of imitation games. Already during the first year a child slowly starts to mix what can be perceived as spontaneous babbling and some early imitations of the caregivers. During the second year as much as one third of the utterances produced by the child can be perceived as imitations, [55]. Broadly speaking, these “imitation games" where the child may play either the role of demonstrator or that of the imitator, serve two different purposes: (i) learning elementary speech units or speech vocabulary and (ii) gaining inter-speaker invariance for recognition.

In the remainder of the chapter we first take a closer look at the interaction between the infant and a caregiver and discuss how these interactions can guide the infant’s language acquisition. We then discuss the learning architecture and necessary resources that has to be implemented in a robot for it to be able to acquire early language structures through the described interactions. Finally we show how a humanoid can use the architecture and interaction scenarios to learn word-object relations and extract target positions for a number of vowels.

2 The Role of Interaction

In this section we take a closer look at how interaction can facilitate language acquisition, more specifically we look at the interaction between a caregiver and the child (or robot). When interacting with a child, adults tend to adapt their speech signal in

a way that is attractive to the infant and in the same time can be helpful for learning the language. It is therefore natural to start this section with a review of some of the typical aspects of IDS.

This interaction it typcially multimodal in its nature. The child does not only hear the caregiver, but receives multimodal information such as combinations of speech, vision, and tactile input. In this section we discuss how this multimodal information can be used to ground speech. This kind of interaction, based on shared attention, starts even before the infant can produce speech itself. However, as the infant starts to explore its capacity to produce sound, verbal interaction will become more and more important. Initial verbal interaction is mainly based on imitations. Here we separate between two types of imitation scenarios, one where the infant imitates its caregiver, and one where the caregiver imitates the infants. Both have been found equally common in adult-infant interactions, but they serve two different goals (i) to learn how to produce sounds that are useful for communication and (ii) to learn how to map the sound of the caregiver to the own articulatory positions.

2.1 Infant Directed Speech

An important portion of the physical signal in the ambient language of almost every infant is in the form of Infant Directed Speech (IDS), a typical speech style used by adults when communicating with infants. IDS is found in most languages [2] [31] [10] and is characterized by long pauses, repetitions, high fundamental frequency, exaggerated fundamental frequency contours [11] and hyperarticulated vowels [31]. A very similar speech style is found in speech directed to pets [22] [4], and to some degree also in speech directed to humanoid robots [17], and pet robots [3].

The function of IDS seems to change in accordance with the infant's developmental stages, phonetic characteristics in the adult's speech are adjusted to accommodate the communicative functions between the parents and their infants, for example a gradual change in consonant specifications associated with the infants communicative development was found in a study by Sundberg and Lacerda [50]. In longitudinal studies it has been shown that parents are adapting their speech to their infants linguistic and social development the first post-natal year. On the whole they use higher fundamental frequency, greater frequency range, shorter utterance duration, longer syllable duration, and less number of syllables per utterance when speaking to their infants as compared to speaking to adults. Sundberg [51] suggests that these phonetic modifications might be an intuitive strategy adults use automatically that is both attractive and functional for the infant.

In a study more directly related to infants word learning, Fernald and Mazzie [13] found that target words in infant directed speech were typically highlighted using focal stress and utterance-final position. In their study 18 mothers of 14-month-old infants were asked to tell a story from a picture book called Kelly's New Clothes, both to their infants and to an adult listener. Each page of the book introduced a new piece of clothes that was designated as a target word. When telling the story to the infants target words were stressed in 76% of the instances, and placed in

utterance-final position in 75% of the instances. For adult speech the same values were 40% and 53% respectively. Albin [1] found that an even larger portion of the target words (87% - 100% depending of subject) occured in final position when the subjects were asked to present a number of items to an infant. There are some indications that infants take advantage of these features and have easier to learn target words with focal stress and utterance final position [15]. It can therefore be motivated for a robot that mimics infant learning to also have a preference for words with focal stress and in utterance final position.

2.2 Multimodal Interaction

Whereas communication between adults usually is about exchanging information, speech directed to infants is of a more referential nature. The adult refers to objects, people and events in the world surrounding the infant [33]. When for example playing with a toy, the name of the toy is therefore likely to be mentioned several times within a relatively short time as the infant is being introduced to toy. Finding such recurrent patterns in the sound stream coming from the caregiver can help the infant to extract potential word candidates that can be linked to the visual representation of the object. On the other hand, also words that are not directly related to the object may be mentioned repeatably to the same extent or even more often than the target word itself. By linking all recurrent sound patterns to the most salient object in the visual field we are likely to end up with a large number of false word-object links. However, if the same wordlike pattern consistantly appears when a certain object is observed, and only to a less degree when the object is not present, it is highly likely that the pattern is actually related to the object. While many of the recurrent patterns are likely to be found closely in time, it may be necessary to look for cross-modal regularities over a relatively long time in order to separate target words from other recurrent word pattern.

A robot may therefore need both a short term memory that is searched for recurrent patterns and a long-term memory that is searched for cross-modal regularities in order to form word-object associations, as is the case in the in the CELL-model [46].

2.3 Babbling and Imitation

One of the first key observations regarding a child's language development is the use of babbling [36], an exploration process that allows the child to explore different actuations of the vocal-tract and the corresponding acoustic consequences. This process allows to build sensorimotor maps that associate the articulatory parameters of the vocal tract and the produced sounds. In other words, through babbling the child (or the robot) learns the physics of the vocal tract and how to produce sounds. While babbling was first seen as an isolated process, it has later been shown to have a continuous importance for the vocal development [54]. It has also been shown that in order to babble normally, children need to be able not only to hear both themselves and other con-specifics [49], but also to establish visual contact

with others [43]. Speech babbling could therefore be seen as an early type of interaction, instead of just a self exploration task. When a caregiver is present, he or she is likely to imitate the sound of the infant, giving the infant the possibility to also create a map between its own utterances and that of the caregiver. Imitation studies performed at Stockholm University has shown that in about 20% of the cases, an eventual response from the caregiver is seen as an imitation of the infant's utterance when judged by other adult listeners. We have previously shown that it is possible to get a good estimation of when there is an imitation or not by comparing a number of prosodic features [25].

By creating speech sound and detecting possible imitations, the child or robot can overcome differences between its own voice and that of the caregiver, and by repeating this kind of "imitation game" with several different caregivers it is also possible to overcome inter-speaker variations.

As stated in the introduction of this section, there are two different goals with the imitation games. Apart from the goal to overcome the inter-speaker differences and allow the robot to correctly map sounds from different speakers to its own motor positions in order to reproduce or recognize those sounds, the second goal is to help the robot to separate between sounds that are useful for communication (thus becoming part of a motor vocabulary of speech gestures) and sounds that can be considered as noise and should be forgotten.

The latter goal can be obtained by having the caregiver repeating certain utterances, words or sounds that the child tries to imitate. To imitate a sound the robot uses its sensor-motor maps to calculate the corresponding vocal tract positions and then reproduces the same utterance with its own voice. Depending on how well the robot is able to map the voice of the caregiver to its own vocal tract positions the reproduced sound may or may not be perceived as an imitation by the caregiver. During this type of interaction the caregiver may need to actively change his or her voice in order to facilitate the task. This behaviour can also be found in the interaction between a child and its parents and has been studied in [8]. When the child eventually succeeds in producing an "acceptable" imitation, the adult provides some sort of reinforcement signal. Through this reinforcement, the child identifies elementary speech units (e.g. phonemes or words) that play a role in the communication process and will form a (motor) speech vocabulary that can be used in future interactions. In addition, the child can learn how to better produce those specific terms in the vocabulary.

3 Embodiment

In order to interact with the caregiver as explained in the previous section, the robot must be able to see, hear, and also to produce sounds. To mimic the way human infants interact with their caregivers it is of course an advantage to have a robot that looks and acts as would be expect from an infant. However, the embodiment is important not only to evoke emotions and make it more attractive for interaction, but also from a learning perspective. Having a similar body structure facilitates

imitations since it allows for a more direct mapping between the demonstrator and the imitator while at the same time limiting the search space and hence the risk of incorrect mappings.

In this section we describe the architecture and the system components that are implemented in the humanoid robot in order to allow the robot to engage in the interaction tasks necessary to acquire early language structures. The described architecture is an extension of the architecture described in [23] and [24], which uses mirror neurons to create a unified model for both speech production and recognition. A similar architecture can also be found in [28], and some earlier work can be found in the DIVA model [19]. In this work we have also included a memory model, similar to that in [46]. The resulting architecture includes a speech production module, a sensing unit, a sensorimotor coordination module and a memory unit, as shown in Figure 1.

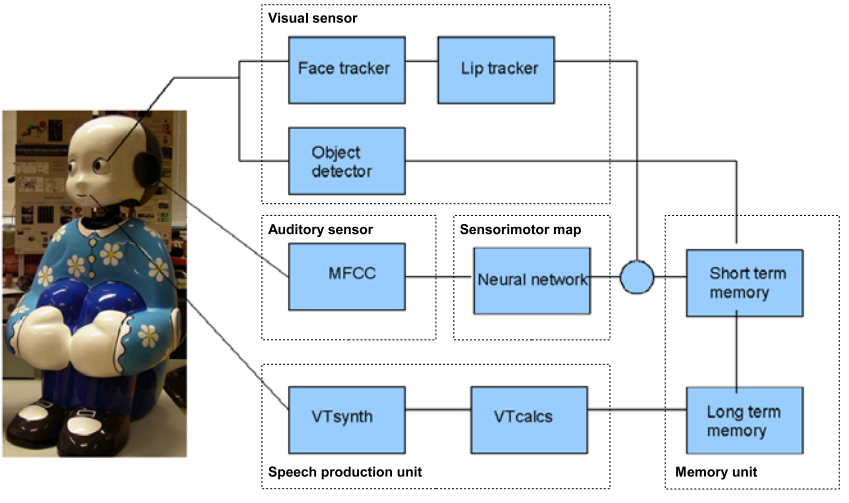


Fig. 1 System architecture for language acquisition.

3.1 Speech Production Unit

The speech production unit consists of an articulatory model of the human vocal tract and a position generator. There has been several attempts to build mechanical models of the vocal tract [21] [18]. While these can produce some human like sounds they are still rather limited and there are no commercially available mechanical solutions. The alternative is to simulate the vocal tract with a computer model. Such simulators are typically based on the tube model [40] where the vocal tract is considered to be a number of concatenated tubes with variable diameter. On top of the tube model an articulator is used that calculates the diameter of the tubes for different configurations of the vocalization units. In this work we have chosen to simulate the vocal tract by using VTcals developed by Maeda [41]. This model has been developed by studying x-rays from two women articulating French words, and

has six parameters that can be used to control the movements of the vocal tract. One parameter is used for controlling the position of the yaw, one for the extrusion of the lips, one for lip opening, and three parameters for controlling the position of the tongue. A synthesizer converts the vocal tract positions into sound. While the synthesizer works well for vowel-like sounds, it is unable to produce fricatives sounds and can hence only produce a limited set of consonants.

Apart from creating sound, the lip parameter is also used to control a number of Light Emitting Diods (LEDs) in the robot's face in order to simulate the movements of the lips. The current lip model is very simple and only shows the mouth as either open or closed.

3.2 Sensing Units

As explained in the previous section, not only ears but also other sensing modalities are useful when learning to speak. Here we have implemented two sensing modalities, an auditory sensor unit and a visual sensor unit that extract features from the acoustic and visual spaces respectively.

The auditory sensor consist of two microphones and artificial pinnas [26]. To get a more compact representation of the sound signal it is transformed into a tonotopic sound representation (sound features). There exists various representations that can be used for this. For production and recognition of vowels, formants are commonly used [57]. However, formants only contain information useful for vowels so its application is rather narrow. In other related work, LPC has been used [30] [44]. LPC are more generally applicable than formants, but still require rather stationary signals to perform well. Here we use Mel frequency cepstral coefficients (MFCC) [7] as our speech features since these do not require a stationary signal. To calculate the MFCC each utterance is first windowed using 25 ms windows with 50% overlap between the windows, and MFCC are then calculated for each window.

As visual sensors the robot uses two cameras with pan and tilt controls. However, in the case of language acquisition only one camera is used. The visual sensor is used both for finding objects during multimodal interaction, and for tracking faces and lipmovements to provide information on the lip-opening when trying to imitate the caregiver.

Starting with the object detector, the robot takes a snapshot of the camera's view and segments the image in order and look for the object closest to the center of the image. The segmentation is done by background subtraction followed by morphological dilation. Using the silhouette of the object we create a representation of its shape by taking the distance between the center of mass and the perimeter of the silhouette. This is done for each degree of rotation creating a vector with 360 columns. The tranformation of an image to the object representation is illustrated in Figure 2.

To estimate the opening of the mouth, the visual sensor takes a video sequence of the speaker's face as input and calculates the openness of the mouth in each frame. The openness is defined as the distance between the upper and lower lip, normalized with the height of the head. We use a face detection algorithm, based on [56] and

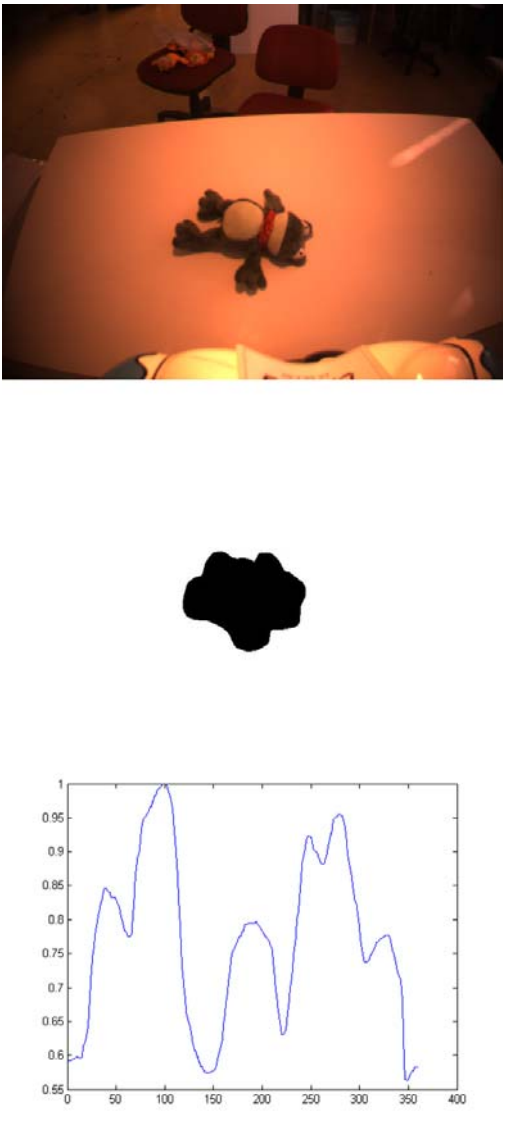


Fig. 2 Original image (top), silhouette image after background substraction and morphologic operations (center), and the silhouette perimeter in polar coordinates (bottom).

[39], to calculate the size of the head and the initial estimate of the position of the lips. Within the estimated position we use colour segmentation methods to select candidate pixels for the lips based on their redness. While there are methods to find the exact contour of the lips, like snakes or active contour methods [29], we are only

interested in the openness of the mouth and have chosen to represent the lips with an ellipse. To fit the ellipse to the lip pixels we use a least square method described in [16]. Finally we calculate a new estimate for the lip position in the next frame of the video sequence by using the method in [38].

3.3 Sensorimotor Maps

The sensorimotor maps are responsible for retrieving the vocal tract position from the given auditory and visual features. We use two separate neural networks to model the sound-motor and visuomotor maps. The sound-motor map is the more complicated of the two, mapping the 12 cepstral coefficients back to the 6 articulatory parameters of the vocal tract model. The added difficulty of the problem lies in the fact that several positions of the vocal tract may result in the same sound, hence giving several possible solutions for a given set of input features. These ambiguities have to be solved through the interaction with a caregiver. For the sound-motor map we use an artificial neural network with 20 hidden neurons.

The vision-motor map is a very simple unit, performing a linear mapping from the mouth opening to the lip height parameter of the synthesizer.

Since the output from both the sound-motor map and the vision-motor map consist of vocal tract positions, the integration of those sensor outputs becomes very simple. Here we simply use a weighted average of the lip height calculated from the two maps. The weight is currently set by hand, but should preferably be set automatically according to the quality and intensity of the visual and auditory stimuli.

3.4 Short Term Memory

The short-term memory receives a continuous stream of visual and auditory data that are stored during a fixed time (here we have used 10-20 s).

The auditory sound stream is sequenced into utterances. This is done automatically when the sound level is under a certain threshold value for at least 200 ms. Each utterance within the short term memory at a given time is compared pair-wise with all other utterences in the memory in order to find recurrent patterns. For each utterance-pair we first make sure that the utterances have the same length by padding the shortest utterance. The utterances are then aligned in time and we calculate the sum of differences between their mel coefficients creating a vector with the acoustic distance between the two utterances at each window. The second utterance is then shifted forward and backward in time and for each step a new distance vector is calculated. These vectors are averaged over 15 windows, i.e. 200 ms, and combined into a distance matrix as illustrated in Figure 3. By averaging over 200 ms we exclude local matches that are too short and can find word candidates by simply looking for minimas in the distance matrix. Starting from a minima we find the start and the end points for the word candidate by moving left and right in the matrix while making sure that the distance metric at each point is always below a certain critical threshold.

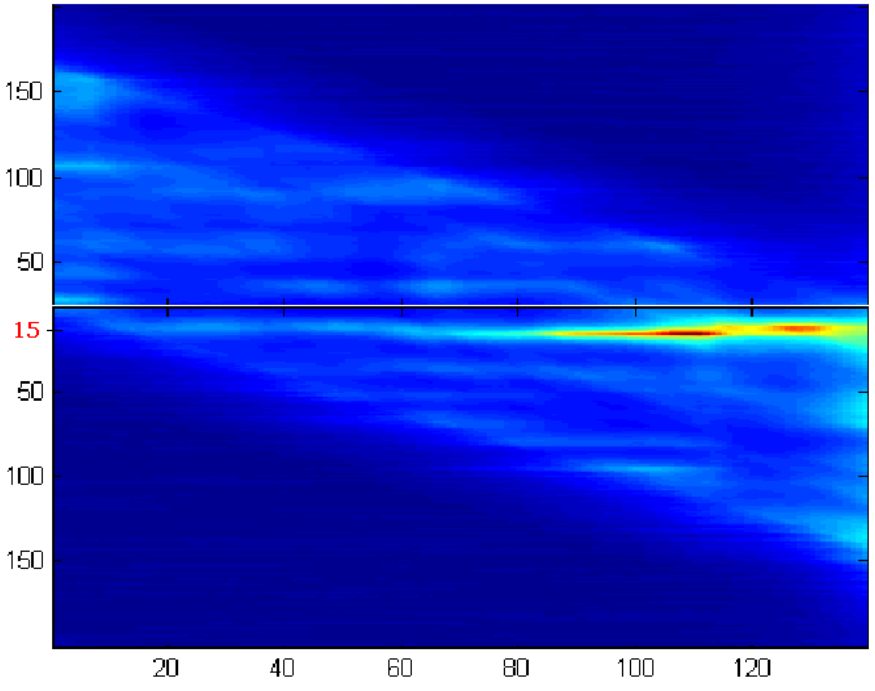


Fig. 3 Finding word candidates in sentences “titta här är den söta dappan" and “se på lilla dappan". Blue color indicates a weak match and red a strong match. The strongest match is found when shifting the second sentence 15 windows to the right and corresponds to the word “dappan".

In order to take advantage of the structure of infant directed speech and to mimic infants’ apparent bias towards target words in utterance-final position and focal stress, we also check for these features. For a word candidate to be considered to have utterance-final position we simply check that the end of the candidate is less than 15 windows away from the end of the utterance. To find the focal stress of an utterance we look for the F0-peak. While there are many ways for adults to stress words (e.g. pitch, intensity, length) it has been found that F0-peaks are mainly used in infant directed speech [13]. If the F0-peak of the utterance as a whole is within the boundaries of the word candidate, the word candidate is considered to be stressed. If a word candidates is not stressed and in utterance-final position we may reject it with a specified probability.

The same pattern matching technique is also be used to compare visual objects. When comparing two object representations with each other we first normalize the vectors and then perform a pattern maching, much in the same way as for the auditory representations, by shifting the vectors one step at a time. By doing this we get a measurement of the visual similarity between objects that is invariant to both scale and rotation.

When both a word candidate and a visual object are found, their representations are paired and sent to the long term memory.

3.5 Long Term Memory

The long term memory is used to store both word candidates, visual objects, and vocal tract positions that are found interesting during the interaction with the caregiver. To organize the information we use an hierarchical clustering algorithm [20]. Word candidates, visual objects, and vocal tract positions are organized independently into three different tree clusters. The algorithm starts by creating one cluster for each item. It then iteratively joins the two clusters that have the smallest average distance between their items until only one cluster remains.

While the algorithm is the same for all three trees, the distance measure varies slightly between them. The distance between the visual objects is measured directly through the pattern matching explained above. For the acoustic similarity we use Dynamic Time Warp (DTW) [48] to measure the distance between different word candidates. The reason to use DTW instead of directly applying the pattern matching described earlier is to be less sensitive to how fast the word candidate is pronounced. For the vocal tract position we simply use the Euclidean distance to measure the distance between each target position.

The resulting hierarchical trees can now be analyzed in a second step to determine the correct number of clusters. For each level of the clustering process, we have different relationships between data groupings. The question is then to find the "natural" grouping for this dataset. To estimate the adequate number of clusters in the dataset we have used the Gap statistic [53]. This function compares the within-cluster dispersion of our data with that obtained by clustering a reference uniform distribution. This is to compare the gain of raising the cluster number in a structured data with that arising from adding another cluster to a non-informative and not structured set of points. To find the optimal number of clusters we look for the first maximum in the Gap. Each position within the same cluster is considered to be part of the same phoneme or pseudo-phoneme in the motor speech vocabulary.

When we have interconnected multimodal representations, which is the case for the word candidates and visual objects that assumingly refers to the same object we can make use of these connections, not only to create associations, but also to find where we should cut the trees in order to get a good representations of the words and the objects. In order to find which branch in the word candidate tree that should be associated with which branch in the object tree we use the mutual information criterion [5]. In the general form this can be written as

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \left(\frac{p(x,y)}{p_1(x) p_2(y)} \right) \quad (1)$$

Where $p(x,y)$ is the joint probability distribution function of X and Y, and $p_1(x)$ and $p_2(y)$ are the marginal probability distribution functions of X and Y respectively.

We want to calculate $I(X;Y)$ for all combinations of clusters and objects in order to find the best word representations. For a specific word cluster A and visual cluster V we define the binary variables X and Y as

$$X = \{1\ if\ observation \in A;\ 0\ otherwise\}$$
$$Y = \{1\ if\ observation \in V;\ 0\ otherwise\}$$

The probability functions are estimated using the relative frequencies of all observations in the long-term memory, i.e. $p_1(x)$ is estimated by taking the number of observations within the cluster A and dividing with the total number of observations in the long-term memory. In the same way $p_2(y)$ is estimated by taking the number of observations in the cluster V and again dividing with the total number of observations. The joint probability is found by counting how many of the observations in cluster A that is paired with an observation in cluster V and dividing by the total number of observations.

4 Humanoid Robot Experiments

In this section we examplify how the interaction strategies and the described architecture works in practice by showing the results from two different experiments.

In the first experiment we show how the robot can extract useful word-candidates and associate those to visual object. This is done both by interacting directly with the robot and by using recordings of real Infant Directed Speech taken from adult-infant interactions.

In the second experiment we teach the robot a number of vowels and see how well the robot can recognize the same vowels when pronounced by different human speakers. Here we especially look at the effect that the different developmental stages of babbling and interaction have for the recognition rate and the role played by the visual input with respect to the recognition rate.

4.1 Experiment 1: Word Learning

In this experiment the robot makes use of multimodal information in order to learn word-object associations when interacting with the caregiver. The experimental setup is shown in Figure 4.

The caregiver shows a number of toys for the robot and, at the same time, talks about these objects in an infant directed speech style. The objects that were used during the experiment were one ball and two dolls named “Pudde" and “Siffy". The experiment was performed by demonstrating one object at a time by placing it in front of the robot for approximately 20 s, while talking to the robot about the object by saying things like “Look at the nice ball!" and “Do you want to play with the ball?". Each utterance contained a reference to a target word and we made sure that the target word has always stressed and in utterance-final position. For the dolls we referred to them both by using their individual names and the swedish word for doll, “docka". The ball were always referred to using the swedish word “bollen".

Fig. 4 Experimental setup for robot test.

During the length of one demonstration, sound and images are continuously stored in the short-term memory. The sound is then segmented by simply looking for periods of silence between the utterances and each utterance is then compared to the others as explained in the previous section. Each word candidate, i.e. matching sound pattern, in the short-term memory is paired with the visual representation of the object and sent to the long-term memory. After having demonstrated all three objects we repeat the procedure once more, but this time with the objects in slightly different orientations in front of the robot. This is done in order to verify that the clustering of the visual objects is able to find similarities in the shape despite differences in the orientation of the objects.

When word candidates have been extracted from all six demonstrations, the hierarchical clustering algorithm is used to group word candidates in the long-term memory that are acoustically close. The result from the hierarchical clustering of the word candidates and the visual objects can be seen in Figure 5. The numbers at each leaf shows the unique identifier that allows us to see which of the word candidates that was paired with which of the visual objects.

Looking only at the hierarchical tree for the word candidates it is not obvious where the tree should be cut in order to find good word representations. By listening to the word candidates we notice that the cluster containing candidates (25 26 19 20 2 6 18 14 16 1) represent the word “dockan", the cluster (3 7 4 9 5 8 10 12 15 11 13 17) represent the word “Pudde", the cluster (21 22 23 27 28 29 24 31

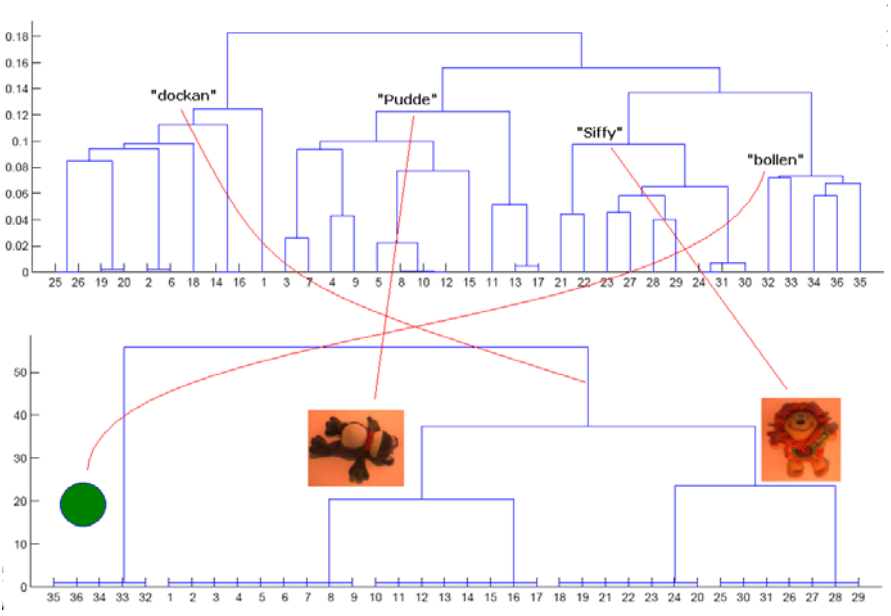


Fig. 5 Above: Clusters of the extracted word candidates during the robot experiment. Word candidates 1-17 are paired with object Pudde, nr 18-29 with object Siffy, and 32-36 with object bollen. Below: Clusters of the extracted visual objects during the robot experiment. Objects 1-17 corresponds to object Pudde, nr 18-29 to object Siffy, and 32-36 to object bollen.

30) represent the word “Siffy", and the cluster (32 33 34 36 35) represent the word “bollen". The hierarchical tree for the visual objects may look more simple and it is tempting to select the five clusters in the bottom as our objects. However, in this case it is actually the clusters one level up that represents our visual objects.

To find out which branch in the respective tree that should be associated with which branch in the other we calculate the mutual information criterion. Calculating the mutual information criterion for all pair of branches shows that we get the highest score for associating the word candidates (32-36) with the same visual objects (32-36). This is what we could expect since all visual observations of “bollen" were also paired with a correct word candidate. In the case of the objects “Pudde" and “Siffy" part of the observations are not paired with the object name, but instead with the word “docka". Still we get the second and third heighest scores by associating word candidates for the word “Pudde" with object Pudde and the word “Siffy" with object Siffy respectively. We can also find that the branch above the visual representations of Pudde and Siffy receives the heighest score for being associated branch containing word candidates for “dockan".

The experiment was repeated without putting any bias on word candidates that were stressed and in utterance-final position. This resulted in four false word candidates for the object Pudde and one for object Siffy. However, this did not affect the word-object associations as these candidates were found in separate branches in

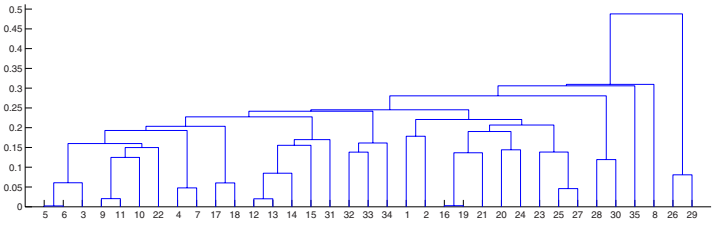


Fig. 6 Cluster formations from word candidates taken from infant directed speech. Word candidates between 1 and 30 are paired with object Kuckan and word candidates between 31 and 35 are paired with Siffy. Using the mutual information criterion, cluster (32 33 34) gets associated with Siffy and cluster (5 6 3 9 11 10 22 4 7 17 18) gets associated with Kucka.

the word candidate tree and only received low scores by the mutual information criterion.

A second experiment was performed using recordings of interactions between parents and their infants. The recordings were made under controlled forms at the Department of Linguistics, Stockholm University. A lot of care were taken to create a natural interactions. The room was equipped with several toys, among those two dolls called “Kuckan" and “Siffy". The parents were not given any information of the aim of the recordings but were simply introduced to the toys and then left alone with their infants. In this study we have only used a single recording of a mother interacting with her 8 month old infant. The total duration of the recording is around 10 minutes. The audio recording has been segmented by hand to exclude sound coming from the infant. In total the material consists of 132 utterances with time stamps and also object references in those case that an object were present. In 33 of these the doll “Kuckan" was present and in 13 of them the doll “Siffy". In total the word “Kuckan" is mentioned 15 times and “Siffy" is mentioned 6 times.

In this experiment we limit the short-term memory to 10 s. The utterances enter in the short-term memory one at a time and any utterance older than 10 s is erased from the memory. Word candidates that also have an assigned object label are transferred into the long-term memory.

After searching all utterances for word candidates we cluster all the candidates in the long-term memory. The result can be found in Figure 6. Here we don’t have any hierarchical tree for the visual objects. Instead we use the labels assigned by hand that can be used for calculating the mutual information criterion. Doing so gives us that the object Kuckan is best represented by word candidates (5 6 3 9 11 10 22 4 7 17 18) and Siffy by (32 33 34). Listening to the word candidates confirms that they represent the names of the dolls, but the segmentation is not as clear as in the humanoid experiment and there are a few outliers. Among the word candidates associated with Kuckan, nr 22 was unhearable and nr 17 and 18 were non-words but with a prosodic resembly of the word “Kuckan". For the word candidates associated with Siffy all contained parts of initial words.

When repeating the experiment without bias on focal stress and utterance-final position, the number of word candidates grew significantly resulting in lots of outliers being associated with both the objects. In the case of Kuckan it even caused the correct word candidates to be excluded from the branch that was associated with the object. However, it should be stated that the experiment was very small in order to draw any general conclusions.

4.2 Learning Target Positions

The objective of the second experiment is to show how the robot can learn articulatory target positions for a number of Portuguese vowels.

To learn vowels the robot first has to create an initial sound-motor map. Using the initial map it can then try to imitate the caregiver in order to get some first estimated motor configurations that represent vowels in the speech motor vocabulary. Local babbling is used to explore the neighbourhood of the terms in the vocabulary, while the caregiver gives feedback on the result. Finally, the clustering algorithm is used to group all positions learned into a feasible number of elements in the vocabulary.

The initial sound-motor map is created through random babbling. We generated 10000 random positions vectors for this phase. Each vector contains information about the position of the 6 articulators used in Maeda's model. These configurations are used by the speech production unit to calculate the resulting sound, which is coded in MFCC by the auditory unit. The sound-motor-map then tries to map the MFCC back to the original articulator positions that originated the sound. The error resulting from the comparison with the correct motor configuration given by the random articulator generator is used with a back-propagation algorithm to update the map. Repeating this will create an initial map between sound and the articulator positions used to create this sound.

The second step can be seen as a parroting behaviour where the robot tries to imitate the caregiver using the previously learned map. Since the map at this stage is only trained with the robot's own voice, it will not generalize very well to different voices. This may force the caregiver to change his/her own voice in order to direct the robot. There can also be a need to over-articulate, i.e. exaggerate the positions of the articulators in order to overcome flat areas in the maps that are a result of the inversion problem. When two or more articulator positions give the same sound the initial maps tends to be an average of those. However, for vowels the articulator positions are usually naturally biased towards the correct position as the sound is more stable around the correct positions than around the alternative positions. For most of the vowels it was not necessary to adapt the voice too much. Typically between one and ten attempts were enough to obtain a satisfying result. When the caregiver is happy with the sound produced by the robot it gives positive feedback which causes the robot to store the current articulator positions in its speech motor vocabulary. Using this method the caregiver was able to teach the robot prototype positions for nine Portuguese vowels. Visual inspection of the learned articulator positions showed that the positions used by robot are similar to those used by a human speaker, Figure 7.

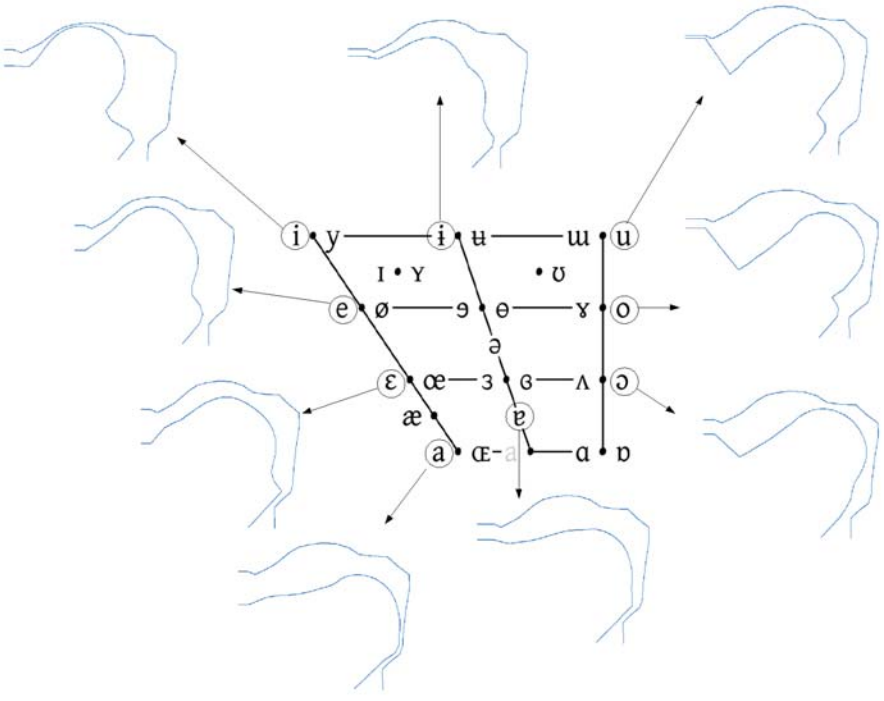


Fig. 7 Articulator positions used by the robot for the Portuguese vowels. In the center we show the positions of the vowels in the International Phonetic Alphabet (IPA). The vertical axis in the IPA corresponds to the vertical position of the tongue and the horizontal axis to the front-back position when the vowel is pronounced by a human speaker. For the simulated articulator positions used by the robot the upper line corresponds to the soft palate and the lower line to the tongue. There is a strong correlation between how the robot and a human articulate the vowels.

Since the vowel positions were learned under controlled forms where only one position was stored for each vowel sound we did not do any clustering of the target positions, but simply let each position represent a pseudo-phoneme. In [24] we did a larger scale experiment where 281 target positions were learned, each representing one of the nine vowels above. We then used the hierarchical clustering algorithm together with GAP statistics to group the target positions into a number of pseudo-phonemes. This showed that the robot automatically would group the positions into nine pseudo-phonemes corresponding to the Portuguese vowels.

Here we instead study how well the robot is able to recognize the learned vowels when those are pronounced by human speakers. We especially look at how the recognition rate is improved as a result of the different stages of babbling and interaction. To study this training and test data were collected with 14 speakers (seven males and seven females) reading words that included the nine Portuguese vowels

above. We used the vowels from seven speakers for training and the other seven for testing. Each speaker reads the words several times, and the vowels were hand labelled with a number 1 to 9. The amplitude of the sound was normalized and each vowel was then divided into 25 ms windows with 50% overlap. Each window was then treated as individual data which resulted in a training set of 2428 samples, and a test set of 1694 samples.

During training, we simulated the interaction where the humans imitate the robot by having the robot pronouncing one of its vowels at the time, and then present the robot with the same vowel from one of the humans in the training set. In this step we used both auditory and visual input. The auditory input consisted of a single window of 25 ms sound, and the visual input is an image showing the face of the human at the same instant of time. The robot then mapped these inputs to its vocal tract positions, compared the result with the position used by the robot to create the same sound, and used the error to update both the auditory-motor map and the vision-motor map.

For testing, we let the robot listen to the vowels pronounced by the speakers in the test set, i.e. speakers previously unknown to the robot. The input was mapped to the robot's vocal tract positions and the mapped positions were compared to the vowel positions stored in the speech motor vocabulary. Based on the minimum Euclidean distance, each position was classified as one of the stored vowel positions.

We performed this test several times using the maps obtained at each of the different stages of babbling and interaction. First we tested how well the robot was able to map the human vowels using maps that had only been trained using the robot's own voice, i.e. after the initial random babbling. As expected at this stage, the estimated positions were relatively far from the correct ones and it was not possible to recognize more than 18% of the human vowels. This is mainly due to the difference between the voice of the robot and the voices of the the human adults in the test set, and it is because of this that the human caregiver may need to adapt his or her voice during the early interaction with the robot.

When the robot has already had some interaction with humans, through the people in the training set, we noticed a significant increase in the performance. The distance between the vocal tract positions estimated from the human utterances in the test set, and the positions used by the robot to create the same utterance, decreased, and the recognition rate improved. Using only sound as input, the recognition rate became close to 58%, and using both sound and visual data the recognition rate reached 63%. A summary of the results is shown in Table 1.

Table 1 Recognition rates at the different stages of development.

Training data	Sum of square distance	recognition rate
Only babbling	9.75	18%
Using interaction	0.52	58%
Using interaction with vision	0.47	63%

5 Conclusions

In this work we have taken a step away from the traditional data-driven approach to speech processing. Instead we have adopted a developmental and ecological approach where embodiment and interactions are seen as enabling factors for the language acquisition.

Our review of some of the typical characteristics found in IDS clearly indicates that the signal directed to infants by their caregivers include several hints that can facilitate the language acquisition. Two of the most important characteristics is the use of repetitions and multimodal information. When demonstrating an object, the caregiver is likely to repeatably mention the name of that object. This provide useful information for finding word-like structures and to associate those with objects in the visual field. Moreover, when talking about an object, the name of the object is typically highlighted using utterance final position and focal stress. It has been shown that children are very sensitive to these kind of cues.

Infants don't only listen, they also produce sounds themselves. This enables for richer types of interactions such as imitation games. We have found that a significant part of adult-infant interactions can be seen as imitation games and that those may serve at least two different purposes. First they allow the infant to find target positions in it vocal tract that are useful in the communication with the adult, second they allow the infant to map adult speech to its own vocal tract positions. The latter is especially important since it allow for making speech recognition in motor space rather than in the acoustic space.

To implement and test this developmental and ecological approach to language acquisition in a machine, embodiment becomes a key factor. In this work we have described the models used for simulating the human ears, eyes, vocal tract, and memory functions. The focus has been to realise the functions needed for the described approach and to implement them in a humanoid platform, rather than to create complete and biological plausible models of the human organs.

The robot has been used in two experiments where it learns word-object relations and articulatory positions for a number of vowels.

In the first experiment we show that the robot successfully is able to extract suitable words for describing the presented objects. In contrast to related earlier work on word-object associations, our model is able to create those associations without any preprogrammed phoneme model. Instead it uses a preference for words in utterance final position and focal stress, which can be motivated from memory issues and a general attention system that triggers on salient events. Furthermore, our hierarchical model make it possible to associate names both for individual objects and groups of objects.

In the second experiment we show how imitation games allow the robot to acquire a set of target positions for vowel production, and that the robot can recognise the same vowels when pronounced by other persons by first mapping the sound to motor space.

While the current vocal tract model is only able to produce vowels and a very limited number of consonants, the robot has not been able to learn a complete phoneme model, which would be necessary in order to connect the two experiments and

allow the robot both to produce complete words and to recognise these words by first mapping to motor space. Still, the results from the individual experiments are encouraging and provide some initial steps towards a developmental and ecological approach to language acquisition in robots.

Acknowledgements. This work was partially supported by EU Project CONTACT and by the Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the POS Conhecimento Program that includes FEDER funds.

References

1. Albin, D.D., Echols, C.H.: Stressed and word-final syllables in infant-directed speech. Infant Behavior and Development 19, 401–418 (1996)
2. Andruski, J.E., Kuhl, O.K., Hayashi, A.: Point vowels in Japanese mothers' speech to infants and adults. The Journal of the Acoustical Society of America 105, 1095–1096 (1999)
3. Batliner, A., Biersack, S., Steidl, S.: The Prosody of Pet Robot Directed Speech: Evidence from Children. In: Proc. of Speech Prosody 2006, Dresden, pp. 1–4 (2006)
4. Burnham, D.: What's new pussycat? On talking to babies and animnals. Science 296, 1435 (2002)
5. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley, Chichester (2006)
6. Crystal, D.: Non-segmental phonology in language acquisition: A review of the issues. Lingua 32, 1–45 (1973)
7. Davis, S.B., Mermelstein, P.: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. IEEE Transactions on Acoustics, speech, and signal processing ASSP-28(4) (August 1980)
8. de Boer, B.: Infant directed speech and evolution of language. In: Evolutionary Prerequisites for Language, pp. 100–121. Oxford University Press, Oxford (2005)
9. Fadiga, L., Craighero, L., Buccino, G., Rizzolatti, G.: Speech listening specifically modulates the excitability of tongue muscles: a TMS study. European Journal of Neuroscience 15, 399–402 (2002)
10. Ferguson, C.A.: Baby talk in six languages. American Anthropologist 66, 103–114 (1964)
11. Fernald, A.l.: The perceptual and affective salience of mothers' speech to infants. In: The origins and growth of communication, Norwood, N.J, Ablex (1984)
12. Fernald, A.: Four-month-old infants prefer to listen to Motherese. Infant Behavior and Development 8, 181–195 (1985)
13. Fernald, A., Mazzie, C.: Prosody and focus in speech to infants and adults. Developmental Psychology 27, 209–221 (1991)
14. Gallese, V., Fadiga, L., Fogassi, L., Rizzolatti, G.: Action Recognition in the Premotor Cortex. Brain 199, 593–609 (1996)
15. Gustavsson, L., Sundberg, U., Klintfors, E., Marklund, E., Lagerkvist, L., Lacerda, F.: Integration of audio-visual information in 8-months-old infants. In: Proceedings of the Fourth Internation Workshop on Epigenetic Robotics Lund University Cognitive Studies, vol. 117, pp. 143–144 (2004)
16. Fitzgibbon, A., Pilu, M., Risher, R.B.: Direct least square fitting of ellipses. Tern Analysis and Machine Intelligence, 21 (1999)
17. Fitzpatrick, P., Varchavskaia, P., Breazeal, C.: Characterizing and processing robotdirected speech. In: Proceedings of the International IEEE/RSJ Conference on Humanoid Robotics (2001)

18. Fukui, K., Nishikawa, K., Kuwae, T., Takanobu, H., Mochida, T., Honda, M., Takanishi, A.: Development of a New Humanlike Talking Robot for Human Vocal Mimicry. In: Proc. International Conference on Robotics and Automation, Barcelona, Spain, April 2005, pp. 1437–1442 (2005)
19. Guenther, F.H., Ghosh, S.S., Tourville, J.A.: Neural modeling and imaging of the cortical interactions underlying syllable production. Brain and Language 96(3), 280–301
20. Hastie, T.: The elements of statistical learning data mining inference and prediction. Springer, Heidelberg (2001)
21. Higashimoto, T., Sawanda, H.: Speech Production by a Mechanical Model: Construction of a Vocal Tract and Its Control by Neural Network. In: Proc. International Conference on Robotics and Automation, Washington DC, May 2002, pp. 3858–3863 (2002)
22. Hirsh-Pasek, K.: Doggerel: motherese in a new context. Journal of Child Language 9, 229–237 (1982)
23. Hörnstein, J., Santos-Victor, J.: A Unified Approach to Speech Production and Recognition Based on Articulatory Motor Representations. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, USA (October 2007)
24. Hörnstein, J., Soares, C., Santos-Victor, J., Bernardino, A.: Early Speech Development of a Humanoid Robot using Babbling and Lip Tracking. In: Symposium on Language and Robots, Aveiro, Portugal, (December 2007)
25. Hörnstein, J., Gustavsson, L., Santos-Victor, J., Lacerda, F.: Modeling Speech imitation. In: IROS-2008 Workshop - From motor to interaction learning in robots, Nice, France (September 2008)
26. Hörnstein, J., Lopes, M., Santos-Victor, J., Lacerda, F.: Sound localization for humanoid robots - building audio-motor maps based on the HRTF. In: IEEE/RSJ International Conference on intelligent Robots and Systems, Beijing, China, October 9-15 (2006)
27. Jusczyk, P., Kemler Nelson, D.G., Hirsh-Pasek, K., Kennedy, L., Woodward, A., Piwoz, J.: Perception of acoustic correlates of major phrasal units by young infants. Cognitive Psychology 24, 252–293 (1992)
28. Kanda, H., Ogata, T.: Vocal imitation using physical vocal tract model. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, USA, October 2007, pp. 1846–(1851)
29. Kass, M., Witkin, A., Terzopoulus, D.: Snakes: Active contour models. International Journal of Computer Vision (1987)
30. Krstulovic, S.: LPC modeling with speech production constraints. In: Proc. 5th speech production seminar (2000)
31. Kuhl, P., Andruski, J.E., Christovich, I.A., Christovich, L.A., Kozhevnikova, E.V., Ryskina, V.L., et al.: Cross-language analysis of Phonetic units in language addressed to infants. Science 277, 684–686 (1997)
32. Kuhl, P., Miller, J.: Discrimination of auditory target dimensions in the presence or absence of variation in a second dimension by infants. Perception and Psychophysics 31, 279–292 (1982)
33. Lacerda, F., Marklund, E., Lagerkvist, L., Gustavsson, L., Klintfors, E., Sundberg, U.: On the linguistic implications of context-bound adult-infant interactions. In: Genova: Epirob 2004 (2004)
34. Lacerda, F., Klintfors, E., Gustavsson, L., Lagerkvist, L., Marklund, E., Sundberg, U.: Ecological Theory of Language Acquisition. In: Genova: Epirob 2004 (2004)
35. Lacerda, F.: Phonology: An emergent consequence of memory constraints and sonsory input. Reading and Writing: An Interdisciplinary Journal 16, 41–59 (2003)
36. Lenneberg, E.: Biological Foundations of Language. Wiley, New York (1967)

37. Liberman, A., Mattingly, I.: The motor theory of speech perception revisited. Cognition 21, 1–36 (1985)
38. Lien, J.J.-J., Kanade, T., Cohn, J., Li, C.-C.: Detection, tracking, and classification of action units in facial expression. Journal of Robotics and Autonomous Systems (1999)
39. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: IEEE ICIP, pp. 900–903 (2002)
40. Liljencrants, J., Fant, G.: Computer program for VT-resonance frequency calculations. In: Liljencrants, J., Fant, G. (eds.) STL-QPSR, pp. 15–20 (1975)
41. Maeda, S.: Compensatory articulation during speech: evidence from the analysis and synthesis of vocat-tract shapes using an articulatory model. In: Hardcastle, W.J., Marchal, A. (eds.) Speech production and speech modelling, pp. 131–149. Kluwer Academic Publishers, Boston
42. Moore, R.K.: PRESENCE: A Human-Inspired Architecture for Speech-Based Human-Machine Interaction. IEEE Transactions on Computers 56(9) (September 2007)
43. Mulford, R.: First words of the blind child. In: Smith, M.D., Locke, J.L. (eds.) The emergent lexicon: The child's development of a linguisticvocabulary. Academic Press, New York (1988)
44. Nakamura, M., Sawada, H.: Talking Robot and the Analysis of Autonomous Voice Acquisition. In: Proc. International Conference on Intelligent Robots and Systems, Beijing, China, October 2006, pp. 4684–4689 (2006)
45. Nowak, M.A., Plotkin, J.B., Jansen, V.A.A.: The evolution of syntactic communication. Nature 404, 495–498 (2000)
46. Roy, D., Pentland, A.: Learning words from sights and sounds: A computational model. Cognitive Science 26, 113–146 (2002)
47. Saffran, J.R., Johnson, E.K., Aslin, R.N., Newport, E.: Statistical learning of tone sequences by human infants and adults. Cognition 70, 27–52 (1999)
48. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech and Signal Processing 26(1), 43–49 (1978)
49. Stoel-Gammon, C.: Prelinguistic vocalizations of hearing-impaired and normally hearing subjects: a comparison of consonantal inventories. J. Speech Hear Disord. 53(3), 302–315 (1988)
50. Sundberg, U., Lacerda, F.: Voice onset time in speech to infants and adults. Phonetica 56, 186–199 (1999)
51. Sundberg, U.: Mother tongue – Phonetic aspects of infant-directed speech, Department of Linguistics, Stockholm University (1998)
52. ten Bosch, L., Van hamme, H., Boves, L.: A computational model of language acquisition: focus on word discovery". In: Interspeech 2008, Brisbane (2008)
53. Tibshirani, R., Walther, G., Hastie, T.: Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63(2) (2001)
54. Vihman, M.M.: Phonological development. Blackwell, Oxford (1996)
55. Vihman, M., McCune, L.: When is a word a word? Journal of Child Language 21, 517–542 (1994)
56. Viola, P., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: IEEE CVPR (2001)
57. Yoshikawa, Y., Koga, J., Asada, M., Hosoda, K.: Primary Vowel Imitation between Agents with Different Articulation Parameters by Parrot-like Teaching. In: Proc. Int. Conference on Intelligent Robots and Systems, Las Vegas, Nevada, October 2003, pp. 149–154 (2003)

Human-Robot Cooperation Based on Interaction Learning

S. Lallee, E. Yoshida, A. Mallet, F. Nori, L. Natale, G. Metta, F. Warneken, and P.F. Dominey

1 Introduction

Robots are now physically capable of locomotion, object manipulation, and an essentially unlimited set of sensory motor behaviors. This sets the scene for the corresponding technical challenge: how can non-specialist human users interact with these robots for human robot cooperation? Crangle and Suppes stated in [1] : "the user should not have to become a programmer, or rely on a programmer, to alter the robot's behavior, and the user should not have to learn specialized technical vocabulary to request action from a robot." To achieve this goal, one option is to consider the robot as a human apprentice and to have it learn through its interaction with a human. This chapter reviews our approach to this problem.

An apprentice is an able-bodied individual that should interactively assist an expert, and through this interaction, they should acquire knowledge and skill in the given task domain. In other words, the expert teaches the apprentice by sharing a

S. Lallee and P.F. Dominey
Stem Cell and Brain Research Institute, INSERM U846. 18 avenue Doyen Lépine, 69675 Bron Cedex, France

E. Yoshida and A. Mallet
Laboratoire d'Analyse et Architecture des Systèmes, CNRS. 7 avenue du Colonel Roche, 31077 Toulouse, France

F. Nori, L. Natale, and G. Metta
Italian Institute of Technology, Central Research Labs, Genoa Headquarter. Via Morego, 30. Genoa, Italy

F. Warneken
Department of Developmental and Comparative Psychology,
Max Planck Institute for Evolutionary Anthropology. Deutscher Platz 6, D-04103 Leipzig, Germany

O. Sigaud, J. Peters (Eds.): From Mot. Learn. to Inter. Learn. in Rob., SCI 264, pp. 491–536.
springerlink.com

task with him, or by direct demonstration or explanation of what to do. In this context, the robot owns a repertoire of useful actions which execution can be requested by the user. While the human uses theses actions in order to achieve task, the robot should extract information about the human goal and how actions can imbricate to reach this goal.

The implementation of this apprentice architecture is one of our long term goal for which we developed the Spoken Language Programming system (SLP). First ([2],[3]) we established a primary mapping between sentences and actions, allowing verbal command of a robot and online creation of new commands. By including some visually guided actions this provides already a large repertoire of behaviors. However, even if the SLP allows fluent commanding of the robot, everything that the robot does has first to be predefined by the user. This is the reason that leads us to add anticipation ability to the system in [4]. This ability allows the robot to not only wait for commands, but to predict or even execute these commands without any order received from the human. In this chapter, we will review our previous work relative to the SLP development and extend it so that the user and the robot will have shared representations not only for actions, but also for objects and high level tasks that can imply sub-tasks. These shared representations and "hierarchical actions" are part of a more global cooperation skill used by humans which is described in [5]. In the second part of this chapter we will describe some results about the human cooperation ability and how we can use them to improve the SLP so it will provide to robots a human like cooperation ability.

1.1 Linking Words to Actions – Spoken Language Programming

Robots will integrate gradually everyday life over the next century. It will require a way for non specialists to use them for a wide range of tasks that are not predefined and that will occur in an unknown environment. The most common ability used by humans to communicate is spoken language; so it can provide a very rich vector for communication between a user and a robot. Through grammatical constructions, complex meanings can be communicated. Construction grammar (CxG) provides a linguistic formalism for achieving the required link from language to meaning [6].Meaning is represented in a predicate-argument structure as in [6], based on generalized abstract structures as in [7]. The power of these constructions is that they are based on abstract "variables" that can take an open set of arguments.

1. "John put the ball on the table."
2. Transport(John, Ball, Table)
3. Event(Agent, Object, Recipient)

We can thus use the Predicate - Argument structures to extract robot commands from natural language, and to generate natural language descriptions of physical events extracted from video scenes ([7-12]). In this section we review our work [2, 3, 7] aimed to the devellopment of a Spoken Language Programming system (SLP). The objective of the SLP is to to use natural language in order to allow human users to both command, program or teach the robot with spoken language. In a related context, Nicolescu and Mataric [12] employed spoken language to

allow the user to clarify what the robot learned by demonstration. In order to explore how language can be used more directly, Lauria et al. [13] asked naïves subjects to provide verbal instructions to a robot in a visual navigation task. Their analysis of the resulting speech corpora yielded a set of verbal action chunks that could map onto robot control primitives. They demonstrated the effectiveness of such instructions translated into these primitive procedures for actual robot navigation [13]. This indicates the importance of implementing the mapping between language and behavioral primitives for natural language instruction or programming [14, 15]. Learning by imitation and/or demonstration likewise provide methods for humans to transmit desired behavior to robots [11]. The SLP extends such methods in a complimentary way. We established a representative scenario where human and robot have to interact using spoken language in order to accomplish a task. We will first present this scenario and the robotic platform requirements for it execution, and then we will use it to benchmark successive versions of the SLP.

1.2 A Scenario for Human-Robot Cooperation

In order to test the SLP a task that involve human – robot cooperation has to be defined. We first introduced the Cooperative Table Construction Task in [2] and

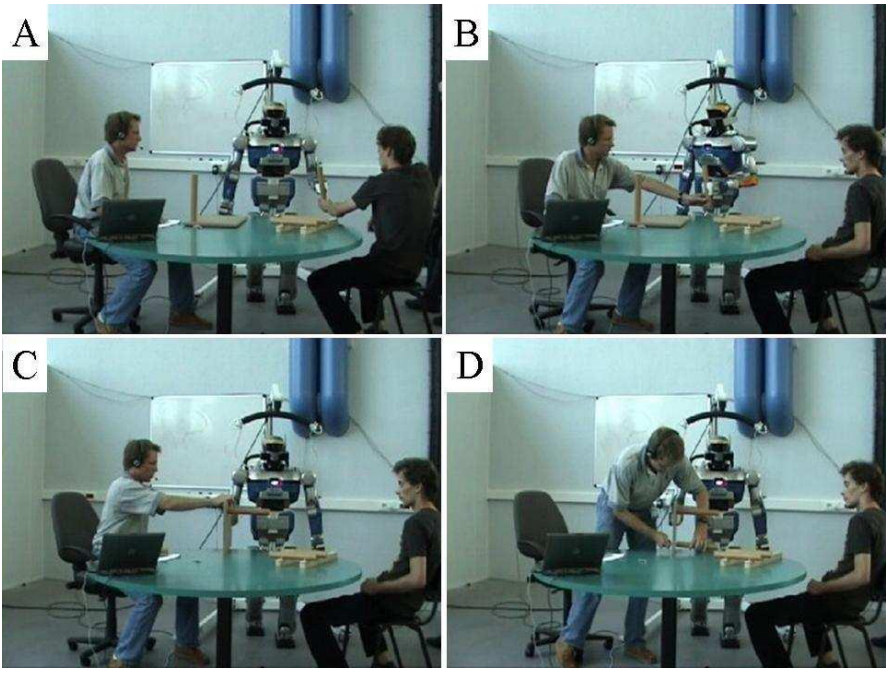


Fig. 1 The Cooperative Table Construction Task with HRP-2 : robot and human have to cooperate in order to build a table (A,D). The robot has to pass legs to the user (B) and hold the table while the user screws the legs (C).

refined it in [3, 4]. In the Cooperative Table Construction Task a robot and a human have to cooperate in order to build a small table (Fig. 1). The user has first to ask the robot for the table's legs, and then he will have to screw then to the table's surface which will be hold by the robot. All the interaction between the two participants are done using spoken language. One major interest of this task is that its execution includes regularities : the same sequence of actions will occur several times. For example, the user will have first to ask "grasp a leg", then "pass it to me" and finally "hold the surface". Since this sequence will be the same for the 4 legs, it can be easily extracted by the user, which will be able to speed up the task by programming the robot. Moreover, this task involves a lot of knowledge that can be hard coded or learned by the robot in real time. It involves mainly two domains: mapping words to actions (what does "pass" mean?) and mapping words to arguments (what is a "leg"?), the combination of these two mappings allows the execution of a concrete physical action based on a Predicate - Argument structure extracted from spoken language.

1.3 The Robotic Platforms

One of the main interests of the SLP is that it is not robot specific. The speech interface (synthesis & recognition) is provided by the CSLU Rapid Application Development (RAD) Toolkit (http://cslu.cse.ogi.edu/toolkit/) which runs on an independent computer remotely connected to the robot. RAD provides a state based dialog system capability, in which the passage from one state to another occurs as a function of recognition of spoken words or phrases; or evaluation of Boolean expressions. It also allows scripting in TCL which allows implementing complex behaviors. The vision system is also robot independent and need only that a YARP interface (http://eris.liralab.it/yarp/) has been defined to access the robots cameras. Then the object recognition is then provided using Spikenet, a commercial system based on research of Thorpe and al. [16] related to vision processing in the cotex. The only components of the SLP that are robot specifics are the 3D object localization (since all robots don't have the same camera system) and the basic set of atomic actions. Since all robots have different bodies and abilities we assume that they provide some basic actions that the SLP will be able to call (e.g: "grasp" will not be the same according to whether you are using a humanoid robot or a dog robot.). Anyway, these requirement can be easily implemented on any robot and we already used successfully the SLP on multiple robot platforms including :

- Kawada Industries HRP-2 humanoid robot [17] under the control of the OpenHRP controller [18]
- AIBO ERS7 with a WIFI interface
- Lynxmotion 6DOF robot arm
- Khepera mobile robots with a serial port controller [8]
- iCub humanoid robot [19] using a YARP [20] based architecture

However, the Cooperative Table Construction Task scenario is more suited to a humanoid robot, so most of the researches presented in this chapter are made using the HRP-2 or the iCub which are robots with many effective degrees of freedom (respectively 30 and 54) and possibilities for rich cooperative interaction.

2 The Development of Spoken Language Programming

The SLP is not static and has significantly evolved since its creation. The first implementation was a simple system that allowed the creation of macros, this system was then refined to turn these macros into procedures and to include the use of vision and arguments. The most recent evolution, presented here for the first time, allows the creation of hierarchical functions (functions that call functions) and the establishment of a common vocabulary between the robot and the user regarding visual object names. In this section we will present all these successive evolutions of the SLP and suggest what can be improved in future work.

2.1 Macro style Programming

The first implementation of SLP is described in [2]. In this study we used the HRP-2 and an early version of the Cooperative Table Construction Task. There was no vision included in the system and only a small set of available actions. However this provided already enough material to achieve a proof of concept for the SLP.

2.1.1 Atomic Actions and Learning Commands

The central idea in SLP is compositionality: based on a finite set of atomic action primitives the user should be able to compose arbitrary new cooperation behaviors. As we stated above, one of the few robot specific components of the system is the set of atomic actions. These are actions which are available to the user without any need of programming, in the case of the HRP2 theses actions were a set of static postures corresponding to required function in the task (table 1). The set of atomic actions for this study is presented in table 1. Each atomic action is called by the user through the SLP using a verbal command, thus allowing the user to control the robot using speech.

In addition to the set of atomic actions, the system requires a set of commands that allow the user to control the actual programming and program execution. These commands and their consequences are presented in table 2. When the user invokes the “Learn” command, the dialog system begins to encode the sequence

Table 1 HRP-2 Set of Atomic Actions.

Verbal command	Resulting actions
Prepare	Move both arms to neutral position, rotate chest to center, elevate left arm, avoiding contact with the work surface (5 DOF)
OpenLeft	Open left hand (1 DOF)
OpenRight	Open right hand (1 DOF)
Give it to me	Rotate hip to pass the object in left hand to User (1 DOF)
Hold	Center hip, raise right arm preparing to hold table top (5 DOF)
Right open	Open right hand (1 DOF)
Right close	Close right hand (1 DOF)

of the subsequent commands that are issued. The user proceeds to issue action commands to effect the desired task that will make up this sequence. When the user has finished the part of the task he wants to program, he issues the "OK" command. This results in the action sequence being written to a file. Now, when the "Macro" command is issued, this file is read into an array, and the commands are sequentially executed. During these executions, the behavioral scenarios above also identified the requirement for a conditional wait, in which the execution of a stored sequence waits for the user to finish what he is doing which the user signifies with the "continue" command. Thus, when the "wait" condition is issued, the system pauses until the "continue" command is issued.

Table 2 SLP Learning and control commands.

Commands	Correspondence
Learn	Begin encoding subsequent commands
OK	Store encoded command sequence in macro
Macro	Execute the stored macro
Wait	Interrupt command execution until a spoken « continue » command is issued
Continue	Terminate the « wait » pause and resume execution.

By using both the set of atomic actions and the learning commands, the user is able to make the task execution easier and faster as we show in the next section.

2.1.2 Experimental Results

In order to evaluate the implemented system, we used the Cooperative Table Construction Task in two different ways. First the user and the robot had to assemble the table, and then they had to cooperate again but to take the table apart. In our initial development, only the first experiment was planned, but when it was successfully accomplished we wanted to determine if the set of atomic actions was indeed general enough to extend to a new but related task of disassembling the table. Each of the two experiments will be presented, followed by an analysis of the effects of the ability to program on measures of efficiency of task completion.

Assembling the table

In the following paragraphs we provide the literal transcription of the human-robot dialog, annotated with explanation about the corresponding human and robot behavior and key system functions in <> brackets. Robot language will be indicated in *italics*, and human language in plain text, with main commands in **bold.**

The user begins its first interaction with the robot. He controls the task via spoken language. Because the task is new to him, he proceeds for the moment only by commanding the robot, with no programming.

<Robot initiates the interaction> *I am ready*.
Prepare.
You said prepare? Yes. <Because of possible errors in speech recognition, we choose to have an explicit validation on each command as illustrated here>.
Preparing to grasp.<Robot raises left hand >
I am ready.
Left open.
You said left open? Yes.
Opening left hand. <Robot opens left hand>
I am ready. <A third party user places the table leg in the robot's left hand>
Left close.
You said left close? Yes.
Closing left hand. <Robot closes left hand to grasp the table leg>
I am ready.
Give it to me.
You said give it to me? Yes.
Giving you the object. <Robot turns to its right, holding the table leg out to User1>
I am ready.
Left open.
You said left open? Yes.
Opening left hand. <As the Robot opens the hand, User reaches for and grasps the table leg>
I am ready.
Hold. <The first of several primitives required for the robot to hold the table while User attaches the leg.>
You said give it to me? No. <This is an example of a speech recognition error.>
I am ready.
Hold.
You said hold? Yes.
Getting ready to hold. <The robot orients to the work surface and raises its right hand.>
I am ready.
Right open.
You said right open? Yes.
Opening right hand. <With the robot's right hand open, User places the table top within the robot's grasp>
I am ready.
Right close.
You said right close? Yes.
Closing right hand. I am ready.
Wait.

You said wait? Yes.
Waiting for your signal <During this wait period, User attaches the leg to the table and then gives the "continue" command when ready.>
Continue.
I am ready.
Right open.
You said right open? Yes.
Opening right hand. <User takes the table from the robot and prepares for the next leg.>
I am ready.

At this point, the user has used spoken language tele-operation to enlist the robot's assistance in attaching the first leg to the table. Given the knowledge of the task he has gained, he is now ready to "program" the robot for the repetitive component of the task by using the "learn" command.

Learn.
You said learn? Yes. *I'll start learning now. I am ready.*
Prepare. <The interaction continues as for the previous leg until the end >
Continue.
I am ready.
Right open.
You said right open? **Yes**.
Opening right hand. <User takes the table>
I am ready.
OK.
You said OK? **Yes**. <"OK" indicates the macro end (table 2) >
OK we will store this plan. I am ready.

At this point, the sequence of commands including the conditional wait are now stored as an SLP macro for immediate re-use.

Macro
You said macro ? Yes.
Running the macro.
Preparing to grasp.
Waiting for your signal.
Continue
Closing left hand. <Robot takes leg>
Giving you the object.
Opening left hand. <User takes leg>
Getting ready to hold.
Waiting for your signal.
Continue
Closing right hand.
Waiting for your signal <User attaches the leg>
Continue
Opening right hand. In line macro finished. I am ready. < >

Third leg is now attached. User takes the table. Same procedure with macro for attaching fourth leg.

Performance Analysis of SLP Effects

During the assembly of the first two legs, the user issued 10 distinct commands for each of the two legs. Installing legs 1 and 2 took 3min 25sec and 3min 24sec. Once the program was learned, for legs 3 and 4, a single command initiated the program, and the user was only required to issue 3 "continue" commands in order to indicate to the robot that he was ready to proceed. Execution time was reduced to 2:11 and 2:33 respectively. The ability to execute a suite of primitive behaviors with a single command had a significant impact on the execution time for completion of the task. Fig 2 indicates the average execution times for the individual commands under the explicit command conditions without the programmed macro (CMD) at 25.14 sec/operation, and under macro program execution conditions (PRG) at 15.00 sec/operation. We performed a statistical analysis of the variance (ANOVA) in these completion times examining the effects of Repetition (i.e. first and second trial in either the CMD or PRG mode), and Programming condition (i.e. CMD vs PRG). Only the Programming condition had a significant effect on the completion times (ANOVA, Programming Effect: $F(1,6) = 109$, $p < 0.0001$). These results demonstrate that SLP not only eliminates the need to enumerate the successive commands, it also significantly reduces the task completion time.

Disassembling the table

Here we test whether the system that was developed for assembling the table could equally well serve to allow the user to program the robot to aid in disassembling the table. Disassembly resembles the reverse process of assembly. User will first ask the robot to hold the table while he removes the leg. He will then put the robot into the "give it to me" posture, with the left hand open, give it the leg, and then issue to "prepare" command so that the robot passes the leg to the third party user who then puts it away. For ease of reading, we dispense with the portion of the dialog in which the robot asks for confirmation of each command.

I am ready.
Hold.
Getting ready to hold. I am ready.
Right open.
Opening right hand. I am ready. <User2 puts table in robot's hand>
Right close.
Closing right hand.<Robot grasps the table>
I am ready.
Wait.
Waiting for your signal. <User1 unscrews the first leg while the robot holds the table>
Continue.
I am ready.

Right open.
Opening right hand. <Robot releases table, User1 takes the table and places it on the work surface> *I am ready.*
Prepare.
Preparing to grasp. <Right hand descends, left hand raises, preparing to take the leg> *I am ready.*
Give it to me
Giving you the object. <This is a bit counter-intuitive. With this command robot rotates its hip to face User1>.
I am ready.
Left open.
Opening left hand. I am ready. <User1 places the first table leg within the robot's left hand.>
Left close.
Closing left hand. <Robot grasps the leg> *I am ready*.
Prepare. Preparing to grasp <Robot orients to User2> *I am ready*.
Left open. <User2 takes the leg.> I am ready.

As in the previous experiment, after one run with the first leg, the user is now ready to program the robot. Again, he initiates the program storage by saying "Learn" and then executes step-by-step the procedure for taking a leg off and passing it to the third party user with the help of the robot, and finally storing this program by saying "OK". The important point is that by using exactly the same primitives but in a different sequence we were able to generate a new stored macro on the fly for a different, but related, task, thus demonstrating the generalization capability of the SLP system.

I am ready.
Macro.
Running the macro. Getting ready to hold. <User1 places the table in the robot's right hand> *Closing right hand. Waiting for your signal.* < User1 unscrews the leg and then tells the robot to continue>.
Continue.
Opening right hand <Robot releases table, user1 places it on table surface> *Preparing to grasp.* <Right hand descends, left hand raises, preparing to take the leg> *Giving you the object.* <Robot rotates hip to face User1>. *Closing left hand.* <Robot takes the leg from User1>*Preparing to grasp*. <Robot orients to User2> *Opening left hand* <Robot gives the leg to User2. The second execution of the macro for the final leg is identical, and the table is thus taken apart. >

Performance analysis

As in Experiment 1, the use of the programming capability for the third and fourth leg (executed in 2:51 and 2:51 respectively) yielded significant reductions in execution time as compared with the first two legs (executed in 3:57 and 4:11

respectively). To compare performance in the two experiments we performed a 3 way ANOVA with the factors Experiment (Exp1 vs. Exp2), Programming vs simple voice Commanding (PRG vs CMD), and Repetition (First vs. second repetition in each condition). Fig 2 indicates that both for Exp1 and Exp2 the completion times were elevated for the CMD vs PRG conditions, i.e. action execution was slower when programming was not used. The ANOVA reveled that only the Programming effect was significant ($F(1,6) = 277$, $p < 0.0001$).

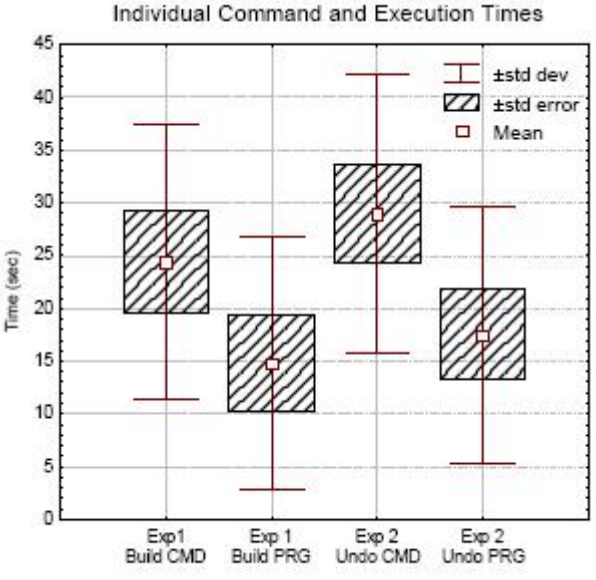


Fig. 2 Average command execution times for the Building (Exp1) and Undoing (Exp2) task using spoken language for on-line commanding (CMD) and for macro programming (PRG).

2.1.3 Conclusion

Despite the speed improvement induced by the SLP we have not yet fully exploited the potential richness of the predicate-argument structure of grammatical constructions. There are two important considerations to note here. First, a 3 month field study with an interacting robot [21] concluded that expectations on language-based interfaces have often been too high, and that "we need rich empirical experience of the use of robust and simple systems in order to formulate new and relevant questions," justifying our simplified (and successful) approach. Second, this simplified approach has aided us in generating requirements for higher level predicate argument representations for robot action and perception that will allow us to more deeply exploit the communicative richness of natural spoken language. Communicative interaction that will allow humans to truly cooperate with robots is an open and active area of research. Progress towards this objective is being made in part via well-documented methods for action learning

that include demonstration and imitation [15, 22]. Language has been used in this context for correcting and clarifying what is being learned by demonstration [12]. One of the fundamental requirements is to establish the grounded meaning at the base of the communication, that is the link between human language, and robot action and perception. This has recently been explored and developed in the domain of language based navigation [13, 23]. Roy and colleagues further establish these links via an amodal Grounded Situation Model that integrates perception, action and language in a common framework for language based human-robot cooperation [14]. We have made progress with a system that can learn grammatical constructions which make the mapping between predicate argument representation of action as perceived by a robot vision system, and natural language sentences that describe that action, generalizing to new action scenes [8, 9]. In this context of language-based human-robot cooperation, this research demonstrated - for the first time - the capability for a human user to tell a humanoid what to do in a cooperative task so that in real time, the robot performs the task, and acquires new skills that significantly facilitate the ongoing cooperative human-robot interaction.

2.2 Vision Based Procedures with Arguments

The first implementation of the SLP was limited: multiple macros were not allowed, and the actions offered were quite rigid. The first refinement of the SLP has been done in [3]: we integrated vision and motion planning into the SLP framework, providing a new level of flexibility in the behavior that can be created. Most important we possibility to the user to create "generic" functions with arguments (e.g. Give me X), and we allowed multiple function creations. We thus demonstrated again with the HRP-2 equipped with vision based grasping the ability to acquire multiple sensory motor behavioral procedures in real-time through SLP in the context of a cooperative task. The humanoid robot thus acquired new sensory motor skills that significantly facilitate the cooperative human-robot interaction.

2.2.1 Adding Visually Guided Action to the Robot

One of the most impressive improvements of the SLP in [3] is the fact that the robot is able to localize an object based on vision and to reach/grasp it wherever it is in the reachable workspace. In the original study, we used the OpenCV library and some basic color recognition algorithm in order to recognize the different colored legs. We then used stereovision triangulation to get the object coordinates in a 3D space. However, the only requirement of the vision module is to provide 3D coordinates of the object to the SLP, it can be achieved in different ways. For example, in our latest studies we use a commercial program (Spikenet) [16, 24] which allows the recognition of arbitrary objects based on their shape; but one can think about using other 3D localization systems using markers, etc.

The second requirement is to be able to use these 3D coordinates in order to guide certain actions. Basic actions that involve vision are for example turning the head to gaze at an object, or moving the arm and hand in order to grasp it. To do so we used an inverse kinematic solver which gives, for a given 3D position, the

values of the arm's joints angles which will bring the hand to this position. This approach is simple but sufficient for our purposes, however, usage of more complex motor control algorithm is allowed but they are behind the scope of our research.

Combining object vision and inverse kinematics thus provides the robot with a behavioral capability to localize a specified object and grasp it. This allowed the creation of an extended set of atomic action described in table 3.

Table 3 Set of atomic action. "Take the leg" is a vision-based action.

Motor Command	Resulting Actions
Ready Position	Move bith arms to a ready position above the workspace (6DOF)
Take the $VAR1 leg.	Visually localize the $VAR1 colored object, and grasp it. $VAR1 = {green, yellow, rose, orange}. (6DOF)
Open(Right,Left)	Open right/left hand (1DOF)
Close(Right,Left)	Close right/left hand (1DOF)
Turn(Right,Left,Center)	Rotate chest right, left or center (1DOF)
Reach(Right,Left)	Extend right/left arm (5DOF)

2.2.2 Learning Arguments Taking Procedures

In the previous study, the learned programs were fixed sequences of actions, and thus had strong requirements on the invariant conditions of execution. In [3] we extended such methods in a complimentary way by using spoken language to create complex motor procedures or behaviors that are flexible to changes in the environment. This robustness is provided through the learning of procedures (i.e. actions) that take arguments (i.e. objects that are manipulated in the action). The learned behaviors correspond to procedures that take arguments, e.g. "Give me X", where the robot uses vision and motion planning to localize X which can be arbitrarily located within the robots perceivable works space. As in the previous study, this procedures definition is controlled by a set of learning and interaction commands presented in table 4. The updated version of the SLP allowed multiple procedures to be created, and allows these procedures to take arguments. While the general solution is to allow an arbitrary number of procedures to be created and named, we will describe only two specific procedures in this section: "**Give me the X**" and "**Hold this**".

When the user issues a command such as "**give me the green leg**", the SLP determines if it has a corresponding procedure, by looking up that command in the command database. If there is no procedure defined, the robot asks the user to "show me how to give you the green block". The user proceeds to issue action commands to achieve the desired task that will make up this sequence. Of particular interest in this case, if the user issues a command with an argument as a component of the new behavior to be learned, such as "grasp the green leg," then the argument to the grasp command becomes bound to the argument for the give

command. When the user has finished the part of the task he wants to program, he issues the "OK" command. This results in the action sequence being written to a file. Now, when "**give me the rose leg**" is issued, this file is read into an array, and the commands are sequentially executed. Importantly, the value "rose" is bound to variable $VAR1 which is passed to any commands that take arguments in the procedure, notably the "**take the X leg**" command. This "natural" programming of argument-taking procedures gives the user a powerful tool for specifying generalized behaviors.

Table 4 Programming commands.

Commands	Correspondence
Give me the $VAR1 leg	Learned procedure
Hold this	Learned procedure
OK	Store encoded command sequence
Wait	Interrupt command execution until a spoken "continue" command is issued
Continue	Terminate the "wait" pause and resume execution

2.2.3 Experimental Results

Here again we used the Table Construction Task as a benchmark for the SLP. We will describe first how the user teach the two procedures "give me X" and "hold this", then we will show how to use them and study the effect of these procedures on the task execution.

"Give me the X" Procedure

<Robot initiates the interaction> *Describe an action.*
Give me the green leg.
Please show me how to give you the green leg.
Take the green leg.
Attempting to grasp the green leg. <System executes the Vision and Motion Planning to place the hand above the leg.>
Is that good ? <Robot asks user if the hand is correctly placed>. Yes.
Attempting to grasp. <Robot descends hand, grasps the object and raises the hand – executing the Take command>
Describe an action.
Turn right. <Robot turns right>
Open right hand. <Robot opens right hand, giving object to user>.
Ready position. <Robot pulls right hand back to ready position>
OK - Finish learning.
You said OK, so I will store the learned program.

At this state in the dialog, the system writes the sequence of commands to a file, in the format: *Give me the $VAR1 leg { Take the $VAR1 leg Turn right Open right hand Ready position }* When the "give" procedure is subsequently invoked with an argument, as in "**give me the orange leg**", the word "orange" that is recognized in the $VAR1 position for the "give" command will be passed as an argument to the take command.

"Hold This" Procedure

At this point in the cooperative interaction, the user needs the robot to hold the table while he attaches the leg.

Hold this <Robot does not have a stored procedure for "hold this">
Can you show me how to hold?
Reach left <Robot extends left arm, and user places table in the hand (Fig 2C)>.
Close left hand <Robot closes left hand to hold the table (Fig 2C)>
Wait <User requests robot to wait while he attaches the leg to the table.>
Continue <User indicates he is finished>
Open left hand <Robot opens left hand to release the table (Fig 2C)>
OK - Finish learning
You said OK, so I will store the learned program.

At this point the sequence of commands including the conditional wait are now stored as an SLP procedure for immediate re-use.

Execution of Learned Procedures

Now the user can apply these learned procedures for the 2-4th legs.

Give me the orange leg <System executes the learned procedure to place the hand above the orange leg.>
Is that good? <Robot asks user if the hand is correctly placed>
Yes.
Attempting to grasp <Robot descends hand, grasps the object and raises the hand – executing the Take command>
Turning right
Opening right hand <to give the leg to the user>
Moving to ready position <"Give" procedure finished>
Describe an action
Hold this
Reaching left <Robot begins to execute the stored procedure and reaches with left hand>
Closing left hand <User places the table in the robot's hand, and robot closes hand to hold the table.>

Waiting for your signal <User attaches the leg and then indicates to the robot to go on.> **Continue**
Opening left hand <Robot releases the table, assumes the ready position and turns to the center>
Moving to ready position
Turning to center <Leg is now attached, using the "give" and "hold" procedures>

Performance Analysis of SLP Effects

In order to quantify the benefits of the learning capability, we can analyzed the number of commands and the time required to perform the "give" and "hold" procedures when each command was individually enumerated during learning, vs. when the learned procedures were employed. During the attachment of the first leg, the user issued 13 distinct commands, and getting the leg and holding the table took approximately 2:00 and 2:16 respectively. Once the procedures were learned, each of the remaining 3 legs was attached using only 4 commands, and the total execution time for the three legs was 5:24. In other words, this means that in a total task completion time of less than 10 minutes, the human was able to program two different behavior procedures and then immediately use them in order to assemble the table, yielding a significant reduction in the number of commands and execution time required.

Another important point that improves the performances on a long term point of view is that the user can create and store multiple procedures. These procedures will be available for further tasks: for example we can imagine that the "**Give me X**" procedure can be used for a drawing interaction (i.e: "**give me orange pen**") which will speed up even more the task because the robot will not have to learn again how to give an object.

2.2.4 Conclusion

In [3] we demonstrated how the functional vocabulary of the robot is extended by learning a new command and a new grammatical construction. Part of the richness of grammatical constructions in language is that they encode predicate – argument relations in a natural and understandable manner. These predicates can be complex relations between (or operations on) the set of accompanying arguments. This provides a vastly richer communication ability than does the use of single word commands. This allows the procedure to operate on an entire class of objects (i.e. those that can be "taken" by vision-based grasping capability), and makes the procedure robust to variability in the position of the argument objects. This implies more than just plugging together the specific components for vision, motor control, language and sequence learning. In particular, the "form to meaning mapping" aspect of construction grammar provides a principled method for linking of language to "meaning" in the context of predicate-argument meanings.

2.3 Flexible Common Ground

Even if the SLP as it is described in [2, 3] already provide nice programming possibilities to a naïve user, it is still possible to improve it in order to allow more powerful human robot interaction. The goal of the system is that the user can arrive with a new set of objects, and some ideas about the names for these objects, and what he wants the robot to do to help him. The construction of the common interaction framework can be divided in three different parts: setting visual recognition, teaching non predefined postures and finally having new complex behaviors learned by the robot. In this section we describe how any naive user can achieved each of these teaching processes.

2.3.1 Vocal Interaction Prerequisite

One of the main limitation of the SLP as it was until now, is that the user has to know in advance the robot's vocabulary. He had to know how were named the manipulated objects and the robot specifics actions. In this version, we propose to the user to be able to name these objects and also to create his own new postures. In order to do that, we have to address a technical limitation. To provide speech recognition, RAD needs to be fed by every word it can have to recognize. We do that using three text files (objects' names, postures' names, and behaviors' names) that contain all the words that one can use. These three files will describe the complete vocabulary of the SLP and can contain a huge set of words. However, they can be edited manually by the user if he needs to do so.

2.3.2 Real-Time Object Learning

As we mentioned in the previous section, we use for the vision of the SLP a system call Spikenet [16, 24]. The main advantage of this system is that it allows very fast object recognition based on template learning. In other words, you give to the system some models of the objects you want to use, and it recognize them in real time on the video stream from the webcam. First step is the creation of objects' models: the user displays the set of objects on the workspace and uses the mouse to circle on the camera windows the objects he wants the robot to recognize. During the spoken language interaction, the user then presents the objects one at a time, and tells the system the name of the object via spoken language as described in this sample.

Learn object names
Please, show me only one object <User removes all object from the workspace but one>
Ok.
I can see an object, please tell me its name.
Leg
So this is a leg. Do you want to teach me another object ?
No.
Learning finished, what should I do now ?
Describe the scene. <User ask the robot to describe what he see**>**
I can see a leg.

2.3.3 Improved Set of Atomic Actions

In previous studies, the set of atomic actions was completely static and it was not possible for the user to create his own postures. In this study we refined the SLP architecture to allow such a possibility. The set of atomic actions does not contain anymore fixed postures, but only atomic procedures that are defined for every robot and that can be used to build more complex procedures. We describe the set we used in our experiment on a humanoid torso in table 5. Note that for a complete humanoid robot this set can be much more extensive.

Table 5 Actual version of the set of atomic action.

Atomic Action	Correspondence
Posture $VAR	Move the robot into the posture described by $VAR (see 2.3.3.4 Posture Creation)
Move <up/down> $VAR	Move the actual arm along the vertical axis by $VAR centimeters
Move <left/right> $VAR	Move the actual arm along the horizontal axis by $VAR centimeters
Move <in/out> $VAR	Move the actual arm along the depth axis by $VAR centimeters
Open/Close	Open or close the hand of the actual arm
Reach $VAR	Visually locate the object $VAR and reach it with the actual arm
Describe	The robot tell the name of every objects that are present on the workspace
Wait	Stop execution of all commands until a "Continue" is issued
Continue	Resume execution of commands when being in a "Wait" state.

One can already notice that this set of atomic actions does not include the action "grasp X". This is because "grasp X" is not atomic and is composed by: *{reach X, move down 5, close, move up 10, posture neutral position}*. We will see how to define such a composed action later. First we describe how user can create their own postures, like *neutral position*.

2.3.4 Posture Creation

User can ask the robot to go to any posture by telling him "Posture Posture_Name". In the case of an unknown posture, the robot goes in a "relative movement command" state. This state allow the user to teach the desired posture by verbally moving the gripper with the Move (up|down; left|right; in|out) commands (Table 5). This provides a highly flexible method to create useful postures (e.g. for going to a neutral position)

Posture neutral position
Sorry, I don't know what is neutral position . Do you want to teach it to me?
Yes
Where do you want me to go?
Move up 20
Did you say to move up of 20 centimeters?
Yes <Robot moves**>**
Where do you want me to go?
Move right 20
Did you say to move right of 20 degree?
Yes <Robot moves**>**
Where do you want me to go?
Save

The user thus positions the robot using spoken language in less than 30 seconds. The posture is then added to the robot knowledge and can be embedded in a behavioral sequence. Note that in the case of a posture definition, the absolute values for the joints angles are recorded as a posture, so asking the robot to reach this posture will put him exactly as it was (not just 20cm upper/right of its actual position).

2.3.5 Complex Behavior Creation

Even if the user is able to call the atomic actions from the set of atomic actions, it is much more valuable for him to define his own actions. We exploit simple grammatical constructions that allow the mapping of natural language sentences onto specification of procedures and their arguments. This improves the SLP capabilities and make it able provide more speed improvement. In this section we show how to create a simple behavior, and then we use more complex ones in order to benchmark the system.

Learning to Grasp X

As we mentioned before, the grasp X behavior can be easily defined by the user in the case of a robot working on a flat table. Of course the grasping function can be hardcoded in the robot, and then it should be part of the set of atomic actions, but in our case it's a useful way to illustrate the power of the new SLP architecture. Here is how we defined "Grasp X" for our experiments:

Grasp leg
Sorry, I don't know how to grasp. Do you want to teach it to me?
Yes
What should I do?
Reach leg
Reaching **<**Robot moves his hand over the table's leg**>**
What should I do?

Move down 5
<Robot lay his hand on the table's leg**>**
What should I do?
Close
<Robot closes his hand and grasp the object**>**
What should I do?
Move up 10
<Robot grasp the table's leg**>**
What should I do?
Posture neutral position
<Robot moves to the previously learned neutral position**>**
What should I do?
Save

The behavior "grasp X" is now saved in the robot knowledge. The SLP inferred that the procedure took an argument and it automatically found that this argument was the same the one for the atomic action "reach". So when the user will call "grasp ball", the system will look and reach the ball and not the leg. In a very few steps the user defined a behavior that will be useful in a lot of task, and he will even be able reuse it in more complex behaviors like "Put X at Y".

Learning to Put X at Y

Here the user asks the robot to put the object "front" (it is the front part of a helicopter wood model) on a visual target labeled "3". The system does not know "put", and infers there are two arguments. As the user teaches the robot, known procedures "grasp" and "reach" have arguments that match the first and second arguments for "put" and so this argument binding order is established for the learned sequence. Note that the learning procedure takes only 71 seconds.

12:11:05 User : **put front 3**
12:11:07 Robot : *Do you mean put front 3 ?*
12:11:09 User : **yes**
12:11:16 Robot : *Tell me what to do.*
12:11:18 User : **grasp front**
12:11:22 Robot : *Loading grasp front. Tell me what to do.*
12:11:36 User : **reach 3**
12:11:39 Robot : *Loading reach. Tell me what to do.*
12:11:57 User : **open**
12:12:00 Robot : *Loading open. Tell me what to do.*
12:12:07 User : **move neutral**
12:12:10 Robot : *Loading move neutral. Tell me what to do.*
12:12:16 User : **save**

Learning to Swap X and Y

Once the "put" procedure is learned it can be used in a more complex. "swap" procedure, which itself is learned in just over two minutes.

12:38:58 User : **swap train front**
12:39:11 Robot : *Tell me what to do.*
12:39:15 User : **put train 3**
12:39:19 Robot : *Loading put train 3. Tell me what to do.*
…. Speech recognition error and recovery …
12:39:59 User : **put front 1**
12:40:02 Robot : *Loading put front 1 Tell me what to do.*
12:40:30 User : **put train 2**
12:40:34 Robot : *Loading put train 2 Tell me what to do.*
12:41:01 User : **save**

At this point the "swap" operation has been learned, using graphical markers (learned as any other object) as place holders for the objects to be swapped. The new behavior is then demonstrated to transfer to new objects, and the complex behavior is achieved in just slightly over 1 minute

12:42:10 User : **swap tail nose**
12:42:13 Robot : *Loading swap tail nose . Is it ok ?*
12:42:16 User : **yes**
12:43:15 Robot : *Tell me what to do.*

Then it's possible to use swap in an even more high level new sequence and so on. After a while of teaching the robot what he wants to do, the user is able to command the robot in a very efficient way without taking care of the most basic interactions.

2.4 Discussion

With the SLP implementation, we started to build a framework which facilitates the cooperation between human beings and robots. It allows any user to program automated behavior on any kind of robot in a very easy and natural manner. The robot is able to "learn" what the human wants to teach to him and then the user is able to take benefit of this knowledge in order to improve the way he uses the robot. Since the SLP is a kind of programming language, we can imagine multiple ways to improve it (e.g: adding "if…then" conditional commands, loops, etc.). However we can't really call that a cooperative interaction: the robot interacts with the human but it does not share any goal with him, he doesn't show any intentionality. Indeed, the SLP provides the user with a vocal puppet mastering ability but it doesn't give to the robot any choice regarding his own behavior. A great improvement of the system could be to enable it to learn even without direct teaching, only by interacting with the user. Such an ability can really make robots useful to humans and avoid the "rigid" aspect of their behavior. In the second part of this chapter we will present our efforts to bring to the SLP this human-like cooperation ability.

3 Beyond SLP – Anticipation and Cooperation

In his well known test [25], Turing defined a way to test artificial intelligence by having a human being chatting with it. Spoken language is a hard problem; however we can generalize this test to any kind of collaborative ability between an artificial system and a human. When the artificial system owns a physical body, like a robot, the interactions framework provides a perfect platform to test how the system should behave. Collaborative tasks, like the Table Construction Task that we defined previously, are situations from which we can extract primates' specific behaviors. Such behaviors should be implemented into collaborative artificial systems like robots (especially humanoids one) if we want them to be useful and friendly to human beings. In this context we started to add to the SLP anticipation ability, which is the first step toward more human-like cooperation ability. Interface between robotic and psychology gave us the fruitful possibility to go further in this direction in [5].

3.1 Anticipation – Extraction Behavior Regularities

If robots are to engage with humans in useful, timely and cooperative activities, they must be able to learn from their experience, with humans. Ideally this learning should take place in a way that is natural and comfortable for the user. The results of such learning should be that during the course of an interaction, as the robot continuously acquires knowledge of the structure of the interaction, it can apply that knowledge in order to anticipate the behavior of the user. This anticipation can be expressed both in terms of the actions performed by the robot, as well as by its style of verbal communication. Anticipation is the hallmark of cognition: von Hofsten for example says that: Actions are directed to the future and must predict what is going to happen next [26].

We have previously developed cooperation systems that allow a hands-free condition in which the user can actively perform one role in a cooperative task, while instructing the robot at the same time, such that the robot acquired new behaviors via its interaction with the human [2, 3]. The current research thus takes place in the continuity of our studies of human-robot cooperation in the context of a cooperative construction task. The Cooperative Table Construction Task has repetitive subtasks (attaching the 4 legs) which provide an opportunity for learning and performance improvement within the overall task. In previous research, the user was required to explicitly instruct the robot about when to initiate and terminate behavior learning, that is, the user was required to keep track of the segmentation of the overall task into subtasks. The goal, and novelty of the current work, is to extend the learning and anticipation capabilities of the robot within this interactive context by allowing the robot to automatically analyze ongoing behavior with respect to its internal representation of its past. This will allow the robot to anticipate what the user will say (thus improving the spoken language interface), and to take a progressively more proactive role in the interaction. Most importantly, this frees the user from requirement to explicitly segment tasks into subtasks for teaching the robot, as this is now performed automatically.

3.1.1 Multi-level Anticipation Capability

In order to achieve this anticipatory and adaptive capability we will exploit the notion of the interaction history, as the temporally extended personal sensory motor history of the robot in its interaction with the world and the human [27]. By comparing ongoing interactions with previous experience in the interaction history, the system can begin to anticipate. Depending on the level of stability of these encoded interactions, the system can commit to different levels of anticipatory and initiative-taking behavior. The stability of an interaction will increase as a function of its reuse over time. Here we describe the distinct levels of anticipation and initiative taking that will be implemented and tested. Level 1 anticipation allows the system to predict what the user will say, and thus eliminate the need for verification when the prediction holds. At Level 2 allows the system to take initiative to propose the predicted next event. At Level 3, the robot is highly confident and takes initiative to perform the predicted action.

3.1.1.1 Level 1: Dialog Anticipation

While the speech recognition provided by the CSLU RAD system is quite reliable, we systematically employ a subdialog in which, after the user makes a statement the system asks "Did you say … ?", in order to correct for recognition errors. Most often, the recognition works correctly, and so this verification step is unnecessary, and most of all, it is tiresome for the user.

When the system has recognized that the current sequences of actions matches with a sequence that has been previously executed and stored in the Interaction History, then it can anticipate what will be said next by the user. If this item matches with what is actually recognized as the user's next statement, then the system can dispense with the need to explicitly validate. This can significantly improve the smoothness of the flow of interaction.

3.1.1.2 Level 2: Action Proposition

Once a sequence has been validated at level 1 (i.e. the system correctly predicts what the user will say), then that sequence is elevated to level 2. At this level, again, when the system detects that a level 2 sequence is being executed, it will take initiative and propose to the user the next element in the predicted sequence. The user can then accept or decline the offer. In the context of a repetitive task, this is actually quite helpful as the user can rely on the record of his own previous history with the robot in order to guide ongoing action.

3.1.1.3 Level 3: Action Initiation

Once the sequence has been validated at Level 2, (i.e. the user has accepted the succession of actions proposed by the system), then it attains Level 3. At this level, when the sequence is detected, the robot takes full imitative and begins to execute the subsequent actions in the Level 3 sequence. At this level, the user is truly aided by the "apprentice" who has successively gained confidence, and can now proceed with its part of the interaction without the need to confer by language.

Fig. 3 The cooperative table construction task using iCub platform. Robot and human have to cooperate in order to build a table (A,D). The robot has to pass legs to the user (B) and hold the table while the user screws the legs (C).

3.1.1.4 Technical Implementation

The table construction task has been tested using the iCub platform for this research (Fig 3). Part of the system we developed for the current research is illustrated in Fig 4. It describes the control flow that implements the multilevel anticipation capability.

3.1.2 Experimental Results

As in the first part of this chapter, we use the Cooperative Table Construction Task to test the system. In this case we just want to test the anticipation ability and not any programming function of the SLP, so we just use a basic set of atomic actions (Reach, Grasp, Lift, Pass, Open, Hold, Wait, Release).

3.1.2.1 Assembling the table

The Cooperative Table Construction Task is interesting as it involves cooperation (the robot must pass elements to the user, and hold things while the user works), and it has a repetitive structure that allows learning. For each leg, the user will ask the robot to reach to and grasp the leg, lift it, pass it to the user and open the hand, and then finally hold the table while the user attaches the leg. In the following interaction, note how the corresponding interaction becomes streamlined and fluid as the robot acquires knowledge of the task.

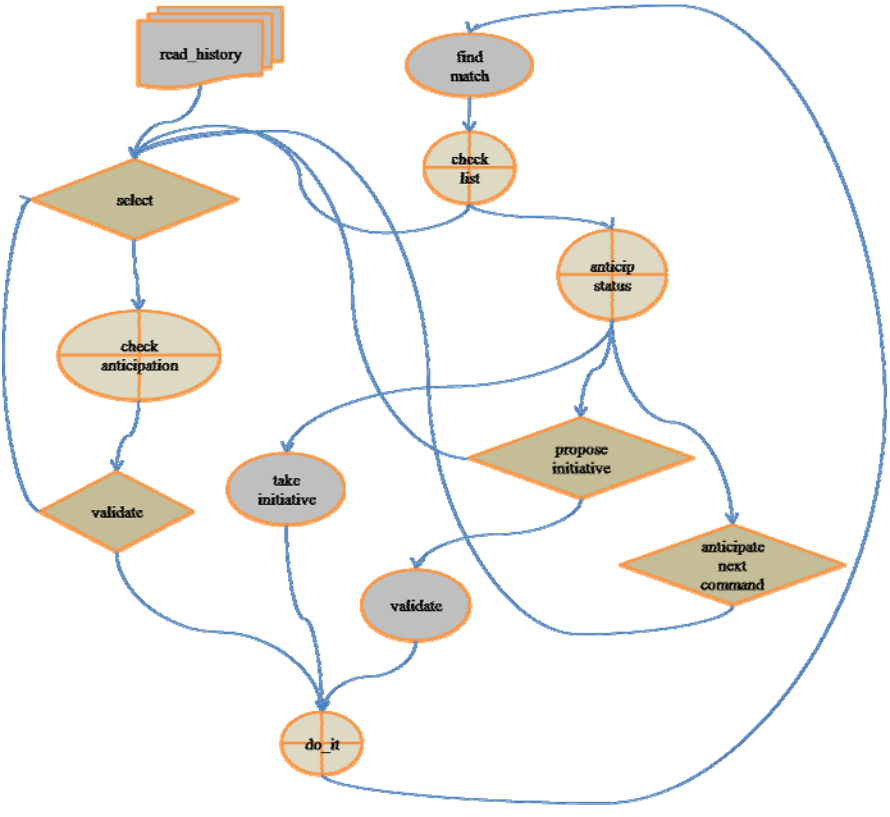


Fig. 4 Anticipation's specific part of the SLP. The invariant situation is that the last action commanded has been added to the ongoing Interaction History, and the previous two actions are continuously compared in a sliding window with the Interaction History at *find_match*. If a match is found, the next action in the Interaction History is a candidate for anticipation. At *anticp_status*, if the sequence is recognized for the first time, then at *anticipate_next_command* the next command is identified as a target for speech recognition, and the sequence anticipation level is incremented for the current sequence element. If the sequence is recognized for the second time, at *propose_initiative*, the system proposes this action to the user. If it has been successfully recognized and validated more than twice, the next element is taken for execution at *take_initiative*. At the main node, *Select*, the user chooses actions from the set of atomic actions. *check_anticipation*: Once the action is selected, the system determines what level of anticipation can be applied. If the current action is not part of a sequence of at least two elements recognized in the Interaction History, then there is no anticipation, and the system asks the user to confirm her command. If the command is part of a sequence that has been recognized for the first time – then the system will skip the verbal confirmation if command matches prediction. As already stated, at the 2nd recognition – propose to anticipate, and for 3rd + recognition: take initiative directly.

Reach.
Did you say reach? **Yes** *Reaching*
Grasp.
Did you say grasp? **Yes.** *Grasping.*
Lift.
Did you say lift? **Yes.** *Lifting*
Pass.
Did you say pass? **Yes.***Passing*
Open.
Did you say open? **Yes.** *Opening hand.*
Hold the table.
Did you say hold? **Yes.** *Holding*
Wait. Did you say wait? **Yes.** *Waiting for your signal;*
OK.
Release.
Did you say release? **Yes.** *Releasing*
Wait.
Did you say wait? **Yes.** *Waiting for your signal;*
OK.
Reach.
Did you say reach? **Yes.** *Reaching*
Grasp.
Did you say grasp? **Yes.** *Grasping.*
Lift.
Lifting.
Pass.
Passing
Open.
Opening hand
Hold the table.
Holding
Wait.
Waiting for your signal.
OK
Release.
Releasing.
Reach.
Did you say reach? **Yes.** *Reaching.*
Grasp.
Did you say grasp. **Yes.** *Grasping.*
Shall I do lift? **Yes.** *Lifting.*
Shall I do pass? **Yes.** *Passing.*
Shall I do open? **Yes.** *Opening hand.*
Shall I do hold? **Yes.** *Holding.*
Shall I do wait? **Yes.** *Waiting for your signal.*

OK
Shall I do release? **Yes.** *Releasing.*
Shall I do wait? **No**
Reach.
Did you say reach? **Yes.** *Reaching.*
Grasp.
Grasping.
I know what to do. Lifting.
I know what to do. Passing,
I know what to do. Opening hand,
I know what to do. Holding
I know what to do. Waiting for your signal
OK
I know what to do. Releasing

3.1.2.2 Performance Analysis of SLP Effects

Previous dialog provides evidence that as the levels of anticipation increase, so does the efficiency of the interaction. In order to quantify the effects of this form of anticipatory learning, we measured command processing time as the duration from the spoken onset of the user's command to the robot's completion of that action. We then grouped these actions based on the four assembly phases corresponding to the four legs of the table.

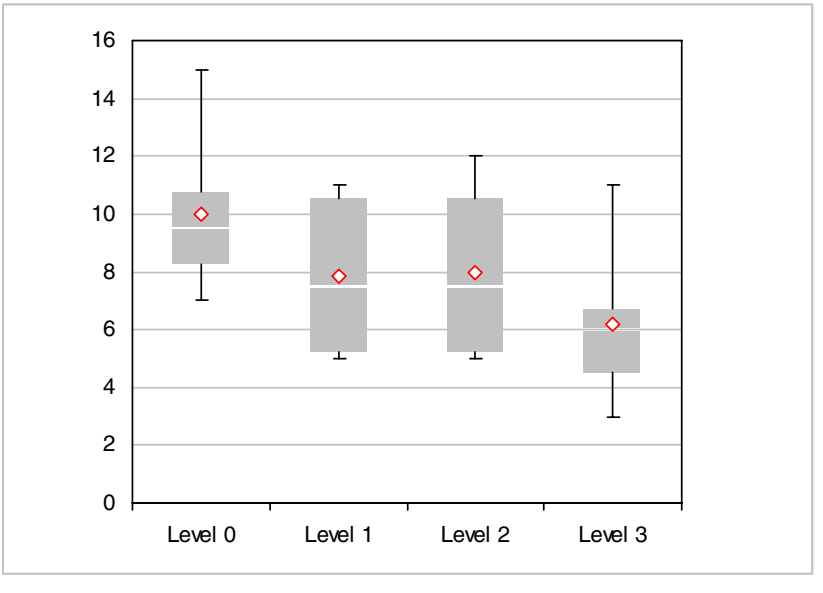


Fig. 5 Action execution times (seconds) for robot actions in the assembly of the table, based on level of anticipation. These four levels correspond to the repetitive structure of the task for the 4 legs, respectively. Whiskers indicate max and min times, Boxes 25th and 75th percentile, point – mean, white line – median.

Because of the repetitive structure of the task, the four legs correspond to four levels of anticipation – starting at a naïve state with the first leg at Level 0. Figure 5 presents the mean values and their ranges. We see a reduction in processing time over the course of the task. A repeated measures ANOVA confirmed a significant effect of Level of anticipation on completion time, F(3,15)=4,65; p<,0172.

3.1.3 Conclusion

The current research focused on the use of the interaction history for allowing the system to automatically identify useful behaviours. The user no longer needs to manage this learning explicitly, but instead can rely on the robot to extract pertinent regularities from their interactions. However, this ability mimics only a small part of the full cooperation capability of a human being. We believe that this global skill should be investigated in order to guide future human robot interaction research.

3.2 Shared Plans – Cooperation towards a Common Goal

There is a fundamental difference between robots that are equipped with sensory, motor and cognitive capabilities, vs. simulations or non-embodied cognitive systems. Via their perceptual and motor capabilities, these robotic systems can interact with humans in an increasingly more "natural" way, physically interacting with shared objects in cooperative action settings. Indeed, such cognitive robotic systems provide a unique opportunity to developmental psychologists for implementing their theories and testing their hypotheses on systems that are becoming increasingly "at home" in the sensory motor and social worlds, where such hypotheses are relevant. The current section reviews our work related to cooperation and learning [5]. It results of interaction between research in computational neuroscience and robotics on the one hand, and developmental psychology on the other. One of the key findings in the developmental psychology context is that with respect to other primates, humans appear to have a unique ability and motivation to share goals and intentions with others. This ability is expressed in cooperative behavior very early in life, and appears to be the basis for subsequent development of social cognition. Here we attempt to identify a set of core functional elements of cooperative behavior and the corresponding shared intentional representations. We then begin to specify how these capabilities can be implemented in a robotic system, the Cooperator, and tested in human-robot interaction experiments. Based on the results of these experiments we discuss the mutual benefit for both fields of the interaction between robotics and developmental psychology.

3.2.1 Introduction

There is a long history of interaction between theoretical aspects of psychology and the information and computer sciences. The "information processing" model of cognitive psychology developed by Neisser [28] and Broadbent [29] borrowed notions such as input, representation, processing and output from computer

science and applied them to the analysis of mental processes. Whether or not one holds with specific application of computing metaphors to psychological theories, it appears clear that the use of such metaphors is useful in that it confronts psychological theory with specific questions to be addressed, related to representations and processes underlying cognitive functions. Today the psychological and computing sciences are entering a new period of interaction that is linked to new technological developments in the domain of robotics. Unlike simulation and traditional artificial intelligence programs that are constrained at best to "live" in simulated artificial worlds, robots are equipped with sensory and motor capabilities that allow them to exist in the physical world of the humans that they can interact with. That is, robots can provide experimental platforms to cognitive scientists for implementing and testing theories about the intricate relation between a developing system and its physical environment. Likewise, from the robot technology perspective, robotics scientists have reasoned that the most complex behavior cannot be exclusively programmed by hand, but rather should result from adaptive and developmental mechanisms that are based on those identified in the development of physiological systems [30-32]. One of the most interesting opportunities provided by this interaction between robotics and psychology will be in the domain of developmental psychology. Research in this domain is beginning to focus in on the functional aspects of social cognition that make humans unique in the animal world. It appears that part of the uniquely human aspects concern the ability and motivation to shared intentional states with others [33].The objective of the current research is to begin to identify some of the core elements of the human ability to share intentions based on experimental and theoretical results from developmental psychology, and to then begin to determine how these elements can be implemented on a corresponding robotic system designed for interacting and cooperating with humans. We believe that this work is important because it motivates psychologists to formalize their hypotheses in sufficient detail that they can lead to implementation and testing in artificial but naturally inspired cognitive systems. Of particular interest are the underlying representations required for these shared intentions. We also believe that this work is important because it will begin to endow robots with human-like abilities to cooperate. Tomasello proposed in [33] that the human ability to share intentions develops via the interaction of two distinct capabilities. The first concerns the ability to "read" or determine the intentions of other agents through observation of their behavior, and more generally the ability to represent and understand others as intentional goal directed agents. The second capability concerns the motivation to share intentions with others. While non-human and human primates are skilled at the first - reading the intentions of others based on action and gaze direction, only humans seem to possess an additional capability that will make a significant difference. This is the motivation to cooperate: to share mental states, including goal based intentions which form the basis of cooperation. Perhaps one of the most insightful methods of establishing the properties of human social cognition is the comparison of human and great ape performance in equivalent conditions [34]. In this context, Warneken, Chen and Tomasello [35] engaged 18-to-24 month old children and young chimpanzees in goal-oriented tasks and social games which required cooperation. They were

interested both in how the cooperation would proceed under optimal conditions, but also how the children and chimps would respond when the adult stopped performing the task. The principal finding was that children enthusiastically participate both in goal directed cooperative tasks and social games, and spontaneously attempt to reengage and help the adult when he stops. In contrast, chimpanzees are uninterested in non-goal directed social games, and appear wholly fixed on attaining food goals, independent of cooperation. Warneken et al. [36, 37] thus observed what appears to be a very early human capacity for actively engaging in cooperative activities just for the sake of cooperation, and for helping or reengaging the perturbed adult. In one of the social games, the experiment began with a demonstration where one participant sent a wooden block sliding down an inclined tube and the other participant caught the block in a tin cup that made a rattling sound. This can be considered more generally as a task in which one participant manipulates an object so that the second participant can then in turn manipulate the object. This represents a minimal case of a coordinated action sequence. After the demonstration, in Trials 1 and 2 the experimenter sent the block down one of the tubes three times, and then switched to the other, and the child was required to choose the same tube as the partner. In Trials 3 and 4 during the game, the experimenter interrupted the behavior for 15 seconds and then resumed. Behaviorally, children successfully participated in the game in Trials 1 and 2. In the interruption Trials 3 and 4 they displayed two particularly interesting types of response that were (a) to reengage the experimenter with a communicative act (on 38% of the interruption trials for 24 month olds), or less often, (b) to attempt to perform the role of the experimenter themselves (on 22% of interruption trials for 24 month olds). Though (b) was considered a non-cooperative behavior, i.e. as an attempt to solve the task individually, it still indicates that the children had a clear awareness both of their role and that of the adult in the shared coordinated activity. Importantly, after only a few demonstrations of the game (and only one demonstration for the 24 month children) it was apparent that the children had a "bird's eye view" or third person representation of the interaction, allowing them to subsequently take either role in the game – that of the launcher or of the receiver of the sliding block. This implies a rather clever representation scheme which can keep track of the goal directed actions of multiple agents, and their interaction, allowing the observer to then take the role of either of the observed agents. In a related study, Warneken & Tomasello [35] demonstrated that 18 and 24 month old children spontaneously help adults in a variety of situations. This is interpreted as evidence for an altruistic motivation to help, and an ability to understand and represent the goals and intentions of others. Indeed, such helping represents a mutual commitment to the shared activity which is one of the defining features of shared cooperative activity [38].The ability to represent the action from multiple perspectives was examined more directly in a study of role reversal imitation conducted by Carpenter et al. [39]. In one experiment of this study, children observed the experimenter cover a "Big Bird" figurine with a cloth. The experimenter then asked the child "Where is big bird? Can you find him?" and the child (or the experimenter) lifted the cloth to reveal the toy. After three such demonstrations, the experimenter handed the cloth to the child and said "It's your turn now."

Approximately 70% of the 21 18 month old children tested successfully performed the role reversal. Again, this suggests that the child maintains a representation of the alternating roles of both participants in a third-person perspective that can then be used to allow the child to take on either of the two roles. In order to begin to think about how such a system has come to be (and could be built), we can look to recent results in human and primate neurophysiology and neuroanatomy. It has now become clearly established that neurons in the parietal and the premotor cortices encode simple actions both for the execution of these actions as well as for the perception of these same actions when they performed by a second agent [40, 41]. This research corroborates the emphasis from behavioral studies on the importance of the goal (rather than the details of the means) in action perception [33, 42-44]. It has been suggested that these premotor and parietal "mirror" neurons play a crucial role in imitation, as they provide a common representation for the perception and subsequent execution of a given action. Interestingly, however, it has been clearly demonstrated that the imitation ability of non-human primates is severely impoverished when compared to that of humans [33, 41]. This indicates that the human ability to imitate novel actions and action sequences in real time (i.e. after only one or two demonstrations) relies on additional neural mechanisms to those found in non-human primates. In this context, a recent study of human imitation learning [45] implicates Brodmann's area (BA) 46 as responsible for orchestrating and selecting the appropriate actions in novel imitation tasks. We have recently proposed that BA 46 participates in a dorsal stream mechanism for the manipulation of variables in abstract sequences and language [46]. Thus, variable "slots" that can be instantiated by arbitrary motor primitives during the observation of new behavior sequences are controlled in BA 46, and their sequential structure is under the control of corticostriatal systems which have been clearly implicated in sensorimotor sequencing [46]. This allows us to propose that this evolutionarily more recent cortical area BA 46 may play a crucial role in allowing humans to perform compositional operations (i.e. sequence learning) on more primitive action representations in the ventral premotor and parietal motor cortices. In other words, ventral premotor and parietal cortices instantiate shared perceptual and motor representations of atomic actions, and BA46 provides the capability to compose arbitrary sequences of these atomic actions, while relying on well known corticostriatal neurophysiology for sequence storage and retrieval. The functional result is the human ability to observe and represent novel behavioral action sequences. We further claim that this system can represent behavioral sequences from the "bird's eye view" or third person perspective, as required for the cooperative tasks of Warneken et al. [36]. That is, it can allow one observer to perceive and form an integrated representation of the coordinated actions of two other agents engaged in a cooperative activity. The observer can then use this representation to step in and play the role of either of the two agents. This is a "dialogic cognitive representation," or "we intention" in that it represents the "dialog" of interaction between agents. Given this overview of some of the core functional elements of cooperative behavior and the corresponding representations (including the "bird's eye view"), we can now take on the task of beginning to specify how these capabilities can be implemented in a robotic system, and tested

in human-robot interaction experiments. When making the transition from human behaviour to technological implantation, there is the risk that the implementation will be biased in terms of specific computational or functionalist solutions. In this context, we are making a concerted effort in "cognitive systems engineering," a process in which the cognitive robotics systems we build are constrained by (1) functional requirements (i.e. specification of how the system behaves) derived from behavior from developmental psychology, and (2) architectural constraints from the neurosciences. To as large a degree as possible, we avoid arbitrary constraints from the purely computational aspects of the implementation platform.

3.2.2 The Robotic System – The Cooperator

In the current experiments the human and robot cooperate by moving physical objects to different positions in a shared work-space as illustrated in Figures 6 and 7. The cooperative activity will involve interactive tasks that preserve the important aspects of the "block launching" task of Warneken et al. [36], transposed into a domain of objects suitable for our robot system. The 4 moveable objects are pieces of a wooden puzzle, representing a dog, a pig, a duck and a cow. These pieces can be moved by the robot and the user in the context of cooperative activity. Each has fixed to it a vertically protruding metal screw, which provides an easy grasping target both for the robot and for humans. In addition there are 6 images that are

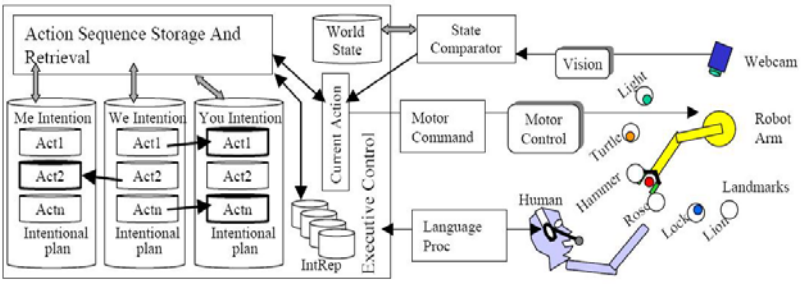


Fig. 6 Cooperation System. In a shared workspace, human and robot manipulate objects (green, yellow, red and blue circles corresponding to dog, horse, pig and duck), placing them next to the fixed landmarks (light, turtle, hammer, etc.). *Action*: Spoken commands interpreted as individual words or grammatical constructions, and the command and possible arguments are extracted using grammatical constructions in Language Proc. The resulting Action(Agent,Object,Recipient) representation is Current Action. This is converted into robot command primitives (Motor Command) and joint angles (Motor Control) for the robot. *Perception:* Vision provides object location input, allowing action to be perceived as changes in World State (State Comparator). Resulting Current Action used for action description, imitation, and cooperative action sequences. *Imitation*: The user performed action is perceived and encoded in Current Action, which is the used to control the robot under the supervision of Executive Control. *Cooperative Games:* During observations, individual actions are perceived, and attributed to the agent or the other player ("Me" or "You"). The action sequence is stored in the "We" Intention structure, that can then be used to separately represent self vs. other actions.

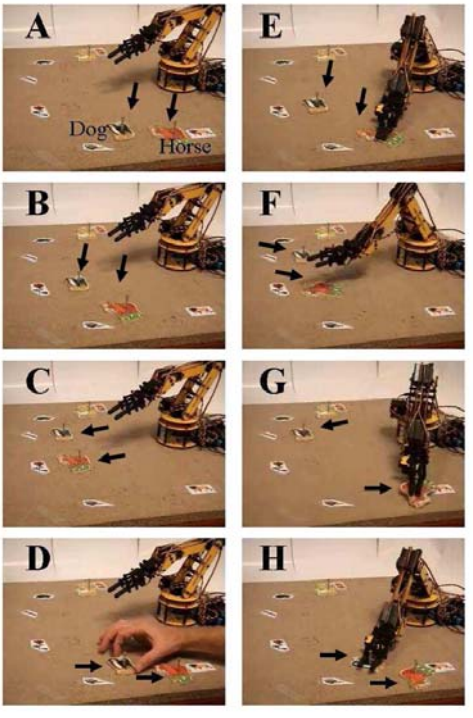


Fig. 7 Cooperative task of Exp 5-6. Robot arm Cooperator, with 6 landmarks (Light, turtle, hammer, rose, lock and lion from top to bottom). Moveable objects include Dog and Horse. In A-D, human demonstrates a "horse chase the dog" game, and successively moves the Dog then Horse, indicating that in the game, the user then the robot are agents, respectively. After demonstration, human and robot "play the game". In each of E – F user moves Dog, and robot follows with Horse. In G robot moves horse, then in H robot detects that the user is having trouble and so "helps" the user with the final move of the dog. See Exp 5 & 6.

fixed to the table and serve as landmarks for placing the moveable objects, and correspond to a light, a turtle, a hammer, a rose, a lock and a lion, as partially illustrated in Fig 6 & 7. In the interactions, human and robot are required to place objects in zones next to the different landmarks, so that the robot can more easily determine where objects are, and where to grasp them. Fig 6 provides an overview of the architecture, and Fig 7, which corresponds to Experiment 6 provides an overview of the actual physical state of affairs during a cooperative interaction.

3.3.2.1 Representation

The structure of the internal representations is a central factor determining how the system will function, and how it will generalize to new conditions. Based on the neurophysiology reviewed above, we use a common representation of action for both perception and production. In the context of the current study, actions involve moving objects to different locations, and are identified by the agent, the object,

and the target location the object is moved to. As illustrated in Fig 6, by taking the "short loop" from vision, via Current Action Representation, to Motor Command, the system is thus configured for a form of goal-based action imitation. This will be expanded upon below. In order to allow for more elaborate cooperative activity, the system must be able to store and retrieve actions in a sequential structure, and must be able to associate each action with its agent. We thus propose that the ability to store a sequence of actions, each tagged with its agent, provides an initial capability for dialogic cognitive representation. This form of real time sequence learning for imitation is not observed in non-human primates[41]. In this context, an fMRI study [45] which addressed the human ability to observe and program arbitrary actions indicated that a cortical area (BA46) which is of relatively recent phylogenetic origin is involved in such processes. Rizzolatti and Craighero [41] have thus suggested that the BA 46 in man will orchestrate the real-time capability to store and retrieve recognized actions, and we can further propose that this orchestration will recruit canonical brain circuitry for sequence processing including the cortico-striatal system (see [46, 47] for discussion of such sequence processing). An additional important representational feature of the system is the World Model that represents the physical state of the world, and can be accessed and updated by vision, motor control, and language, similar to the Grounded Situation Model of Mavridis and Roy [14]. The World Model encodes the physical locations of objects and is updated by vision and proprioception (i.e. robot action updates World Model with new object location). Changes observed in the World Model in terms of an object being moved allows the system to detect actions in terms of these object movements. Actions are represented in terms of the agent, the object and the goal of the action, in the form MOVE(object, goal location, agent). These representations can be used for commanding action, for describing recognized action, and thus for action imitation and narration, as seen below. In the current study we address behavioral conditions which focus on the observation and immediate re-use of an intentional (goal directed) action plan. However, in the more general case, one should consider that multiple intentional action plans can be observed and stored in a repertory (IntRep or Intentional Plan Repertory in Fig 7). When the system is subsequently observing the behavior of others, it can compare the ongoing behavior to these stored sequences. Detection of a match with the beginning of a stored sequence can be used to retrieve the entire sequence. This can then be used to allow the system to "jump into" the scenario, to anticipate the other agent's actions, and/or to help that agent if there is a problem.

3.3.2.2 Cooperation Control Architecture

As in the SLP, the spoken language control architecture illustrated in Fig 8 is implemented with the CSLU RAD. This system provides a state-based dialog management system that allows interaction with the robot (via the serial port controller) and with the vision processing system (via file i/o). Most importantly it also provides the spoken language interface that allows the user to determine what mode of operation he and the robot will work in, and to manage the interaction via spoken words and sentences. Fig 8 illustrates the flow of control of the interaction management. In the Start state the system first visually observes where all of the

objects are currently located. From the start state, the system allows the user to specify if he wants to ask the robot to perform actions via spoken commands (Act), to imitate the user, or to play (Imitate/Play). In the Act state, the user can specify actions of the form "Put the dog next to the rose" and a grammatical construction template [8-11, 46] is used to extract the action that the robot then performs, in the form *Move(object, location)*. In the Imitate state, the robot first verifies the current state (Update World) and then invites the user to demonstrate an action (Invite Action). The user shows the robot one action. The robot then begins to visually observe the scene until it detects the action, based on changes in object locations detected (Detect Action). This action is then saved and transmitted (via Play the Plan with Robot as Agent) to execution (Execute action). A predicate(argument) representation of the form Move(object, landmark) is used both for action observation and execution. Imitation is thus a minimal case of Playing in which the "game" is a single action executed by the robot. The more general case corresponds to "games" in which the robot and human will take turns in the execution of a shared plan. In the current implementation of this, the user

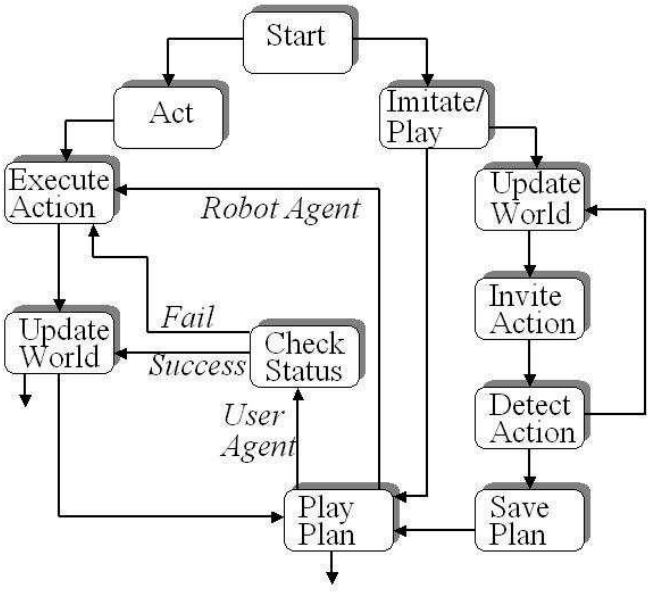


Fig. 8 Spoken Language Based Cooperation flow of control. Interaction begins with proposal to act, or imitate/play a game. Act – user says an action that is verified and executed by robot. World Model updated based on action. Downward arrow indicates return to Start. Imitate/Play – user demonstrates actions to robot and says who the agent should be when the game is to be played (e.g."You/I do this"). Each time, system checks the state of the world, invites the next action and detects the action based on visual object movement. When the demo is finished, the plan (of a single item in the case of imitation) is stored and executed (Play Plan). If the user is the agent (encoded as part of the game sequence), system checks execution status and helps user if failure. If robot is agent, system executes action, and then moves on to next item.

can demonstrate multiple successive actions, and indicate the agent (by saying "You/I do this") for each action. Improvements in the visual processing will allow the more general case in which the system can observe two agents interacting and attribute each action to its respective agent. The resulting intentional plan specifies what is to be done by whom. When the user specifies that the plan is finished, the system moves to the Save Plan, and then to the Play Plan states. For each action, the system recalls whether that action is to be executed by the robot or the user. Robot execution takes the standard Execute Action pathway. User execution performs a check (based on user response) concerning whether the action was correctly performed or not. Interestingly, the ability of the robot to "help" the user comes quite naturally, based on the shared intentional plan. If the user action is not performed, the robot "knows" the failed action based on its own representation of the plan. The robot can thus communicate with the user, and if the user agrees, the robot can help by performing the action itself. Thus, "helping" was quite naturally implemented by combining an evaluation of the user action, with the existing capability to perform a stored action representation. Still, it is worth noting that one crucial difference between the helping by the robot and what Warneken et al. tested in the helping study [35] was that the children and chimpanzees helped the other *with* their action, not just performing the other's action completely, but complementing the other's action.

3.3.2.3 Bird's Eye View and Role Reversal

In an initial set of experiments (Experiments 1-6 below), the "intentional plan" was represented for the robot as a sequence of actions in the "We Intention" of Fig 6, with the attribution of the agent fixed for each action. We know however from the experimental results of Warneken et al. [36], and from the role reversal studies of [39] that this representation is flexible, in the sense that the child can take on the role of either of the two represented agents. Once the adult indicates the role he takes, the child then spontaneously adapts and takes the other role. In the current system, we thus introduce a new capability in which, prior to the playing of the game, the roles can be determined and modified. When control reaches the "Plan Play" node in the controller, i.e. after a new game has been demonstrated, or after the user chooses to play the old game, the robot now asks the user if he wants to go first. If the user responds yes, then the roles of user and robot remain as they were in the demonstration. If the user says no, then the roles are reversed. Reversal corresponds to systematically reassigning the agents (i.e. robot or user) associated with each action. Indeed, technically it would be possible that based upon the first move by the user (or the users insistent waiting for the robot to start), the robot infers who does what (i.e. whether to reverse roles or not) and what role it will take in the cooperative plan, though this has was not implemented in the current version of the system.

3.2.3 Experimental Results

For each of the 6 following experiments, equivalent variants were repeated at least ten times to demonstrate the generalized capability and robustness of the system. In less than 5 percent of the trials overall, errors of two types were observed to

occur. Speech errors resulted from a failure in the voice recognition, and were recovered from by the command validation check (Robot: "Did you say …?"). Visual image recognition errors occurred when the objects were rotated beyond 20° from their upright position. These errors were identified when the user detected that an object that should be seen was not reported as visible by the system, and were corrected by the user re-placing the object and asking the system to "look again". At the beginning of each trial the system first queries the vision system, and updates the World Model with the position of all visible objects. It then informs the user of the locations of the different objects, for example "The dog is next to the lock, the horse is next to the lion." It then asks the user "Do you want me to act, imitate, play or look again?", and the user responds with one of the action-related options, or with "look again" if the scene is not described correctly.

3.2.3.1 Experiment 1: Validation of Sensorimotor Control

In this experiment, the user says that he wants the "Act" state (Fig 6), and then uses spoken commands such as "Put the horse next to the hammer". Recall that the horse is among the moveable objects, and hammer is among the fixed landmarks. The robot requests confirmation and then extracts the predicate-argument representation - *Move(X to Y)* - of the sentence based on grammatical construction templates. In the Execute Action state, the action *Move(X to Y)* is decomposed into two components of *Get(X)*, and *Place-At(Y)*. *Get(X)* queries the World Model in order to localize X with respect to the different landmarks, and then performs a grasp at the corresponding landmark target location. Likewise, *Place-At(Y)* simply performs a transport to target location Y and releases the object. Decomposing the *get* and *place* functions allows the composition of all possible combinations in the *Move(X to Y)* space. Ten trials were performed moving the four objects to and from different landmark locations. In these ten experimental runs, the system performed correctly. Experiment 1 thus demonstrates that the system has (1) the ability to transform a spoken sentence into a Move(X to Y) command, (2) the ability to perform visual localization of the target object, and (3) the sensory-motor ability to grasp the object and put it at the specified location.

3.2.3.2 Experiment 2: Imitation

In this experiment the user chooses the "imitate" state. As stated above, imitation is centered on the achieved ends – in terms of observed changes in state – rather than the detailed trajectory or means by which these ends were achieved [39, 42]. Before the user performs the demonstration of the action to be imitated, the robot queries the vision system, and updates the World Model (Update World in Fig 6) and then invites the user to demonstrate an action. The robot pauses, and then again queries the vision system and continues to query until it detects a difference between the currently perceived world state and the previously stored World Model (in State Comparator of Fig 6, and Detect Action in Fig 8, corresponding to an object displacement. Extracting the identity of the displaced object, and its new location (with respect to the nearest landmark) allows the formation of an *Move(object, location)* action representation. Before imitating, the robot operates

on this representation with a meaning-to-sentence construction in order to verify the action to the user, as in "Did you put the dog next to the rose?" It then asks the user to put things back as they were so that it can perform the imitation. At this point, the action is executed (Execute Action in Fig 8). In ten experimental runs the system performed correctly. This demonstrates the ability of the system to detect the final "goal" of user-generated actions as defined by visually perceived state changes, and the utility of a common representation of action for perception, description and execution.

3.2.3.3 Experiment 3: A Cooperative Game

The cooperative game is similar to imitation, except that there is a sequence of actions (rather than just one), and the actions can be effected by either the user or the robot in a cooperative, turn taking manner. In this experiment, the user responds to the system request and enters the "play" state. In what corresponds to the demonstration in Warneken et al. [36] the robot invites the user to start showing how the game works. Note that in these experiments, two experimenters demonstrate the game and the subject is observing this interaction from a third-person-perspective. The experimenters invite the child to see how the game works by showing it to them first and then have them participate afterwards. For technical limitations of the visual system, we currently use the following modification: The user then begins to perform a sequence of actions that are observed by the robot. For each action, the user specifies who does the action, i.e. either "you do this" or "I do this". The intentional plan is thus stored as a sequence of action-agent pairs, where each action is the movement of an object to a particular target location. Note that because the system can detect actions, if it is capable of identifying distinct users (by some visual cue on their hands for example) then the system could observe two humans perform the task, thus adhering more closely to the protocol of Warneken et al. [36]. In Fig 6, the resulting interleaved sequence is stored as the "We intention", i.e. an action sequence in which there are different agents for different actions. When the user is finished he says "play the game". The robot then begins to execute the stored intentional plan. During the execution, the "We intention" is decomposed into the components for the robot (Me Intention) and the human (You intention). In one run, during the demonstration, the user said "I do this" and moved the horse from the lock location to the rose location. He then said "you do this" and moved the horse back to the lock location. After each move, the robot asks "Another move, or shall we play the game?" When the user is finished demonstrating the game, he replies "Play the game." During the playing of this game, the robot announced "Now user puts the horse by the rose". The user then performed this movement. The robot then asked the user "Is it OK?" to which the user replied "Yes". The robot then announced "Now robot puts the horse by the lock" and performed the action. In two experimental runs of different demonstrations, and 5 runs each of the two demonstrated games, the system performed correctly. This demonstrates that the system can learn a simple intentional plan as a stored action sequence in which the human and the robot are agents in the respective actions.

3.2.3.4 Experiment 4: Interrupting a Cooperative Game

In this experiment, everything proceeds as in experiment 3, except that after one correct repetition of the game, in the next repetition, when the robot announced "Now user puts the horse by the rose" the user did nothing. The robot asked "Is it OK" and during a 15 second delay, the user replied "no". The robot then said "Let me help you" and executed the move of the horse to the rose. Play then continued for the remaining move of the robot. This illustrates how the robot's stored representation of the action that was to be performed by the user allowed the robot to "help" the user.

3.2.3.5 Experiment 5: A More Complex Game

Experiment 3 represented the simplest behavior that could qualify as a cooperative action sequence. In order to more explicitly test the intentional sequencing capability of the system, this experiment replicates Exp 3 but with a more complex task. In this game, the user starts by moving the dog, and after each move the robot "chases" the dog with the horse, until they both return to their starting places. Action User identifies agent User Demonstrates Action Ref in Figure 6 1. I do this Move dog from the lock to the rose B 2. You do this Move the horse from the lion to the lock B 3. I do this Move the dog from the rose to the hammer C 4. You do this Move the horse from the lock to the rose C 5. You do this Move the horse from the rose to the lion D 6. I do this Move the dog from the hammer to the lock D Table 1. Cooperative "horse chase the dog" game specified by the user in terms of who does the action (indicated by saying) and what the action is (indicated by demonstration). Illustrated in Fig 6. As in Experiment 3, the successive actions are visually recognized and stored in the shared "We Intention" representation. Once the user says "Play the game", the final sequence is stored, and then during the execution, the shared sequence is decomposed into the robot and user components based on the agent associated with each action. When the user is the agent, the system invites the user to make the next move, and verifies (by asking) if the move was OK. When the system is the agent, the robot executes the movement. After each move the World Model is updated. As in Exp 3, two different complex games were learned, and each one "played" successfully 5 times. This illustrates the learning by demonstration [15] of a complex intentional plan in which the human and the robot are agents in a coordinated and cooperative activity.

3.2.3.6 Experiment 6: Interrupting the Complex Game

As in Experiment 4, the objective was to verify that the robot would take over if the human had a problem. In the current experiment this capability is verified in a more complex setting. Thus, when the user is making the final movement of the dog back to the "lock" location, he fails to perform correctly, and indicates this to the robot. When the robot detects failure, it reengages the user with spoken language, and then offers to fill in for the user. This demonstrates the generalized ability to help that can occur whenever the robot detects the user is in trouble.

3.2.3.7 Experiment 7: Role reversal in the Complex Game

Carpenter et al. [43] demonstrated that 18 month old children can observe and participate in a cooperative turn-taking task, and then reverse their role, indicating that they develop a third person "bird's eye view" perspective of the interaction.

The current experiment tests the ability of the system to benefit from the "bird's eye view" representation of the shared intentional plan in order to take either role in the plan. In one test, the same "old game" from experiments 5 and 6 was used, with the modified version of the system that asks, prior to playing the game "do you want to go first". To test the role reversal, the human responds "no". In the demonstrated game, the human went first, so the "no" response constitutes a role reversal. The system thus systematically reassigns the You and Me actions of the We intention in Fig 6. Once this reassignment has been made, then the shared plan execution mechanism proceeds in the standard manner. The system successfully performed this role reversal. Again, it is technically feasible for the robot to infer its own role based upon only what the user does, by detecting whether or not the user initiates the first action in the game, and such an implementation will be pursued in our future work.

3.3 Discussion

Significant progress has been made in identifying some of the fundamental characteristics of human cognition in the context of cooperative interaction, particularly with respect to social cognition [48-51]. Breazeal and Scassellati [52] investigate how perception of socially relevant face stimuli and object motion will both influence the emotional and attentional state of the system and thus the human-robot interaction. Scassellati [53] further investigates how developmental theories of human social cognition can be implemented in robots. In this context, Kozima and Yano [50] outline how a robot can attain intentionality – the linking of goal states with intentional actions to achieve those goals – based on innate capabilities including: sensory-motor function and a simple behavior repertoire, drives, an evaluation function, and a learning mechanism. The abilities to observe an action, determine its goal and attribute this to another agent are all clearly important aspects of the human ability to cooperate with others. The current research demonstrates how these capabilities can contribute to the "social" behavior of learning to play a cooperative game, playing the game, and helping another player who has gotten stuck in the game, as displayed in 18-24 month old children [35, 36]. While the primitive basis of such behavior is visible in chimpanzees, its full expression is uniquely human. As such, it can be considered a crucial component of human-like behavior for robots. The current research is part of an ongoing effort to understand aspects of human social cognition by bridging the gap between cognitive neuroscience, simulation and robotics [7-11, 46, 47]. The experiments presented here indicate that functional requirements derived from human child behavior and neurophysiological constraints can be used to define a system that displays some interesting capabilities for cooperative behavior in the context of imitation. Likewise, they indicate that evaluation of another's progress, combined with a representation of his/her failed goal provides the basis for the human characteristic of "helping." This may be of interest to developmental scientists, and the potential collaboration between these two fields of cognitive robotics and human cognitive development is promising. The developmental cognition literature lays out a virtual roadmap for robot cognitive development [33, 47]. In this context, we are currently investigating the development of hierarchical means-end action

sequences [44]. At each step, the objective will be to identify the characteristic underlying behavior and to implement it in the most economic manner in this continuously developing system for human robot cooperation. Here we begin to address the mechanisms that allow agents to make changes in perspective. In the experiments of Warneken et al. the child watched two adults perform a coordinated task (one adult launching the block down the tube, and the other catching the block). At 18-24 months, the child can thus observe the two roles being played out, and then step into either role [43]. This indicates a "bird's eye view" representation of the cooperation, in which rather than assigning "me" and "other" agent roles from the outset, the child represents the two distinct agents A and B, and associates one of these with each action in the cooperative sequence. Then, once the perspective shift is established (by the adult taking one of the roles, or letting the child choose one) the roles A and B are assigned to me and you (or vice versa) as appropriate. This is consistent with the system illustrated in Fig 6. We could improve the system: rather than having the user tell the robot "you do this" and "I do this," the vision system can be modified to recognize different agents who can be identified by saying their name as they act, or via visually identified cues on their acting hands. In the current system we demonstrate that the roles associated with "you" and "me" can be reversed. More generally, they can also be dissociated from "you" and "me" and linked with other agents. The key is that there is a central representation corresponding to the "We intention" in Figure 6, which allows the "bird's eye view", and a remapping mechanism that can then assign these component actions to their respective agents (corresponding to the Me and You intentions in Fig 6). Clearly there remains work to be done in this area, but the current results represent a first step in specifying how these intentional representations could be implemented. Indeed, we take a clear position in terms of internal representational requirements, defined by a hybrid form of representation. At one level, online action and perception are encoded in an "embodied" form in terms of joint angles, and continuous values from the visual system. At a different level, "we intentions" which allow an extension in time, are distinct sequences of predicate-argument propositional elements. Thus there is a continuum of embodiment and representation. In the context of representing a joint activity through observation – the action perception is linked to the sensorimotor system, but the system that stores and replays these sequences can be considered to be independent. Indeed, it is this simulation capability that might well provide the basis for abstract processing [54]. More broadly speaking, though the demands of requiring implementation, robot experiments such as these can help us to shed further light on the nature and necessity of internal representations. An important open issue that has arisen through this research has to do with inferring intentions. The current research addresses one cooperative activity at a time, but nothing prevents the system from storing multiple such intentional plans in a repertory (IntRep in Fig 6). In this case, as the user begins to perform a sequence of actions involving himself and the robot, the robot can compare this ongoing sequence to the initial subsequences of all stored sequences in the IntRep. In case of a match, the robot can retrieve the matching sequence, and infer that it is this that the user wants to perform. This can be confirmed with the user and thus provides the basis for a

potentially useful form of learning for cooperative activity. Indeed, this development in the robotics context provides interesting predictions about how these inferences will be made that can be tested with children. A potential criticism of this work could hold that while it might demonstrate an interesting and sophisticated simulation, everything of interest seems to be built in rather than emergent or developed, thus of relatively thin relevance to psychologists. We would respond that any implementation must make choices about what is built in and what is emergent. Here we have built in functions that provide the ability to perceive actions, encode action-agent sequences, and to use these sequences in behaviour. What results is the open ended capability to learn arbitrary cooperative behaviors, to help, and to changes perspectives/roles. The relevance to psychologists is twofold, in terms of what the resulting system can do, and in terms of where it fails. Thus, while we have begun to implement some aspects of these intention representations, we should also stress how the robot's capabilities still differ from what these young children already do, including the following. (1) Children learn intentional plans quickly without direct teaching, but just by observing from the outside how two people interact. (2) They are not told who performs which role, but they themselves are able to parse the interaction into roles. (3) They spontaneously provide help without the experimenter asking them for help and without them asking the experimenter whether he wants help. (4) They not only help the other with his role but they insist on the partner performing his role when he interrupts. In other words, they seem to insist on the joint commitment to perform the respective roles. For the most part, these differences are "peripheral" in that they are related to the perception and action capabilities, rather than to the structure of internal representations. Point (1) will rely on a "salience" system that determines what behavior is interesting and merits learning (perhaps any behavior between multiple agents operating on the same objects). Point (2) will require vision processing that allows identification of different individuals. For points (3) and (4), the behavior is currently available, i.e. it is wholly feasible for the robot to help and to insist that the other partner participates spontaneously as the situation requires. In conclusion, the current research has attempted to build and test the Cooperator, a robotic system for cooperative interaction with humans, based on behavioral and neurophysiological requirements derived from the respective literatures. The interaction involves spoken language and the performance and observation of actions in the context of cooperative action. The experimental results demonstrate a rich set of capabilities for robot perception and subsequent use of cooperative action plans in the context of human-robot cooperation. This work thus extends the imitation paradigm into that of sequential behavior, in which the learned intentional action sequences are made up of interlaced action sequences performed in cooperative and flexible alternation by the human and robot. While many technical aspects of robotics (including visuomotor coordination and vision) have been simplified, we believe that this work makes a useful contribution in demonstrating how empirical and theoretical results in developmental psychology can be formalized to the extent that they can be implemented and tested in a robotic system. In doing so, we gain further insight into the core functions required for cooperation, and help to increase the cooperative capabilities of robots in human-robot interaction.

4 Conclusion

In the future, robots will cooperate with human beings in everyday life. Even (and especially) people that are not involved in any kind of programming or computer science should be able to command, ask for help or play with a robot. In this context, robots will have to adapt to human desires through social interaction, which include spoken language, behavior and intention recognition, etc. In this chapter, we reviewed our work in this direction. We believe that the most natural modality for social interaction is spoken language, which led us to build the Spoken Language Programming system. This tool provides a way to control a robot efficiently, but the lack of free will and intentionality of the robot is somehow restrictive and result in an interaction that is too artificial and rigid. Cooperative abilities like the anticipation of actions or the will to help people have to be embedded in the SLP in order to build a complete robotic collaborative system. Such skills will elevate the robot behaviors to a new level of fluency, allowing the robot to become a partner instead of just being commanded. Our studies have already extracted some of these skills from human interaction based experiments and we begin to implement them within the context of human-robot cooperation. Such collaborations between domains of psychology and robotic are still novel and will lead to most interesting results in the future.

Acknowledgements

This work has been supported in part by the European FP7 ICT project CHRIS, and by the French ANR Amorces.

References

1. Crangle, C., Suppes, P.: Language and Learning for Robots. In: CSLI lecture notes, Stanford (1994)
2. Dominey, P., Mallet, A., Yoshida, E.: Progress in Programming the HRP-2 Humanoid Using spoken Language. In: ICRA (2007)
3. Dominey, P., Mallet, A., Yoshida, E.: Real-Time Cooperative Behavior Acquisition by a Humanoid Apprentice. In: IEEE/RAS International Conference on Humanoid Robotics, Pittsburg, Pennsylvania (2007)
4. Dominey, P., Metta, G., Nori, F., Natale, L.: Anticipation and Initiative in Human-Humanoid Interaction. In: Humanoids (2008)
5. Dominey, P., Warneken, F.: The Basis of Shared Intentions in Human and Robot Cognition (2008) (in press)
6. Goldberg, A.: Constructions: A new theoretical approach to language. Trends in Cognitive Sciences 7, 219–224 (2003)
7. Dominey, P.: Spoken Language and Vision for Adaptive Human-Robot Cooperation. In: Hackel, M. (ed.) Humanoid Robotics ARS International, Vienna (2007)
8. Dominey, P., Boucher, J.D.: Learning To Talk About Events From Narrated Video in the Construction Grammar Framework. Artificial Intelligence 167, 31–61 (2005)

9. Dominey, P.: Learning grammatical constructions from narrated video events for human–robot interaction. In: Proceedings IEEE Humanoid Robotics Conference, Karlsruhe, Germany (2003)
10. Dominey, P., Boucher, J.D., Inui, T.: Building an adaptive spoken language interface for perceptually grounded human–robot interaction. In: Proceedings of the IEEE-RAS/RSJ international conference on humanoid robots (2004)
11. Dominey, P., Alvarez, M., Gao, B., Jeambrun, M., Weitzenfeld, A.,, M.: A Robot Command, Interrogation and Teaching via Social Interaction. In: IEEE Conference on Humanoid Robotics (2005)
12. Nicolescu, M., Mataric, M.: Learning and Interacting in Human-Robot Domains. IEEE Trans. Sys. Man Cybernetics
13. Lauria, S., Kyriacou, B.G.: T, Klein E, Mobile robot programming using natural language. Robotics and Autonomous Systems 38(3-4), 171–181 (2002)
14. Mavridis, N., Roy, D.: Grounded Situation Models for Robots:Where Words and Percepts Meet. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS (2006)
15. Zöllner, R., Asfour, T., Dillman, R.: Programming by Demonstration:Dual-Arm Manipulation Tasks for Humanoid Robots. In: Proc. IEEE/RSJ Intern. Confon Intelligent Robots and systems (IROS). (2004)
16. Thorpe, S., Delorme, A., Van Rullen, R.: Spike-based strategies for rapid processing. Neural Netw. 14(6-7), 715–725 (2001)
17. Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., Isozumi, T.: Humanoid robot hrp-2. In: IEEE International Conference on Robotics & Automation (2004)
18. Kanehiro, F., Miyata, N., Kajita, S., Fujiwara, K., Hirukawa, H., Nakamura, Y., Yamane, K., Kohara, I., Kawamura, Y., Sankai, Y.: Virtual Humanoid Robot Platform to Develop Controllers of Real Humanoid Robots without Porting. In: Int. Conference on Intelligent Robots and Systems (2001)
19. Tsagarakis, N.G., Metta, G., Sandini, G., Vernon, D., Beira, R., Becchi, F., RIghetti, L., Victor, J.S., Ijspeert, A.J., Carrozza, M.C., Caldwell, D.G.: iCub - The Design and Realization of an Open Humanoid Platform for Cognitive and Neuroscience Reseach. Advanced Robotics (Robotic platforms for Research in Neuroscience) 21 (2007)
20. Fitzpatrick, P., Metta, G., Natale, L.: Towards Long-Lived Robot Genes. Journal of Robotics and Autonomous System. Special Issue on Humanoid Technologies 56, 1–3 (2008)
21. Severinson-Eklund, K., Green, A., Hüttenrauch, H.: Social and collaborative aspects of interaction with a service robot. Robotics and Autonomous Systems 42, 223–234 (2003)
22. Calinon, S., Guenter, F., Billard, A.: On Learning the Statistical Representation of a Task and Generalizing it to Various Contexts. In: IEEE/ICRA 2006 (2006)
23. Kyriacou, T., Bugmann, G., Lauria, S.: Vision-based urbannavigation procedures for verbally instructed robots. Robotics and Autonomous Systems 51, 69–80 (2005)
24. Delorme, A., Thorpe, S.J.: SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons. Network 14(4), 613–627 (2003)
25. Turing, A.M.: Computing Machinery and Intelligence, `http://www.abelard.org/turpap/turpap.htm`
26. von Hofsten, C.: An action perspective on motor development. Trends in Cognitive Sciences 8(6), 266–272 (2004)

27. Mirza, N., Nehaniv, C.L., Dautenhahn, K., Boekhorst, R.: Grounded Sensorimotor Interaction Histories in an Information Theoretic Metric Space for Robot Ontogony. Adaptive Behavior 15, 167–187 (2007)
28. Neisser, U.: Cognitive psychology Appleton-Century-Crofts, New York (1967)
29. Broadbent, D.E.: Information processing in the nervous system. Science 150(695), 457–462 (1965)
30. Brooks, R.: Elephants Don't Play Chess. Robotics and Autonomous Systems 6(3), 3–15 (1990)
31. Pfeifer, R., Scheier, C.: Understanding Intelligence. The MIT Press, Cambridge (1999)
32. Pfeifer, R., Gómez, G.: Interacting with the real world: design principles for intelligentsystems. Artificial Life and Robotics 9(1), 1–6 (2005)
33. Tomasello, M., et al.: Understanding and sharing intentions: the origins of cultural cognition. Behav. Brain Sci. 28(5), 675–691 (2005); discussion 691–735
34. Tomasello, M., Carpenter, M.: Shared intentionality. Dev. Sci. 10(1), 121–125 (2007)
35. Warneken, F., Tomasello, M.: Altruistic helping in human infants and young chimpanzees. Science 311(5765), 1301–1303 (2006)
36. Warneken, F., Chen, F., Tomasello, M.: Cooperative activities in young children and chimpanzees. Child Dev. 77(3), 640–663 (2006)
37. Warneken, F., et al.: Spontaneous Altruism by Chimpanzees and Young Children. PLoS Biol. 5(7), e184 (2007)
38. Bratman, M.: Shared cooperative activity. The Philosophical Review 101(2), 327–341 (1992)
39. Carpenter, M., Tomasello, M., Striano, T.: Role reversal imitation and language in typicallydeveloping infants and children with Autism. Infancy 8(3), 253–287 (2005)
40. di Pellegrino, G., et al.: Understanding motor events: a neurophysiological study. Exp. Brain Res. 91(1), 176–180 (1992)
41. Rizzolatti, G., Craighero, L.: The mirror-neuron system. Annu. Rev. Neurosci. 27, 169–192 (2004)
42. Bekkering, H., Wohlschlager, A., Gattis, M.: Imitation of gestures in children is goal-directed. Q J. Exp. Psychol. A 53(1), 153–164 (2000)
43. Carpenter, M., Call, J.: The question of 'what to imitate': inferring goals and intentions from demonstrations. In: Nehaniv, C.L., Dautenhahn, K. (eds.) Imitation and Social Learning in Robots, Human sand Animals. Cambridge University Press, Cambridge (2007)
44. Sommerville, J.A., Woodward, A.L.: Pulling out the intentional structure of action: the relation between action processing and action production in infancy. Cognition 95(1), 1–30 (2005)
45. Buchine, G., Vogt, S., Ritzl, A., Fink, G.R., Zilles, K., Freund, H.-J., Rizzolatti, G.: Neural circuits Underlying Imitation Learning of Hand Actions: An Event-Related fMRI Study. Neuron 42, 323–334 (2004)
46. Dominey, P.F., Hoen, M., Inui, T.: A neurolinguistic model of grammatical construction processing. J. Cogn. Neurosci. 18(12), 2088–2107 (2006)
47. Dominey, P.: Toward a construction-based account of shared intentions in social cognition. Behav. Brain Sci. 28(5), 696 (2005)
48. Fong, T., Nourbakhsh, I., Dautenhaln, K.: A survey of socially interactive robots. Robotics and Autonomous Systems 42(3-4), 143–166 (2003)
49. Goga, I., Billard, A.: Development of goal-directed imitation, object manipulation and language in humans and robots. In: Arbib, M.A. (ed.) Action to Language via the Mirror Neuron System. Cambridge University Press, Cambridge (2005)

50. Kozima, H., Yano, H.: A robot that learns to communicate with human caregivers. In: International Workshop on Epigenetic Robotics (2001)
51. Lieberman, M.D.: Social cognitive neuroscience: a review of core processes. Annu. Rev. Psychol. 58, 259–289 (2007)
52. Breazeal, C., Scassellati, B.: Challenges in building robots that imitate people. K. Dautenhahn (2001)
53. Scassellati, B.: Theory of mind for a humanoid robot. Autonomous Robots 12(1), 13–24 (2002)
54. Barsalou, L.W.: Perceptual symbol systems. Behav. Brain Sci. 22(4), 577–609 (1999); discussion 610–660

Author Index