Cognition Rehearsed

Recognition and Reproduction of Demonstrated Behavior

Erik A. Billing

PhD Thesis, January 2012
Department of Computing Science
Umeå University
Sweden

Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden

billing@cs.umu.se
www.cs.umu.se/personal/erik-billing

ISBN 978-91-7459-349-5
ISSN 0348-0542
UMINF 11.16
December 21, 2011

Front cover by Johan Billing, Mena Abd Mohammed, and Pär Andersson.
Printed by Print & Media, Umeå University, 2011.

Abstract

The work presented in this dissertation investigates techniques for robot *Learning from Demonstration (LFD)*. LFD is a well established approach where the robot is to learn from a set of demonstrations. The dissertation focuses on LFD where a human teacher demonstrates a behavior by controlling the robot via teleoperation. After demonstration, the robot should be able to reproduce the demonstrated behavior under varying conditions. In particular, the dissertation investigates techniques where previous behavioral knowledge is used as bias for generalization of demonstrations.

The primary contribution of this work is the development and evaluation of a semi-reactive approach to LFD called *Predictive Sequence Learning (PSL)*. PSL has many interesting properties applied as a learning algorithm for robots. Few assumptions are introduced and little task-specific configuration is needed. PSL can be seen as a variable-order Markov model that progressively builds up the ability to predict or simulate future sensory-motor events, given a history of past events. The knowledge base generated during learning can be used to control the robot, such that the demonstrated behavior is reproduced. The same knowledge base can also be used to recognize an on-going behavior by comparing predicted sensor states with actual observations. Behavior recognition is an important part of LFD, both as a way to communicate with the human user and as a technique that allows the robot to use previous knowledge as parts of new, more complex, controllers.

In addition to the work on PSL, this dissertation provides a broad discussion on representation, recognition, and learning of robot behavior. LFD-related concepts such as *demonstration, repetition*, *goal*, and *behavior* are defined and analyzed, with focus on how bias is introduced by the use of behavior primitives. This analysis results in a formalism where LFD is described as transitions between information spaces. Assuming that the behavior recognition problem is partly solved, ways to deal with remaining ambiguities in the interpretation of a demonstration are proposed.

The evaluation of PSL shows that the algorithm can efficiently learn and reproduce simple behaviors. The algorithm is able to generalize to previously unseen situations while maintaining the reactive properties of the system. As the complexity of the demonstrated behavior increases, knowledge of one part of the behavior sometimes interferes with knowledge of another parts. As a result, different situations with similar sensory-motor interactions are sometimes confused and the robot fails to reproduce the behavior.

One way to handle these issues is to introduce a context layer that can support PSL by providing bias for predictions. Parts of the knowledge base that appear to fit

the present context are highlighted, while other parts are inhibited. Which context should be active is continually re-evaluated using behavior recognition. This technique takes inspiration from several neurocomputational models that describe parts of the human brain as a hierarchical prediction system. With behavior recognition active, continually selecting the most suitable context for the present situation, the problem of knowledge interference is significantly reduced and the robot can successfully reproduce also more complex behaviors.

Sammanfattning

Den här avhandlingen presenterar en undersökning av metoder för robotinlärning från demonstrationer (LFD). LFD är en väl etablerad teknik för att lära robotar nya beteenden. Avhandlingen fokuserar på LFD där en mänsklig lärare fjärrstyr roboten medan motorkommandon och sensoravläsningar spelas in. Efter demonstrationen ska roboten kunna reproducera beteendet under varierande förhållanden. Möjligheten att använda tidigare motorisk kunskap för att tolka demonstrationen undersöks. Denna information kan underlätta generalisering av demonstrationen, så att beteendet kan reproduceras även när förhållandena i omgivningen förändrats.

Det huvudsakliga vetenskapliga bidraget i den här avhandlingen är en semireaktiv algoritm för LFD benämnd *Predictive Sequence Learning (PSL)*, samt en serie utvärderingar av denna. PSL har flera intressanta egenskaper när den appliceras som metod för LFD. PSL kräver endast begränsad anpassning till nya applikationer och få antaganden introduceras. Algoritmen kan ses som en Markovmodell som anpassar tillståndsrymden efter det data som den tränas på. Genom träning genereras en modell som kan användas för att predicera eller simulera sensor- och motortillstånd som spelats in vid demonstrationer. Modellen kan användas för att kontrollera roboten så att det demonstrerade beteendet reproduceras. Modellen kan också användas för att känna igen ett pågående beteende. Detta görs genom att predicerade sensortillstånd jämförs med observerade. Denna förmåga att känna igen beteenden är viktig för LFD, både som ett sätt att kommunicera med användaren men också som en teknik som möjliggör användandet av tidigare kunskap för att tolka demonstrationer.

Utöver arbetet med PSL presenteras en diskussion om representation, igenkänning och inlärning av robotars beteende. LFD-relaterade koncept som demonstration, repetition, mål och beteende definieras och analyseras, med fokus på hur förkunskap kan introduceras genom beteendeprimitiv. Analysen resulterar i en formalism där LFD beskrivs i termer av övergångar mellan informationsrymder. Flera sätt att hantera tvetydigheter i tolkningen av demonstrationer föreslås.

Utvärderingen av PSL visar att algoritmen är användbar som en reglermetod för robotar. PSL kan på ett effektivt sätt representera och reproducera enklare beteenden, samt generalisera till nya situationer. För mer komplexa beteenden ökar dock risken att delar av den genererade modellen stör andra delar, och det inlärda beteendet kan inte reproduceras på ett korrekt sätt. Ett sätt att hantera detta problem är att introducera ett kontextlager. Kontextlagret kan stödja PSL genom att aktivera de delar av modellen som hör till den aktuella kontexten, medan övriga delar inhiberas. Den prediktiva modellen kan användas för att beräkna hur den aktuella situationen är förenlig med

olika kontexter. Roboten kan på så vis automatiskt aktivera den kontext som bäst passar den aktuella situationen. Denna metod är inspirerad av flera beräkningsmässiga modeller av nervsystemet vilka beskriver hjärnan som ett hierarkiskt prediktionssystem. När kontextlagret används minskar risken att delar av modellen stör andra delar, och roboten kan framgångsrikt reproducera mer komplexa beteenden.

Preface

This thesis consists of an introduction, an overview of relevant research, and the following seven articles.

Paper I Erik A. Billing. Cognitive Perspectives on Robot Behavior. In *Proceedings of the Second International Conference on Agents and Artificial Intelligence, Special Session on Languages with Multi-Agent Systems and Bio-Inspired Devices*, p. 373–382. INSTICC Press. Valencia, Spain, January 22–24, 2010.

Paper II Erik A. Billing and Thomas Hellström. Behavior Recognition for Segmentation of Demonstrated Tasks. In Vladimír Mařík, Jeffery M. Bradshaw, Joachim Meyer, William A. Gruver, and Petr Benda (Eds.), Proceedings of *IEEE SMC International Conference on Distributed Human-Machine Systems*, p 228–234. IEEE. Athens, Greece. March 9–12, 2008.

Paper III Erik A. Billing and Thomas Hellström. A Formalism for Learning from Demonstration. *Paladyn: Journal of Behavioral Robotics*. 1:1, p. 1–13. Versita, co-published with Springer Verlag. March 2010.

Paper IV Erik A. Billing, Thomas Hellström, and Lars-Erik Janlert. Predictive learning from demonstration. In Joaquim Filipe, Ana Fred, and Bernadette Sharp (Eds.), *Agents and artificial Intelligence: Revised Selected Papers*, p. 186–200. Springer Verlag. Communications in Computer and Information Science, 129. 2011.

Paper V Erik A. Billing, Thomas Hellström, and Lars-Erik Janlert. Behavior Recognition for Learning from Demonstration. In Proceedings of *IEEE International Conference on Robotics and Automation*, p. 866–872. IEEE. Anchorage, Alaska, May 3–8, 2010.

Paper VI Erik A. Billing, Thomas Hellström, and Lars-Erik Janlert. Robot Learning from Demonstration using Predictive Sequence Learning. To appear in A. Dutta (Ed.), *Robotic Systems - Applications, Control and Programming*. InTech. 2011.

Paper VII Erik A. Billing, Thomas Hellström, and Lars-Erik Janlert. *Simultaneous Control and Recognition of Demonstrated Behavior*. Technical Report, UMINF 11.15. Department of Computing Science. Umeå University. Sweden. 2011.

Additional work

Minor additional contributions can be found in the following papers by the author.

1. Erik A. Billing. *Simulation of Corticospinal Interaction for Motor Control*. Master Thesis. Cognitive Science Programme, Department of Integrative Medical Biology, Umeå University, Umeå, Sweden. 2004.

2. Erik A. Billing and Thomas Hellström. Behavior and Task Learning from Demonstration. In *Proceedings of the 23rd Annual workshop of the Swedish Artificial Intelligence Society (SAIS06)*, p. 151. Umeå, Sweden. May 10-12, 2006.

3. Erik A. Billing. *Representing Behavior - Distributed theories in a context of robotics*. Technical Report, UMINF 07.25. Department of Computing Science. Umeå University. Sweden. 2007.

4. Erik A. Billing. Cognition Reversed - Robot Learning from Demonstration. Licentiate Thesis. Department of Computing Science. Umeå University. Sweden. 2009.

5. Erik A. Billing, Thomas Hellström, and Lars-Erik Janlert. Model-free Learning from Demonstration. In *Proceedings of the Second International Conference on Agents and Artificial Intelligence*, p. 62-71. INSTICC Press. Valencia, Spain, January 22–24, 2010.

6. Erik A. Billing. *Achilles' heel of cognitive science*. Technical Report, UMINF 11.14. Department of Computing Science. Umeå University. Sweden. 2011.

Path to dissertation

When I started my PhD studies in 2006 I was convinced that robots able to act and learn like humans do were science fiction and not a realistic research topic. I had taken what I saw as a mature perspective on artificial intelligence, aligning to a weak AI perspective. During my undergraduate studies at the Cognitive Science Program[1], I was taught that cognition is about how humans, animals and artificial systems perceive information, process it and finally respond with some output or action. Since I had not even seen computers able to solve the perception problem in any way comparable to humans' and animals' perceptual abilities, I could not see how we could even approach the problems of implementing human-like information processing and action abilities in robots. Of course there were many specific applications were robots were successful, but my interest lay, and still lies, in a general understanding of cognition. In this context, robot learning appeared as one area where general solutions where still in focus.

I directed my attention to robot *Learning From Demonstration (LFD)*, where the robot is to learn from a set of examples or demonstrations. I focused on scenarios where a human teacher is controlling the robot pupil via teleoperation. In this context, a *demonstration* is a sequence of sensor readings and motor commands issued by the teacher during execution of the desired behavior. While this kind of scenario may not resemble the way humans teach each other, they constitute practically useful settings generalizable to many kinds of robots.

I was initially interested in how behavior should be represented in robots. When reviewing the literature on intelligent robotics and robot learning, leading up to Paper I, I had problems to find a clear consensus on what methods to use. Many of the proposed methods appeared to fit the particular application well, but it was difficult to get an understanding of which methods that would work best in the general case. Together with my supervisor Thomas Hellström[2], I decided to direct my attention to approaches that used so called behavior primitives or skills as a method for LFD. A *behavior primitive* is a simple controller that can be combined with other controllers to form more complicated behaviors. Without specifying how each primitive was to be implemented, we could still reason about how they could be combined. If we could create a system able to combine primitives on several levels, such that combined skills could constitute primitives for even more complex behaviors, a hierarchical structure would emerge able to gradually increase the robot's knowledge.

[1] Cognitive Science Program, Department of Psychology, Umeå University, Umeå, Sweden
[2] Assoc. Prof. Thomas Hellström, Department of Computing Science, Umeå University, Umeå, Sweden

We realized the importance of behavior recognition, i.e., that the robot must be able to recognize some part of a demonstration corresponding to a known behavior primitive. We developed and evaluated three techniques for behavior recognition, presented in Paper II. During this work we realized that behavior recognition was a very hard problem. Even simple demonstrations could be manifestations of a great variety of different behaviors. Small changes in the environment or the controller could result in a completely different sequence of sensory-motor events constituting the demonstration. Me and Thomas Hellström put a lot of work into analyzing and formalizing these issues, resulting in Paper III.

The conclusion was that some assumptions (biases) had to be introduced to make learning possible. Even though this was an obvious conclusion for anyone with some experience in machine learning, I couldn't help but finding it really annoying. If we have to introduce information about the behavior prior to learning, then what good does learning do? One could of course argue that we must rely on some very basic assumptions, applicable in many situations and behaviors, but this wasn't how it was done in practice. The kind of assumptions that we, and many other researchers in the field, introduced were specific things, like what aspects of objects were relevant, how positions of the robot and objects in the environment should be represented, and with which granularity the sensors could perceive the world. All these assumptions are typical examples of ontological information that is necessary for any knowledge representation. It seemed to me that what we did was building more and more information into the robot until the interpretation became obvious. This was in direct conflict with the kind of incremental learning that we aimed for when using behavior primitives.

In the middle of all this, a colleague, Daniel Sjölie[3], directed me to a book called *On Intelligence* by Jeff Hawkins. For me, this book became the first step into a field of research investigating high level computational aspects of the brain. I had been working with computational neuroscience for my Master Thesis[4], and was happy to find a book that actually put knowledge from both neuroscience and computing science together. About the same time, Ben Edin[5], supervisor for my Master Thesis, directed me to the work by Brandon Rohrer at Sandia National Laboratories. Both the work by Rohrer and Hawkins focus less on *where* in the brain it happens, and more on *how* it happens. Two things in Hawkins's book really caught my attention.

1. Cortex is primarily a memory system
2. The whole cortex performs one and the same basic computation, referred to as the *common cortical algorithm (CCA)*

If the idea about CCA is right, it should be possible to formulate it in computational terms and implement it in a computer, allowing robots to learn like humans and other animals do. While the brain does not work like a computer and a computer may not

[3] Daniel Sjölie, Department of Computing Science, Umeå University, Umeå, Sweden

[4] Erik A. Billing. *Simulation of Corticospinal Interaction for Motor Control.* Master thesis. Department of Integrative Medical Biology, Umeå University, Sweden

[5] Prof. Ben Edin, Department of Integrative Medical Biology, Umeå University, Umeå, Sweden

be an efficient platform for implementing the kind of computations performed by the brain, the brain does learn without a programmer telling it what is important and I got convinced that the best way to figure out how to do the same in robots is to understand how the brain works.

During autumn 2008 and spring 2009 I studied several models of the brain which resulted in an overview constituting large parts of the introduction chapters to my Licentiate thesis[6]. Inspired by Rohrer's work on modeling motor control, we also developed the algorithm *Predictive Sequence Learning (PSL)* which forms the basis for papers IV to VII of this dissertation. PSL is a dynamical temporal difference algorithm that introduces very few assumptions into learning. In the work presented in Paper IV, PSL was applied to an LFD-problem, learning to control a Khepera miniature robot. Based on PSL, we also developed two algorithms for behavior recognition. The new algorithms were compared with our previous work on behavior recognition. The results are presented in Paper V. The work with Paper IV and Paper V showed that PSL could be used both as a controller and as a method for behavior recognition, but also revealed a number of problems and limitations with the algorithm.

In December 2009, I presented my Licentiate thesis and during the spring that followed we explored several ways to continue the work on PSL. In order to allow larger knowledge bases, I spent some time on implementing a version of PSL that could store the knowledge base in a standard relational database. This implementation did however prove to be too slow to be useful for robotic applications. Almost half a year was spent on applying PSL to a reinforcement learning task. The idea was to use the growing knowledge base of PSL as basis for generalizing rewards, potentially creating a system that dynamically constructed a state space suitable for the particular task. This proved to be much more difficult than expected and also directed me away from LFD, that was the main focus of my dissertation. We therefore decided to cancel this direction and unfortunately I've not found the time to pick it up within the time of my PhD studies.

We also put work into a new version of PSL based on Fuzzy Logic (presented in papers IV and VII). The new version handles data with many dimensions in a better way than the original algorithm, which made it possible to scale up the evaluation environment from the Khepera robot to a human size Kompai robot. While all results presented in papers VI and VII are taken from the simulated environment, experiments on the physical robot were made parallel to this work. We were however not able to finish the experiments on the physical robot in time for this dissertation.

Inspired by the neurocomputational models reviewed in my Licentiate thesis, we also explored the possibility to create a hierarchical system based on the original PSL algorithm. While a complete implementation of such an architecture has not been done within the timespan of this dissertation, several components have been implemented and evaluated. In Paper VII, a context layer for PSL is introduced. The behavior recognition abilities of PSL is used to continuously select the most suitable context while the robot is driving. The context layer provides bias to PSL by activating some parts of the knowledge base, while inhibiting other parts. The architecture presented

[6] Erik A. Billing. Cognition Reversed - Robot Learning from Demonstration. Licentiate Thesis. Department of Computing Science. Umeå University. Umeå. Sweden. 2009.

in Paper VII could potentially be extended with a second instance of PSL running at the context level, further supporting selection of suitable contexts. Such a two-layer architecture could be further extended with more layers, producing a dynamic hierarchical system making predictions at multiple levels of abstraction (see Chapter 3 for details).

The results presented in Paper VII are promising and I am now finishing this dissertation with a feeling that I want to do so much more. I want to fully explore the possibilities of the kind of learning architecture that this dissertation embraces. In a couple of years, if I look back on this thesis, the text will probably appear different to me. My brain may have rehearsed the arguments yet a number of times, hierarchical learning systems may not appear as thrilling as they do now, and I will hopefully see their limitations much clearer. I may use different knowledge to interpret these words, and they may mean different things to me, than they do now. If that is so, I will be happy.

Acknowledgements

Wednesday, June first 2005, I made a mistake. I wrote and failed the exam on the course Intelligent Robotics at the Department of Computing Science, Umeå University. I had not studied enough, obviously. Even though I found the subject very interesting, I could not see how it would ever contribute to my future career. The failure was a close cut, and the course responsible Thomas Hellström gave me the opportunity to do a project work rather than taking the re-exam. The project went well, and when I was finished, Hellström asked me if I would like to become PhD student. And I did.

With all my heart, I now, more than six years later, express my great gratitude to my supervisor Assoc. Prof. Thomas Hellström for believing in me and giving me this opportunity to become a PhD. Thank you for all the long discussions, our many arguments, and for being there when I needed you.

Even though Hellström was the one who pulled me into PhD studies and was most present in my work for the first years, my secondary supervisor Prof. Lars Erik Janlert has also been an invaluable mentor during my PhD studies. Thank you for providing guidance and detailed comments on my work. Thank you for many interesting discussions, especially during the last part of my PhD studies. But most of all, thank you for always trusting in me and providing a sense of calm when needed. And thank you for pulling me into Swecog.

The National Graduate School of Cognitive Science (Swecog) has been a very important platform for me. I would like to thank Christian Balkenius, Nils Dahlbäck, and the other members of Swecog, for providing a very inspiring community for discussion and reflection which have helped me enforce the cognitive direction of my research. I also acknowledge Brandon Rohrer for valuable input to this work.

I thank Ola Ringdahl and Johan Tordsson who all since I came to the department have been close colleagues, helping me out with all these small, daily things that are so important. I would also like to thank Daniel Sjölie who provided some of the most valuable directions for the work presented in this thesis, and Benjamin Fonooni who made important work on the software platform used for papers VI and VII. Thanks also go to Stefan Holmgren, Lennart Edblom, and Lena Kalin Westin for giving me the opportunity to teach as much as I have during these years. It has been hard at times, but very rewarding. I also express my gratitude to all other colleagues at the department for providing a warm environment which makes it easy to go to work in the morning. Special thanks goes to Tommy Eriksson, Roland Johansson, Yvonne Löwstedt, Anne-Lie Persson, Inger Sandgren, and the department's support group, for

always keeping a positive attitude and helping out with all the practical things.

Finally, thanks goes to my dear Mena, my friends, my parents, and my brother. Thank you for dragging me out of the office, and for letting me stay at times. Thank you for helping me forget work when I need to, and for reminding me that there are other things than robots worth exploring. Without you, this dissertation would probably be more comprehensive, but I would not be wiser.

Contents

1 Introduction **1**

2 Learning from demonstration **5**
2.1 Level of imitation 6
2.2 Control 7
2.3 Recognition 9

3 Hierarchical models for learning **13**
3.1 Motivation for hierarchies 14
3.2 Hierarchical predictive learning 16

4 Summary of articles **21**

5 Contributions **23**

Paper I **33**

Paper II **47**

Paper III **59**

Paper IV **77**

Paper V **97**

Paper VI **109**

Paper VII **129**

CHAPTER 1
Introduction

Robots are more present in our society than ever before. The first autonomous cars are now driving on public streets (Taylor III, 2011), the first humanoid robots are helping customers in shopping malls (Pal Robotics, 2011) and robots are becoming increasingly important for industry (Bischoff & Guhl, 2009). It is challenging to develop robots for several reasons. Robots are often aimed for applications that require high precision, handling of heavy loads, and are typically expected to execute tasks faster and more reliably than humans. Most tasks require that actions are executed in relation to the environment, in a safe way. Furthermore, the robot is expected to act on its own, without being directly controlled by a human user. To accomplish this, the robot is given sensors, actuators, and a computational unit. A computer program, referred to as a *controller*, reads and processes information from the sensors and controls the robot by sending signals to the actuators. The work presented in this dissertation is concerned with how to design controllers for robots acting in an everyday changing environment.

A controller π is defined as a mapping from a state $x \in X$ to an action $u \in U$:

$$\pi : X \rightarrow U \tag{1.1}$$

The state space X comprises all information necessary to select an action from the action space U. What information that is necessary depends on the particular behavior, but also on the robot's physical properties and its set of sensors and actuators. The current state x_t is defined by the use of information from sensors, mapping physical measures to a sensor state $y_t \in Y$, at time t.

Robots for special purposes, like lawn mowers and vacuum cleaners, are becoming common products. More flexible, multi-purpose, robots are however still far from the market. Robots could potentially be of great support in our daily lives; at work, and in our homes. Supporting us when we grow old, or serving as fun and interesting toys to play with as kids. Research on robots is also one way to better understand ourselves. Robots can be used as models of humans and other animals, supporting research on how we perceive information and control our actions (Berthouze & Metta, 2005).

One of the things that still prevent robots from entering everyday environments, like homes and office areas, are robots' limited ability to adapt to the environment. In an engineered environment, like a factory floor, the robot can be perfectly tuned to the application at design time, or through an iterative process where the robot is tested

and modified by the developers. However, in most other environments, the developers do not have complete information about the environment in which the robot is going to be used. For example, a robot that is to support a person by fetching things in that person's home, must know where he or she usually places things. This kind of information can only to limited extent be introduced at design time and the robot must therefore be able to store and use information gained from interaction with the environment in order to change π. We say that the robot must be able to *learn*.

Robot learning is not only about adapting an existing controller to a particular environment, but also about creating new behavior. The user may for example want his robot to also place things back at the right place in the apartment. In this case, the human user must be able to describe the desired behavior for the robot. There are at least two major approaches to robot learning. The robot can try out new things by itself, while the human gives feedback in terms of reward and punishment. This kind of learning is called *Reinforcement Learning (RL)*. Alternatively, the teacher may demonstrate the desired behavior to the robot. This kind of learning is called *Learning from Demonstration (LFD)* and is the primary focus of the work presented in this dissertation. The research problem of LFD can be formulated as how to represent information gained from demonstrations in such a way that the robot can reproduce the demonstrated behavior under varying conditions.

The term *behavior* is used to denote an agent's actions in relation to the environment and the term *demonstration* is used to refer to the information gained from the teacher showing how to execute a particular behavior. The teacher often has to demonstrate a behavior several times in order to allow the robot to generalize the behavior to new situations. The question of how a set of demonstrations should be generalized is central in LFD and is the main research question investigated in this dissertation.

In order to generalize demonstrations, bias is needed. That is, some basis on which the robot can choose one generalization over another. This very general claim, illustrated by the "no free lunch" theorems (Wolpert & Macready, 1997), applies not only to LFD, but to any learning or optimization system. In the context of robot learning, the "no free lunch" can be interpreted as an argument that it is not possible to create a general learning system that is always better than another. This can be seen as a good argument for conducting research on LFD only in limited domains, where domain specific knowledge can be introduced into the system. The work presented here focuses however on general approaches to LFD. The main thesis of this work is that such a general approach to learning is both possible and desirable, with the goal of showing why, and how, general learning can be achieved.

The techniques for LFD proposed in this dissertation use previous knowledge, gained from earlier learning sessions, as bias in future learning. Such an approach changes the generalization biases as learning progresses and allows the robot to progressively learn more complex behaviors. This ability to learn by the use of existing skills, possessed by humans and many animals, is illustrated by the *zone of proximal development*. This notion was introduced by Vygotsky (1978) in an argument against the use of standardized tests as a measure of students' intelligence. Vygotsky argues that a better gauge of intelligence is obtained by contrasting the results of students solving problems with, and without, guidance from others. The basic idea is that

learning can only take place when the task is not too easy and not too hard, but within the zone of proximal development. Taking a pupil through the zone of proximal development is called *scaffolding*, a term that has also become popular in robot learning (Berk & Winsler, 1995; Otero et al., 2008). In a scaffolded learning process, the pupil takes active part by exploiting new solutions based on known skills, while the teacher is supporting, scaffolding, the learning environment such that the pupil is able to complete the task. As learning progresses, teacher support is gradually reduced until the pupil can complete the task by itself.

The idea of using the result from previous learning sessions as basis for future learning is also present within the machine learning community, for example in form of *learning to learn* (e.g. Thrun & Pratt, 1998). These techniques aim to represent knowledge gained from learning such that it increases the performance of future learning. In the field of LFD, one common approach that implements this idea is the use of so called *behavior primitives* or *skills* (Fod et al., 2002; Matarić, 2002; Nakaoka et al., 2003; Nicolescu, 2003; Peters II et al., 2003; Bentivegna, 2004; Koenig & Matarić, 2006). A behavior primitive is a pre-programmed or previously learned controller that can execute a behavior, or some part of a behavior. A demonstration is matched with known primitives, transforming LFD into the problem of selecting a set of primitive controllers that can produce the demonstrated behavior. This process can be divided into three activities:

1. **Behavior segmentation** where a demonstration is divided into smaller segments.

2. **Behavior recognition** where each segment is associated with a primitive controller.

3. **Behavior coordination**, referring to identification of rules or switching conditions for how the primitives are to be combined.

These three activities were identified during the work on Paper III and remain central through most of the work presented in this dissertation. Specifically, the problem of *behavior recognition* is studied in detail. Behavior recognition can be seen as a classification problem of sequential data and is, just like the original generalization problem, in need of bias. While many solutions exist for specific controllers, a system able to recognize learned behaviors needs a generic solution to the problem of behavior recognition. One approach that may provide a general solution is to use a forward model (predictor) in combination with the inverse model (controller). The last four papers included in this dissertation comprise an investigation of one method along these lines. We call the proposed algorithm *Predictive Sequence Learning (PSL)*.

An introduction to LFD is given in Chapter 2. Chapter 3 introduces hierarchical models for LFD. A summary of the seven papers included in this dissertation is found in Chapter 4 and primary contributions of presented work are summarized in Chapter 5.

CHAPTER 2
Learning from demonstration

Successful *Learning from Demonstration (LFD)* requires that, given a set of demonstrations, a controller is generated such that the robot can reproduce the demonstrated behavior under varying conditions. This generalization process is difficult to formalize since it is often far from obvious how a particular set of demonstrations should be generalized. The relevance of different features in the demonstrations depend on how the behavior is demonstrated, what the purpose of the behavior is, and what sensors and actuators the robot has. It is therefore difficult to provide a precise formulation of how to generalize a demonstration, or a set of demonstrations. The notion of behavior is often used in a very general sense to describe some action in response to stimuli (e.g. Arkin, 1998) where it is basically up to the human teacher to freely decide whether a certain robot behavior is successful or not.

One way to structure the research field of LFD was made by the formulation of four questions of imitation learning: *what-to-imitate*, *how-to-imitate*, *who-to-imitate,* and *when-to-imitate* (Alissandrakis et al., 2002). The first question, *what-to-imitate*, was originally introduced in a classical work by Nehaniv & Dautenhahn (1999):

> An action or sequence of actions is a successful component of imitation of a particular action if it achieves the same subgoal as that action. An entire sequence of actions is successful if it successively achieves each of a sequence of abstracted subgoals.

In other words, successful LFD requires that the goal of the demonstrated behavior be identified. In robotics, the formulation of a goal is often not trivial since it relies on background knowledge. Even if the goal state itself may be easily identified, for example a particular location in the environment, we usually demand that the robot be able to reach that location in a particular way. We may implicitly introduce the requirement that the robot be able to reach the target location within a certain time, and without hitting walls or objects on the way. It appears that there is no obvious limit to the amount of background knowledge required and it is therefore difficult to directly use a human's description of a goal as basis for the robot's behavior.

The second question, how-to-imitate, captures the problem of reproducing the behavior. Even if a suitable level of imitation is selected and the goal is identified, it is often not trivial to generate a controller that fulfills the goal. This problem is critical when the pupil has a different body structure. In this case, the teacher's actions

have to be transformed to corresponding actions for the robot pupil, introducing the *correspondence problem* (Nehaniv & Dautenhahn, 1999).

Who and when to imitate are the questions of selecting another agent that would be valuable to imitate and selecting the right time for imitation, respectively. Humans and animals do not just go around and imitate others all the time, but are able to identify parts of another agent's behavior that are valuable to copy. Parts of these problems are to identify a beginning and an end of demonstrations. Another related problem is to temporally align demonstrations such that common features can be identified even when the length of demonstrations varies. One common technique for temporal alignment is *dynamic time warping* (Myers & Rabiner, 1981), for example applied to kinesthetic demonstrations of a chess-piece moving task (Calinon et al., 2007).

Behaviors can be demonstrated to a robot in many different ways. Argall et al. (2009) outline four types of demonstrations: A direct recording of motor commands and sensor readings is referred to as an *identity record mapping.* In this case, the robot is controlled via tele-operation or by physically moving the robot's limbs (kinesthetic teaching). An external observation, e.g. a video recording of the teacher, is called a *non-identity record mapping*. This type of demonstrations poses a difficult sensing problem of detecting how the teacher has moved, but also allows much more flexible demonstration settings. The teacher may have a body identical to that of the pupil (*identity embodiment*) or a body with a different structure (*non-identity embodiment*). The latter case introduces the *correspondence problem* mentioned above. The work presented in this dissertation focuses on LFD via tele-operation. Sensor data and motor commands are recorded while a human teacher demonstrates the desired behavior by tele-operating the robot, producing demonstrations with identity in both record mapping and embodiment. In papers II to V, a miniature Khepera robot (K-Team, 2007) is used. The robot is controlled using a keyboard while motor commands and sensor readings are recorded. In papers VI and VII, a human size Kompai robot (Robosoft, 2011) in a simulated apartment environment is used. This robot is controlled using a joypad while motor commands and sensor readings are recorded in a similar way as with the previous work.

2.1 Level of imitation

One part of solving the what-to-imitate question is to identify suitable levels of abstraction at which the behavior is imitated. Stressing the hierarchical structure of behavior, Byrne & Russon (1998) identifies two distinct levels. The first level, corresponding to copying of the action sequence, is called *action-level imitation*. Actions may be executed in relation to stimuli, but with a fixed sequential structure. The second level, called *program-level imitation,* corresponds to imitation where the exact sequence of actions varies, while the overall structure of the behavior is copied. A set of demonstrations of a behavior that should be imitated at the action level is expected to have a fairly linear variability, with common statistical features. In contrast, demonstrations of a behavior at the program level may differ drastically considering sequence of actions, making it much harder to extract common features among sev-

eral demonstrations. In these situations it is often necessary to introduce high level knowledge about the behavior, often leading to specialized systems directed to LFD in limited domains.

A third level, the *effect-level imitation*, was introduced by Nehaniv & Dautenhahn (2001) in order to better describe imitation between agents with dissimilar body structures. With an effect level imitation, the imitation may look very different, both in terms of executed actions and their structure. An effect-level imitation is regarded successful if the produced effects on the environment, or the agent's relation to the environment, matches thaws of the demonstration.

Demiris & Hayes (1997) proposed three slightly different levels: *1) basic imitation* with strong similarities to the notion of action-level imitation, 2) *functional imitation* that best corresponds to effect-level imitation and 3) *abstract imitation* that represents coordination based on the presumed internal state of the agent rather than the observed behavior. Demiris and Hayes give the example of making a sad face when someone is crying. In cases like this, it is clear that quite specific external information is required to draw the connection between the demonstration and the imitation, or between two sequences of actions that the human would argue are demonstrations of the same behavior.

The quality of an imitation at a specific imitation level, or a combination of imitation levels, has been formalized as a *metric of imitation*. The metric of imitation is defined as a weighted sum over all strategy-dependent metrics on all imitation levels (Billard et al., 2003). A *strategy* should be understood as an assumption of what is relevant in the demonstrated behavior. For example, a) to move a specific object, b) to move the objects in a specific direction, c) to move the objects in a specific sequence, d) to perform a specific gesture. The approach takes the perspective that the most frequent features of demonstrations are the most important and selects a strategy with optimal agreement among demonstrations. The metric of imitation was originally demonstrated on a manipulation task with a humanoid robot and has later been applied to a number of LFD applications. With focus on the correspondence problem, Alissandrakis et al. (2005) propose an approach to imitation of manipulation tasks. The what-to-imitate problem is approached by maximizing trajectory agreements of manipulated objects, using several different metrics. Some metrics encoded absolute trajectories while other metrics encoded relative object displacement and the relevant aspects of the behavior were in this way extracted as the common features in the demonstration set.

2.2 Control

Assuming that a suitable level of imitation has been identified, that the correspondence problem is solved, and that beginnings and ends of demonstrations have been found, we are left with the problem of deriving a controller, also referred to as control policy or forward model, π (Equation 1.1). The current state x_t is taken to be the determinant of action, which implies that the state must satisfy the Markov assumption. The Markov assumption states that given the state-action pair (x_t, u_t), x_i is independent of

x_j for all $j < t < i$. In other words, (x_t, u_t) must encapsulate all information available to predict the future state x_i optimally. Argall et al. (2009) divide methods for control policy derivation into three classes:

1. **Mapping functions** use the demonstration to directly approximate a mapping from underlying states to actions.

2. **System models** use the demonstration to derive a model of the world dynamics. The model is often combined with a reward function that specifies the value of being in a certain model state, or taking an action in a certain state.

3. **Plans** use the demonstration to identify a set of pre- and post-conditions for each action. A sequence of actions can then be planned using a model of the state dynamics. The approach is often used together with additional user feedback.

As mentioned in the introduction, the information required by the controller varies with what kind of behavior it is to produce. It is therefore difficult to define a state space X suitable for any behavior. A large X, comprising very much information about the world, will introduce sensing problems when the robot is to identify the current state. Conversely, a small state space will limit the range of behaviors that the robot is able to learn. π is therefore often redefined as a function from the most recent observation $y_t \in Y$, or the agent's history of sensor and motor experiences $\eta_t = (e_1, e_2, \ldots, e_t)$, where $e_i = (u_{t-1}, y_t)$:

$$u_t = \pi(y_t) \tag{2.1}$$

$$u_t = \pi(\eta_t) \tag{2.2}$$

Both Equation 2.1 and 2.2 are typical examples of mapping functions. They have the advantage that X is not explicitly represented and less prior assumptions are introduced into the system. Approaches based on system models or plans are however using different kinds of world representations. These approaches have the advantage that complex behavior can be represented more efficiently than possible with mapping functions, but usually require a state representation that is partly predefined. See Paper III and the work by Argall et al. (2009) for longer discussions.

The present work primarily investigates techniques associated with mapping functions. In this category, a number of classification and regression approaches can be found. Billard & Hayes (1999) use a recurrent neural network trained with Hebbian learning to encode control policies for mobile robots. Hovland et al. (1996) use a *Hidden Markov Model (HMM)* trained from human demonstrations to encode a controller for an assembly task. More recently, Calinon & Billard (2005) apply an HMM to encode gestures of a humanoid robot. Another technique that has recently become popular is to encode controllers with *Gaussian Mixture Models (GMM)*. Calinon et al. (2007) applies a mixture of Gaussian/Bernoulli distributions to a chess-piece moving task for a humanoid robot. GMM was also used by Chernova & Veloso (2007) to encode controllers for a Sony AIBO robot dog and a simulated driving task. One interesting feature of this work is that the system continuously evaluates the uncertainty

of the learned Gaussian mixture set and is able to stop and ask for further directions when the uncertainty is high, reducing the need for repetitive demonstrations of simple parts of the behavior.

de Rengervé et al. (2010) compared a GMM based encoding strategy with a Neural Network (NN) based controller, using a simple robot navigation task. The NN based controller appears to perform better with limited training data, and can give more direct feedback to the teacher. In contrast, the GMM does better in well known domains, when statistical features and variations can be properly estimated. The authors suggest that the two methods could be used complementarily, applying the NN during early learning sessions, with the GMM taking over when more training has been done.

2.3 Recognition

It is difficult to imagine a situation where one knows how to execute a certain behavior, but is unable to recognize someone else doing the same thing. However, for a robot, the ability to recognize behavior does not directly come with the ability to execute that behavior. Even though a controller generated through LFD may work well for executing the demonstrated behavior, it does not automatically provide a way to recognize that behavior.

A number of approaches to segmentation and recognition of behaviors can be found in the literature. Several measures have been proposed, including variance thresholding for certain sensor modalities (Peters II et al., 2003; Koenig & Matarić, 2006) and thresholding the mean velocity of joints (Fod et al., 2002; Nakaoka et al., 2003). Nicolescu (2003) recognizes behavior primitives by matching their pre- and postconditions with current sensory states. Support Vector Machines have been used for recognition of upper body postures (Ardizzone et al., 2000) and hand grasps (Zollner et al., 2002). Bentivegna (2004) uses a nearest-neighbor classifier on state data to identify skills in a marble maze task. Pook & Ballard (1993) present an approach where sliding windows of data are classified using Learning Vector Quantization (Kohonen, 2003) in combination with a nearest-neighbor classifier. *Hidden Markov Models (HMM)* are frequently used for recognition of gestures. One example is the work by Park et al. (2005) using a camera-based system to track the position of hands an head of a human, and an HMM for recognition of gestures based on hand and head positions. Fujie et al. (2004) use an HMM in a similar way, recognizing head gestures based on the optical flow in the visual scene.

In *Paper II*, we present and evaluate three additional techniques; β*-comparison*, *AANN-comparison* and *S-comparison*. β*-comparison* compares the outcome of a controller in response to the stimuli with observed actions in the demonstration. *AANN-comparison* is based on Autoassociative Neural Networks that model each skill such that the reconstruction error can be used for behavior recognition. Finally, *S-comparison is* based on S-Learning (Rohrer & Hulet, 2006b,a) and uses the sequence length as a measure of behavior similarity.

Even though several of these techniques work well for recognizing many types of skills, none provide a general solution to the problem. When seen as a classification

problem, behavior recognition appears ill posed. Which features of the demonstration that are relevant depends a lot on the particular behavior to be recognized. A demonstration may contain relevant features ranging from simple statistical properties to symbol-level conditions such as relations between objects (Calinon, 2009). Features that frequently appear in many demonstrations are often, but not always, more relevant. Overall, it appears that there is a significant need for task-specific biases also in the recognition process.

The problem of behavior recognition may be better approached by using information provided by the controller. One common way to do this is to implement a set of modules consisting of a controller π_M paired with a forward model ϕ_M, (e.g. Billard & Hayes, 1999; Demiris, 1999; Wolpert & Kawato, 1998; Haruno et al., 2001):

$$u_t = \pi_M(x_t) \tag{2.3}$$

$$\hat{y}_{t+1} = \phi_M(x_t, u_t) \tag{2.4}$$

where $x \in X_M$ is the state of module M, tuned for the particular behavior implemented by π_M. The forward model (predictor) computes the expected observation $\hat{u}_{t+1}$ as a result of the action y_t taken in state x_t under the module's context. Simultaneously, the prediction $\hat{u}_t$, (made in previous time step $t-1$) is compared to the actual observation u_t, producing a prediction error $\Delta_t = |u_t - \hat{u}_t|^2$. A large Δ_t indicates that the forward model is not tuned to the controller, or that observations do not correspond to the behavior implemented by π_M.

Successful behavior recognition using this approach requires that there be a module implementing the behavior to be recognized. However, the approach also supports learning of modules. Consider a set of modules initially implementing random forward and inverse models. When presented some data, one module will by chance receive the smallest prediction error and be selected as the responsible module. If the controller of this module is implemented as a mapping from observations to actions, it can be directly trained from a demonstration. Similarly, the predictor can be trained to minimize the prediction error for the presented data. As long as the mapping between observations and actions does not change, the active module will benefit from training and produce decreasing prediction errors, causing it to remain active. However, if large changes in world conditions appear, previous knowledge will no longer be applicable and the active module will perform worse than random modules. As a result, another module takes over responsibility and is tuned to the new conditions.

This mechanism is often put forward as an explanation of the role of the motor system in perception and imitation (e.g. Oztop et al., 2006; Rizzolatti & Craighero, 2004). As Demiris and Hayes put it:

> The imitator is not imitating because it is understanding what the demonstrator is showing, but rather, it is understanding it because it is imitating. Imitation is used as a mechanism for bootstrapping further learning and understanding. (Demiris & Hayes, 2002)

Each module can be said to implement an internal simulation of sensor consequences in response to action, and action in response to current state. The module that produces the best prediction of observed events is selected as the best interpretation of observed events. That should be understood as one way to give the robot an inner world, a simulation of the physical world that does not rely on a pre-defined physics simulator but generated from interactions with the world. Such a simulation is inherently grounded in the robot's sensors and actuators and is consequently not subject to the symbol grounding problem (Harnad, 1990). A minimalistic implementation of this approach can be found in the work by Ziemke et al. (2005). This approach also has tight connections with the work by Barsalou and colleagues (e.g. Barsalou et al., 2003; Barsalou, 2009), describing the human cortex as a system simulating sensor percepts in relation to motor activity.

The use of a predictive measure to match the internal model with the actual world takes further support in that prediction error is proportional to the amount of free energy in the system (Friston & Stephan, 2007). Free energy is a thermodynamic measure describing the amount of work that a thermodynamic system can perform. In information theory, the term is used as a quantity that bounds the evidence of a model. For humans and other animals, the model is encoded by the brain and the data it models is the organism's interactions with the world. An organism that minimizes free energy will minimize the risk for unexpected exchanges with the environment, and consequently control entropy. This can be understood as an argument that any organism in the physical world must act to prevent surprises that can lead to potentially harmful states. Prediction error, with a specific measure of precision, is useful as a method for behavior recognition, not because forward models are very powerful classifiers, but because the purpose of acting, on a very fundamental level, is to minimize surprise (Friston, 2009).

CHAPTER 3
Hierarchical models for learning

This dissertation is a continuation of the work presented in the Licentiate thesis (Billing, 2009). The Licentiate thesis provides an analysis of several neurocomputational models of the brain (Riesenhuber & Poggio, 1999; Hawkins & Blakeslee, 2002; Haruno et al., 2003; Wolpert, 2003; Demiris & Simmons, 2006; George, 2008), all with some emphases on hierarchical structures and connections to the mirror system (Rizzolatti & Craighero, 2004). In an attempt to identify common features among these models, four criteria for general learning ability are proposed (Billing, 2009):

1. **Hierarchical structures**

   Knowledge gained from learning should be represented in hierarchies.

2. **Functional specificity**

   Knowledge gained from learning should be organized in functionally specialized modules.

3. **Forward and inverse**

   Prediction error reflects how well the state definition satisfies the Markov assumption, and by consequence a forward model can be used to improve knowledge representation when paired with an inverse model.

4. **Bottom-up and top-down**

   Both bottom-up and top-down signals must be propagated through the hierarchical structure. Bottom-up signals represent the state of modules, and top-down signals specify context.

Criteria 2 and 3 have already been discussed in the previous chapter, but criteria 1 and 4 may need some elaboration. In Section 3.1, some arguments for introducing hierarchies with the specific information flow given by Criterion 4 is presented. The argumentation leads up to a specific hierarchical architecture based on PSL, presented in Section 3.2.

3.1 Motivation for hierarchies

Hierarchical structures should be thought of as a useful and very general bias for representing knowledge about the physical world. Hierarchies are found almost everywhere in nature. Humans and animals consist of several body parts, which in turn consist of even smaller units down to cells and atoms. Scaling upwards, any organism lives in some kind of environment which can be viewed on many levels up to a scale where the earth is one component in an even greater ecosystem. The hierarchical structure of vegetables is even more apparent. Trees consists of branches which consist of even smaller branches which have leaves. Leaves have their own hierarchical structure which in fact resembles the tree itself in many ways. This apparent self-similarity of many natural structures has been extensively studied from a theoretical perspective as fractals, with works by Mandelbrot (1983) and Wolfram (2002) as prominent examples.

Hierarchies also exist in the temporal domain. Natural systems tend to have a nested organization with large-scale system variables and small-scale sub-system level variables. Large-scale system variables are often changing slower than the variables of its sub-systems Werner (1999). A common example of this temporal and spatial hierarchy can be drawn from weather. On a large scale, one can observe long term variations, such as seasons or even global warming. Simultaneously, there are local variations in the weather, such as storms and rain, which change much faster (George, 2008, p. 95).

Functional hierarchies appear to be a critical aspect of neural information processing, both spatially (Felleman & Van Essen, 1991; Hilgetag et al., 2000) and temporally (Boemio et al., 2005; Fuster, 2001). Further support comes from behavioral studies, for example the work by Byrne & Russon (1998) reporting hierarchical structure in voluntary behavior of great apes.

In a more technical respect, Criterion 1 can be motivated through an efficient division of labor between different parts of the system. A flat architecture implementing a set of functionally specific modules, as proposed in Section 2.3, relies only on forward models in order to control the switching between modules. Putting these modules into a hierarchy produces a system able to represent complex behaviors while keeping each module relatively simple. Modules at the bottom layer interacts directly with sensors and actuators of the robot at high temporal resolution. Modules higher up in the hierarchy have decreasing resolution, allowing these modules to efficiently express dependencies over longer periods of time. State variables that change slowly compared to a specific module's resolution are not included in the state description for that module, but are assumed to be provided by modules higher in up the hierarchy. This information is often referred to as the module's *context* (Wolpert & Ghahramani, 2000). Since modules higher up in the hierarchy are working at a lower temporal resolution, they can more easily capture slow variables, providing contextual information to lower modules. Conversely, variables that change quickly in comparison to the temporal resolution are handled lower in the hierarchy. This allows each module to be implemented as a semi-reactive controller, where the behavior depends on relatively recent states. Another advantage of this kind of architecture is that updates of a single

behavior or parts of a behavior will only require updates of a few modules and will not propagate changes to other parts of the system.

Wolpert & Ghahramani (2000) take the example of picking up a milk carton. The controller executing the behavior must take the amount of milk in the carton into consideration. It would be possible to consider the amount of milk as part of the module's state space, but that would pose a difficult sensing problem since the amount of milk is not directly visible. More importantly, since the amount of milk normally is constant during the operation of the module, it is not necessary to include as part of the state description. Instead, the amount of milk in the carton is treated as contextual information and handled by modules higher up in the hierarchy.

As the controller starts executing the lifting behavior, the forward model will produce predictions of expected sensor consequences of executed actions. If the expectation about the amount of milk is wrong, it will show up as large prediction errors when grasping the carton. This information is propagated upwards in the hierarchy, leading to a change in contextual information that better matches the actual amount of milk in the carton.

For this hierarchy of modules to be useful the system must be able to compute the system state at every level in the hierarchy. With the exception of modules at the bottom level (Layer 1), the input to each module is a context match for modules at the layer directly below. A context match should be understood as an estimate of how well present circumstances match the module's context, normally computed as a function of prediction error. The output of each module at Layer 2 and higher represents prior probabilities for each module in the layer below. This top-down information is often called a *responsibility signal* (e.g. Haruno et al., 2001) and is used both for control and training. During learning, only active modules are updated to fit present circumstances, while inactive modules remain unchanged. Ideally, this leads to a system where each module is in control only when the world satisfies its context. This interaction between layers in the hierarchy is one way to fulfill Criterion 4.

While the architecture presented here describes functional specificity as a set of distinct modules, it should be pointed out that this notion of module is quite different from how the term is used in many other robotic architectures, for example hybrid and deliberate systems (Murphy, 2000). These architectures often emphasize modularization in the sense that it should be possible to develop and test each module separately. This is not at all emphasized in the architecture presented here. In contrast, each module is to a high degree working in the environment constituted by the other modules and the overall behavior of the system is an emergent property of the interaction between modules, rather than a strict division of labor.

Criterion 3 was formulated to put emphasis on the functional specificity rather than on crisp modularization. An architecture based on crisp modules requires that each demonstrated sequence is interpreted as a specific sequence of selected modules, introducing a conflict between generalization and segmentation (Yamashita & Tani, 2008). This kind of architecture can potentially benefit from a partial overlap between modules. The important property of functional specificity is that errors identified under a certain context should not automatically propagate through the whole system, but only take effect within that particular context. A similar argument applied to re-

inforcement learning is put forward by Winberg & Balkenius (2007). Most work on reinforcement learning treats rewards and penalties as positive and negative changes of a single state value variable, respectively. Winberg and Balkenius demonstrate that learning time is decreased if rewards are generalized to similar states, while penalties are only applied to the current state. The reasoning behind this argument is that any over-generalizations of rewards will eventually be identified, since the agent will act to reach these states. In contrast, states with negative value will be avoided, causing any over-generalizations to remain.

One architecture able to share knowledge between different primitive behaviors is the recurrent neural network *RNNPB* (Tani et al., 2004). Both input and output layer of the network contain sensor and motor nodes, as well as nodes with recurrent connections. The input layer is given a set of extra nodes, representing *parametric bias* (PB). The network is trained to minimize prediction error, both by training the network using back-propagation and by changing the PB input vector. The PB vector is however updated slowly, such that it organizes into what could be seen as a context layer for the rest of the network. In addition to giving the network the ability to represent different behaviors that share knowledge, the PB vector can be used for behavior recognition. In a continuation of this work, Yamashita & Tani (2008) demonstrate the emergence of a functional hierarchy in an architecture based on a *continuous time recurrent neural network* (CTRNN). Context nodes (nodes that are not connected to inputs or outputs) of the network are given different time constants. Through training, slow nodes drive switching between behaviors, while fast nodes express dynamics within each behavior.

While the present work is directed to methods for robot learning, similar arguments can be found in several fields of research. The four criteria presented above embody the assumption that a wide range of cognitive phenomena can be explained through one, or a few, basic mechanisms. As mentioned in the previous section, Friston (2009) uses the notion of free-energy to describe such a fundamental mechanism, applying to any living organism. In a less mathematical formulation, Hawkins & Blakeslee (2002) introduce the notion of a *common cortical algorithm* as a basic computational description of the brain. *Activity Theory* (Kaptelinin & Nardi, 2006) describes high level cognitive function, including consciousness, as inherently grounded in action and learning as a progressive internalization of the physical world leading to hierarchical representations. A longer discussion on the brain as a prospective organ is found in the work by Sjölie (2011).

3.2 Hierarchical predictive learning

Based on the four criteria outlined in the previous section, this section describes a general purpose architecture for LFD. The architecture presented here is not fully implemented and evaluated, but builds on the results from Papers IV to VII. A central part of this architecture is the PSL algorithm. The algorithm treats control as a prediction problem, such that the next action is selected based on the sequence of recent sensory-motor events. In addition, PSL also produce predictions of expected sensor

states. While these are not directly useful for control, predictions of sensor states appear to serve well as a method for behavior recognition, as proposed in Section 2.3.

PSL has many interesting properties as a learning algorithm for robots. It is model free, meaning that it introduces very few assumptions into learning and needs little task specific configuration. PSL can be seen as a variable-order Markov model. Starting out from a reactive (first-order) model, PSL estimates transition probabilities between sensory-motor states. For states that do not display the Markov property, the order is increased and PSL models the transition probability based on several passed events. In this way, PSL will progressively gain memory for the parts of the behavior that cannot be modeled in a reactive way. Knowledge is stored as hypotheses h, describing a statistical relation between a sequence of passed events $\left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)$, and a future event e_{t+1}:

$$h : \left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right) \overset{C(h)}{\Rightarrow} e_{t+1} \tag{3.1}$$

where $C(h)$ is the confidence of h and $|h|$ denotes the length of h, i.e., the number of left-hand-side conditions. In Paper VII, we present a version of PSL where $C(h)$ is context dependent. A hypothesis that is fairly weak in the general case can in this way still receive high confidence in a particular context. Given a number of contexts C, defined as fuzzy sets over all h, the belief in each context $\lambda_t(C)$ at time t is given by Bayes' rule:

$$\lambda_t(C) = \frac{\lambda_{t-1}(C) \exp\left(\frac{\left|e_t - \hat{e}_t^C\right|^2}{2\sigma^2}\right)}{\sum_{i=1}^{N}\left[\lambda_{t-1}(C_i) \exp\left(\frac{\left|e_t - \hat{e}_t^C\right|^2}{2\sigma^2}\right)\right]} \tag{3.2}$$

where N is the number of contexts and $\hat{e}_t^C$ is the prediction produced by PSL within context C. The variance σ^2 is used as a scaling constant controlling the size of confidence changes in relation to prediction error size.

$\lambda_t(C)$ can be used to directly control which context is to be responsible at a certain time (Paper VII). Transitions between contexts may however also be controlled by another instance of PSL, running at a context level (Figure 3.1). In each time step, PSL queries the knowledge base selecting the best hypothesis based on how well each hypothesis matches observed events, and respective confidence $C(h)$. At the context level, a similar computation is made, selecting a context hypothesis h' based on the sequence of transitions between contexts. The instance of PSL running at the context level handles data with lower temporal resolution, allowing context hypotheses with relatively few left hand side conditions to stretch over fairly long periods of time.

While PSL at the observation level works directly with the sequence of sensory-motor events, the context layer works with information that can not be predicted at the observation level (Figure 3.2). Sensory-motor events that can be correctly predicted by the observation layer does not contain any new information, and is therefore filtered away. Conversely, prediction error represents new information that could not be explained by the observation layer, and is therefore regarded as more valuable. This

argument relates to the discussion in Section 2.3. A successful prediction at the context level will reduce prediction errors at the observation level, producing a system that acts to reduce predictions at both levels of abstraction.

In Paper VII, we use manually defined contexts trained from demonstrations of different behaviors. However, the teacher's interpretation of what is one behavior, and what is another, does not necessarily correspond to the robot's need to categorize experiences. It may consequently be an advantage not to rely on manually defined contexts. Automatic identification of suitable contexts could possibly be formulated as an optimization problem where the total entropy of the model, over all contexts, is to be minimized.

If a context layer can be organized dynamically, it opens the possibility to add a second context layer that interacts with the first context layer in a similar way that the first context layer interacts with the observation layer (Figure 3.2). Potentially, the hierarchy could be extended even further, to create a fully hierarchical system minimizing prediction errors at many levels of abstraction.

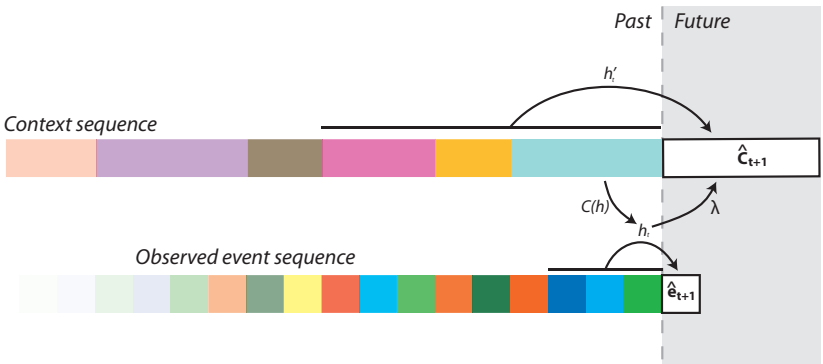

Figure 3.1: Interaction between PSL at observation level and PSL at context level. h is the selected hypothesis at observation level and h' is a hypothesis selected by PSL running at context level. $\hat{e}_{t+1}$ is the predicted event at observation level and $\hat{C}_{t+1}$ represents predicted context responsibility.

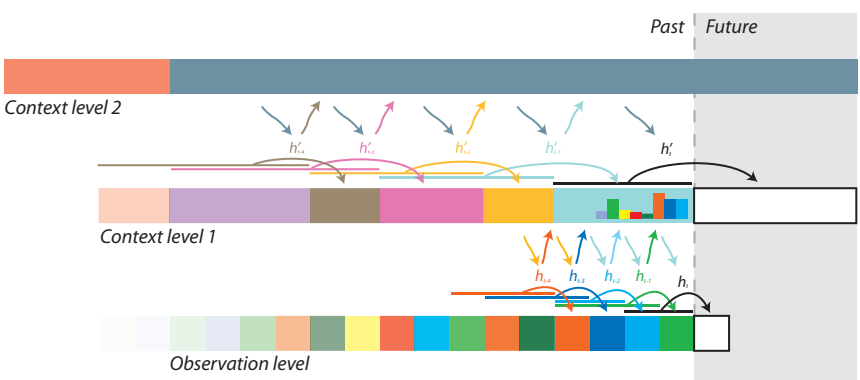

Figure 3.2: Three layer architecture with one instance of PSL is running at each level.

CHAPTER 4
Summary of articles

Starting out from a cognitive perspective, Paper I reviews the problem of robot learning and representation of behavioral knowledge in a broad sense. Several robotic paradigms are compared and put in relation with the theory of *extended mind* (Clark & Chalmers, 1998), *distributed cognition* (Hutchins, 1995) and *situated action* (Suchman, 1987). The paper was an early attempt to identify how these cognitive theories apply to robotics. The work recognizes the importance of separating the agent's perspective from the perspective of the observer and argues that many of the fundamental problems of artificial intelligence can be solved if the robot's knowledge is represented as a coordination between sensors and actuators, rather than as an explicit world model. At the same time, notions of emergent behavior and self-organization, that are often brought up in response to traditional representations, are criticized since they rarely provide a solution to the kind of problems that explicit representations do.

Paper II presents an evaluation of several techniques for behavior recognition. The use of behavior primitives with behavior recognition is one way to use previous knowledge in order to generalize the demonstration such that the behavior can be executed under varying conditions. While behavior recognition is possible to implement for particular behaviors or classes of behaviors, it appears as a very difficult problem in the general case. One reason for this may lie in a vague definition of behavior and lack of general bias for selecting one interpretation over another. Large parts of this dissertation argues for the use of a predictive measure in order to approach the problem of behavior recognition in a general way.

Paper III presents a formalization of LFD, specifically directed to LFD when using behavior primitives. Central concepts in LFD are defined, including behavior, demonstration, sensors and actuators. A possibility to use behavior primitives as a way to interpret and display a demonstration for a human user is also explored. A visualization of the robot's interpretation of the behavior could serve both as a way for the human user to evaluate the effect of learning, and as a way to give feedback to the robot.

Papers IV to VII present and evaluate a specific technique for LFD called *Predictive Sequence Learning (PSL)*. In Paper IV, PSL is used as a controller for a miniature Khepera robot (K-Team, 2007). The result from this work indicates that PSL is applicable as a controller for simpler behaviors. PSL is however unable to correctly reproduce more complex behaviors. The reason for this is that knowledge of one part of the behavior may interfere with knowledge of other parts, causing PSL to mix up situ-

ations and select inappropriate actions. This problem, called knowledge interference, could potentially be solved by integrating PSL with a high-level controller that selects which part of the PSL knowledge base that should be active in a certain situation. One such high-level controller, based on semantic networks, is proposed by Fonooni et al. (2012). Parts of the PSL knowledge base is in this architecture integrated as nodes in the semantic network.

In Paper V, an evaluation of two ways to apply PSL as a method for behavior recognition is presented, including a comparison with previous work from Paper II. The evaluation shows that PSL is able to recognize previously learned behaviors in a demonstration. The recognition is based on the same model as PSL uses for control. If this results holds in the general case, PSL could potentially constitute one component in a dynamic hierarchical system similar to the one presented in Chapter 3.

In Paper VI, a new version of PSL based on Fuzzy Logic is presented and compared to the original algorithm. Fuzzy PSL handles dimensionality better than the original, discrete, version of the algorithm. This allows us to scale up the problem to a Robosoft Kompai robot (Robosoft, 2011) acting in a simulated apartment environment. Paper VII extends this work by integrating on-line behavior recognition with the PSL-based controller. We are able to reproduce the results from both Paper IV and V using Fuzzy PSL and the new evaluation environment. The on-line behavior recognition is used to compute a responsibility signal, controlling which part of the PSL knowledge base that is active at a certain point in time. In this way, the problem of knowledge interference appears more manageable.

CHAPTER 5
Contributions

The primary contribution of this dissertation lies in the design and evaluation of *Predictive Sequence Learning (PSL)* (Paper IV to VII). The algorithm has a straight forward design that introduces the right kind of biases for general learning ability in robots. Two important attributes are the semi-reactive behavior and a progressive extension of the knowledge base. The semi-reactive behavior could probably be captured just as well by many other algorithms, for example a recurrent neural network. The progressive extending knowledge base is less common and reduces the risk for catastrophic forgetting present in many other approaches, including neural networks. On the negative side, PSL in its current form can not cope with high-dimensional data in a good way. Investigating how this could be solved could be one focus for a continuation of this work. Improved handling of high-dimensional data could possibly be achieved in a similar way that the combinatorial explosion in the temporal dimension is handled through the use of contexts (Paper VII). While we have shown that PSL does work as controller and as a method for behavior recognition in some fairly realistic robotic applications, much more work is needed to evaluate the algorithm. Comparisons with a recurrent neural network such as RNNPB (Tani et al., 2004) or Dynamic Field Theory (Amari, 1977) could be interesting directions for future work.

In Paper III, we present a formalization of LFD. This formalism itself is not revolutionary in any way, to large extent it adapts descriptions that have been standard in the field for a long time. However, the paper describes important problems of LFD, especially with a focus on introduction of bias in learning, in a novel way.

Another contribution of this dissertation is the hierarchical architecture for general purpose LFD presented in Section 3.2 that builds on four criteria for general learning ability originally presented in the Licentiate thesis (Billing, 2009). In many respects, the presented architecture is not new, similar approaches have been around in the literature for more than ten years. However, the present work adds new properties by emphasizing semi-reactive modules and by proposing a concrete system allowing partial knowledge overlap between functionally specific modules. One interesting direction for future work is to evaluate how human notions of behavior correspond to the hierarchical system's need to categorize information into contexts on different levels. Above all, I hope that the present work approaches the question of general machine learning in a realistic way. Through the work with this dissertation I have been convinced that it is possible to create computer systems able to learn in a similar way that humans and other animals do. I believe that this line of research can support develop-

ment of useful robots and other machines, but also contribute to a better understanding of human cognition.

Bibliography

Alissandrakis, A., Nehaniv, C. L., & Dautenhahn, K. (2002). Imitation With ALICE: Learning to Imitate Corresponding Actions Across Dissimilar Embodiments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 32, 482–496.

Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., & Saunders, J. (2005). An Approach for Programming Robots by Demonstration: Generalization Across Different Initial Configurations of Manipulated Objects. In *Proceedings of 2005 International Symposium on Computational Intelligence in Robotics and Automation* (pp. 61–66). Espoo, Finland.

Amari, S. (1977). Dynamics of Pattern Formation in Lateral-Inhibition Type Neural Fields. *Biological Cybernetics*, 27(2), 77–87.

Ardizzone, E., Chella, A., & Pirrone, R. (2000). Pose classification using support vector machines. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* (pp. 317–322 vol.6).

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5), 469–483.

Arkin, R. C. (1998). *Behaviour-Based Robotics*. MIT Press.

Barsalou, L. W. (2009). Simulation, situated conceptualization, and prediction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521), 1281–1289.

Barsalou, L. W., Simmons, K. W., Barbey, A. K., & Wilson, C. D. (2003). Grounding conceptual knowledge in modality-specific systems. *Trends in Cognitive Sciences*, 7(2), 84–91.

Bentivegna, D. C. (2004). *Learning from Observation using Primitives*. PhD thesis, Georgia Institute of Technology.

Berk, L. E. & Winsler, A. (1995). *Scaffolding Children's Learning: Vygotsky and Early Childhood Education*. National Association for the Education of You.

Berthouze, L. & Metta, G. (2005). Epigenetic robotics: modelling cognitive development in robotic systems. *Cognitive Systems Research*, 6(3), 189–192.

Billard, A., Epars, Y., Cheng, G., & Schaal, S. (2003). Discovering imitation strategies through categorization of multi-dimensional data. In *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, volume 3 (pp. 2398–2403 vol.3). Las Vegas, Nevada.

Billard, A. & Hayes, G. (1999). DRAMA, a Connectionist Architecture for Control and Learning in Autonomous Robots. *Adaptive Behavior*, 7(1), 35–63.

Billing, E. A. (2009). *Cognition Reversed - Robot Learning from Demonstration*. Licentiate thesis, Umeå University, Department of Computing Science, Umeå, Sweden.

Bischoff, R. & Guhl, T. (2009). Robotic Vision to 2020 and Beyond - The Strategic Research Agenda for Robotics in Europe.

Boemio, A., Fromm, S., Braun, A., & Poeppel, D. (2005). Hierarchical and asymmetric temporal sensitivity in human auditory cortices. *Nature neuroscience*, 8(3), 389–95.

Byrne, R. W. & Russon, A. E. (1998). Learning by Imitation: A Hierarchical Approach. *The Journal of Behavioral and Brain Sciences*, 16(3).

Calinon, S. (2009). *Robot Programming by Demonstration - A Probabilistic Approach*. EFPL Press.

Calinon, S. & Billard, A. (2005). Recognition and reproduction of gestures using a probabilistic framework combining PCA, ICA and HMM. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 105–112). Bonn, Germany: ACM.

Calinon, S., Guenter, F., & Billard, A. (2007). 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.

Chernova, S. & Veloso, M. (2007). Confidence-based policy learning from demonstration using Gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems - AAMAS '07* (pp. 1315–1322). New York, New York, USA: ACM Press.

Clark, A. & Chalmers, D. J. (1998). The Extended Mind. *Analysis*, 58, 10–23.

de Rengervé, A., D'halluin, F., Andry, P., Gaussier, P., & Billard, A. (2010). A study of two complementary encoding strategies based on learning by demonstration for autonomous navigation task. In *Proceedings of the Tenth International Conference on Epigenetic Robotics* Lund, Sweden.

Demiris, J. & Hayes, G. (1997). Do robots ape? In *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents* (pp. 28–31).

Demiris, J. & Hayes, G. R. (2002). Imitation as a dual-route process featuring predictive and learning components: A biologically plausible computational model. In K. Dautenhahn & C. L. Nehaniv (Eds.), *Imitation in animals and artifacts* (pp. 327–361). Cambridge, MA, USA: MIT Press.

Demiris, Y. (1999). *Movement Imitation Mechanisms in Robots and Humans*. PhD thesis, University of Edinburgh.

Demiris, Y. & Simmons, G. (2006). Perceiving the unusual: Temporal properties of hierarchical motor representations for action perception. *Neural Networks*, 19(3), 272–284.

Felleman, D. J. & Van Essen, D. C. (1991). Distributed Hierarchical Processing in the Primate Cerebral Cortex. *Cereb Cortex*, 1, 1–17.

Fod, A., Matarić, M., & Jenkins, O. C. (2002). Automated derivation of primitives for movement classification. *Autonomous Robots*, (pp. 39–54).

Fonooni, B., Hellström, T., & Janlert, L. E. (2012). Learning High-Level Behaviors from Demonstration through Semantic Networks. In *Proceedings of the 4th International Conference on Agents and Artificial Intelligence (ICAART) (to appear)* Vilamoura, Algarve, Portugal.

Friston, K. J. (2009). The free-energy principle: a rough guide to the brain? *Trends in Cognitive Sciences*, 13(7), 293–301.

Friston, K. J. & Stephan, K. E. (2007). Free-energy and the brain. *Synthese*, 159(3), 458, 417.

Fujie, S., Ejiri, Y., Nakajima, K., Matsusaka, Y., & Kobayashi, T. (2004). A conversation robot using head gesture recognition as para-linguistic information. In *13th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN 2004)*. (pp. 159–164).

Fuster, J. M. (2001). The Prefrontal Cortex - An Update. *Neuron*, 30(2), 319–333.

George, D. (2008). *How the Brain might work: A Hierarchical and Temporal Model for Learning and Recognition*. Phd thesis, Stanford University.

Harnad, S. (1990). The Symbol Grounding Problem. *Physica*, D(42), 335–346.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10), 2201–2220.

Haruno, M., Wolpert, D. M., & Kawato, M. (2003). Hierarchical MOSAIC for movement generation. In *International Congress Series 1250* (pp. 575– 590).: Elsevier Science B.V.

Hawkins, J. & Blakeslee, S. (2002). *On Intelligence*. Times Books.

Hilgetag, C. C., O'Neill, M. A., & Young, M. P. (2000). Hierarchical organization of macaque and cat cortical sensory systems explored with a novel network processor. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 355(1393), 71–89.

Hovland, G., Sikka, P., & McCarragher, B. (1996). Skill acquisition from human demonstration using a hidden Markov model. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3 (pp. 2706–2711).

Hutchins, E. (1995). *Cognition in the Wild*. Cambridge, Massachusetts: MIT Press.

K-Team (2007). Khepera robot. www.k-team.com.

Kaptelinin, V. & Nardi, B. A. (2006). *Acting with Technology: Activity Theory and Interaction Design*. The MIT Press.

Koenig, N. & Matarić, M. J. (2006). Behavior-Based Segmentation of Demonstrated Tasks. In *Proceedings of the International Conference on Development and Learning* Bloomington, USA.

Kohonen, T. K. (2003). Learning vector quantization. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 631–638). MIT Press.

Mandelbrot, B. B. (1983). *The Fractal Geometry of Nature*. W. H. Freeman.

Matarić, M. J. (2002). Sensory-motor primitives as a basis for imitation: linking perception to action and biology to robotics. In *Imitation in animals and artifacts* (pp. 391–422). MIT Press.

Murphy, R. R. (2000). *Introduction to AI Robotics*. MIT Press.

Myers, B. C. S. & Rabiner, L. R. (1981). A Comparative Study of Several Dynamic Time-Warping. *The Bell System Technical Journal*, 60(7), 1389–1409.

Nakaoka, S., Nakazawa, A., Yokoi, K., & Ikeuchi, K. (2003). Recognition and generation of leg primitive motions for dance imitation by a humanoid robot. In *Proceedings of 2nd International Symposium on Adaptive Motion of Animals and Machines* Kyoto, Japan.

Nehaniv, C. L. & Dautenhahn, K. (1999). Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. *Interdisciplinary Approaches to Robot Learning*, 24, 136–161.

Nehaniv, C. L. & Dautenhahn, K. (2001). Like Me? - Measures of Correspondence and Imitation. *Cybernetics and Systems*, 32, 11–51.

Nicolescu, M. (2003). *A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains*. PhD thesis, University of Southern California.

Otero, N., Saunders, J., Dautenhahn, K., & Nehaniv, C. L. (2008). Teaching robot companions: the role of scaffolding and event structuring. *Connection Science*, 20, 111–134.

Oztop, E., Kawato, M., & Arbib, M. (2006). Mirror neurons and imitation: a computationally guided review. *Neural networks : the official journal of the International Neural Network Society*, 19(3), 254–71.

Pal Robotics (2011). REEM Humanoid Service Robot. www.pal-robotics.com.

Park, H., Kim, E., Jang, S., Park, S., Park, M., Kim, H., Marques, J., Pérez de la Blanca, N., & Pina, P. (2005). HMM-Based Gesture Recognition for Robot Control. In J. S. Marques, N. Pérez de la Blanca, & P. Pina (Eds.), *LNCS Pattern Recognition and Image Analysis*, volume 3522 of *Lecture Notes in Computer Science* (pp. 695–716). Berlin, Heidelberg: Springer Verlag.

Peters II, R. A., Campbell, C. L., Bluethman, W. J., & Huber, E. (2003). Robonaut Task Learning through Teleoperation. In *Proceedings of the 2003 IEEE Intl. Conference on Intelligent Robots and Automation* (pp. 23–27). Taipei, Taiwan.

Pook, P. K. & Ballard, D. H. (1993). Recognizing Teleoperated Manipulations. In *Proceedings on 1993 IEEE International Conference on Robotics and Automation, 1993.* (pp. 578–585).

Riesenhuber, M. & Poggio, T. (1999). Hierarchical Models of Object Recognition in Cortex. *Nature Neuroscience*, 2(11), 1019–25.

Rizzolatti, G. & Craighero, L. (2004). The Mirror-Neuron System. *Annual Review of Neuroscience*, 27, 169–192.

Robosoft (2011). Kompai Robot, www.robosoft.com.

Rohrer, B. & Hulet, S. (2006a). A learning and control approach based on the human neuromotor system. In *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics.* (pp. 57–61).

Rohrer, B. & Hulet, S. (2006b). *BECCA - A Brain Emulating Cognition and Control Architecture*. Technical report, Cybernetic Systems Integration Department, Sandria National Laboratories, Alberquerque, NM, USA.

Sjölie, D. (2011). *Reality-based brain-computer interaction*. Licentiate thesis, Department of Computing Science, Umeå University, Umeå, Sweden.

Suchman, L. A. (1987). *Plans and Situated Actions*. Cambride University Press.

Tani, J., Ito, M., & Sugita, Y. (2004). Self-Organization of Distributedly Represented Multiple Behavior Schemata in a Mirror System : Reviews of Robot Experiments Using RNNPB. *Neural Networks*, 17, 1273–1289.

Taylor III, A. (2011). Is Google Motors the new GM? *CNN Money*.

Thrun, S. & Pratt, L. Y., Eds. (1998). *Learning to Learn*. Kluwer Academic Publishers.

Vygotsky, L. S. (1978). *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press, 14th edition.

Werner, B. T. (1999). Complexity in Natural Landform Patterns. *Science*, 284(5411), 102–104.

Winberg, S. & Balkenius, C. (2007). Generalization and Specialization in Reinforcement Learning. In L. Berthouze, C. G. Prince, M. Littman, H. Kozima, & C. Balkenius (Eds.), *Proceedings of the Seventh International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems.* Lund, Sweden.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media, 1 edition.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.

Wolpert, D. M. (2003). A unifying computational framework for motor control and social interaction. *Phil. Trans. R. Soc. Lond.*, B(358), 593–602.

Wolpert, D. M. & Ghahramani, Z. (2000). Computational principles of movement neuroscience. *Nature Neuroscience*, 3, 1212–1217.

Wolpert, D. M. & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8), 1317–1329.

Yamashita, Y. & Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS computational biology*, 4(11), e1000220.

Ziemke, T., Jirenhed, D. A., & Hesslow, G. (2005). Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68, 85–104.

Zollner, R., Rogalla, O., Dillmann, R., & Zollner, M. (2002). Understanding users intention: programming fine manipulation tasks by demonstration. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System*, volume 2 (pp. 1114–1119).

I

Paper I

Cognitive Perspectives on Robot Behavior*

Erik Billing

Dept. Computing Science, Umeå University, SE-901 87 Umeå, Sweden
billing@cs.umu.se

Abstract: A growing body of research within the field of intelligent robotics argues for a view of intelligence drastically different from classical artificial intelligence and cognitive science. The holistic and embodied ideas expressed by this research promote the view that intelligence is an emergent phenomenon. Similar perspectives, where numerous interactions within the system lead to emergent properties and cognitive abilities beyond that of the individual parts, can be found within many scientific fields. With the goal of understanding how behavior may be represented in robots, the present review tries to grasp what this notion of emergence really means and compare it with a selection of theories developed for analysis of human cognition, including the extended mind, distributed cognition and situated action. These theories reveal a view of intelligence where common notions of objects, goals, language and reasoning have to be rethought. A view where behavior, as well as the agent as such, is defined by the observer rather than given by their nature. Structures in the environment emerge by interaction rather than recognized. In such a view, the fundamental question is how emergent systems appear and develop, and how they may be controlled.

Keywords: Behavior based control, Cognitive artificial intelligence, Distributed cognition, Ontology, Reactive robotics, Sensory-motor coordination, Situated action.

COGNITIVE PERSPECTIVES ON ROBOT BEHAVIOR*

Erik A. Billing
Department of Computing Science, Umeå University, Umeå, Sweden
billing@cs.umu.se

Keywords: Behavior based control, Cognitive artificial intelligence, Distributed cognition, Ontology, Reactive robotics, Sensory-motor coordination, Situated action.

Abstract: A growing body of research within the field of intelligent robotics argues for a view of intelligence drastically different from classical artificial intelligence and cognitive science. The holistic and embodied ideas expressed by this research promote the view that intelligence is an emergent phenomenon. Similar perspectives, where numerous interactions within the system lead to emergent properties and cognitive abilities beyond that of the individual parts, can be found within many scientific fields. With the goal of understanding how behavior may be represented in robots, the present review tries to grasp what this notion of emergence really means and compare it with a selection of theories developed for analysis of human cognition, including the extended mind, distributed cognition and situated action. These theories reveal a view of intelligence where common notions of objects, goals, language and reasoning have to be rethought. A view where behavior, as well as the agent as such, is defined by the observer rather than given by their nature. Structures in the environment emerge by interaction rather than recognized. In such a view, the fundamental question is how emergent systems appear and develop, and how they may be controlled.

1 INTRODUCTION

During the last decades, intelligent robotics has drawn towards a pragmatic view where no single design philosophy is clearly dominating. On the one hand, low level interaction with the world is often implemented with a reactive design philosophy inspired by Rodney Brooks' work, (Brooks, 1986; Brooks, 1990; Brooks, 1991a; Brooks, 1991b). On the other hand, classical AI-elements such as cartographers and planners are common modules for the high level control. Simultaneously, increasing system size and complexity raises requirements on well structured and modular system designs. Colored by an object-oriented programming approach, the system behavior is implemented through composition of modules. This kind of systems is commonly referred to as hybrid architectures. (Gowdy, 2000; Murphy, 2000; Doherty et al., 2004)

In a wider perspective hybrid systems propose a view of intelligence where simple behavior, like walking or grasping objects are typically reactive, while more complex tasks, like choosing a path or selecting objects are products of reasoning upon internal representation. "The robot can think in terms of a closed world, while it acts in an open world", (Murphy, 2000).

This view is not totally distant from the one proposed by modern cognitive science. The information processing model is still dominant for describing high level cognition (Stillings et al., 1995), while more reactive models have become popular for describing lower levels of control, especially within cognitive neuroscience (Shea and Wulf, 1995; Kaiser and Dillmann, 1996).

Even though hybrid architectures are today clearly dominating the field of intelligent robotics, there are several alternatives. A fundamentally different standpoint is taken by researchers proposing an embodied and holistic approach, (Matarič, 1997; Pfeifer and Scheier, 1997; Pfeifer and Scheier, 2001; Nicolescu, 2003). As these theories enforce concepts of distributed and emergent behavior, the present work is an attempt to analyze these notions of emergence actually mean. Similar ideas can be found within a vari-

*Parts of this text also appear as a technical report: Billing, E. A. (2007). Representing Behavior - Distributed theories in a context of robotics, *UMINF 07.25*, Department of Computing Science, Umeå University, Sweden.

ety of fields, including the *extended mind* (Clark and Chalmers, 1998), *distributed cognition* (Hutchins, 1995) and *situated action* (Suchman, 1987). All these theories are, in a general sense, studies of behavior beyond that of the individual, in groups or in interaction with artifacts. As such, these theories provide new perspectives on what it is we are in fact trying to achieve when building intelligent robots, and how we should get there.

1.1 Differences Between the Reactive and Deliberative Views

The reactive view grew during the 1980 as a reaction against classical artificial intelligence and cognitive science, and was in many ways a step back towards behavioristic ideas, (Braitenberg, 1986; Brooks, 1986; Georgeff and Lansky, 1987; Maes, 1991). The early reactive trend argued strongly against representations, but due to the obvious limitations of such an attitude, later work within the reactive field incorporate representations, but of a different type than the ones typically found within deliberating systems.

Deliberative architectures implement a *domain ontology*, that is, a definition of what things that exist in the world, but without a precise description of their properties and interrelations, (Russell and Norvig, 1995). This corresponds to a reductionist perspective also found within cognitive science and classical artificial intelligence.

Reactive systems are instead defined by a *low-level specification* that corresponds to the inputs and outputs of the system, referred to as the *sensory-motor space* (Pfeifer and Scheier, 1997). I here refer to a quite large variety of approaches, including the *subsumption architecture* (Brooks, 1986; Brooks, 1991b), *behavior based systems* (Mataric̀, 1997; Arkin, 1998; Nicolescu, 2003) and *sensory-motor coordination* (Pfeifer and Scheier, 2001; II and Campbell, 2003; Bovet and Pfeifer, 2005). Without claiming that all these approaches are one and the same, I use the term *reactive* as a common notion for these approaches proposing an embodied and holistic view.

The low-level specification defines the sensory-motor space as an entity which is related to the external world via a physical sensor or actuator on the robot. For a simple robot with eight proximity sensors and two independently controlled wheels, the sensory-motor space is ten-dimensional where each dimension corresponds to one sensor or motor. Many sensors provide multi-dimensional data. For example, a camera with a resolution of 100x100 pixels would increase the sensory-motor space with 10 000 new dimensions, where each pixel in the camera image corresponds to one dimension in the sensory-motor space. Cameras and other complex sensors could also be viewed as providing a single, complex, dimension in the sensory-motor space, but the amount of pre-processing or interpretation of data is always very limited in a low-level specification implement.

Pfeifer and Scheier (Pfeifer and Scheier, 1997) point out that a system using a low-level specification has a much larger input space than deliberative systems specified by a domain ontology, which also allows much greater complexity. Another interesting difference lies in information content. On the one hand, each dimension in the input space of a deliberative system is fairly informative. It could be the horizontal position of the robot on a map or the height of an object in front of the robot. On the other hand, most dimensions in the sensory-motor space are essentially meaningless if not viewed in the context of other dimensions. A single pixel in an image says very little about the content of the scene when that pixel is viewed alone, but in the context of the other pixels, it may be very informative. One could easily argue that such a large and complex sensory-motor space is the result of an ill chosen representation. With no doubt it is much easier to create readable representations using a deliberative approach, where sensor data have been processed so that it much better reflects *our own* understanding of what is going on.

Even though this criticism is correct and important, the sensory-motor space should not be understood as an unprocessed version of objects and other aspects in the world, but the representation of something else. The original argument against representations found in early reactive research has the last decade been replaced with a more accepting attitude towards representations, but representations of behavior rather than representations of the world. According to Pfeifer and Scheier, many things can be solved in a much simpler and more robust way without the use of high-level percepts. In general, the sensory-motor space appears to be a more suitable frame for representations than the kind of world models found in classical AI. (Pfeifer and Scheier, 1997; Pfeifer and Scheier, 2001; Dawson, 2002)

Low-level specifications have the great advantage that each dimension in the sensory-motor space is directly mapped to the corresponding sensors or motors, while the inputs to a deliberative system, such as position and size of objects, are often very hard to acquire. By assuming the necessity of high-level percepts we impose our own *frame of reference* upon the agent. Our notions of objects and states in the world are for

sure handy when reflecting upon an agent's behavior, but may not be necessary, or even desirable, when performing the same acts.

The principle of the frame of reference may be illustrated through the parable with the ant, presented by Herbert A. Simon, (Simon, 1969). Imagine an ant making its way over the beach, and that the way it chose was traced. When observing all the twists and turns the ant made, one may be tempted to infer a fairly complex internal navigation process. However, the complexity of the path may not be the result of the complexity of the ant, but the result of interaction between a relatively simple control system, and a complex environment.

Long before Brooks presented his ideas on reactive robotics (Brooks, 1986; Brooks, 1990; Brooks, 1991b), it was shown that complex behavior could emerge from simple systems, for example through the Homeostat (Ashby, 1960) and Machina speculatrix (Walter, 1963). Furthermore, Braitenberg's Vehicles (Braitenberg, 1986) was one of the most important inspiration sources for Brooks's work.

This discussion constitutes a central part of the criticism against deliberative systems and the motivation for a reactive approach. However, since reactive systems do not define any ontology with meaningful inputs, many types of, typically sequential tasks, are very hard to represent in this manner. Even though several examples of reactive systems showing deliberative-like behaviors exist, for example *Toto* (Mataric̀, 1992) and the reactive categorization robot by Pfeifer and Scheier (Pfeifer and Scheier, 1997), both the systems and the task they solve are typically handcrafted, making them appear more as cute examples of clever design than solutions to a real problem.

The difference between reactive and deliberative systems has been described as the amount of computation performed at run-time, (Mataric̀, 1997). A reactive control system can be derived from a planner, by computing all possible plans off-line beforehand, and in this way create a universal plan (Schoppers, 1987).

This argument about on-line computation beautifully points out how similar the two approaches of reactive and deliberative control may be. Still, when proposing the reactive approach, Rodney Brooks pointed out a number of behavioral differences to classical deliberative systems: "robots should be simple in nature but flexible in behavior, capable of acting autonomously over long periods of time in uncertain, noisy, realistic, and changing worlds", (Brooks, 1986). So if a reactive controller is merely a precomputed plan, why these differences in behavior?

One critical issue is speed. Brooks often points out the importance of real-time response and that the cheap design of reactive systems allows much faster connections between sensors and actuators than the deliberative planners, (Brooks, 1990). Even though this was an important point in the early nineties, the last years' increase in computational power allows continuous re-planning within a reactive time frame, (Dawson, 2002).

Another reason may be that reactive controllers are typically not derived from planners. Rather, reactive controllers are handcrafted solutions specialized for a certain type of robot. Achieving a specific complex behavior in a reactive manner can be a challenge, which may be one important reason for the limited success of reactive systems in solving more complex, sequential tasks (Nicolescu, 2003). Taking Matarić's point about run-time computation into account, the reactive approach still does not propose a clear way to achieve a desired controller; it only shows that the deliberative part can be removed when intelligence has been compiled into reactive decision rules.

Hybrid systems do obstacle avoidance using reactive controllers not because re-planning is computationally heavy, but because re-planning is difficult to implement. Even though one could imagine a planner generating exactly the same behavior as one of Braitenberg's vehicles avoiding obstacles, the structure of such a planner would probably be much more complicated than the corresponding controller formulated in reactive terms. This may in fact, at least from an engineer's point of view, be the most suitable distinction between the reactive and deliberative perspectives. It appears that behaviors like obstacle avoidance and corridor following is easily formulated in reactive terms, while selecting a suitable path from a known map is better formulated using a planner. Other things, actually most things, are too hard to manually design using any of these two approaches.

1.2 Emergence of Behavior

As mentioned in the previous section, supporters of the reactive approach freely admit that the implementation of high-level deliberative-like skills in reactive systems is very difficult, (Pfeifer and Scheier, 2001; Nicolescu, 2003). The route to success is often said to be emergence, (Maes, 1990; Mataric̀, 1997; Pfeifer and Scheier, 1997). But what exactly does this mean?

The term *emergent* is commonly described as *something that is more than the sum of its parts*, but apart from that it is in fact hard to arrive at a definition suitable for all uses of the term, (Corning, 2002). Within the field of intelligent robotics, emergence is used to point out that a robot's behavior is not explicitly defined in the controller, but something that ap-

pears in the interaction between the robot and its environment. Pfeifer and Scheier (Pfeifer and Scheier, 2001) proposes a number of design principles for autonomous robots. The critical points are shortly summarized below.

1. Behavior should emerge out of a large number of parallel, loosely coupled processes.
2. Intelligence is to be conceived as sensory-motor coordination, i.e., the sensory-motor space serves as a structure for all representations, including the categorization and memory.
3. The system should employ a *cheap* design and exploit the physics of its ecological niche.
4. The system must be redundant.
5. The system should employ the principles of self-organization.

Not surprisingly, those principles are well aligned with those found in literature discussing emergence, (Corning, 2002; Flake, 1998). Consequently, modern reactive architectures should constitute a good approach for design of systems showing emergent properties, but this is yet far from a unified theory on which robotics architectures could be built, (Heylighen et al., 2004). Before a theory of emergent behavior could actually be used, much more has to be understood about the theoretical properties of emergence, but such an analysis is seldom found in robotic literature.

1.3 Criticism Against the Hybrid View

After looking a bit closer at the principles of the reactive and deliberative approaches, the philosophy behind hybrid approaches seems to be much closer to the latter. Hybrid systems clearly align to a reductionist view, enforcing the importance of system modularity and hierarchical structures. The promoters of hybrid systems motivate this design effort in completely different ways than the supporters of reactive systems argue for a holistic perspective. Obviously, from an engineering perspective, it is very important to be able to build larger systems in some kind of modules, so that each part can be tested and refined separately. While this strongly contradicts the holistic perspective, reactive supporters have no solution to, and are generally not interested in, these issues.

So what exactly are the problems with combining the reactive in the small, and deliberative in the large? Since the hybrid approach is so wide and generally friendly towards anything that works, it is hard to actually say something about these systems which truly applies to all of them. Still, some common criticism has been raised against the hierarchical approach, especially from the field of embodied cognitive science. The core issues are summarized bellow.

• Even though hybrid systems adopt an embodied view for interaction with the world, they still define a domain ontology and are consequently bound by the limitations of this approach. One critical point of the reactive approach is that no concepts or symbols should be pre-defined. This point is lost when hybrid systems use reactivity as an interface to the world rather than the source of intelligence. (Pfeifer and Scheier, 2001)

• The kind of information produced by the reactive layer in a hierarchical system is often fundamentally different from that required by the deliberative subsystems, making it hard to design suitable interaction between the two layers. For this reason, the sensing part in the deliberative layer is often designed in a non-reactive way, reintroducing the problem of how objects, and concepts in general, should be recognized in complex and noisy data. (Pfeifer and Scheier, 2001; Dawson, 2002)

• While a critical aspect of modularity is to be able to test and control the function of each module before inserting it in the complete system, one important goal of reactive approaches is to achieve emergent properties which by definition do not appear in the modules alone. Even though hybrid systems successfully employ simpler reactive behavior, they do not leave room for emergent properties. (Pfeifer and Scheier, 2001; Brugali and Salvaneschi, 2006)

2 AN EXTENDED PERSPECTIVE

The fundamental differences between modern approaches within intelligent robotics have now been outlined. The rest of this paper present a number of different views on cognition, and apply them in a context of intelligent robotics. These views will make deliberative and reactive perspectives appear less like the two extremes, and more like one dimension within the multi-dimensional study of cognition and behavior.

Cognitive science has received significant amounts of criticism for its undivided focus on the individual, where a proper analysis of the social aspects of interaction is missing, (Greeno, 1993; Heylighen et al., 2004; Hutchins, 1995; Suchman, 1987). Similar criticism has been raised towards classical AI and deliberative robotics, but to some extent it also applies to reactive systems. Both deliberative and reactive approaches share a view of the single agent as one conceived unit, interacting

with the outside world through input and output interfaces. Even though this might seam like a safe assumption for many systems like humans, animals, computers and robots, I will in this chapter present a couple of theories where the object of analysis is changed to incorporate fundamentally different types of *cognitive systems*. As will be illustrated, many aspects of these systems are strikingly similar to the architectures proposed within robotics.

2.1 The Extended Mind

Clark and Chalmers (Clark and Chalmers, 1998) describe two people, Inga and Otto, who are both going to visit the Museum of Modern Art which lies in the 53rd street. Inga heard from a friend that there is an exhibition at the Museum of Modern Art and she decides to go there. She looks up the address and remembers it. She forms the belief that the Museum of Modern Art is on 53rd street. But now consider Otto who has Alzheimer's disease and instead of remembering the address writes it down in his notebook. Clark and Chalmers argue that Inga and Otto in principle have the same belief, even though parts of Otto's belief in a very strong sense are outside his body. This is an example of *the extended mind*.

Interestingly, Otto's behavior may easily be described in reactive terms. Otto changes the environment, his notebook, in a manner which later will lead him to the correct address. In contrast, Inga's behavior is a classical example which can't be described within a purely reactive architecture. Still, Clark and Chalmers point out that Otto's and Inga's cognitive processes are essentially the same. I would argue that the reason why we usually view Inga's and Otto's cognitive processes as quite different is our usual concept of an individual. If we chose to view the person as an entity enclosed by the skin, the use of a notebook as memory is very different from the use of nerve structures for the same purpose. But, if we instead follow Clark and Chalmers' argument and widen our notion of a person to include the notebook, the two types of memory appear very similar.

This point could also be illustrated by a computer. What exactly is a computer? Most people would probably say that it's the screen, the key-board and mouse, and of course the box on the floor which you connect all the cables to. If one uses a Memory Stick to store things on, that is not a part of the computer, but a different object. Still, technically speaking, the Memory Stick, when connected, is very similar to the hard drive within the computer. The Memory Stick might, just as Otto's notebook, have lower storage capacity, a bit slower access speed, and not always be available. Still, it fills the same function as the internal memory. Admittedly, our common notion of a person, where the notebook is not included, is very convenient, but we should be aware that it is merely a convention.

This discussion opens up the notion of an agent. We choose to see one agent as separated from its environment not because it *is* different from the environment, but because it, from our perspective, is convenient to view it like that. This does not imply that the notion of agents and objects is totally arbitrary, but neither is it totally predefined.

I mentioned earlier Pfeifer and Scheier's notion of *frame of reference*, pointing out that an agent's behavior is always seen from an observer's perspective. The view presented here takes one step further by saying that even the notion of agent is dependent of the observer. This distinction is important since Pfeifer and Scheier strongly argue towards representations within a sensory-motor space. If the agent is an entity created by the observer, so are sensors and motors, and with these, the sensory-motor space.

Following this discussion, the notion of an agent should be able to divide into smaller, sub-agents, with different sensors and motors. The functions of these sub-agents may differ drastically from the function of the combined agent, just like Otto and his notebook can do more things together, than neither of them can do separately. Consequently, the question of how behavior is represented is transformed into how Otto figures out that he should use a notebook? Or more generally: How does *purposeful* emergent behavior among agents appear?

2.2 A Universe of Possibilities

One of the inspiration sources to Clark and Chalmers' work came from *distributed cognition*, (Hutchins, 1995). Hutchins points out the importance of viewing cognitive processes as something that goes on both in the environment and within the individual, but in contrast to Clark and Chalmers, Hutchins' focus lies on group level dynamics. In this context, the agent, or the *cognitive system*, is expanded not only to incorporate one person and his tools, but many people and artifacts in cooperation.

Hutchins takes a few steps further than Clark and Chalmers by not only proposing an extended view, but also using it to analyze systems. Distributed cognition has been applied to many systems, including ship navigation (Hutchins, 1995), human-computer interaction (Hollan et al., 2000), various aspects of airplane control (Hutchins and Klausen, 1996; Hutchins and Holder, 2000; Hutchins et al., 2002) and more re-

cently clinical systems (Galliers et al., 2006). While distributed cognition as used in these examples have no apparent application to robotics, the result of this research might still shed some light on what we want to achieve when designing for purposeful emergent behavior.

From a deliberative perspective, a large and difficult problem is to recognize objects and their properties in complex and noisy data. From a reactive view, the same problem is instead described as how to arrive with suitable emergent properties. And finally, form the perspective of distributed cognition, this problem is strongly related to the formation of interpretations within a group. Hutchins investigates the properties of interpretation formation on group level using constraint satisfaction networks, (Hutchins, 1995). The weights of the networks were arranged so that each network could arrive at only two stable states, or interpretations. One interpretation corresponds to the activation pattern 111000 while the other interpretation corresponds to 000111, see Figure 1.

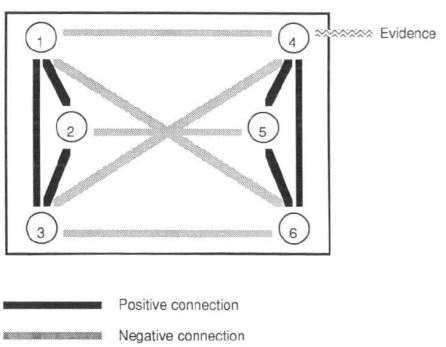

Figure 1: Constraint satisfaction network, (Hutchins, 1995, p. 244). Black and gray lines represent positive and negative connections, respectively. Strong activation in left side nodes will consequently inhibit activation in nodes to the right, and the other way, driving the network towards one of two stable states. Republished with permission.

The initial activation of the nodes is here viewed as *confirmation bias*. When the network is executed alone, it will always arrive with the interpretation closest to its initial state. However, when the networks are connected so that the activity of some nodes propagates to other nodes of a different network, their decision properties change due to interaction between the networks. As Hutchins puts it, this illustrates how interpretation of information changes due to the organization of the group. In a robotics context, this might be applied as having several reactive controllers creating a virtual "environment" for each other. More specifically, controllers do not only take input from sensors and send commands to actuators, but might also sense and act upon other controllers, drastically increasing the complexity of the system as a whole. The organization of such a system should be similar to the organizational properties of a group.

Hutchins shows that having this kind of organization, where multiple connected agents try to make their individual interpretations, produces a system that efficiently explores interpretation space. Furthermore, even after reaching a common interpretation, such a system is much more likely to re-evaluate the interpretation in case of new evidence. However, in case of too much interaction within the group, interpretation space is not explored properly. In contrast, too little interaction will result in that no single interpretation is achieved. Hutchins calls this *the fundamental tradeoff in cognitive ecology*.

This discussion is not only interesting as analysis of group behavior, but also as a way to understand interpretations within the individual. Here, the process of transforming sensor data into symbols with meaning is replaced with a continuous constraint satisfaction between sensors, actuators and internal states. In this view, we do not perceive what is in the environment; instead we are striving towards the closest stable state, pushed in one direction or another by changes in the environment.

There are mainly two advantages with this model of cognition, compared to the classical view of information processing. First, the interpretation always arrives from the current state of the system. Consequently, each interpretation is based on much more information than when sensor data is seen as a series of discrete readings which should be understood more or less separately. Secondly, when the notion of symbols is replaced with that of attractors, the number and meaning of these entities can be changed dynamically. This opens up the possibility for a solution to one key problem of deliberative processing; the creation of new symbols.

This view of cognition as the propagation of representational states across representational media might provide a new and powerful tool for understanding interpretation and decision making also in robots. However, even though Hutchins provides an extensive analysis of several existing distributed systems, a general understanding of how such a system appears and develops is still missing, (Heylighen et al., 2004).

2.3 Situated Action

When studying interactions between people, the analysis of language becomes one critical sub field. Clas-

sical cognitive science literature describes language as "a system that uses some physical signal (a sound, a gesture, a mark on paper) to express meaning", (Stillings et al., 1995). In other words, language is viewed as a communication channel where some meaning, i.e. an internal representation, is encoded into a physical signal using some grammar and then decoded by the listener into a similar internal representation. However intuitive this view may appear, it is not the only one. A fundamentally different perspective was presented in late eighties by Lucy A. Suchman under the name *situated action*, (Suchman, 1987).

Suchman's claim is that the traditional view of language includes several fundamental problems. One of the most important issues is the discussion around *shared knowledge*. If the cognitive view of language is correct, a speaker must not only encode the representation into words, but also take into account what the listener already knows. As Suchman puts it, this is equivalent with having a body of shared knowledge that we assume all individuals within our society to have. When speaking, only the specifics of the internal representation are transformed into words, leaving out everything covered by the shared knowledge.

To exemplify the problem, Suchman uses the result from an exercise assigned by Harold Garfinkel, (Suchman, 1987). Students were asked to write down a description of a simple conversation. On the left hand side of the paper the students should write what was said, and on the right hand side what they understood from the conversation. While the first part of the assignment was of course easy, the second part seemed to grow without limit. Many students asked how much they were supposed to write, and when Garfinkel imposed accuracy, clarity and distinctness the students finally gave up with the complaint that the task was impossible. The point here is not that the body of shared knowledge is too large to write down on a paper, but that the task resulted in a continually growing horizon of understandings to be accounted for.

The assignment, it turned out, was not to describe some existing content, but to generate it. As such, it is an endless task. The students' failure suggests not that they gave up too soon, but that what they were assigned to do was not what the participants in the conversation themselves did in order to achieve shared understanding. (Suchman, 1987).

Even though there might be several other ways to explain the result from Garfinkel's assignment, Suchman's point is striking. If knowledge is not preexisting to language as much as it is generated by it, it puts our understanding of internal representations in a fundamentally different light. The meaning of a spoken phrase does not appear to exist in any stronger sense than obstacles exists for one of Braitenberg's vehicles.

Situated action is not at all limited to analysis of language. In fact, situated action tries to unify all kinds of behavior where language is seen as a very specialized sub field. In such a view, spoken words has the same relation to semantics as an agent's actions has to intentions. However, it should be remembered that Suchman's work is presented as a theory of human-computer interaction rather than a theory of behavior or intelligence.

If language and other complex behavior do not begin with an internal representation or intent, how is it produced? The view proposed by Suchman begins in the context of the agent: "every course of action depends in essential ways upon its material and social circumstances," (Suchman, 1987). The circumstances or *situation of actions* can, at least in a context of intelligibility, be defined as "the full range of resources that the actor has available to convey the significance of his or her own actions, and to interpret the actions of others," (Suchman, 1987). This could be interpreted as if the world is understood in terms of actions. The fact that we know how to walk makes us really good at recognizing such behavior. In the domain of human-computer interaction, the same argument implies that our understanding of a computer is represented in terms of what we can do with it, not as a structural model of the computer as such. As a consequence, a selection of the best path towards a desired goal is not dependent on a representation of roads, but on the availability of the actions for turning left or right.

Furthermore, the *goal* of situated action is not represented in any other way than as preferences to some actions given a specific situation. Our common understanding of plans and goals is in this context nothing but a way to reflect on past events. As Suchman points out, a declaration of intent generally says very little about the precise actions to follow, it is the obscurity of plans that makes them so useful for everyday communication (Suchman, 1987).

This discussion puts not only notions of intentions and plans in a secondary position, but conscious thoughts in general appear to be less the driving force behind action than an artifact of our reasoning about action. Seen in the robotics context, deliberative processes should in a very strict sense be emerging from lower levels of interaction, not something predefined that supervises the lower levels.

One interesting implication of these theories is that observed sensor data bears a very loose connection to its semantic content. The interpretation is *cre-*

ated by the observer in interaction with the data rather than *extracted* from the observed data. The creation of an interpretation is in this view more about generating information, than processing it.

3 DEVELOPMENT AND DESIGN

The theories presented above depict a perspective of intelligence where cognitive ability emerges out of interactions between multiple parts of an agent. The agent is very loosely defined as a cognitive system, i.e., a large number of physically and/or socially distributed entities which interact and in this way achieve something more than any of them could do alone. More explicitly, intelligent behavior of a cognitive system is produced from entities which are totally unaware of the dynamics of that system as a whole.

In such a view, elements in the world are never explicitly represented, but appear in terms of possibilities for situated actions. Information does not flow from inputs to outputs, but back and forth through numerous representational states, coordinating sensors and actuators rather than controlling them. The meaning of actions, symbols or data in general is achieved through interaction among elements, not given by a grammar. Conscious processes are not the fundamental foundation for intelligent behavior, but its fundamental phenomenon. The question still remaining is: *How does one create a system based on the principles presented above?*

3.1 Evolution of Self-organization

Emergent properties which in the previous discussion so gracefully are said to explain intelligence do, from an engineering perspective, often cause more problems than they solve. What is seldom mentioned is that guidelines like those presented by Pfeifer and Scheier, (Pfeifer and Scheier, 2001) only address half the question of emergent behavior. We are normally not interested in just any emergent behavior, but specifically in those emergent effects which fulfill the task for which the robot is designed. This may prove to be much more difficult to achieve than just emergence in general.

There are a number of theories approaching this problem. The most frequently mentioned within robotics is self organization. The principle of self organization means that the system spontaneously develops functional structure through numerous interactions between its parts. The basic mechanism behind this structure is mutual benefit, symbiosis. Parts in the system will continue to rearrange until both find a relative state which is satisfactory. A frequently used interaction pathway will grow stronger while rarely used pathways will weaken or disappear. The mechanisms controlling what is satisfactory will as a consequence have direct influence on the emergent properties of the system as a whole. (Heylighen and Gershenson, 2003)

While the principle of self-organization provides a clearer understanding of the mechanisms controlling emergence, it still does not explain how useful behavior emerges. The fact that favorable interactions are reinforced on the micro level will certainly not lead directly to favorable behavior on the macro level.

For natural systems the obvious answer is evolution. The fundamental mechanism of natural selection will, given enough time, lead to a solution. However, this gives us little hope for training robots. The problem space for a robot acting in the real world grows extremely fast. Allowing a robot to try out a population of randomly chosen behaviors will, even for very simple problems, most likely never lead to a solution. Consequently, the evolutionary process won't work since nothing can be reinforced.

Interestingly, a robot acting in the real world is, by definition, in the same situation as many biological systems, which obviously have evolved. The discussion above makes an evolutionary explanation to the problem of grabbing and moving objects appear highly unlikely. Even when including the billions of years preceding the human era, the chance of combining all the biological structures required for object manipulation to work appears very small. And yet evolution has given us a wonderful tool in the form of the hand, and the neural structures underlying its control.

The explanation for our highly flexible and dexterous ability to manipulate objects is of course that it did not evolve from nothing. As such, the human hand is not an optimal solution, nor is it anything close to optimal. Instead is it a result of what came before it. Small incremental changes of the mammal front legs, which at each stage were reinforced through natural selection, have eventually led to the human arm and hand. (Wolfram, 2002)

Why this divergent discussion about evolution? The manipulator of a robot has to be designed. We are simply interested in teaching the robot to use it, given a fairly short period of time. The point of this sidestep into evolution is that the physical shape of the human hand did not evolve alone, but together with the neural system controlling it. The human child is born with a large amount of basic reflexes, which are all fairly simple. In robotic terms, we would probably call them purely reactive behaviors. (Thelen and

Bates, 2003; Grupen, 2006)

Some of these innate behaviors are certainly critical for the survival of the infant, such as the grip reflex or the sucking reflex. But many other reflexes do not have an obvious purpose, such as the *asymmetric tonic neck reflex*, *landau reflex* or the *galant reflex*, (Grupen, 2006). Instead, reflexes like these seem to play a key role for learning. As mentioned above, the child is born with a set of reflexes. These basic reflexes are, during the first four years, gradually replaced by new, more complicated behaviors. The child seams to learn through an evolutionary process of behavioral development. New behaviors appear as a modification or combination of more basic behaviors, while other behaviors disappear. In theoretical terms, this incremental development allows the problem space to remain small even as the problems grow more complicated. The full space of possible solutions to the problem is never searched, but only the parts covered by previous knowledge. Sometimes, the small changes to the underlying control structure result in drastic changes in behavior, which we see in the child as the establishment of new behaviors. This kind of incremental development process should favor robust behavior rather than optimal. A behavior which succeeds under many circumstances simply has much greater chance of survival than a perfect solution only succeeding under very special circumstances.

4 CONCLUSIONS

Through out this review there has been a theme of emergent behavior. Notions of objects in the world, goals and concepts in general are said to emerge out of simpler parts. The literature reviewed here frequently points out several aspects critical for a system to show emergent properties. However, it has been much harder to find clear theories of how to control these emergent properties. In fact, one important property of emergence seems to be that it can not be controlled in a supervising way. Without a proper theory of how to arrive at useful emergent properties, the argument that behavior should emerge is very much like saying that we do not know. It is generally admitted that distributed and emergent control systems for robots are very hard both to obtain and control. As such, these approaches do not seem less problematic than their counterparts within classical AI.

Nevertheless, the concepts presented in this review open a new frame for representation of behavior. A troubling and yet thrilling aspect of these theories is that they span over an enormous theoretical horizon. Some fundamental problems are solved, many new are introduced, but just viewing the problem from a different perspective might get us closer to a general understanding: An understanding of what intelligence is and how it can be created.

ACKNOWLEDGEMENTS

I thank Lars-Erik Janlert and Thomas Hellström at the Department of Computing Science, Umeå University for valuable input to this work.

REFERENCES

Arkin, R. C. (1998). *Behaviour-Based Robotics*. MIT Press.

Ashby, R. (1960). *Design for a brain; the origin of adaptive behavior*. Wiley, New York.

Bovet, S. and Pfeifer, R. (2005). Emergence of coherent behaviors from homogenous sensorimotor coupling. In *12th International Conference on Advanced Robotics*, pages 324 – 330.

Braitenberg, V. (1986). *Vehicles - Experiments in Synthetic Psychology*. The MIT Press.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation RA-2*, volume 1, pages 14–23.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15.

Brooks, R. A. (1991a). Intelligence without reason. *Proceedings, 1991 Int. Joint Conf. on Artificial Intelligence*, pages 569–595.

Brooks, R. A. (1991b). New approaches to robotics. *Science*, 253(13):1227–1232.

Brugali, D. and Salvaneschi, P. (2006). Stable aspects in robot software development. *International Journal of Advanced Robotic Systems*, 3(1).

Clark, A. and Chalmers, D. J. (1998). The extended mind. *Analysis*, 58:10–23.

Corning, P. A. (2002). A venerable concept in search of a theory. *Complexity*, 7(6):18–30.

Dawson, M. R. W. (2002). From embodied cognitive science to synthetic psychology. In *Proceedings of the First IEEE International Conference on Cognitive Informatics (ICCI'02)*.

Doherty, P., Haslum, P., Heintz, F., Merz, T., and Persson, T. (2004). A distributed architecture for autonomous unmanned aerial vehicle experimentation. In *Proceedings of the 7th International Symposium on Distributed Autonomous Systems*.

Flake, G. W. (1998). *The Computational Beauty of Nature*. MIT Press, Cambridge, Massachusetts.

Galliers, J., Wilson, S., and Fone, J. (2006). A method for determining information flow breakdown in clinical systems. *Special issue of the International Journal of Medical Informatics*.

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *AAAI*, pages 677–682.

Gowdy, J. (2000). *Emergent Architecture: A Case Study for Outdoor Mobile Robots*. PhD thesis, The Robotics Institute. Carnegie, Carnegie Mellon University.

Greeno, J. G. (1993). Special issue on situated action. In *Cognitive Science*, volume 17, pages 1–147. Ablex Publishing Corporation, Norwood, New Jersey.

Grupen, R. (2006). The developmental organization of robot behavior. Oral presentation during the 6th International UJI Robotics School.

Heylighen, F. and Gershenson, C. (2003). The meaning of self-organization in computing. In *IEEE Intelligent Systems*.

Heylighen, F., Heath, M., and Overwalle, F. V. (2004). The emergence of distributed cognition: a conceptual framework. In *Collective Intentionality IV*, Siena, Italy.

Hollan, J., Hutchins, E., and Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Trans. Comput.-Hum. Interact.*, 7(2):174–196.

Hutchins, E. (1995). *Cognition in the Wild*. MIT Press, Cambridge, Massachusetts.

Hutchins, E., E, B., Holder, R., and P, A. (2002). Culture and flight deck operations. Prepared for the Boeing Company.

Hutchins, E. and Holder, B. (2000). Conceptual models for understanding an encounter with a mountain wave. In *HCI-Aero 2000*, Toulouse, France.

Hutchins, E. and Klausen, T. (1996). Distributed cognition in an airline cockpit. *Cognition and communication at work. Y. E. a. D. Middleton. New York, Cambridge University Press*, pages 15–34.

II, R. A. P. and Campbell, C. L. (2003). Robonaut task learning through teleoperation. In *Proceedings of the 2003 IEEE, International Conference on Robotics and Automation*, pages 23–27, Taipei, Taiwan.

Kaiser, M. and Dillmann, R. (1996). Building elementary robot skills from human demonstration. *International Symposium on Intelligent Robotics Systems*, 3:2700–2705.

Maes, P. (1990). Situated agents can have goals. *Robotics and Autonomous Systems*, 6:49–70.

Maes, P., editor (1991). *Designing Autonomous Agents*. MIT Press, Elsevier.

Mataric̀, M. J. (1992). Integration of representation into Goal-Driven Behavior-Based robots. In *IEEE Transactions on Robotics and Automation*, volume 8, pages 304–312.

Mataric̀, M. J. (1997). Behavior-Based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2–3):323–336.

Murphy, R. R. (2000). *Introduction to AI Robotics*. MIT Press, Cambridge, Massachusetts.

Nicolescu, M. (2003). *A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains*. PhD thesis, University of Southern California.

Pfeifer, R. and Scheier, C. (1997). Sensory-motor coordination: the metaphor and beyond. *Robotics and Autonomous Systems*, 20(2):157–178.

Pfeifer, R. and Scheier, C. (2001). *Understanding Intelligence*. MIT Press. Cambrage, Massachusetts.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, NJ.

Schoppers, M. (1987). Universal plans for reactive robots in unpredictable demains. In *IJCAI-87:*, pages 1039–1046.

Shea, C. H. and Wulf, G. (1995). Schema theory - a critical appraisal and reevaluation. *Journal of Motor Behavior*.

Simon, H. A. (1969). *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts.

Stillings, N. A., Weisler, S. E., Chase, C. H., Feinstein, M. H., Garfield, J. L., and Rissland, E. L. (1995). *Cognitive Science*. MIT Press, Cambridge, Massachusetts.

Suchman, L. A. (1987). *Plans and Situated Actions*. PhD thesis, Intelligent Systems Laboratory, Xerox Palo Alto Research Center, USA.

Thelen, E. and Bates, E. (2003). Connectionism and dynamic systems: Are they really different? *Developmental Science*, 6(4):378–391.

Walter, W. (1963). *The Living Brain*. Norton & Co., New York.

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media, 1 edition.

II

Paper II

Behavior Recognition for Segmentation of Demonstrated Tasks*

Erik Billing and Thomas Hellström

Dept. Computing Science, Umeå University, SE-901 87 Umeå, Sweden
billing@cs.umu.se, thomash@cs.umu.se
www.cs.umu.se/research/robotics

Abstract: One common approach to the robot learning technique Learning From Demonstration, is to use a set of pre-programmed skills as building blocks for more complex tasks. One important part of this approach is recognition of these skills in a demonstration comprising a stream of sensor and actuator data. In this paper, three novel techniques for behavior recognition are presented and compared. The first technique is function-oriented and compares actions for similar inputs. The second technique is based on auto-associative neural networks and compares reconstruction errors in sensory-motor space. The third technique is based on S-Learning and compares sequences of patterns in sensory-motor space. All three techniques compute an activity level which can be seen as an alternative to a pure classification approach. Performed tests show how the former approach allows a more informative interpretation of a demonstration, by not determining ”correct” behaviors but rather a number of alternative interpretations.

Keywords: Learning from demonstration, Segmentation, Generalization, Sequence Learning, Auto-associative neural networks, S-Learning.

Behavior Recognition for Segmentation of Demonstrated Tasks

Erik A. Billing
Department of
Computing Science
Umeå University
Umeå, Sweden
phone: +4690-7869915
e-mail: billing@cs.umu.se

Thomas Hellström
Department of
Computing Science
Umeå University
Umeå, Sweden
phone: +4690-7867759
e-mail: thomash@cs.umu.se

***Abstract*—One common approach to the robot learning technique Learning From Demonstration, is to use a set of pre-programmed skills as building blocks for more complex tasks. One important part of this approach is recognition of these skills in a demonstration comprising a stream of sensor and actuator data. In this paper, three novel techniques for behavior recognition are presented and compared. The first technique is function-oriented and compares actions for similar inputs. The second technique is based on auto-associative neural networks and compares reconstruction errors in sensory-motor space. The third technique is based on S-Learning and compares sequences of patterns in sensory-motor space. All three techniques compute an activity level which can be seen as an alternative to a pure classification approach. Performed tests show how the former approach allows a more informative interpretation of a demonstration, by not determining "correct" behaviors but rather a number of alternative interpretations.**

***Index Terms*—Learning from demonstration, Segmentation, Generalization, Sequence Learning, Auto-associative neural networks, S-Learning.**

I. Introduction

A lot of research in Learning from Demonstration (LFD) deals with the problem of generating behaviors from data recorded during manual demonstrations. Behaviors are in this case direct mappings from sensor states to action states, where actions typically are low-level motor commands [12], [8]. This both challenging and interesting research has natural limitations in many real-world applications. A key problem is generalization, i.e. the robot´s ability to repeat a demonstrated behavior under conditions not identical to those present during the demonstration. One common way to overcome this is to transform the demonstration into a set of pre-programmed higher-level actions, called sub-tasks [20], motion primitives [23], motor primitives [1] or motor skills [18].

Most works in LFD deal with tasks such as robot arm motion, pole balancing, and robot gait, e.g., [2], [13], [4], and the higher-level actions are often short sequences of low-level motor commands. One of the few examples of LFD with complex high-level behaviors is the work presented by Nicolescu and Mataric´ [16], [15], [10]. The work presented in this paper uses similar types of higher-level skills. For this reason, we will adopt the terminology developed by Nicolescu [16], where *skills* can be understood as a relatively simple mapping from sensors to actuators, with an aptitude or ability to achieve or maintain a goal. Several skills can be combined to perform a task. A *task, which* has higher complexity than skills, may involve serveral goals and is represented as a sequence of skills.

The ability to represent a demonstration as a sequence of skills does not only serve as support for generalization, but is also a powerfull way to make the demonstrated data understandable to a human user. A labeled sequence of skills, for example, *following the wall to my right, passing through a door, going straight over the floor avoiding any obstacle*, is significantly easier to interprete than the raw sensor and motor data. A demonstration represented in this way should give the user a better understanding of what the robot did observe, and the opportunity to evaluate the robot's interpretation of the demonstrated behavior.

As such, the generalization part of LFD is in this context understood as describing a task-level demonstration in terms of the previously learned, or pre-programmed skills. This involves determining positions and characteristics of *segmentation points* where the skills change, and identifying the skills themselves. Identifying suitable parameters for parametrized skills may also be part of this process, e.g., [15]. Consequently, the result of performing LFD will be a task representation, i.e., a sequence of identified skills, together with a characterization of the segmentation points. This paper addresses the specific problem of determining positions for segmentation points and the identification of previously learned skills for each segment. However, the problem of determining the characteristics of each segmentation point is not addressed, and will be subject to future work.

Section 2 of our paper describes LFD and introduces a basic notation. Section 3 presents three novel techniques for behavior recognition. In Section 4, the test cases are described and the test results are reported in Section 5. Related work is reviewed in Section 6, and finally conclusions and ideas for future research are given.

II. Learning From Demonstration

The basic principle of the learning technique Learning From Demonstration (LFD) is that a robot should learn to repeat

a behavior after being teleoperated through one or several demonstrations performed by a human. Some key concepts will be introduced in this section.

The robot represents its view of itself and of the environment at time t with a sensor vector x_t comprising observed sensor variables. Some variables represent exteroceptive physical sensors, such as infra-red distance sensors, ambient light sensors, cameras, bump sensors, accelerometers, gyros, and GPS. Other variables represent interoceptive sensors, such as joint angles, wheel encoders, actual motor speed, and battery state. Dynamics and time dependencies can be handled by adding lagged variables, derivatives, or sub sequences to the sensor vector. The robot affects the world and itself through its effectors represented by the action vector y_t. Typical actions are motor speed and steering angle controlling propulsion, or joint angles controlling the motion of a robot arm.

A behavior β can be represented by a function from a subset of sensor vector space to a subset of action vector space. Thus, the expression $y_t = \beta(x_t)$ means that β suggests action y_t for a sensor vector x_t. β is sometimes called "policy" or, in the control systems community, "control law". By adding lagged state variables to the sensor vector, the behaviors can be viewed as purely reactive while still being able to handle dynamical models.

Often, no distinction between sensor data and actions is made, and an event e_t is defined as

$$e_t = (x_t, y_t)\,. \tag{1}$$

A demonstration θ is represented by a time series comprising N such events:

$$\theta = (e_1, e_2, ..., e_N)\,. \tag{2}$$

For each t, x_t is the observed sensor vector and y_t is the action vector issued by the demonstrator. Hereby, β most often denotes a relatively simple behavior with a single goal. In these cases, β represents a skill, as defined in the introduction. Common skills are *avoid obstacles, follow wall* or *drive towards goal*.

III. Methods for behavior recognition

In the presented work, three methods for behavior recognition are suggested and evaluated. Each method defines a function f_β for each skill β, mapping a sequence of events $\theta = \{e_1, e_2, ..., e_N\}$ and a time index t in $[1, \ldots, N]$ to a a real number representing the *activity level* $\alpha_t \in [0, 1]$:

$$\alpha_t = f_\beta\left(\theta, t\right). \tag{3}$$

α_t could, informally, be interpreted as the probability of β controlling the robot at time t given the observations θ, i.e.:

$$\alpha_t \sim P\left(\beta|\theta\right). \tag{4}$$

If f_β for all β are computed for each time t, the activation levels can be used both for positioning the segmentation points and determining the most suitable skills for each segment. The three suggested methods for behavior recognition are described below, followed by two examples where the methods are applied and evaluated.

A. β-Comparison

This method is based on the notion that two skills are equal if they produce similar actions given the same sensor input. f_β is defined as the distance between the action y_t observed in θ and the action produced by the specific β :

$$f_\beta(\theta, t) = 1 - d\left(\beta\left(x_t\right), y_t\right) \tag{5}$$

where d is a function computing a relevant distance measure for action vectors in the specific application. d should reflect the *relevant* difference between the two actions, and the implementation of d is as such dependent on both the robot and the behavior which is being learned. This is a limitation compared to the other methods described in Section III-B and III-C, which do not require any application-specific functions. The precise implementation of d used in the present work is described in Section IV.

B. AANN-Comparison

Autoassociative Neural Networks AANNs are regular feed-forward neural networks with the same size of input and output layers. The input and output parts of the training data are identical, such that the net learns to map input values onto the same values in the output layer. With a small hidden layer, the network performs data compression with a least-squares criterion [6]. When exposed to a new data vector, the difference between input and output (reconstruction error) is a measure of how similar the new data vector is to the training data. In this particular case, the network input at time t consists of the vector e_t comprising both sensor data and action data (Equation 1). The network output is denoted τ_t. One network for each skill is created and trained with an event sequence θ_β observed while performing behavior β. In this way, the characteristics of the sensory-motor patterns from β will be modeled by the network. When exposed to a new input vector, the reconstruction error can be used to define the f function and hence the activity level scaled to $[0, 1]$:

$$f_\beta(\theta, t) = 1/\left(1 - \|\tau_t - e_t\|\right). \tag{6}$$

C. S-Comparison

This algorithm is based on S-Learning, a prediction-based control algorithm inspired by the human neuromotor system,[21], [22]. S-Learning is able to extract temporal patterns in presented data, a very attractive property when comparing sequential data, such as sensor readings and motor commands. The temporal dimension allows S-Comparison to make decisions based on several recent samples, in contrast to β-Comparison (III-A) and AANN-Comparison (III-B) which both treat each sample separately.

S-Comparison differs in many respects from the S-Learning algorithm. Some changes are a direct consequence of the algorithm being used to compute a similarity measure rather than future actions. Other modifications improve the handling

of continuous data, since S-Learning was originally designed for discrete data.

Similarly to AANN-Comparison, one model of each skill is first created from a separate demonstration θ_β. Each model, or *pattern library* $\lambda = \{\rho_1, \rho_2, \ldots\}$, is a set of *patterns* $\rho = (e_1, e_2, ..., e_m)$, where each ρ is a sub sequence of θ_β. $\rho(k)$ denotes the k-th element e_k in ρ. The concatenation operator $\rho||e$ combines ρ and e into a new pattern including all elements in ρ and with e as last element.

Algorithm 1 Training of S-Comparison

1) $\lambda = \{\varnothing\}$
2) $\rho_{max} \leftarrow null$; $\delta_{max} \leftarrow H$
3) For each $\rho \in \lambda$ then
   $\delta \leftarrow \sum_{k=1}^{|p|} 1 - \frac{|\rho(k)-\theta(k)|}{\sigma}$
   If $\delta > \delta_{max}$ then $\delta_{max} \leftarrow \delta$; $\rho_{max} \leftarrow \rho$

4) If $\rho_{max} \neq null$ then
   $e_{new} \leftarrow \theta\left(|\rho_{max}| + 1\right)$
   $\rho_{new} \leftarrow \rho_{max}||e_{new}$
   Else $\rho_{new} \leftarrow \theta(1)$

5) Add ρ_{new} to λ then
6) Remove the first $|\rho_{new}|$ elements from θ
7) If $\theta \neq \{\varnothing\}$ then go to 2
   Else: *Training finished*

The initially empty pattern library is populated by traversing θ_β, a detailed description of this training procedure is found in Algorithm 1. Two constants control the result: The *creation threshold* H controls how frequently the algorithm creates completely new patterns, and the *error tolerance* σ is a real number between 0 and 1, balancing pattern length against correctness. A small σ produces many short patterns, resulting in an algorithm less prone to fall into false interpretations, but also with limited ability to recognize temporal patterns.

After training, S-Comparison can be used to define the f_β function, and hence the activity level for each position in θ. In the same way as during the training phase, a similarity measure δ is computed for each $\rho \in \lambda$ given a set of past events, i.e., all elements in θ up to time t. f_β are defined as

$$f_\beta(\theta, t) = \begin{cases} \frac{2}{\pi}\arctan\left(\delta_{max}/d\right): & if\ \delta_{max} > 0 \\ 0: & otherwise \end{cases} \quad (7)$$

where d denotes the dimensionality of θ. Since the similarity measure δ is arbitrary and depends on the amount of training data, the creation level, and the error tolerance, it has no obvious maximum value. For this reason, the $\arctan$ function is used as a squashing function to keep the activity levels between 0 and 1.

IV. Experimental setup

The three methods for behavior recognition presented in Section III were evaluated using a Khepera robot from K-Team [9]. As discussed earlier, the *correct* way to generalize any demonstration depends on, among other things, which skills are available. In the present work, five skills were used, which all produce common movement behaviors: FLW - *Drive along a wall on left side*, FRW - *Drive along a wall on right side*, AVOID - *Go straight ahead but avoid obstacles*, CORRIDOR - *Drive in a narrow corridor without hitting the walls*, and SLALOM - *A slalom drive around circular cones*. Each skill was first demonstrated manually using a standard keyboard interface as remote control. The physical test environment can be seen in Figure 1. Values from the eight infrared proximity sensors constituted the sensor vector x_t, while the speed of the two wheels constituted the action vector y_t. None of the presented recognition methods assumes that the skills are created from a single demonstration, and consequently θ_β should be understood as a general notation for all demonstrations of a specific β. However, in the present work, each skill is created from a single demonstration with a length of about 4000 samples. All values were rescaled to a number between 0 and 1 and logged at about 10 Hz.

Figure 1. Experimental setup. During both training and test sessions, the Khepera robot was placed in a large rectangular box with movable walls and cones, creating a steady and well controlled environment. Sensor readings and motor commands were recorded while the Khepera was remotely controlled, demonstrating one or several behaviors. The present image was taken during demonstration of the Slalom test case, cf. Figure 3.

To ensure that each log file contained all information necessary to achieve the specific goal, one neural network for each skill was created. Each network was trained on its corresponding log file, using the proximity values and wheel speeds as input and output data, respectively. The networks had one hidden layer with five nodes. After training, each network was used as controller β (See Section III) of the robot, which was then able to repeat the corresponding demonstrated behavior.

The three behavior recognition methods were evaluated using two test cases. The first test case, referred to as the *L-demonstration*, involved controlling the robot from start to goal, along the dashed line in Figure 2. Given the skills listed above, it should be generalized into FLW t=0 to 140, FRW t=140 to 215, CORRIDOR t=215 to 320 and finally FLW t=320 to 390.

The second test case, named *Slalom-demonstration*, involves

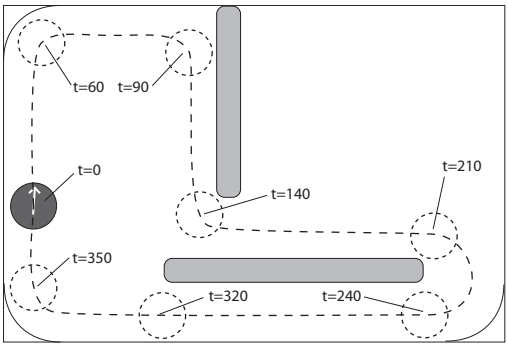

Figure 2. L-demonstration

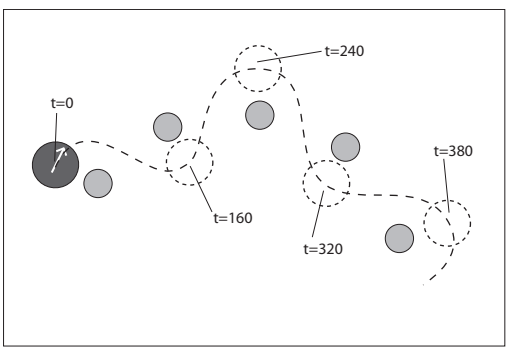

Figure 3. Slalom-demonstration

driving zigzag through a five-cone track, as illustrated in Figure 3. This demonstration differs in several ways from the demonstration used to create the SLALOM skill, both in number of cones and their relative positioning. The demonstration could be understood as an instance of the SLALOM skill, or as a sequence of FLW, FRW, FLW, and so on.

The dark gray circles in Figures 2 and 3 mark the robot's initial position, and the dashed circles mark key positions with the corresponding time stamps. Walls and obstacles in the environment are illustrated as light gray areas. Note that the dashed trajectory simply is a coarse illustration of the robots motion, the real trajectory is significantly more jagged due to the binary behavior of the keyboard control.

For the $\beta - Comparison$, the neural network controllers created from respective θ_β, as described above, were used. All five skills used in the present work are speed invariant in the sense that the goal does not depend on the speed of the robot. Inspired by the work of Olenderski and co-workers [17], the difference function d (Equation 5) is defined as the absolute difference in turning rates:

$$d\left(\beta\left(x_t\right), y_t\right) = \left|w_t - v_t\right| \tag{8}$$

where w_t represents the turning rate produced by $\beta(x_t)$ and v_t represents the turning rate observed in θ at time t. The turning rate is given by the difference in speed between the left and right wheels of the Khepera robot.

The AANN-Comparison and S-Comparison are performed by first training the algorithms with each θ_β. Each AANN has one hidden layer with five nodes, and is trained for 100 epochs, (very similar results are achieved for 50, 150 and 200 epochs). Neither AANN nor S-Comparison distinguishes between sensors and actions, and the action vector is not converted to a turning rate as in β-Comparison.

V. Results

The estimated activity levels produced for the two test cases are visible in Figure 4 to 9.

A. Results from β-Comparison

When looking at the results for the two test cases it is obvious that the β-Comparison approach has problems. The results from the L-demonstration, plotted in Figure4, show roughly correct estimations of FLW for t=20 to 90, FRW for t=130 to 210 and CORRIDOR for 150 to 320. However, during parts of the demonstration, β-Comparison gives the AVOID and SLALOM skills higher activity levels than any of the other, although none of these behaviors is part of the demonstration.

The reason for this poor performance can primarily be derived from the fact that β-Comparison only compares action vectors. For example, when driving along a wall on the left side, β-Comparison has no information about the high values on the leftmost proximity sensor. The algorithm only measures the difference in turning rate between the observed and produced data. When the robot is closer to the wall than the FLW controller is configured for, the controller returns a right turn to increase the distance to the wall. However, the human demonstrator might be more tolerant, accept the current distance to the wall, and consequently not find it necessary to turn. As a result, β-Comparison recognizes a relatively large difference in turning rate, and a low activity level is returned, even though the episodes from the demonstrator's point of view is very similar. This is the reason for the low FLW ratings for t=90 to 140.

B. Results for AANN-Comparison

The activity level computed by AANN-Comparison varies a lot between the different skills. In the L-demonstration (Figure 5), FLW and FRW receive activity levels around 0.8 during their respective parts of the demonstration, wile the CORRIDOR skill is only given an activity level of about 0.1 during the period where it should receive the highest levels. However, when looking only at the maximum levels at each time, the correct skill is identified during almost the entire demonstration. In the Slalom-demonstration (Figure 8), a sequence of FRW, FLW, FRW, FLW, and finally FRW is identified. This is, if not the best, at least a reasonable interpretation of the demonstration.

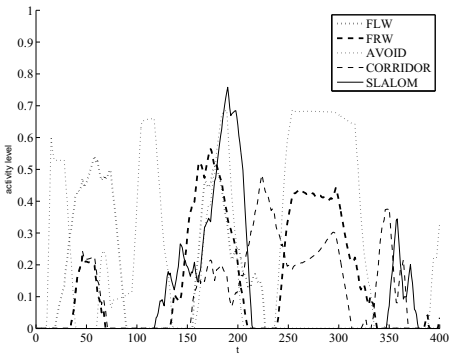

Figure 4. Recognition of the L-demonstration using β-Comparison

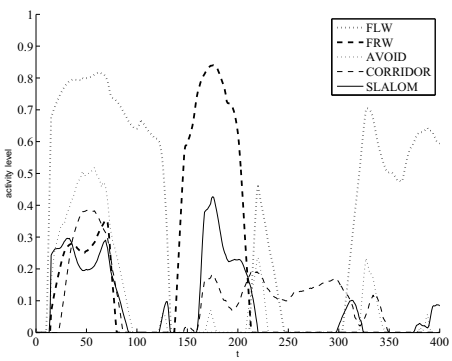

Figure 5. Recognition of the L-demonstration using AANN-Comparison

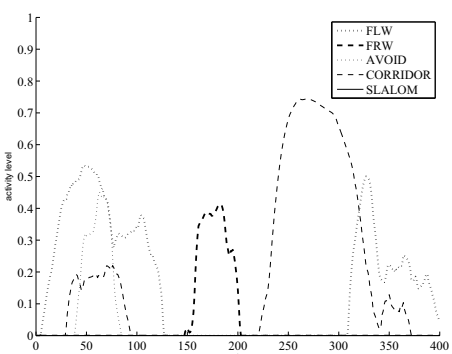

Figure 6. Recognition of the L-demonstration using S-Comparison

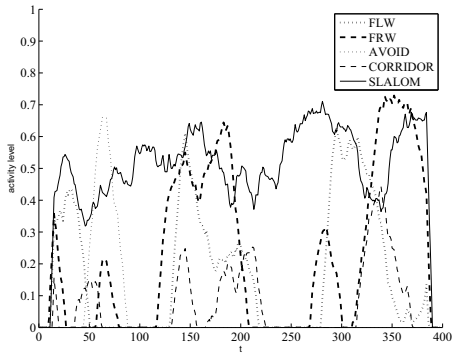

Figure 7. Recognition of the Slalom-demonstration using β-Comparison

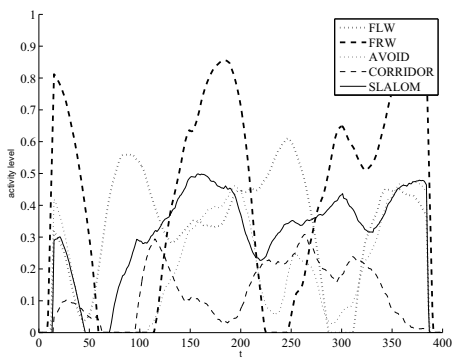

Figure 8. Recognition of the Slalom-demonstration using AANN-Comparison

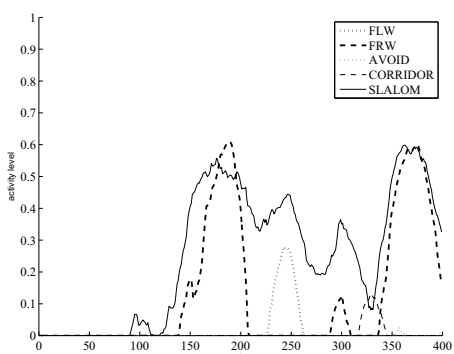

Figure 9. Recognition of the Slalom-demonstration using S-Comparison

C. Results for S-Comparison

S-Comparison is the most restrained comparison method, and causes significantly fewer false alarms than the other methods, in the sense that it does not give activity levels over 0 to behaviors that clearly do not belong to the demonstrated data. With the exception of relatively high ratings for the AVOID and CORRIDOR skills early in the L-demonstration (Figure 6), inappropriate skills are never given an activity level over 0.1 in either of the two test cases.

However, even appropriate skills receive relatively low activity values compared to the other recognition methods. This can be, at least to some extent, explained by the fact that the activity levels computed by S-Comparison are arbitrary, see Section III-C for details.

VI. Related Work

Skills and segmentation points in demonstrated sequences can be identified in several ways. Segmentation points may be identified by statistical features in data, for example thresholding the variance for certain sensor modalities [11], [19], thresholding the mean velocity of joints [7], [14], or entropy measures [5]. Another way is to observe the outcome of the robot actions. In [16], segmentation points are identified by constantly matching current sensory states with post-conditions for all pre-programmed skills. Once a post-condition is matched, both segmentation point and skills are identified. Other techniques try to directly identify the skills in the demonstrated data. Bentivegna [3] uses a nearest-neighbor classifier on state data to identify skills in a marble maze task. Pook and Ballard [20] present an approach where sliding windows of data are classified using Learning Vector Quantization in combination with a k-nn classifier.

To compare the approaches [11], [19], [7], [14], [5], [16], [3], [20] mentioned above with the work presented in this paper, it is important to first observe that the complexity of the skills is a crucial factor when choosing techniques for segmentation and skill identification. Approaches that look for general statistical features in data to detect segmentation points are not sufficient for the high-level behaviors that we are using. Our second test case, presented in Section IV, shows how both the location of segmentation points, and the identity of the actual skills depend on much more than the fluctuations in data. The Slalom demonstration can be understood as both an instance of the SLALOM primitive, and a sequence of FLW and FRW primitive, and the segmentation points have to be placed differently for these two cases. The activation levels suggested in this paper serve as a tool to deal with this ambiguity, e.g. by behavior arbitration or parallel behaviors, based on fuzzy logic, for example.

Nicolescu's approach using post-conditions [16] focuses on the segmentation points and ignores the actual behavior. This works well for behaviors where the goal determines the wanted behavior completely, but would fail with other types of behaviors.

In contrast, the sliding window k-nn classifier presented by Pook and Ballard [20] focuses directly on the identification of skills, and is in this sense similar to our approach. In fact, S-Comparison can be understood as a 1-nn classifier with a dynamic sliding window, even though it has not been used as a classifier in this case .

VII. Conclusions and Future Work

We have developed and evaluated three different techniques for behavior recognition in an LFD setting. All techniques compute an activity level which can be seen as an alternative to a pure classification approach. The examples show how the former approach allows a more informative interpretation of a demonstration, by not determining "correct" behaviors, but rather a number of alternative interpretations. As shown in Figure 8 for example , there is no reason to claim that a sequence of FLW, FRW, FLW, ... is more, or less "correct" than the SLALOM skill. The final decision of how to interpret the observed event sequence should be left to higher cognitive levels in a final LFD system. This decision depends, among other things, on the meaning of "generalization", which we intend to deal with in our future research.

The three presented techniques differ in the way they utilize the demonstrated data for behavior recognition. β-Comparison focuses on actions and ignores the input part entirely. As discussed in Section V, this leads to unavoidable problems. The technique is not suitable for segmentation purposes, but the average activity levels for an entire demonstration may be useful for work with behavior fusion, such as described in [17].

AANN-Comparison models the sensory-motor space for each skill and performs recognition based on how well the demonstrated data fits into these models. The results are clearly better than for β-Comparison, both in terms of skill identification and localization of segmentation points. This can be explained by AANN-Comparison using the sensor vectors directly in the comparison process, and consequently it has much more information available than does β-Comparison.

S-Comparison uses most information of the three evaluated algorithms. Similarly to AANN-Comparison, it treats sensors and actuators as a single event vector. In addition, it models temporal patterns in the event stream. However, the modeling power of S-Comparison does not show up as increased performance in the results, compared to AANN-Comparison. S-Comparison is less noisy than the other two methods and has both fewer false positives (high activity level where it should be low) but also more false negatives (low activity level where it should be high) compared with AANN-Comparison. This is clearly visible in the first part of the Slalom test case, Figure 9.

Furthermore, S-Comparison fails to identify exact positions of the segmentation points. This can be seen as a direct consequence of the temporal dimension of S-Comparison. Since S-Comparison has not been trained on data describing the transitions between different skills, such periods yield a low similarity measure. As seen in Figures 6 and 9, this problem shows up as gaps in the activity level curves. Similar problems have been reported by Pook and Ballard [20], evaluating their window-based approach.

Even though some effort has been put on comparing these techniques, this work should be seen as an attempt to eval-

uate a concept of behavior recognition, rather than test the exact performance of presented algorithms. Furthermore, the problem of identifying characteristics of segmentation points required to autonomously repeat a demonstrated behavior is not addressed here. However, by identifying the location of the segmentation points, we have drastically narrowed down the problem.

Both skills and task level representations [16] are created in this approach from manual demonstrations. A thrilling possibility is to use the task level representations, i.e., sequences of skills, as primitives in even more complex tasks. This would allow the robot to reuse its experience, both from manual demonstrations and successful automatic drives. These issues will be subject to future work.

The present work should also be seen as a step towards a developed interaction between the robot and its user. The presented methods transform a complex, multi-dimensional stream of data into a relatively simple sequence of named skills, easily read and understood by a human user. From an interaction perspective, this feature appears as one of the strongest motivations behind the use of previously learned skills in LFD, and possibly also in other areas of robotics.

REFERENCES

[1] R. Amit and M. Mataric. Parametric primitives for motor representation and control, 2002.

[2] C. G. Atkeson and S. Schaal. Learning tasks from a single demonstration. In *In Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 1997.

[3] Darrin C. Bentivegna. *Learning from observation using primitives*. PhD thesis, 2004. Director-Christopher G. Atkeson.

[4] E.U. Braun, H. Mayer, I. Nagy, A. Knoll, S.M. Wildhirt, R. Lange, and R. Bauernschmitt. An instrumentation system with force feedback, automatic recognition and skills for cardiac telemanipulation. In *Proceedings 33rd Annual International Conference of Computers in IEEE Comp Cardiol*, volume 33, pages 553–556, 2006.

[5] Paul Cohen, Niall Adams, and Heeringa Brent. Voting experts: An unsupervised algorithm for segmenting. *To appear in Journal of Intelligent Data Analysis*.

[6] K. I. Diamantaras and S. Y. Kung. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[7] A. Fod, M. Mataric, and O. Jenkins. Automated derivation of primitives for movement classification, 2000.

[8] Thomas Hellström. Teaching a robot to behave like a cockroach. In *Proceedings of the Third International Symposium on Imitation in Animals and Artifacts in Hatfield UK*, pages 54–61, 2005.

[9] K-Team. Khepera ii mobile robot. www.k-team.com, 2007.

[10] N. Koenig and M. J. Matarić. Demonstration-based behavior and task learning. Working Notes, AAAI Spring Symposium, 2006.

[11] Nathan Koenig and Maja J Matarić. Behavior-based segmentation of demonstrated tasks. In *In International Conference on Development and Learning (ICDL),*, Bloomington, IN., May 2006.

[12] Paul Martin and Ulrich Nehmzow. Programming by teaching: Neural network control in the manchester mobile robot. In *Proceedings Intelligent Autonomous Vehicles*, 1995.

[13] J. Nakanishi, J. Morimoto, G. Endo, S. Cheng, G. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:2–3:79Ü81, 2004.

[14] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi. Recognition and generation of leg primitive motions for dance imitation by a humanoid robot, 2003.

[15] Monica Nicolescu and Maja Matarić. Linking perception and action in a control architecture for human-robot domains. In *Thirty-Sixth Hawaii International Conference on System Sciences, HICSS-36*, Hawaii, USA, January 2003.

[16] Monica Nocolescu. *A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains*. PhD thesis, University of Southern California, 2003.

[17] Adam Olenderski, Monica Nicolescu, and Sushil Louis. Robot learning by demonstration using forward models of schema-based behaviors. In *Proceedings of the Second International Conference on Informatics in Control, Automation, and Robotics.*, volume 3, pages 263–26, 2005.

[18] J. Peters and S. Schaal. Policy learning for motor skills. In *Proceedings of 14th international conference on neural information processing (iconip)*, 2007.

[19] Richard Alan Peters II and Christina L. Campbell. Robonaut task learning through teleoperation. In *Proceedings of the 2003 IEEE, International Conference on Robotics and Automation*, pages 23 — 27, Taipei, Taiwan, September 2003.

[20] Polly K. Pook and Dana H. Ballard. Recognizing teleoperated manipulations. In *ICRA (2)*, pages 578–585, 1993.

[21] B. Rohrer and S. Hulet. Becca – a brain emulating cognition and control architecture. Technical report, Cybernetic Systems Integration Department, Sandria National Laboratories, Alberquerque, NM, USA, 2006.

[22] B. Rohrer and S. Hulet. A learning and control approach based on the human neuromotor system. In *Biomedical Robotics and Biomechatronics, 2006. BioRob.*, pages 57–61, February 2006.

[23] H. Urbanek, A. Albu-Schäffer, and P. Smagt van der. Learning from demonstration repetitive movements for autonomous service robotics. In *IROS 2004 IEEE RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, Sept. 28-Oct.2, 2004*, 2004. LIDO-Berichtsjahr=2004.

III

Paper III

A Formalism for Learning from Demonstration*

Erik Billing and Thomas Hellström

Dept. Computing Science, Umeå University, SE-901 87 Umeå, Sweden
billing@cs.umu.se and thomash@cs.umu.se
www.cs.umu.se/research/robotics

Abstract: The paper describes and formalizes the concepts and assumptions involved in *Learning from Demonstration (LFD)*, a common learning technique used in robotics. LFD-related concepts like *goal*, *generalization*, and *repetition* are here defined, analyzed, and put into context. Robot behaviors are described in terms of trajectories through information spaces and learning is formulated as mappings between some of these spaces. Finally, behavior primitives are introduced as one example of good bias in learning, dividing the learning process into the three stages of *behavior segmentation*, *behavior recognition*, and *behavior coordination*. The formalism is exemplified through a sequence learning task where a robot equipped with a gripper arm is to move objects to specific areas. The introduced concepts are illustrated with special focus on how bias of various kinds can be used to enable learning from a single demonstration, and how ambiguities in demonstrations can be identified and handled.

Keywords: Learning from demonstration, Ambiguities, Behavior, Bias, Generalization, Robot learning.

Research Article · DOI: 10.2478/s13230-010-0001-5 · JBR · 1(1) · 2010 · 1-13

A Formalism for Learning from Demonstration*

Erik A. Billing†, Thomas Hellström‡

Department of Computing Science, Umeå University, Umeå, Sweden

Received 12 October 2009

Accepted 26 February 2010

Abstract

The paper describes and formalizes the concepts and assumptions involved in ***Learning from Demonstration (LFD)***, a common learning technique used in robotics. LFD-related concepts like ***goal, generalization***, and ***repetition*** are here defined, analyzed, and put into context. Robot behaviors are described in terms of trajectories through information spaces and learning is formulated as mappings between some of these spaces. Finally, behavior primitives are introduced as one example of good ***bias*** in learning, dividing the learning process into the three stages of ***behavior segmentation***, ***behavior recognition,*** and ***behavior coordination***. The formalism is exemplified through a sequence learning task where a robot equipped with a gripper arm is to move objects to specific areas. The introduced concepts are illustrated with special focus on how bias of various kinds can be used to enable learning from a single demonstration, and how ambiguities in demonstrations can be identified and handled.

Keywords

learning from demonstration · ambiguities · behavior · bias · generalization · robot learning

1. Introduction

Learning From Demonstration (LFD) is a well established technique for teaching robots how to perform useful tasks. The basic idea is that the robot learns a behavior from one or several demonstrations performed by a, most often human, teacher. The research area is attractive, both in its intuitive approach to human robot interaction and as a framework for a theoretical analysis of knowledge representation and transfer of knowledge between intelligent agents.

Research on LFD is influenced by a variety of fields, including control theory, artificial intelligence, psychology, ethology, and neuro physiology. While primarily being a big asset, the multidisciplinary nature of LFD also contributes to the lack of a unified formalism for the different components constituting the research field. It should not come as a surprise that the terminology used differs for works conducted by researchers from various areas. In this paper, we define and formalize the common ideas and principles involved in LFD. The presented work is both a survey of how these concepts are used in research, and an attempt to describe them in the light of related concepts in machine learning, planning theory, and psychology. To our knowledge this has not been previously done in a unified way and the result can be used both as a theoretical introduction to the field and as framework for further development and research. In contrast to other surveys of the area [4, 12], the present work specifically focuses on LFD where the robot is directly controlled during demonstration, e.g. via teleoperation or kinematic teaching. While this direction removes some of the hard and important issues in LFD, it allows increased focus on other aspects, specifically how bias is introduced into the LFD process.

The formalism is applied to a sequence learning task in which the introduced concepts are illustrated with a special focus on how bias of various kinds can be used to enable learning from a single demonstration, and how ambiguities in demonstrations can be handled.

The formal approach is inspired by the work on planning and actuation by LaValle [53] and therefore does not always follow the terminology and notation found in common literature on LFD. Where this is the case, it is highlighted and the commonly used terms are referred.

In Section 2, a few fundamental concepts that form the basis for the rest of the paper are introduced. Section 3 gives a formal description of the learning process using these concepts. In Section 4, the introduced formalism is applied on a sequence learning task using a Khepera robot equipped with a gripper arm. Section 5 summarizes the paper and discuss directions for future research. A symbol index summarizing introduced notations can be found in Table 5.

2. Basic concepts

2.1. State space

One fundamental component in classical AI is the concept of a ***state space*** X, described by a ***world ontology*** [77, p.222]. The state space can be defined as a set of all possible situations that could arise in the world [53, p.17]. More specifically, the state space only includes the ***relevant*** aspects of the world, given a particular task or limited set of tasks. However, if the task is unknown it is very difficult to identify which aspects of the world are relevant. One could of course try to include all aspects that might be of interest, but even if possible, that would result in a huge and complex state space, implying tremendous sensing requirements when applied to a field such as LFD. Furthermore, defining a state space introduces many unnecessary assumptions about the world, and requirements for information which make the problem much more complex than necessary. This observation is nicely illustrated by

*Parts of this text also appear as a technical report: E. A. Billing and T. Hellström. Formalising Learning from Demonstration, *UMINF 08.10,* Department of Computing Science, Umeå University, Sweden, 2008.

†E-mail: billing@cs.umu.se

‡E-mail: thomash@cs.umu.se

Simons' ant [81] and is also related to the *frame problem* [47, 62].
For these reasons, it is desirable to create new spaces, less task-specific and sensor-demanding, in which behaviors can be represented. Such a redefined representation is referred to as an *information space* [53, ch.11]. The concept of information spaces is also common within LFD, but appears under different names. In order to facilitate learning, approaches to LFD often utilize so called *primitives* or *skills*. These primitives can be seen as building blocks from which more complex behaviors can be composed, which results in moving the learning process away from the state space into a new representational space composed of the available skills, e.g. [8, 34, 46, 51, 65, 68]. Many of these approaches relate strongly to *Behavior Based Control (BBC)* [5, 58, 60]. BBC has its roots in the reactive paradigm, but emphasizes parallel, loosely connected *behaviors* for control of the robot as an emergent property, rather than a single stimuli-response loop.
The possibility of applying the concept of information spaces within LFD is further investigated in Section 3, but first a few other basic concepts have to be introduced.

2.2. Sensing and acting

Imagine an agent interacting with the environment. It perceives the world through its sensors and acts upon the world with its actuators. The sensors are defined as a function $h : X \rightarrow Y$ transforming a state $x \in X$ into a sensor state $y \in Y$ [53, p. 561]. Y denotes the *observation space*, i.e., the set of all possible readings returned by the agent's sensors. Each $y \in Y$ is a vector $(y(1), y(2), \ldots)$ comprising simultaneous values from all sensors. Typical examples are a thermometer that maps physical temperatures x to numbers $y(1) \in \mathbb{R}$ or a GPS receiver that maps physical positions to latitude and longitude, $y(2) \in \mathbb{R}^2$. Y corresponds to the *stimulus domain* in behavior-based robotics [5].
On the actuator side, actions can be said to transform a state into another state. Hence, actuators implement the function $f : X \times U \rightarrow X$ where U denotes the *action space*, i.e., the set of all possible actions the agent can execute. A typical example is the requested velocity for each motor of the robot. Note that this does not specify the actual motor velocity, and only the outgoing information is represented in U. The actual velocity is usually represented in state space X.
Now a description of how the agent behaves, i.e. generates actions, can be introduced. In general, such a description is referred to as a *controller*, but is also known as a *plan* [53, p.560], *behavior mapping* [5, 27, 68, 71], *motor primitive* [3], *control policy* [4] or *inverse model* [39]. Several important differences between these terms do exist, for example in terms of abstraction level and temporal extension, but for now they can all be said to implement the function π:

$$\pi : X \rightarrow U. \tag{1}$$

Hence, π maps states $x \in X$ to actions $u \in U$. As mentioned before, X is not explicitly represented in the agent. Still, the physical sensors and actuators can be said to implement the functions h and f, respectively. In contrast, π can not be implemented without an explicit definition of and access to X. To solve this issue, π is later redefined and then controls the agent based on the information space instead of the state space.

2.3. Information space

The observation and action spaces are widely used by the robotics community. These spaces are often combined into a *information space* $I = U \times Y$, also known as the *sensory-motor space* [73]. In each stage k the robot experiences a *sensory-motor event* $e_k = (u_{k-1}, y_k) \in I$. The action at $k-1$ is used since u_k changes the current stage to $k+1$.
One approach that extensively uses representations in I is *sensory-motor coordination (SMC)* [72]. From an SMC perspective, sensing and acting are not two separate processes. In contrast to classical reactive systems, SMC does not view the information flow purely as going from sensors to actuators. Actions give rise to stimuli, just as much as stimuli influences actions. If the agent can predict these relations, it can intentionally control its interactions with the world. Hence, control is seen as a problem of coordination. Similar views are common within psychology, anthropology and cognitive science, [37, 45, 82].
The sensory-motor space I has several advantages when compared to the state space. Most importantly, it is easily defined. If an agent is designed with a fixed number of sensors and actuators, the size of I remains constant independently of environment and task. Of course this limits the possibility of adding new sensors or actuators to the agent without changing the robot's representational space and as a consequence affects previous representations, but for many applications this is a reasonable limitation. The sensory motor space also has a number of drawbacks. In contrast to state space, I does not necessarily contain all information necessary to make a control decision at each moment. A decision, i.e., selection of the next action, may have to be based not on the most recent sensor and motor readings, but on complex patterns of previously observed sensory-motor events. Let $\tilde{Y}_k$ denote the *history observation space*, i.e., the set of all possible observation histories $\tilde{y}_k$ until current stage k:

$$\tilde{y}_k = (y_1, y_2, \ldots, y_k) \in \tilde{Y}_k \tag{2}$$

where each vector $y_i \in Y$ is provided by the sensors at stage i. Similarly, let $\tilde{U}_k$ be the *history action space*, i.e., the set of all possible action histories until current stage k:

$$\tilde{u}_k = (u_1, u_2, \ldots, u_k) \in \tilde{U}_k \tag{3}$$

where each $u_i \in U$ is a particular action vector issued at stage i.
The histories $\tilde{y}_k$ and $\tilde{u}_k$ in combination with the initial conditions η_0 form a *history information state* η_k, also referred to as an *event history*. η_k includes all accumulated information up to stage k [53, p.566]:

$$\eta_k = (\eta_0, \tilde{u}_{k-1}, \tilde{y}_k) \in I_k \tag{4}$$

The initial conditions η_0 describe presumptions about the state of the world X before stage 1. The history information state is a central concept in the formalism since it represents all the information the agent has received, and as a consequence η_k is always known in stage k. I_k is known as the *history information space* and should be understood as the set of all possible event histories up until stage k [53, p.565]:

$$I_k = I_0 \times \tilde{U}_{k-1} \times \tilde{Y}_k \tag{5}$$

where I_0 represents the set of all possible initial conditions.
The definition of I_k becomes impractical in cases where the number of stages is not fixed. Instead, we normally refer to the *information history space* I_{hist}, which has an unspecified length [53, p.657]:

$$I_{hist} = I_0 \cup I_1 \cup I_2 \cup \ldots \tag{6}$$

I_{hist} includes all possible combinations of everything the agent could possibly observe and do. Most $\eta \in I_{hist}$ will of course never appear, due to limitations imposed by the environment and the physical

shape of the robot. For example, imagine a simple robot, equipped with a proximity sensor on each of its four sides, placed in an empty large square box. In this environment, the robot never observes a y_k with high activation of all proximity sensors simultaneously. This is a simple consequence of physical properties of the environment and the robot itself. The same reasoning could easily be applied to a human agent. There is a huge amount of patterns the human senses theoretically could perceive, but only a fraction of these will actually be observed.

Most of the formal definitions in this paper take place in history information space I_{hist}. You might ask why representations take place in such a huge and complex space when only a fraction of its representational power is actually used. I_{hist} should not be understood as ***the*** representational space, but ***a*** representational space, a very basic one. Any information the agent can acquire is representable as an event history $\eta \in I_{hist}$. Furthermore, I_{hist} is, in contrast to state space X, both well defined and completely task invariant and is as such very suitable for learning purposes. However, in many other respects I_{hist} is not the best representational space. I_{hist} contains a lot of redundant information, making it difficult to extract features relevant to the specific task. For this reason, a new ***derived information space*** I_{der} may be created. I_{der} should be seen as a simplification of I_{hist}, where relevant features are represented, while irrelevant information is not contained, [53, p.571]. The observant reader may think this sounds disturbingly similar to the formulation of state space. This observation is highly relevant and reflects to some extent the purpose of inferring I_{der}. The use of derived information spaces as bias in learning, and its relation to the state space, is further discussed in Sections 3.2 and 3.4.

2.4. Controller

The controller defined in Equation 1 can now be reformulated in a form that allows it to be used without full access to state space X :

$$u_k = \pi(\eta_k) \tag{7}$$

where $u_k \in U$ is the action vector issued at stage k and $\eta_k \in I_k$ is the agent's event history at stage k. π is defined here as a function from information history space to action space:

$$\pi : I_{hist} \rightarrow U. \tag{8}$$

In simple cases, a controller can be modeled as a function of only the most recent sensory-motor event. Systems based purely on such single-event controllers are called ***reactive systems*** [21]. Formally, these systems implement π as

$$u_k = \pi(y_k) \tag{9}$$

which can be seen as a special case of Equation 7. This definition of π is similar to Arkin's ***behavior mapping*** $\beta : S \rightarrow R$, where S and R are stimulus and response, respectively [5]. However, in the general case we use the definition of π given in Equation 7.

2.5. Behavior

The word behavior is commonly understood as an agent's actions in relation to the environment, but in the robotics community it has many different meanings. In the present work, behavior is understood as a purposeful way of acting. This does not imply that behaviors include explicit representations of goals, but from an observer's point of view, the behavior can be said to implement some kind of purpose, or goal. This argument is developed in Section 3.3.

Using the introduced terminology, a ***behavior*** B is defined as a subset of information history space $B \subset I_{hist}$. Each element in B is an event history η that represents one instance of the desired behavior.

Often, no explicit distinction is made between the observable interactions with the world, and the mechanisms producing these interactions. However, B describes nothing about how the behavior is produced, and therefore this notion of behavior is different than the terminology commonly used within behavior-based robot architectures [5, 27, 58, 68]. B is purely an intrinsic definition and describes exclusively the behavior from the agent's perspective.

3. Learning From Demonstration

Learning From Demonstration (LFD) is a well established technique for robot learning. An overview of early work is found in the work by Bakker and Kuniyoshi [6] while recent work and classification of the field is found in the survey by Argall et al. [4]. Another excellent survey of the area can be found in a recent book by Billard et al. [12]. The basic idea in LFD is that the robot learns to do things by observing other agents, be it human beings or other robots. Several flavors of this approach exist and the terminology used differs somewhat in published research. Similar approaches are presented under terms like ***Imitation Learning***, ***Learning From Experience***, ***Learning From Observation*** and ***Robot Programming by Demonstration***. See the work by Argall et al. [4] for more details on terminology.

Research on LFD has been divided into four key problems: what, how, when and who to imitate [11, 12]. ***What to imitate*** refers to the problem of identifying which aspects of the demonstration are relevant for the task [20]. ***How to imitate*** is the question of how the skill is to be encoded. A central part of this issue is the ***correspondence problem*** [66, 67] which refers to the process of mapping the observed sequence of events to corresponding actions of the pupil. In most practical situations the pupil is not given an explicit set of demonstrations, but the pupil must detect when the teacher is doing something related to the task to be learned. This problem is known as ***when to imitate***. Finally, ***who to imitate*** refers to the identification of the teacher, which is also a difficult issue in many applications. These four questions are very general and can also be applied to learning situations with human or animal pupils. In practice, what and how to imitate are the most frequently studied problems within LFD.

New behavior can be demonstrated to a robot in many ways, for example by having the robot pupil watch the teacher demonstrate the desired behavior. Here we focus on LFD where the teacher directly controls the robot, e.g. by teleoperation. The recorded data sequence from such a control session, including both executed motor commands and sensor readings, is denoted ***demonstration***. The purpose of LFD is to create a controller π capable of reproducing the demonstrated behavior. While there are many other ways to demonstrate a new behavior to a robot, LFD via teleoperation constitutes a well defined setting that can be generalized to many practical applications. Formally, a demonstration is, in this setting, an event history $\eta_k \in I_{hist}$ (refer to Equation 4) where $\tilde{u}_{k-1}$ is the sequence of actions issued by the teacher up to stage $k-1$ and $\tilde{y}_k$ is the sequence of observations up to stage k.

In this setting, a direct correspondence between recorded events in a demonstration and sensors and actuators is assumed (a direct record mapping and no embodiment mapping, following the terminology by Argall et al. [4]). The observations y_k in the demonstration are assumed to correspond to the observations that are generated in real-time by the sensors and sent to the controller. Furthermore, the observed action variables u_k are assumed to directly correspond to the actuator signals generated by the controller π. This relates to ***self-***

imitation, i.e., the pupil learns by performing the actions itself, with help from a teacher [78, 79]. Self-imitation, in contrast to imitation of others, avoids two difficult problems. Firstly, the problem of observing the teacher's actions, and secondly, the correspondence problem.
LFD has its roots in the more general approach to create computer programs from demonstrations, known as *Programming By Demonstration (PBD)* or *Programming By Example (PBE)*, e.g. [26, 54]. However, modern LFD is far from these general approaches. This paper presents a formalism for robot learning through demonstration, which, while it can be seen as the creation of a specific kind of computer programs, does not aim at the wider interpretations of PBD or PBE.
The goal of LFD is, in this context, to generate a controller π that enables a robot to *repeat* a demonstrated behavior B. π may be a state-action mapping, a model of the world dynamics (system model) or a model of action pre- and postconditions (plans), see the work by Argall et al. [4] for details. If successful, the robot is said to have learned behavior B. Formally, the process of learning B from a set of N demonstrations b is understood as selecting π from the *controller space* Π using a *learning function* λ:

$$\pi = \lambda(b) \in \Pi \tag{10}$$

where b is the set of event histories η that constitute the demonstration. The LFD process is illustrated in Figure 1. Π contains all possible controllers for a specific chosen observation space and action space. This is of course a huge space that is never computed explicitly.
The selected controller π must have specific qualities for the learning to be regarded successful. These qualities are related to the event histories η that may be generated by a robot using controller π. The *realization space* $R \subset I_{hist}$ for a π is defined as the set of all such event histories, generated by the *realization function* Λ:

$$R = \Lambda(\pi) \in I_{hist} \tag{11}$$

Λ can be seen as an abstraction of the physical robot placed in a particular environment and controlled by a specific π, able to produce the set of all possible trajectories through I_{hist}. Of course, the robot can not control the produced event histories $\eta \in R$ entirely on its own, but relies on an external component, the environment. This creates a nice analogy to λ, which also relies on an external component, called *bias*. Thus the learning function λ can be seen as the inverse function of the robot represented by Λ. λ maps a set of event histories to a controller and Λ maps a controller to a set of event histories. This is further developed in Section 3.2.
The process of selecting π has many similarities to system identification, where a model of the system is constructed from observed input and output data [55]. The system, consisting of the agent and its environment, is modeled such that the system output u_{k+1} can be predicted given a sequence of previous inputs and outputs η_k until stage k. However, the aim of system identification is in one sense much more ambitious than LFD, since the system's response to any input y_k is to be predicted. In LFD, we are satisfied with a π producing an action that, if possible, leads to an event sequence $\eta_{k+1} \in B$ given that $\eta_k \in B$. In other words, LFD does not necessarily model the outcome of all possible actions u_k in each state, only the ones that occur for the robot in a particular environment.
B should be understood as the set of event histories the human teacher associates with a particular desired behavior. For example, if the teacher wants to teach the robot to move to a door, B would contain all event histories where the robot ends up by a door, in an acceptable way. The behavior must be formulated such that the robot is able

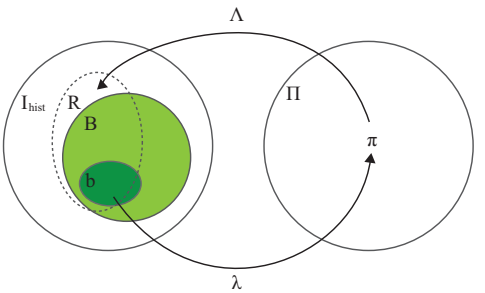

Figure 1. The LFD process. The light-colored area represents the wanted behavior B which is demonstrated with N training demonstrations $b = \{\eta^{(1)}, \ldots, \eta^{(N)}\} \subset B$ represented by the dark-colored area. The learning function λ creates a controller $\pi \in \Pi$. In interaction with the environment, π realizes (repeats) the learned behavior. The realization set $R \subset I_{hist}$ is marked by the dashed line.

to reproduce the behavior in all desired environments. There may be situations in which the robot can not distinguish between significant aspects of the world. In these cases, the robot's sensing capabilities or other aspects of the behavior have to be modified. Assume that the *move-to-door* behavior is to be applied to a robot in a hotel environment. The robot must now be able to separate between doors. One alternative is to add a new sensor allowing the robot to directly identify each door it approaches, resulting in a redefined I_{hist}. Another alternative is to change the behavior such that the robot can use existing sensors, e.g. wheel odometry, in order to distinguish different doors by their locations. This corresponds to a modification of B.
The quality of the generated π is typically described as the ability to "repeat a behavior", which is the topic of the next section.

3.1. What does it mean to repeat a behavior?

The goal of LFD is to generate a controller π that enables a robot to *repeat* a demonstrated behavior B given a set of demonstrations b. This may sound like a well defined mission, but is actually both vague and ambiguous. Consider the following example of a seemingly trivial demonstration.

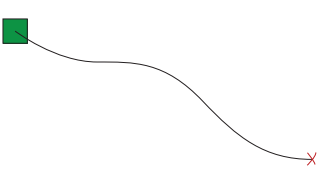

Figure 2. A simple demonstration where the tip of a robot arm starts at the red cross in the lower right corner and moves over the table until it is positioned over the green cube. The demonstration can be interpreted in a number of fundamentally different ways.

Observe a sequence of sensory-motor events describing a robot arm moving over a table, finally stopping when positioned above a green cube (Figure 2). What does it mean to repeat this sequence of events?

One could imagine a vast number of interpretations. Here are a few examples.

1. Assuming that the path is the important aspect of the demonstration, a successful controller may be written as $u = \pi_{PATH}(y)$ where the function π_{PATH} computes an action u for each pose y, such that the arm follows the demonstrated path. This kind of learning scenario refers to traditional programming of industrial robot arms, as well as path-tracking autonomous vehicles, e.g. [43].

2. Instead, if the demonstration is seen as an example of how to reach the final position, the path itself becomes irrelevant and the controller described above would not be suitable. In this case, a successful controller could be written as $u = \pi_{TARGET}(y)$ where the function π_{TARGET} uses inverse kinematics to compute actions such that the tip of the robot arm reaches the target.

Case 1 corresponds to what is often called *action-level imitation* [22] where the robot carries out the same actions as the demonstrator. Case 2 is often called *functional imitation* [29] in which the robot is supposed to achieve the same effect on the environment [67]. In the work by Alissandrakis et al. [2], the quality of action-level imitation is measured in state and action metrics while functional imitation is measured in effect metrics. State and action metrics define the similarity of behaviors in terms of the state and/or actions of the agent, while effect metrics define behavior in terms of their effect on the environment.

Within these two categories one could imagine a vast number of interpretations. Should the observed sequence of positions be understood as fixed coordinates, or relative to the robot arm's starting position? Is the green cube really the relevant target, or is the target defined by an absolute position? Is the target a cube of any color, or or is the target perhaps any green object? Using many demonstrations of the same behavior reduces some of the ambiguity, but in general it is impossible for the learner to tell which interpretation is "correct" without further information. In fact, the learner can not even enumerate a set of possible interpretations without a specification of state variables relevant for the task to be learned. The discussion about what it means to repeat a behavior becomes complicated further when the robot acts in a dynamic, non-deterministic and partially accessible [77, ch.2] environment. Demonstrated event sequences may be both incomplete and contain mistakes that should not be learned or repeated [28].

If the robot manages to successfully repeat a demonstrated behavior under different conditions than during the demonstration we say that the robot is able to *generalize* the demonstrated behavior. More specifically, we refer to the robot's ability to produce an event history $n_k \in B$, under conditions n_{k-1} not identical to the ones appearing during the demonstrations in b. This can be formally described as how well the realization space R corresponds to the desired behavior B, e.g. as a minimization of $R \setminus B$ and $B \setminus R$ (refer to Figure 1).

Generalization can also be viewed as an extension of b by interpolation or extrapolation of the demonstrated event histories. For this to work one has to specify the aspects of the demonstrated data that are important, i.e., the previously mentioned problem of *what to imitate* (Section 3). One approach is to introduce a *metric of imitation performance* [1, 2, 10]. Repeating a demonstration means minimizing the distance between the demonstrations and the repetitions using this metric. To find the metric, the variability in many demonstrations is exploited such that the essential components of the task can be extracted. One promising approach to construct such a metric is to use the demonstrations to impose constraints in a dynamical system [24, 38, 44]. Giovannangeli and Gaussier [35] use human-robot interaction to improve generalization when learning sensory-motor behaviors for homing and path following. In the described work, teaching by error correction (proscriptive learning), is shown to give superior generalization compared to a regular demonstration (prescriptive learning). The generalization problem is also acknowledged outside the LFD community. In *Machine Learning*, the term *generalization performance* of a learning algorithm relates to "its prediction capability on independent test data" [41, p.193] which is identical to the common usage of the term in robotics. The general problem with machine learning in high-dimensional spaces is often expressed as the curse of dimensionality [33, p.170], and is highly relevant also for robots with high-dimensional observation and action spaces. Learning in such situations becomes inherently difficult since the demonstrated data fills history information space very sparsely and interpolation and extrapolation become highly risky operations. The situation is related to the *No Free Lunch Theorem* [85], which states that for a large class of machine learning algorithms, there is no universal best algorithm to solve all problems. Instead, an algorithm has to be specialized to the problem under consideration to guarantee its superiority over any random algorithm. This specialization consists of additional task-dependent information that has to be supplied to the learning algorithm as bias. In the case of LFD, possible sources of bias are the robot's prior knowledge, feedback from the environment when the robot tries to repeat the demonstrated behavior and human feedback before, during, and after learning. The bias concept is further investigated in the next section.

3.2. Bias

The bias of a machine learning algorithm is defined as "any basis for choosing one generalization over another, other than strict consistency with the observed training instances" [63]. The basis may be seen as form of pre-evidential judgment, or prejudice regarding the structure of the data or the data generating process. In the case of numerical regression, assuming a linear relation between input and output corresponds to a high bias, while a cubic model corresponds to a lower bias. In the case of LFD, bias can be applied to three different parts of the problem definition:

1. Sensor variables. This can involve selection of relevant sensors, or extraction of specific features that are judged relevant for the specific task. It may also involve creation of intelligent sensors to facilitate feature extraction.

2. Action variables. Most often this involves restricting the output of the controller π to one or a few actuators. For example when learning a grip operation, the actions for moving the robot may be regarded irrelevant while the gripper motion is highly relevant. This reduces the size of action space.

3. Controller function π. Bias can restrict the functional form of π, e.g. to an artificial neural network of a specific size and architecture. Bias can also be expressed as general requirements of π, such as smoothness criterion or lower/upper bounds. The use of predefined skills as described below is another example.

Bias can be introduced into the learning process in a number of ways. First of all, it may be hard-coded into the learning algorithm, e.g. by choosing a specific neural network [57] or rule based framework Hellström [42] to represent π. Another common and very powerful technique to introduce bias is to use predefined skills or behavior primitives. Besides being biologically motivated [36, 64], the technique is commonly used in robotics research, e.g. [34, 59, 61, 68]. Learning is in this case reduced to selection of the right primitives and parameter estimation to adjust the primitives to the demonstrated data. The

introduction of primitives is a way to reduce the dimensionality of the learning problem (i.e. to deal with the ***curse of dimensionality*** mentioned above). The set of primitives is obviously much smaller than Π which clearly simplifies learning. An analogy is numerical regression with a large feed-forward neural network compared to a low-level polynomial. The polynomial introduces bias that makes learning much easier, at the price of limiting the solution to the specific functional form of the bias.

Regarding bias for sensors and actuators, it is common to hard-code a set of relevant sensors and action variables for the task at hand, or to pre-process the data before feeding it to the learning algorithm. This kind of bias may also be introduced by interaction with the human teacher who tells the robot to use specific sensor modalities. Saunders and coworkers present an approach where relevant elements of the state vector are weighted based on their information gain and on manual selection from a teacher [70, 79].

Bias may also be subject to meta learning, suitable sensors can for example be selected based on demonstrated data. This relates to ***attention*** and ***saliency*** which are important concepts in theories for human and animal learning. The term ***shared attention*** refers to a teacher's and a learner's simultaneous attention to the same objects. Scassellati used the Cog platform [80] to investigate shared attention between humans and robots. Saliency refers to the components of the environment that are important for a given task, and it clearly introduces a bias by reducing the size of observation space Y. Breazeal and Scassellati, [18] describe the relationship between attention and saliency and how the concepts can be used to facilitate learning in robotics.

These techniques relate to the psychological term ***scaffolding***, which is used to denote interaction between caretakers and infants in order to reduce distractions, marking a task's important attributes and reducing the number of degrees of freedom in the learning task in general [19, 87]. All these operations aim at simplifying the learning task by introducing bias to the problem definition.

From a formal perspective, bias regarding sensor and action variables may be introduced by moving away from I_{hist} into a new, derived information space I_{der} [53, p.571]. I_{der} is a reformulated or pre-processed version of the information in I_{hist}. The mapping from I_{hist} to I_{der} is denoted κ, and may have an arbitrary shape:

$$\kappa : I_{hist} \rightarrow I_{der}. \tag{12}$$

An element of I_{der} is referred to as a ***derived event history*** η_{der} and can be generated from $\eta \in I_{hist}$ using the mapping κ. Therefore, I_{der} does not serve as a general purpose representational space as I_{hist} does, but rather as a task-specific representation where relevant features become salient, while irrelevant information is not retained. The purpose of I_{der} is similar to the purpose of the state space X. In fact, a state space is one possible instance of I_{der}, but there are numerous other possible derived information spaces that do not aim at representing states in the world.

The LFD process with bias included is illustrated in Figure 3. Various ways to introduce bias regarding the control function π result in a reduced set $\Pi_p \subset \Pi$. The learning function λ maps from the derived information space I_{der} instead of straight from I_{hist}. This extended formulation of LFD is further discussed in Section 3.4.

Referring to Figure 3, the ***what to imitate*** question shows up as a transformation problem from I_{hist} to I_{der}, i.e., an identification of the relevant aspects of the task. Since we are focusing on a self-imitation setting, the correspondence problem is not present here. However, there is still the problem of selecting a controller $\pi_p \subseteq \Pi_p$ based on b_{der}, reflecting the remaining parts of the how to imitate question. When to imitate appears as ensuring that $b \subseteq B$, i.e., that everything in the demonstration set b is actually part of the desired behavior.

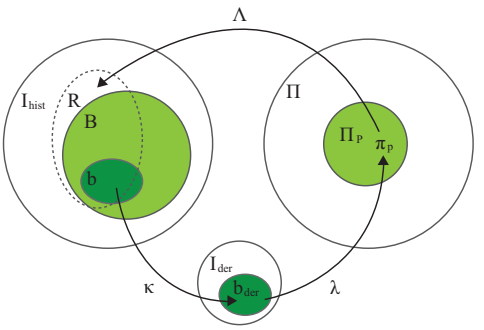

Figure 3. The LFD process with bias introduced. A derived information space I_{der} is introduced as a space where the behavior may be represented in a task-specific way. Training data b is mapped into I_{der} with an information mapping κ. The pre-processed information in I_{der} and various ways to introduce bias in λ result in a reduced set of possible controllers Π_P, illustrated by the light colored area in Π. Compare with Figure 1.

Our discussion about bias has so far been focused on knowledge intentionally introduced into the system to facilitate learning. We like to refer to this kind of information as ***ontological bias***. However, there are also a vast number of restrictions to the problem introduced for other reasons. As mentioned before, selecting a specific type of algorithm to represent π will introduce bias. A particular configuration of the robot's sensors and actuators restricts the ways in which it can solve a particular task. Often the choice of physical platform and software architecture is made for practical reasons rather than for an understanding of ontological implications. We like to phrase these kind of restrictions as ***pragmatical bias***.

Independent of the type of bias being introduced into the system, it limits the behaviors the robot can learn. Consequently bias is not necessarily positive. Instead, one should aim at a suitable level of bias, such that the robot can learn as many interesting behaviors as possible, while still being able to generalize correctly.

As mentioned above, using pre-defined skills or behavior primitives is a common way to define Π_p. The demonstrated data are in such cases used to identify a suitable primitive and may also be used to set parameters for the selected primitive. One way to define such primitives is to associate them with achievement of specific ***goals***. This concept deserves special attention and is analyzed further in the next section.

3.3. Goal

The success or failure to repeat the demonstrated behavior is most often judged by the human demonstrator, and to describe the human intentions we use the word ***goal***. The goal of a behavior is a human concept and can be of two major types [68]:

1. Maintenance goals. A specific condition has to be maintained for a time interval, such as the path-tracking scenario described in Example 1 in Section 3.1.

2. Achievement goals. A specific condition has to be reached, such as the motion to a green cube in Example 2 in Section 3.1.

A behavior B was earlier introduced as a set of event histories that, from a teacher's perspective, fulfills some common purpose. This can be understood as after performing B, specific conditions in the world are satisfied. This is analogous with the common goal formulation from classical AI, where a goal G is a set of states in state space [77]:

$$G \subset X. \tag{13}$$

All the information the agent acquires about G is accumulated over time in $\tilde{y}$ and $\tilde{u}$. Therefore, any goal G which can be measured with the agent's sensors can also be formulated as a set of event histories $\eta \in I_{hist}$:

$$G_I \subset I_{hist}. \tag{14}$$

This should be understood as after observing an $\eta \in G_I$ we know that G is satisfied. A consequence of this formulation is that behaviors and goals are represented in the same way, and since any $\eta \in B$ by definition satisfies the goal of B, G_I and B become identical:

$$G_I = B. \tag{15}$$

This may also be explained from the reversed perspective. When X is viewed as a derived information space, G will cast a pre-image into I_{hist} which per definition will be identical to B. Still, this formulation of goals is not very satisfying. In state space, G most often has an intentional definition, a neat formulation that describes the minimum requirements. However, in the task invariant I_{hist}, a neat goal can not be formulated since no bias has been introduced.
When a human teacher speaks about goals he or she uses task specific information which in principle could be transferred to the robot as bias. This is partly what is done when a state space is defined in classical AI. But the information a human uses to formulate goals may not be necessary for executing the same acts, maybe not even helpful. This argument is nicely illustrated in the ***frame of reference*** [14, 73]. By assuming the necessity for a human goal formulation we impose our own frame of reference upon the agent, and may make representation of the behavior much more complicated than it may be from the agent's perspective.
A common way to introduce this separation between the human's and the robot's frame of reference is to introduce pre-programmed primitives. The set of known primitives creates a space where the human teacher can easily get an understanding of what the robot is doing, while the specific controllers can create local information spaces suitable for the specific primitive. The use of primitives is further developed in the following section.

3.4. Learning with behavior primitives

Based on the concepts of behavior, bias, and goal introduced above, the learning task defined in Equation 10 is here refined. In Section 3.1 it was concluded that λ requires some bias to be able to find a suitable controller, as illustrated in Figure 3. In the most basic form of LFD, λ is simply learned by fitting the demonstrated data to a more or less general functional form, such as a neural network [57] or a rule base framework [42] which in such cases represents the reduced controller set Π_P in Figure 3. The use of primitives, which was introduced in Section 2.1, is fully compatible with this description of learning bias such that learning consists of matching a demonstration with a predefined primitive. This process is denoted ***behavior recognition*** and can be approached in a number of ways as described below.
The description of LFD given above is valid for demonstrations of behaviors that can be repeated by choosing one single primitive. More complex behaviors demand sequences or combinations of primitives. For a given robot and class of learning scenarios, the set of primitives Π_P is normally chosen such that a demonstration may be divided into segments where each segment can be repeated by choosing the right primitive. The general LFD process illustrated in Figure 3 is here extended to include handling of such sequences. Some types of behaviors are better described as combinations of several primitives executed in parallel, e.g. [69]. This organization is common in behavior-based architectures, e.g. [27, 58]. However, recognition of primitives executed in parallel is incredibly complex in the general case. Furthermore, these systems require a coordination function that integrate motor commands from parallel primitives. Due to these issues, parallel primitives are less common in LFD applications and we have therefore chosen to focus on the purely sequential case.
Let us first look from a post learning perspective at how sequence control can be described for a robot using a set Π_P of predefined primitives π_p. To include the assignment of parameters for parameterized primitives into the learning, Π_P is in the following regarded as containing all possible parameterizations of primitives. Control can now be divided into two steps:

1. Action selection where a function π_{sel} selects a primitive $\pi_p \in \Pi_P$:

$$\pi_p = \pi_{sel}(\eta_{der}) \tag{16}$$

where π_{sel} performs the mapping

$$\pi_{sel} : I_{der} \rightarrow \Pi_P \tag{17}$$

$\eta_{der} \in I_{der}$ is a pre-processed or derived version of the original event history $\eta \in I_{hist}$, constructed by an information mapping function κ [53, p.571], defined in Equation 12.

2. Low-level control using the chosen controller π_p to generate an action u_k.

Stepping back to the learning phase, the problem is now reduced to finding the action selection function π_{sel} using demonstrated data b pre-processed with the information mapping κ into the derived information space I_{der} (see Figure 4)[1]. In this way, the dimensionality of the learning problem is drastically reduced since λ is now selecting suitable $\pi_{sel} \in \Pi_{sel}$ based on the pre-processed trajectory information in I_{der} rather than working on the full I_{hist} and Π spaces. Compare with Figures 1 and 3.
While the approaches to sequence learning with primitives vary widely, the process of finding π_{sel} can be divided into three tasks:

1. ***Behavior segmentation*** where a demonstration $\eta^{(i)}$ is divided into smaller segments, referred to as ***task segments***.

2. ***Behavior recognition*** where each segment is associated with a primitive $\pi_p \in \Pi_P$.

[1] *By comparing Equations 16 and 17 with Equations 7 and 8, the primitives π_p may be seen as generalized actions, generated by a controller π_{sel}. Another interesting analogy can be made between action selection and the correspondence problem, i.e., the problem of finding the action(s) that corresponds to an observed event sequence. Viewing the primitives as actions leads to an equivalent problem formulation for action selection; find the primitive that corresponds to an observed event sequence.*

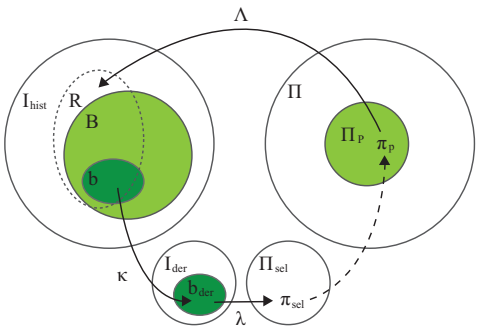

Figure 4. An extended version of the LFD process illustrated in Figure 3. Bias is here introduced into the learning process by restricting Π to a set of primitives Π_P. Primitives π_p are selected by selection function $\pi_{sel} : I_{der} \rightarrow \Pi_P$. Solid lines represent function mappings while the dashed line represents the evaluation of π_{sel}.

3. ***Behavior coordination***, referring to identification of rules or switching conditions for how the primitives are to be combined.

Referring to Figure 4, these tasks are realized by the function λ. In practice, task 1 and 2 are often intertwined. For Task 1, several approaches exist, for example variance thresholding [46, 51], repeated pattern correlation [49, 75, 76], thresholding mean velocity of joints [34, 65] and entropy-based segmentation [25]. Auto-associative neural networks have also been used for segmentation, both by measuring network reconstruction performance [15] and by identifying bifurcations in the network attractor dynamics [49, 50]. Calinon and coworkers [24] used Dynamic Time Warping in combination with Gaussian Mixture Regression to decompose movement trajectories of a humanoid robot.

Task 2 is commonly seen as a classification problem. For example, Bentivegna [8] uses a nearest-neighbor classifier on state data to identify skills in a marble maze and an air hockey game. In both these setups, each primitive is assigned a query point in state space, which is compared with the current system state. Pook and Ballard [74] present an approach where sliding windows of data are classified using Learning Vector Quantization in combination with a k-NN classifier. The complexity of the distance measure is highly dependent on the complexity of B. For simple behaviors, a Euclidean distance function has been shown to work well [9]. However, for more complex behaviors, other measures are necessary. Tani [83, 84] does both recognition of behavior primitives and segmentation with extended recurrent neural networks that model different behavior primitives depending on the parametric bias in the network model. Recognition is done by finding the optimal parametric bias for an observed sensory-motor sequence. Calinon and colleagues use Hidden Markov Models in combination with Principal Component Analysis to compute the likelihood that the observed data was generated by the model [23, 24].

One approach that addresses the complexity of higher level primitives can be found in work by Nicolescu [68], where two behaviors are regarded as being similar if their respective preconditions and goals match, regardless of their internal differences. Nicolescu utilizes the postconditions to recognize primitives in demonstrated data, i.e., task 1 and 2 as described above. Recognized primitives are arranged in a behavior network and during execution the behaviors' preconditions in combination with the network links are used for behavior coordination, Task 3. Formally, any sequence of recognized primitives can be seen as an element in a derived information space I_{der}, and consequently a behavior network, represented as a set of behavior sequences, constitutes a subspace of that I_{der}. In this setting, the definition of postconditions for each primitive constitutes an information mapping κ from I_{hist} to I_{der} and the preconditions take part in the implementation of the coordination function π_{sel}. The primitive controller itself is represented by $\pi_p \in \Pi_p$. Compare with Figure 4.

Demiris and Johnson [31] present a different approach where all primitive controllers are continuously running in parallel, predicting actions in response to incoming sensor data. The prediction errors are then used to estimate how well each primitive represents the demonstrated behavior. This approach is similar to our own method β-***comparison***, which is also used for some primitives in the present example, c.f., Section 4. Even though theoretically appealing and with strong connections to biological findings, see [31] for details, direct comparison of predicted actions become infeasible for complex primitives. The method presented by Demiris and Johnson, as well as our β-***comparison***, has problems capturing the similarity of behaviors that may be executed in many different ways, leading to the same goal. One way to handle these issues is to move from a direct comparison of actions in I_{hist} to more abstract concepts of actions or events in a derived event history $\eta_{der} \in I_{der}$. An evaluation of β-***comparison*** and two other methods for behavior recognition can be found in [15]. In a generalized sense these methods should be seen as an attempt to create a ***metric of imitation performance***, as discussed in Section 3.1.

Sometimes, a demonstrated behavior can not be decomposed into a sequence of known discrete primitives. Several metrics may conflict and cause ambiguities in behavior recognition. In these situations, continuous task representations are preferable since they can better describe a smooth transition from one metric to another, see for instance [24].

A distributed approach to Task 3 is presented by Maes and Brooks [56]. Global feedback is used, allowing the primitives themselves to learn suitable activation conditions by correlating particular stimuli with positive or negative feedback. The feedback functions in combination with the primitives themselves constitute the coordination function π_{sel}. Another approach to behavior coordination is found in the MOSAIC architecture [39, 40, 86]. MOSAIC utilizes forward modes paired with primitive controllers. Each forward model computes a responsibility signal as a measure of how well the paired controller can handle the present situation. When combined with a responsibility predictor this architecture forms a powerful coordination system. MOSAIC is a theoretical framework but the HAMMER architecture [30, 32], which has been implemented and tested on robots, captures many aspects of MOSAIC. Both these architectures are put in relation to LFD in our own recent work [13]. A key aspect of this approach is the pairing of forward models (predictors) and inverse models (controllers) in a model-free way. We are analyzing this issue deeper and propose a possible solution based on the algorithm Predictive Sequence Learning algorithm in other recent publications [16, 17].

There are several approaches to identify relevant aspects of the task that do not employ behavior primitives. While we limit the present review to approaches using primitives, the work by Kulić et al. [52] is worth mentioning even though it does not directly apply behavior primitives. In this approach, demonstrations of movement patterns are encoded in Hidden Markov Models and then clustered into groups using Hierarchical Agglomerative Clustering. Groups are formed incrementally as new demonstrations are added, which makes this approach display many of the advantages with behavior primitives as described here. Furthermore, Kulić et al. put forward the advantage of a hierarchical organization of behavior, a claim we support strongly and discuss deeper in other work [13].

Adding to the motivations presented above, one important reason for the use of primitives in LFD is that primitives constitute high level representations of the demonstrated behavior. Primitives can be labeled in meaningful ways, which helps establish a common understanding between the human teacher and the robot pupil. It is natural for humans to break down sequences of actions into meaningful sections and adults appear to agree upon how segmentation should be made [7]. We therefore believe that identification and recombination of behavior primitives is a critical aspect of LFD.

4. Demonstrator

The concepts and theory introduced above are here illustrated with an experiment in which a Khepera robot [48] is used in an LFD setting. This experimental setup is on purpose simplified to illustrate how ambiguous even a very simple demonstration may be, and how the proposed formalism can be used to describe the LFD process.
The Khepera robot has eight infra-red proximity sensors mounted around the rim of the robot. The limited sensing capabilities have for this experiment been augmented by an external camera mounted above the robot arena. The setup can be seen in Figure 5 and an example image from the top mounted camera can be seen in Figure 6. The robot is equipped with a gripper and is placed in an environment with a number of wood blocks and two colored areas located in one side of the scene.

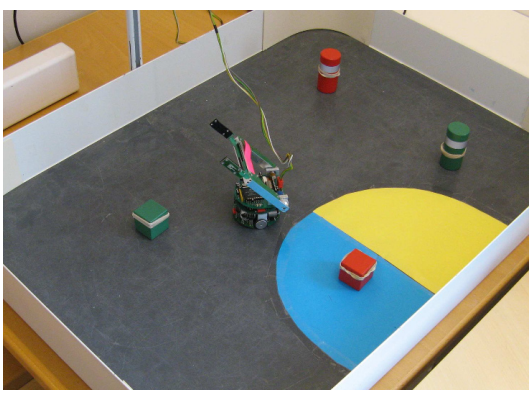

Figure 5. Experimental setup. In the center is a Khepera robot [48] with a gripper that can be raised and lowered. The objects around the scene are painted wood blocks. Rubber bands have been placed around the objects to facilitate gripping. A camera has been mounted directly above the scene, see Figure 6.

The experiment comprises a sequence learning task in which a human intends to teach a robot to pick up cubes and place them in the blue-colored corner area. To demonstrate the wanted behavior, the human tele-operates the robot towards a red cube, grips it, lifts it, moves to the blue area and drops down the cube. The robot should then be able to repeat the demonstrated behavior. The reader is referred to Figure 1 which summarizes much of the discussed formalism.
Observation space Y comprises the camera image (Figure 6), data from the eight proximity sensors, position sensors for gripper and gripper arm and an optical barrier detecting objects in the gripper. *Action space* U comprises the speed of the left and right wheel, and the speeds of the motors controlling gripper lift motion and gripper close motion. Sequences $\tilde{y}_k$ and $\tilde{u}_k$ (Equations 2 and 3) are combined into *history information states* $\eta_k \in I_{hist}$ (Equations 5 and 6). I_{hist} is a huge space comprising all possible sensor and action sequences the robot in principle can experience. Given the task at hand, a more suitable *derived information space* I_{der} is defined. It comprises sequences of the following entities derived from Y and U: Object properties *distance, direction, orientation, type*, and *color* where *type* is either *cube* or *cylinder*. Directions and orientations are given in a coordinate system relative to the robot. Distance and direction to the centroids of the two colored areas are also extracted. Technically, these entities are extracted from the camera image using a combination of image analysis tools, including color segmentation, Sobel edge detection, Hough transform and mathematics morphology. Formally, these techniques are parts of the κ, defined in Equation 12.

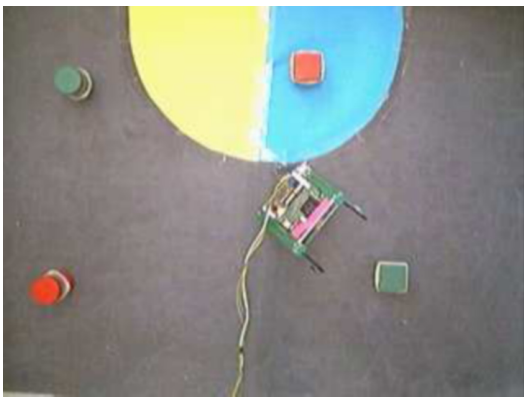

Figure 6. Example image from top mounted camera. A pink tape has been placed on the Khepera gripper to facilitate recognition of the robot's position and orientation.

The generation of I_{der} should be seen as the first of many kinds of *biases* that we introduce in order to make the learning task feasible. This bias depends on the available sensors and actuators and also on the task at hand. It is clear that the dimensionality of the learning problem is significantly reduced by replacing the camera image in I_{hist} by a small number of object properties.
The demonstrator performs the wanted task by tele-operating the robot as described above. The resulting recorded data $b_{der} \subset I_{der}$ is a set of event histories constituting the input to the learning function λ. To support this process, the universe Π of all possible controllers is reduced to a much smaller set Π_p that comprises pre-defined high-level behavior primitives. The following primitives are defined: *move_to_object*, *move_to_area, grip, release, lift* and *put_down*. The *move_to_object* primitive takes two parameters *color* and *type,* where $color = C \subseteq \{red, green, blue, yellow\}$ and $type = T \subseteq \{cube, cylinder\}$. The *move_to_area* primitive takes one argument *color* just like *move_to_object*, but does not have any *type* parameter. One could of course imagine many other possible parameters for the these primitives, e.g. position and size, but the included parameters suffice for the present example. Referring to Section 3.3, each parametrization of the *move_to_object* and *move_to_area* primitives is associated with a specific goal G_l (Equation 14). As been already

mentioned, this is a very efficient way of introducing additional bias in learning such that complex behaviors can be learned by few or even a single demonstration. Conceptually, Π_p comprises all possible parameterizations of the primitives.

To keep the example simple, all primitives are hard-coded into the robot, i.e., both I_{der} and Π_p are defined manually. However, in a realistic setting primitives are often created during a previous learning phase, as has been shown in for example the work by Saunders et al. [70, 79]. The use of primitives should be seen as a way to reuse knowledge that may come either from a programmer manually designing the primitive, or from a previous learning phase. Formally, this is described as a gradual redefinition of I_{der} and Π_p which corresponds to the concept of scaffolding described above.

To learn a sequence of these primitives, the three steps described in Section 3.4 are performed. Behavior segmentation and recognition are executed in one step by continuously matching each primitive against b_{der}. The recognition method differs between different primitives. For *grip, release, lift* and *put_down,* the start and end positions of the gripper are used to *indicate that* the corresponding primitive has been executed. For *move_to_object* and *move_to_area* the behavior recognition method $\beta - comparison$ [15] is used. In this approach, an action vector for each parameterized primitive is computed, creating a set of hypothesis. Each action vector is then compared to the observed b_{der} creating an error measure for each hypothesis. If the error remains low while the robot is approaching a specific target object, the hypotheses is confirmed and a *move_to* primitive with the corresponding parametrization can be inserted into the recognized sequence.

Each primitive specifies a set of finish conditions, e.g. that the robot should be within gripping range of a target object for *move_to_object* to complete. The end result of the learning process λ is a function $\pi_{sel} \in \Pi_{sel}$ that selects an appropriate primitive $\pi_p \in \Pi_p$ given the current event history η_{der} (Equation 16). In this way, π_{sel} acts as a sequencer and the actual control of the robot and gripper motion is done by the currently selected primitive π_p.

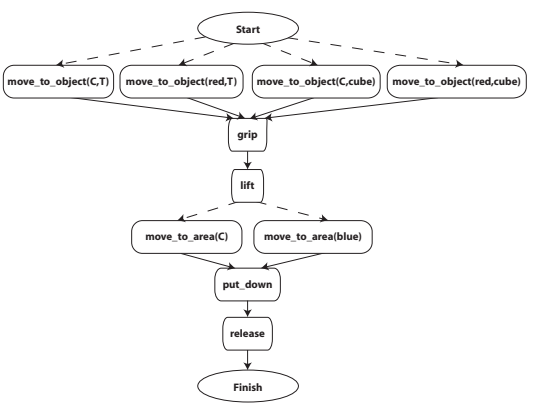

Figure 7. A schematic of the demonstrated sequence going vertically from top to bottom, where each square represents the execution of a primitive. Alternative interpretations of the demonstrated sequence are drawn horizontally, with the most general interpretation to the left and the most specific to the right. C and T are unspecified attributes representing all, or a subset of, possible values for color and type, respectively. Dashed lines mark ambiguous steps in the sequence, that require further information.

Even with the bias introduced by the construction of I_{der} and by the pre-defined behavior primitives in Π_p a substantial uncertainty, similar to the one discussed in Section 3.1, remains. This is illustrated in Figure 7 where alternative interpretations at each step are drawn horizontally and time flows vertically from top to bottom. In the shown example, the first part of the demonstration may be interpreted in four ways; *move_to_object(C,T), move_to_object(red,T), move_to_object(C,cube), move_to(red,cube)*. The second and third primitives *grip* and *lift* are uniquely identified while *move_to_area* is subject to similar ambiguity as *move_to_object*. Finally, the primitives *put_down* and *release* are uniquely identified. The alternatives for each ambiguity represent generalizations along different feature axes. In Section 3.1 this is described as interpolation or extrapolation of the demonstrated event histories η. Figure 7 illustrates a subspace of Π_{sel}. With the ambiguities resolved, through human feedback or other kinds of bias, the resulting sequence represents an instance of $\pi_{sel} \in \Pi_{sel}$ as defined in Equation 17.

Various types of feedback from the human can be applied such that the ambiguous sequence collapses into a single well defined sequence of behavior primitives that will enable repetition of the demonstrated behavior according to the user's intentions. In the described experiment, the human teacher manually selects the appropriate alternatives in a dialog system such that the generated π_{sel} will execute the sequence ž*move_to_object(C,cube), grip, lift, move_to_area(blue), put_down, release*~. The robot is then able to repeat the intended sequence of primitives and autonomously move cubes of any color to the blue area.

To sum up, the present example demonstrates how the huge and complex I_{hist} can be transformed into a significantly smaller and more human interpretable space I_{der}. On the controller side, the set of all possible controllers Π have been reduced by introducing a set of primitives Π_p that can be composed into sequences by Π_{sel}, c.f. Figure 4. A more detailed description of the experimental setup will be presented in future work, including a graphical interface in which the human user is able to give feedback during and after a demonstration, in order to resolve ambiguities such as the one illustrated in Figure 7.

5. Summary

A formalism for robot behaviors and *Learning from Demonstration* (LFD) is presented. Building on terminology from LaValle [53, ch.11], an agent's sensory-motor history is conveniently described by an event history, and a controller maps event histories to actions in action space. As illustrated in Figure 1, a demonstration of a particular behavior can be seen as an event history $\eta \in b$, and the behavior itself as the large set B of allowed event histories, i.e., all possible ways to realize the desired behavior. The quality of the learned controller can be judged by the similarity between B and the realization space R.

The vague and ill-posed meaning of *repeating* a demonstrated behavior is discussed from a machine learning perspective. The concept of *generalization* is defined in the framework of event histories and leads to a discussion of bias in learning. In LFD, bias is essential and can be introduced before, during, and after demonstration as feedback from the human teacher. The huge information history space may be reduced to a derived space, suitable for a limited set of tasks. Behavior primitives are another common way to introduce bias, and are often associated with specific *goals,* which are explicitly or implicitly defined for each primitive. LFD can at a higher level be described as controller selection. In this context, learning consists of finding and tuning a suitable primitive. More complex behaviors can be created by combining several primitives into sequences. LFD can then be described in three

steps, ***behavior segmentation, behavior recognition*** and ***behavior coordination.***

When using primitives created during a previous learning phase, learning can be seen as an evolutionary process where new knowledge is gained through the use of previous knowledge as bias. Formally, this is described as a gradual redefinition of I_{der}, Π_p and Π_{sel} which relates to the concept of scaffolding.

The formalism is applied to a sequence learning task in which the introduced concepts are illustrated with focus on how bias of various kinds can be used to enable learning from a single demonstration. The experiment shows how even a simple demonstration contains almost unavoidable ambiguities that have to be handled one way or another. In context of the presented formalism, these ambiguities appear clearly as a transition problem from behavior $B \subset I_{hist}$ to controller $\pi \in \Pi$, or as a controller selection problem in Π_{sel}. This research problem is believed to be crucial for the development of learning robots and is addressed in our ongoing research.

The presented work is an attempt to structure and formalize general principles and assumptions in LFD. Our aim is not to present the single best way to talk about behaviors, generalization, goals, and other LFD related concepts. Rather, we want to point out the importance of defining these concepts clearly. It is our hope that the presented work will provide useful insights to the mechanisms involved in LFD and thus contribute to further development of this powerful and promising area of robot learning.

Acknowledgments

The authors would like to thank Lars-Erik Janlert for many valuable comments on this paper, and Steven LaValle for inspiring discussions about information spaces.

Table 1. Symbol index

Symbol	Description	Definition
X	State space	Sec. 2.1
Y	Observation space	Sec. 2.2
$\tilde{Y}$	Observation history space	Sec. 2.3
$\tilde{y} \in \tilde{Y}$	Observation history	Eq. 2
U	Action space	Sec. 2.2
$\tilde{U}$	Action history space	Sec. 2.3
$\tilde{u} \in \tilde{U}$	Action history state	Eq. 3
h	Sensor function	Sec. 2.2
f	Action function	Sec. 2.2
Π	Controller space	Sec. 3
$\pi \in \Pi$	Controller	Eq. 1, Eq. 7
I	Information space	Sec. 2.3
$e \in I$	Sensory-motor event	Sec. 2.3
I_k	History information space (up to stage k)	Eq. 5
$\eta \in I_{hist}$	History information state	Eq. 4
$\eta_0 \in I_{hist}$	Initial conditions	Sec. 2.3
I_{hist}	Information history space (unbound)	Eq. 6
$B \subset I_{hist}$	Behavior	Sec. 2.5
$b \subset B$	Demonstration	Sec. 3
$R \subset I_{hist}$	Realization space	Sec. 3
λ	Learning function	Eq. 10
Λ	Realization function	Eq. 11
I_{der}	Derived information space	Sec. 3.2
$G \subset X$	Goal (defined in X)	Sec. 3.3
$G_I \subset I_{hist}$	Goal (defined in I_{hist})	Sec. 3.3
Π_p	Set of primitive controllers	Sec. 3.4
$\pi_p \in \Pi_p$	Primitive controller	Eq. 16
π_{sel}	Controller selection function	Eq. 17
κ	Information mapping	Eq. 12

References

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Imitation with ALICE: learning to imitate corresponding actions across dissimilar embodiments. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 32:482–496, 2002.

[2] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Action, state and effect metrics for robot imitation. In 15th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN 2006), pages 232–237, Hatfield, September 2006.

[3] R. Amit and M. Matarić. Parametric primitives for motor representation and control. In Int. Conf. on Robotics and Automation (ICRA), Washington DC, May 2002.

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

[5] R. C. Arkin. Behaviour-Based Robotics. MIT Press, 1998.

[6] P. Bakker and Y. Kuniyoshi. Robot see, robot do: an overview of robot imitation. In Proceedings of the AISB Workshop on Learning in Robots and Animals, pages 3–11, Brighton, 1996.

[7] D. Baldwin, A. Andersson, J. Saffran, and M. Meyer. Segmenting dynamic human action via statistical structure. Cognition, 106(3): 1382–1407, March 2008.

[8] D. C. Bentivegna. Learning from Observation using Primitives. PhD thesis, College of Computing, Georgia Institute of Technology, 2004.

[9] D. C. Bentivegna, C. G. Atkeson, and G. Cheng. Learning similar tasks from observation and practice. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2677–2683, Beijing, China, October 2006.

[10] A. Billard, Y. Epars, G. Cheng, and S. Schaal. Discovering imitation strategies through categorization of multi-dimensional data. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, volume 3, pages 2398–2403 vol.3, 2003.

[11] A. Billard, Y. Epars, S. Calinon, S. Schaal, and G. Cheng. Discovering optimal imitation strategies. Robotics and Autonomous Systems, 47(2-3):69–77, June 2004.

[12] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, Handbook of Robotics. Springer, 2008.

[13] E. A. Billing. Cognition Reversed - Robot Learning from Demonstration. PhD thesis, Umeå University, Department of Computing Science, Umeå, Sweden, December 2009.

[14] E. A. Billing. Cognitive perspectives on robot behavior. In J. Filipe, A. Fred, and B. Sharp, editors, Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART), Special

Session LAMAS, pages 373–382, Valencia, Spain, January 2010.
[15] E. A. Billing and T. Hellström. Behavior recognition for segmentation of demonstrated tasks. In IEEE SMC International Conference on Distributed Human-Machine Systems, pages 228 – 234, Athens, Greece, March 2008.
[16] E. A. Billing, T. Hellström, and L. E. Janlert. Model-free learning from demonstration. In J. Filipe, A. Fred, and B. Sharp, editors, Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART), pages 62–71, Valencia, Spain, January 2010.
[17] E. A. Billing, T. Hellström, and L. E. Janlert. Behavior recognition for learning from demonstration. In Proceedings of IEEE International Conference on Robotics and Automation, Anchorage, Alaska, May 2010.
[18] C. Breazeal and B. Scassellati. Challanges in building robots that imitate people. In K. Dautenhahn and C. L. Nehahiv, editors, Imitation in Animals and Artifacts. MIT Press, 2002.
[19] C. Breazeal and B. Scassellati. Infant-like social interactions between a robot and a human caretaker. Adaptive Behavior, 8(1): 49–74, 1998.
[20] C. Breazeal and B. Scassellati. Robots that imitate humans. Trends in Cognitive Sciences, 6(11):481–487, November 2002.
[21] R. A. Brooks. New approaches to robotics. Science, 253(13): 1227–1232, 1991.
[22] R. W. Byrne and A. E. Russon. Learning by imitation: a hierarchical approach. The Journal of Behavioral and Brain Sciences, 16(3), 1998.
[23] S. Calinon and A. Billard. Recognition and reproduction of gestures using a probabilistic framework combining PCA, ICA and HMM. In Proceedings of the 22nd international conference on Machine learning, pages 105–112, Bonn, Germany, 2005. ACM.
[24] S. Calinon, F. Guenter, and A. Billard. 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.
[25] P. Cohen, N. Adams, and H. B. Voting experts: An unsupervised algorithm for segmenting. Intelligent Data Analysis, 11(6):607–625, 2007.
[26] A. Cypher, editor. Watch What I Do: Programming by Demonstration. MIT Press, 1993.
[27] T. S. Dahl. Behavior-Based Learning. PhD thesis, Faculty of Engineering, University of Bristol, UK, 2002.
[28] N. Delson and H. West. Robot programming by human demonstration: The use of human inconsistency in improving 3D robot trajectories. In Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems '94. Advanced Robotic Systems and the Real World, IROS '94., volume 2, pages 1248–1255, Munich, Germany, September 1994.
[29] J. Demiris and G. Hayes. Do robots ape? In Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents, pages 28–31, 1997.
[30] Y. Demiris and A. Dearden. From motor babbling to hierarchical learning by imitation: a robot developmental pathway. In Proceedings of the 5th International Workshop on Epigenetic Robotics, pages 31—37, 2005.
[31] Y. Demiris and M. Johnson. Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. Connection Science, 15(4):231–243, 2003.
[32] Y. Demiris and B. Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. Robotics and Autonomous Systems, 54(5):361–369, May 2006.
[33] R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification (2nd Edition). Wiley-Interscience, 2001.
[34] A. Fod, M. Matarič, and O. C. Jenkins. Automated derivation of primitives for movement classification. Autonomous Robots, pages 39–54, 2002.
[35] C. Giovannangeli and P. Gaussier. Human-Robot interactions as a cognitive catalyst for the learning of behavioral attractors. In 16th IEEE International Symposium on Robot and Human interactive Communication (RO-MAN 2007), pages 1028–1033, August 2007.
[36] S. F. Giszter, F. A. Mussa-Ivaldi, and E. Bizzi. Convergent force field organized in the frog's spinal cord. Journal of Neuroscience, 13(2):467–491, 1993.
[37] J. G. Greeno. Special issue on situated action. In Cognitive Science, volume 17, pages 1–147. Ablex Publishing Corporation, Norwood, New Jersey, 1993.
[38] F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. RSJ Advanced Robotics, Special Issue on Imitative Robots, 21(13): 1521–1544, 2007.
[39] M. Haruno, D. M. Wolpert, and M. M. Kawato. MOSAIC model for sensorimotor learning and control. Neural Comput., 13(10): 2201–2220, 2001.
[40] M. Haruno, D. M. Wolpert, and M. Kawato. Hierarchical MOSAIC for movement generation. In International Congress Series 1250, pages 575– 590. Elsevier Science B.V., 2003.
[41] T. Hastie, R. Tibshirani, and J. H Friedman. The Elements of Statistical Learning. Springer, August 2001.
[42] T. Hellström. Teaching a robot to behave like a cockroach. In Proceedings of the Third International Symposium on Imitation in Animals and Artifacts in Hatfield UK, pages 54–61, 2005.
[43] T. Hellström, T. Johansson, and O. Ringdahl. Development of an autonomous forest machine for path tracking. In P. Corke and S. Sukkariah, editors, Field and Service Robotics - Results of the 5th International Conference FSR, volume 25 of Springer Tracts in Advanced Robotics, pages 603–614. Springer, 2006.
[44] M. Hersch, F. Guenter, S. Calinon, and A. Billard. Dynamical system modulation for robot learning via kinesthetic demonstrations. Proceedings of IEEE Transactions on Robotics, 24(6): 1463–1467, 2008.
[45] E. Hutchins. Cognition in the Wild. MIT Press, Cambridge, Massachusetts, 1995.
[46] R. A. Peters II and C. L. Campbell. Robonaut task learning through teleoperation. In Proceedings of the 2003 IEEE, International Conference on Robotics and Automation, pages 23–27, Taipei, Taiwan, September 2003.
[47] L. E. Janlert. Modeling change - the frame problem. In Z. W. Pylyshyn, editor, The Robot's Dilemma, pages 1 – 41. Ablex Publishing, Norwood, New Jersey, 1987.
[48] K-Team. Khepera robot. http://www.k-team.com, 2007.
[49] H. Kadone and Y. Nakamura. Segmentation, memorization, recognition and abstraction of humanoid motions based on correlations and associative memory. In Proceedings of the 6th IEEE-RAS International Conference on Humanoid Robots, pages 1–6, University of Genova, Genova, Italy, 2006.
[50] H. Kadone and Y. Nakamura. Symbolic memory for humanoid robots using hierarchical bifurcations of attractors in nonmonotonic neural networks. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 2900–2905, Edmonton, AB, Canada, 2005.
[51] N. Koenig and M. J. Matarič. Behavior-Based segmentation of demonstrated tasks. In International Conference on Development and Learning (ICDL), Bloomington, USA, May 2006.
[52] D Kulič, W Takano, and Y Nakamura. Incremental learning, clus-

tering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. The International Journal of Robotics Research, 27(7):761–784, July 2008.
[53] S. M. LaValle. Planning Algorithms. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.
[54] H. Lieberman, editor. Your Wish is My Command: Programming by Example. Morgan Kaufmann, San Francisco, 2001.
[55] L. Ljung. System Identification. Prentice-Hall, Simon & Schuster, Englewood Cliffs, New Jersey, 1987.
[56] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In National Conference on Artificial Intelligence (AAAI), pages 796–802, 1990.
[57] P. Martin and U. Nehmzow. Programming by teaching: Neural network control in the manchester mobile robot. In Proc. Intelligent Autonomous Vehicles, Helsinki. Springer Verlag, 1995.
[58] M. J. Mataric̀. Behavior-Based control: Examples from navigation, learning, and group behavior. Journal of Experimental and Theoretical Artificial Intelligence, 9(2–3):323–336, 1997.
[59] M. J. Mataric̀. Designing and understanding adaptive group behavior. Adaptive Behavior, 4(1):51–80, 1995.
[60] M. J. Mataric̀. Integration of representation into Goal-Driven Behavior-Based robots. In IEEE Transactions on Robotics and Automation, volume 8, pages 304–312, 1992.
[61] M. J. Mataric̀ and M. J. Marjanovic. Synthesizing complex behaviors by composing simple primitives. In Proceedings of the European Conference on Artificial Life (ECAL-93), volume 2, pages 698–707, Brussels, Belgium, May 1993.
[62] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, Machine Intelligence 4, pages 463–502. Edinburgh University Press, 1969.
[63] T. M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers Computer Science Department Technical Report, New Brunswick, New Jersey, 1980.
[64] F. A. Mussa-Ivaldi and S. F. Giszter. Vector field approximation: a computational paradigm for motor control and learning. Biological cybernetics, 67:479–489, 1992.
[65] S. Nakaoka, A. Nakazawa, K. Yokoi, and K. Ikeuchi. Recognition and generation of leg primitive motions for dance imitation by a humanoid robot. In Proceedings of 2nd International Symposium on Adaptive Motion of Animals and Machines, Kyoto, Japan, 2003.
[66] C. L. Nehaniv and K. Dautenhahn. The correspondence problem. In K. Dautenhahn and C. L. Nehahiv, editors, Imitation in Animals and Artifacts. MIT Press, 2002.
[67] C. L. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, Learning Robots: An Interdisciplinary Approach, volume 24, pages 136–161. World Scientific Press, 2000.
[68] M. Nicolescu. A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains. PhD thesis, University of Southern California, 2003.
[69] A. Olenderski, M. Nicolescu, and S. Louis. Robot learning by demonstration using forward models of Schema-Based behaviors. In Proceedings of International Conference on Informatics in Control, Automation and Robotics, Barcelona, Spain, 2005.
[70] N. Otero, J. Saunders, K. Dautenhahn, and C. L. Nehaniv. Teaching robot companions: the role of scaffolding and event structuring. Connection Science, 20:111–134, June 2008.
[71] J. Peters and S. Schaal. Policy learning for motor skills. In Proceedings of 14th International Conference on Neural Information Processing (ICONIP 2007), pages 1–10, Berlin, Germany, November 2007. Springer.
[72] R. Pfeifer and C. Scheier. Sensory-motor coordination: the metaphor and beyond. Robotics and Autonomous Systems, 20 (2):157–178, June 1997.
[73] R. Pfeifer and C. Scheier. Understanding Intelligence. MIT Press. Cambrage, Massachusetts, 2001.
[74] P. K. Pook and D. H. Ballard. Recognizing teleoperated manipulations. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 578–585, 1993.
[75] B. Rohrer and S. Hulet. BECCA - a brain emulating cognition and control architecture. Technical report, Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA, 2006.
[76] B. Rohrer and S. Hulet. A learning and control approach based on the human neuromotor system. In Proceedings of Biomedical Robotics and Biomechatronics, BioRob, 2006.
[77] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, NJ, 1995.
[78] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. Using Self-Imitation to direct learning. In 15th IEEE International Symposium on Robot and Human Interactive Communication, pages 244–250, 2006.
[79] J. Saunders, C. L. Nehaniv, K. Dautenhahn, and A. Alissandrakis. Self-Imitation and environmental scaffolding for robot teaching. International Journal of Advanced Robotics Systems, 4(1):109–124, 2007.
[80] B. Scassellati. Imitation and mechanisms of joint attention: A developmental structure for building social skills on a humanoid robot. Lecture Notes in Computer Science, 1562:176–195, 1999.
[81] H. A. Simon. The Sciences of the Artificial. MIT Press, Cambridge, Massachusetts, 1969.
[82] L. A. Suchman. Plans and Situated Actions. PhD thesis, Intelligent Systems Laboratory, Xerox Palo Alto Research Center, USA, 1987.
[83] J. Tani. On the interactions between top-down anticipation and bottom-up regression. Frontiers in Neurorobotics, 1:2, 2007.
[84] J. Tani and M. Ito. Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment. IEEE Trans. on Systems, Man, and Cybernetics Part A: Systems and Humans, 33(4):481–488, 2003.
[85] D. H. Wolpert and W. G Macready. No free lunch theorems for optimization. In IEEE Transactions on Evolutionary Computation, volume 1, pages 67–82, April 1997.
[86] D. M. Wolpert. A unifying computational framework for motor control and social interaction. Phil. Trans. R. Soc. Lond., B(358):593–602, March 2003.
[87] D. Wood, J. Bruner, and G. Ross. The role of tutoring in problem solving. Journal of Child Psychology and Psychiatry, 17:89–100, 1976.

IV

Paper IV

Predictive Learning from Demonstration*

Erik Billing, Thomas Hellström, and Lars-Erik Janlert

Dept. Computing Science, Umeå University, SE-901 87 Umeå, Sweden
billing@cs.umu.se, thomash@cs.umu.se, and lej@cs.umu.se
www.cs.umu.se/research/robotics

Abstract: A model-free learning algorithm called *Predictive Sequence Learning (PSL)* is presented and evaluated in a robot *Learning from Demonstration (LFD)* setting. PSL is inspired by several functional models of the brain. It constructs sequences of predictable sensory-motor patterns, without relying on predefined higher-level concepts. The algorithm is demonstrated on a Khepera II robot in four different tasks. During training, PSL generates a hypothesis library from demonstrated data. The library is then used to control the robot by continually predicting the next action, based on the sequence of passed sensor and motor events. In this way, the robot reproduces the demonstrated behavior. PSL is able to successfully learn and repeat three elementary tasks, but is unable to repeat a fourth, composed behavior. The results indicate that PSL is suitable for learning problems up to a certain complexity, while higher level coordination is required for learning more complex behaviors.

Predictive Learning from Demonstration

Erik A. Billing, Thomas Hellström, and Lars-Erik Janlert

Department of Computing Science, Umeå University, 901 87 Umeå, Sweden
{billing,thomash,lej}@cs.umu.se

Abstract. A model-free learning algorithm called Predictive Sequence Learning (PSL) is presented and evaluated in a robot Learning from Demonstration (LFD) setting. PSL is inspired by several functional models of the brain. It constructs sequences of predictable sensory-motor patterns, without relying on predefined higher-level concepts. The algorithm is demonstrated on a Khepera II robot in four different tasks. During training, PSL generates a hypothesis library from demonstrated data. The library is then used to control the robot by continually predicting the next action, based on the sequence of passed sensor and motor events. In this way, the robot reproduces the demonstrated behavior. PSL is able to successfully learn and repeat three elementary tasks, but is unable to repeat a fourth, composed behavior. The results indicate that PSL is suitable for learning problems up to a certain complexity, while higher level coordination is required for learning more complex behaviors.

1 Introduction

Recent years have witnessed an increased interest in computational mechanisms that will allow robots to *Learn from Demonstrations (LFD)*. With this approach, also referred to as *Imitation Learning*, the robot learns a behavior from a set of good examples, *demonstrations*. The field has identified a number of key problems, commonly formulated as *what to imitate*, *how to imitate*, *when to imitate* and *who to imitate* [3]. In the present work, we focus on the first question, referring to which aspects of the demonstration should be learned and repeated.

Inspiration is taken from several functional models of the brain and prediction is exploited as a way to learn state definitions. A novel learning algorithm, called *Predictive Sequence Learning (PSL)*, is here presented and evaluated. PSL is inspired by *S-Learning* [42, 43], which has previously been applied to robot learning problems as a model-free reinforcement learning algorithm [40, 41].

The paper is organized as follows. In Sect. 2 a theoretical background and biological motivation is given. Section 3 gives a detailed description of the proposed algorithm. Section 4 describes the experimental setup and results for evaluation of the algorithm. In Sect. 5, conclusions, limitations and future work are discussed.

2 Motivation

One common approach to identify what in a demonstration that is to be imitated is to exploit the variability in several demonstrations of the same behavior. Invariants among

J. Filipe, A. Fred, and B. Sharp (Eds.): ICAART 2010, CCIS 129, pp. 186–200, 2011.

the demonstrations are seen as the most relevant and selected as essential components of the task [3, 17]. Several methods for discovering invariants in demonstrations can be found in the LFD literature. One method presented by Billard et al. applies a time-delayed neural network for extraction of relevant features from a manipulation task [4, 5]. A more recent approach uses demonstrations to impose constraints in a dynamical system, e.g. [16, 25].

While this is a suitable method for many types of tasks, there are also applications where it is less obvious which aspects of a behavior should be invariant, or if the relevant aspects of that behavior is captured by the invariants. Since there is no universal method to determine whether two demonstrations should be seen as manifestations of the same behavior or two different behaviors [10], it is in most LFD applications up to the teacher to decide. However, the teacher's grouping of actions into behaviors may not be useful for the robot. In the well known imitation framework by Nehaniv and Dautenhahn [34], it is emphasized that the success of an imitation is observer dependent. The consequence of observer dependence when it comes to interpreting sequences of actions has been further illustrated with Pfeifer and Scheier's argument about the *frame of reference* [35, 36], and is also reflected in Simon's parable with the ant [45]. A longer discussion related to these issues can be found in [6].

Pfeifer and Scheier promotes the use of a *low level specification* [36], and specifically the *sensory-motor space* $I = U \times Y$, where U and Y denotes the *action space* and *observation space*, respectively. Representations created directly in I prevents the robot from having memory, which has obvious limitations. However, systems with no or very limited memory capabilities has still reached great success within the robotics community through the works by Rodney Brooks, e.g., [12–15], and the development of the *reactive* and *behavior based* control paradigms, e.g., [1]. By extending the definition of I such that it captures a certain amount of temporal structure, the memory limitation can be removed. Such a temporally extended sensory-motor space is denoted *history information space* $I^\tau = I_0 \times I_1 \times I_2 \times \ldots \times I_\tau$, where τ denotes the temporal extension of I *[10]*. With a large enough τ, I^τ can model any behavior. However, a large τ leads to an explosion of the number of possible states, and the robot has to generalize such that it can act even though the present state has not appeared during training.

In the present work, we present a learning method that is not based on finding invariants among several demonstrations of, what the teacher understands to be "the same behavior". Taking inspiration from recent models of the brain where prediction plays a central role, e.g. [22, 23, 27, 32], we approach the question of what to imitate by the use of prediction.

2.1 Functional Models of Cortex

During the last two decades a growing body of research has proposed computational models that aim to capture different aspects of human brain function, specifically the cortex. This research includes models of perception, e.g., Riesenhuber and Poggio's hierarchical model [38] which has inspired several more recent perceptual models [23, 32, 37], models of motor control [26, 42, 46–48] and learning [22]. In 2004, this field reached a larger audience with the release of Jeff Hawkins's book On Intelligence [28].

With the ambition to present a unified theory of the brain, the book describes cortex as a hierarchical memory system and promotes the idea of a *common cortical algorithm*. Hawkins's theory of cortical function, referred to as the *Memory-Prediction framework*, describes the brain as a prediction system. Intelligence is, in this view, more about applying memories in order to predict the future, than it is about computing a response to a stimulus.

A core issue related to the idea of a common cortical algorithm is what sort of bias the brain uses. One answer is that the body has a large number of reward systems. These systems are activated when we eat, laugh or make love, activities that through evolution have proved to be important for survival. However, these reward systems are not enough. The brain also needs to store the knowledge of how to activate these reward systems.

In this context, prediction appears to be critical for learning. The ability to predict the future allows the agent to foresee the consequences of its actions and in the long term how to reach a certain goal. However, prediction also plays an even more fundamental role by providing information about how well a certain model of the world correlates with reality.

This argument is supported not only by Hawkins's work, but by a large body of research investigating the computational aspects of the brain [8]. It has been proposed that the central nervous system (CNS) simulates aspects of the sensorimotor loop [29, 31, 33, 47]. This involves a modular view of the CNS, where each module implements one *forward model* and one *inverse model*. The forward model predicts the sensory consequences of a motor command, while the inverse model calculates the motor command that, in the current state, leads to the goal [46]. Each module works under a certain *context* or bias, i.e., assumptions about the world which are necessary for the module's actions to be successful. One purpose of the forward model is to create an estimate of how well the present situation corresponds to these assumptions. If the prediction error is low the situation is familiar. However, if the prediction error is high, the situation does not correspond to the module's context and actions produced by the inverse model may be inappropriate.

These findings have inspired recent research on robot perception and control. One example is the *rehearse, predict, observe, reinforce* decomposition proposed by [18, 20, 44] which adapts the view of perception and action as two aspects of a single process. Hierarchical representations following this decomposition have also been tested in an LFD setting [19] where the robot successfully learns sequences of actions from observation. In work parallel to this, we also investigates PSL as an algorithm for behavior recognition [11], exploring the possibilities to use PSL both as a forward and an inverse model. The present work should be seen as a further investigation of these theories applied to robots, with focus to learning with minimal bias.

2.2 Sequence Learning

PSL is inspired by *S-Learning*, a dynamic temporal difference (TD) algorithm presented by Rohrer and Hulet, [42, 43]. S-Learning builds sequences of passed events which may be used to predict future events, and in contrast to most other TD algorithms it can base its predictions on many previous states.

S-Learning can be seen as a variable order Markov model (VMM) and we have observed that it is very similar to the well known compression algorithm LZ78 [49]. This coincidence is not that surprising considering the close relationship between lossless compression and prediction [2]. In principle, any lossless compression algorithm could be used for prediction, and vice verse [21].

S-Learning was originally developed to capture the discrete episodic properties observed in many types of human motor behavior [39]. Inspiration is taken from the *Hierarchical Temporal Memory* algorithm [24], with focus on introducing as few assumptions into learning as possible. More recently, it has been applied as a model-free reinforcement learning algorithm for both simulated and physical robots [40, 41]. We have also evaluated S-Learning as an algorithm for behavior recognition [9]. However, to our knowledge it has never been used as a control algorithm for LFD.

The model-free design of S-Learning, together with its focus on sequential data and its connections to human motor control makes S-Learning very interesting for further investigation as a method for robot learning. With the ambition to increase the focus on prediction, and propose a model that automatically can detect when it is consistent with the world, PSL was designed.

3 Predictive Sequence Learning

PSL is trained on an *event sequence* $\eta = (e_1, e_2, \ldots, e_t)$, where each *event* e is a member of an alphabet $\sum$. η is defined up to the current time t from where the next event e_{t+1} is to be predicted.

PSL stores its knowledge as a set of hypotheses, known as a *hypothesis library* H. A *hypothesis* $h \in H$ expresses a dependence between an event sequence $X = (e_{t-n}, e_{t-n+1}, \ldots, e_t)$ and a target event $I = e_{t+1}$:

$$h : X \Rightarrow I \tag{1}$$

X_h is referred to as the *body* of h and I_h denotes the *head*. Each h is associated with a *confidence* c reflecting the conditional probability $P(I|X)$. For a given η, c is defined as $c(X \Rightarrow I) = s(X, I)/s(X)$, where the *support* $s(X)$ describes the proportion of transactions in η that contains X and (X, I) denotes the concatenation of X, and I. A transaction is defined as a sub-sequence of the same size as X. The length of h, denoted $|h|$, is defined as the number of elements in X_h. Hypotheses are also referred to as *states*, since a hypothesis of length $|h|$ corresponds to VMM state of order $|h|$.

3.1 Detailed Description of PSL

Let the library H be an empty set of hypotheses. During learning, described in Alg. 1, PSL tries to predict the future event e_{t+1}, based on the observed event sequence η.

If it fails to predict the future state, a new hypothesis h_{new} is created and added to H. h_{new} is one element longer than the longest matching hypothesis previously existing in H. In this way, PSL learns only when it fails to predict.

For example, consider the event sequence $\eta = ABCCABCCA$. Let $t = 1$. PSL will search for a hypothesis with a body matching A. Initially H is empty and consequently

Algorithm 1. Predictive Sequence Learning (PSL)

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_n)$

1: $t \leftarrow 1$
2: $H \leftarrow \emptyset$
3: $M \leftarrow \{h \in H \mid X_h = (e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t)\}$
4: **if** $M = \emptyset$ **then**
5: **let** $h_{new} : (e_t) \Rightarrow e_{t+1}$
6: **add** h_{new} to H
7: **goto** 20
8: **end if**
9: $\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ for\ all\ h' \in M\}$
10: **let** $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ for\ all\ h' \in \hat{M}\right\}$
11: **if** $e_{t+1} \neq I_{h_{max}}$ **then**
12: **let** h_c be the longest hypothesis $\{h \in M \mid I_h = e_{t+1}\}$
13: **if** $h_c = null$ **then**
14: **let** $h_{new} : (e_t) \Rightarrow e_{t+1}$
15: **else**
16: **let** $h_{new} : (e_{t-|h_c|}, e_{t-|h_c|+1}, \ldots, e_t) \Rightarrow e_{t+1}$
17: **end if**
18: **add** h_{new} to H
19: **end if**
20: update the confidence for h_{max} and $h_{correct}$ as described in Sect. 3
21: $t \leftarrow t + 1$
22: **if** $t < n$ **then**
23: **goto** 2
24: **end if**

PSL will create a new hypothesis $(A) \Rightarrow B$ which is added to H. The same procedure will be executed at $t = 2$ and $t = 3$ so that $H = \{(A) \Rightarrow B; (B) \Rightarrow C; (C) \Rightarrow C\}$. At $t = 4$, PSL will find a matching hypothesis $h_{max} : (C) \Rightarrow C$ producing the wrong prediction C. Consequently, a new hypothesis $(C) \Rightarrow A$ is added to H. The predictions at $t = 5$ and $t = 6$ will be successful while $h : (C) \Rightarrow A$ will be selected at $t = 7$ and produce the wrong prediction. As a consequence, PSL will create a new hypothesis $h_{new} : (B, C) \Rightarrow C$. Source code from the implementation used in the present work is available online [7].

3.2 Making Predictions

After, or during, learning, PSL can be used to make predictions based on the sequence of passed events $\eta = (e_1, e_2, \ldots, e_t)$. Since PSL continuously makes predictions during learning, this procedure is very similar to the learning algorithm (Alg. 1). The prediction procedure is described in Alg. 2.

For prediction of a suite of future events, $\hat{e}_t$ can be added to η to create η'. Then repeat the procedure described in Alg. 2 using η' as event history.

Algorithm 2. Making predictions using PSL

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_{t-1})$
Require: the trained library $H = \left(h_1, h_2, \ldots, h_{|H|}\right)$

1: $M \leftarrow \left\{h \in H \mid X_h = \left(e_{t-|h|}, e_{t-|h|+1}, \ldots, e_{t-1}\right)\right\}$
2: $\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ \ for\ all\ h' \in M\}$
3: **let** $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ \ for\ all\ h' \in \hat{M}\right\}$
4: **return** the prediction $\hat{e}_t = I_{h_{max}}$

3.3 Differences and Similarities between PSL and S-Learning

Like PSL, S-Learning is trained on an *event sequence* η. However, S-Learning does not produce hypotheses. Instead, knowledge is represented as *Sequences* ϕ, stored in a *sequence library* κ [43]. ϕ does not describe a relation between a body and a head, like hypotheses do. Instead, ϕ describes a plain sequence of elements $e \in \eta$. During learning, sequences are "grown" each time a matching pattern for that sequence appears in the training data. Common patterns in η produce long sequences in κ. When S-Learning is used to predict the next event, the beginning of each $\phi \in \kappa$ is matched to the end of η. The sequence producing the longest match is selected as a winner, and the end of the winning sequence is used to predict future events.

One problem with this approach, observed during our previous work with S-Learning [9], is that new, longer sequences, are created even though the existing sequence already has Markov property, meaning that it can predict the next element optimally. To prevent the model from getting unreasonably large, S-Learning implements a maximum sequence length m. As a result, κ becomes unnecessarily large, even when m is relatively low. More importantly, by setting the maximum sequence length m, a task-dependent modeling parameter is introduced, which may limit S-Learning's ability to model η.

PSL was designed to alleviate the problems with S-Learning. Since PSL learns only when it fails to predict, it is less prune to be overtrained and can employ an unlimited maximum sequence length without exploding the library size.

4 Evaluation

The PSL algorithm was tested on a Khepera II miniature robot [30]. In the first evaluation (Sect. 4.1), the performance of PSL on a playful LFD task is demonstrated. In a second experiment (Sect. 4.2), the prediction performance during training of PSL is compared to the performance of S-Learning, using recorded sensor and motor data from the robot. During both experiments, the robot is given limited sensing abilities using only its eight infrared proximity sensors mounted around its sides.

One important issue, promoted both by Rohrer et al. [40, 41] and ourselves [10], is the ability to learn even with limited prior knowledge of what is to be learned. Prior knowledge is information intentionally introduced into the system to support learning, often referred to as *ontological bias* or *design bias* [10]. Examples of common design biases are pre-defined state specifications, pre-processing of sensor data, the size of a

neural network or the length of a temporal window. While design biases help in learning, they also limit the range of behaviors a robot can learn. A system implementing large amounts of design bias will to a larger extent base its decisions not on its own experience, but on knowledge of the programmer designing the learning algorithm, making it hard to determine what the system has actually learned.

In addition to design bias, there are many limitations and constraints introduced by other means, e.g., by the size and shape of the robot including its sensing and action capabilities, structure of the environment and performance limitations of the computer used. These kinds of limitations are referred to as *pragmatical bias* [10]. We generally try to limit the amount of ontological bias, while pragmatical bias should be exploited by the learning algorithm to find useful patterns.

In the present experiments, the robot has no previous knowledge about its surroundings or itself. The only obvious design bias is the thresholding of proximity sensors into three levels, *far*, *medium* and *close,* corresponding to distances of a few centimeters. This thresholding was introduced to decrease the size of the observation space Y, limiting the amount of training required. An *observation* $y \in Y$ is defined as the combination of the eight proximity sensors, producing a total of 3^8 possible observations.

An *action* $u \in U$ is defined as the combination of the speed commands sent to the two motors. The Khepera II robot has 256 possible speeds for each wheel, producing an action space U of 256^2 possible actions. However, only a small fraction of these were used during demonstration.

The event sequence is built up by alternating sensor and action events, $\eta = (u_1, y_1, u_2, y_2 \ldots, u_k, y_k)$. k is here used to denote the current stage, rather than the current position in η denoted by t. Even though events is categorized into observations and actions, PSL makes no distinction between these two types of events. From the perspective of the algorithm, all events $e_t \in \sum$ are discrete entities with no predefined relations, where $\sum = Y \cup U$.

In each stage k, PSL is used to predict the next event, given η. Since the last element of η is an observation, PSL will predict an action $u_k \in U$, leading to the observation $y_k \in Y$. u_k and y_k are appended to η, transforming stage k to $k + 1$. This alternating use of observations and actions was adopted from S-Learning [42]. A stage frequency of 10 Hz was used, producing one observation and one action every 0.1 seconds.

4.1 Demonstration and Repetition

To evaluate the performance of PSL on an LFD problem, four tasks are defined and demonstrated using the Khepera II robot. *Task 1* involves the robot moving forward in a corridor approaching an object (cylindrical wood block). When the robot gets close to the object, it should stop and wait for the human teacher to "load" the object, i.e., place it upon the robot. After loading, the robot turns around and goes back along the corridor. *Task 2* involves general corridor driving, taking turns in the right way without hitting the walls and so on. *Task 3* constitutes the "unloading" procedure, where the robot stops in a corner and waits for the teacher to remove the object and place it to the right of the robot. Then the robot turns and pushes the cylinder straight forward for about 10 centimeters, backs away and turns to go for another object. *Task 4* is the combination of the three previous tasks. The sequence of actions expected by the robot

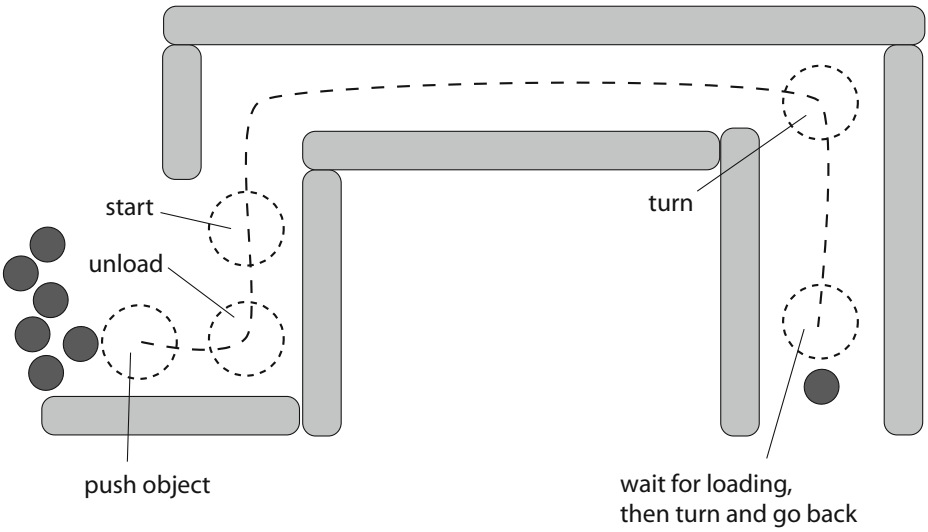

Fig. 1. Schematic overview of the composed behavior (*Task 4*). Light gray rectangles mark walls, dark gray circles mark the objects and dashed circles mark a number of key positions for the robot. See text for details.

is illustrated in Fig. 1. The robot starts by driving upwards in the figure, following the dashed line. until it reaches the object at the loading position. After loading, the robot turns around and follows the dashed line back until it reaches the unload position. When the cylinder has been unloaded (placed to the left of the robot), the robot turns and pushes the object. Finally, it backs away from the pile and awaits further instructions. The experimental setup can be seen in Fig. 2. Even though the setup was roughly the same in all experiments, the starting positions and exact placement of the walls varied between demonstration and repetition.

All tasks capture a certain amount of temporal structure. One example is the turning after loading the object in Task 1. Exactly the same pattern of sensor and motor data will appear before, as well as after, turning. However, two different sequences of actions is expected. Specifically, after the teacher has taken the cylinder to place it on the robot, only the sensors on the robot's sides are activated. The same sensor pattern appears directly after the robot has completed the 180 degree turn, before it starts to move back along the corridor. Furthermore, the teacher does not act instantly. After placing the object on the robot, one or two seconds passed before the teacher issued a turning command, making it more difficult for the learning algorithm to find the connection between the events. Even Task 2 which is often seen as a typical reactive behavior is, due to the heavy thresholding of sensor data, temporally demanding. Even longer temporal structures can be found in Task 3, where the robot must push the object and remember for how long the object is to be pushed. This distance was not controlled in any way, making different demonstrations of the same task containing slightly conflicting data.

After training, the robot was able to repeat Task 1, 2 and 3 successfully. For Task 1, seven demonstrations were used for a total of about 2.6 min. Task 2 was demonstrated

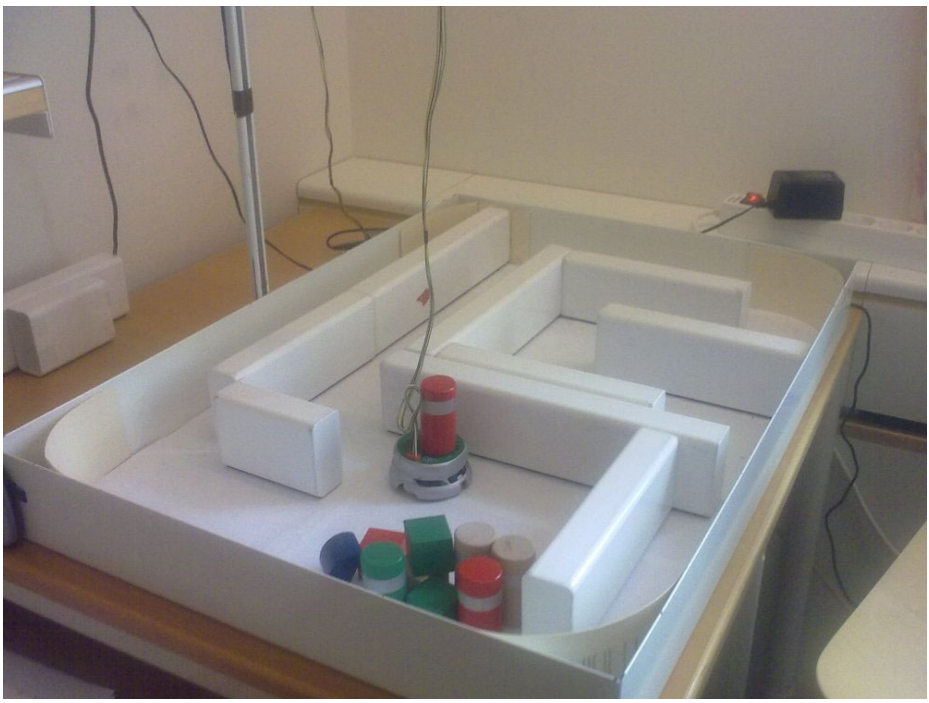

Fig. 2. Experimental setup

for about 8.7 min and Task 3 was demonstrated nine times, in total 4.6 min. The robot made occasional mistakes in all three tasks, reaching situations where it had no training data. In these situations it sometimes needed help to be able to complete the task. However, the number of mistakes clearly decreased with increased training, and mistakes made by the teacher during training often helped the robot to recover from mistakes during repetition.

For Task 4, the demonstrations from all three partial tasks were used, plus a single 2 min demonstration of the entire Task 4. Even after extensive training, resulting in almost 40 000 hypotheses in library, the robot was unable to repeat the complete behavior without frequent mistakes. Knowledge from the different sub-tasks was clearly interfering, causing the robot to stop and wait for unloading when it was supposed to turn, turning when it was supposed to follow the wall and so on. Detailed results for all four tasks can be found in Table 1.

PSL was trained until it could predict about 98% of the demonstrated data correctly. It would be possible to train it until it reproduces all events correctly, but this takes time and initial experiments showed that it did not affect the imitation performance significantly.

4.2 Comparison between S-Learning and PSL

In Sect. 3.3, a number of motivations for the design of PSL were given, in relation to S-Learning. One such motivation was the ability to learn and increase the model size only

Table 1. Detailed statistics on the four evaluation tasks. Training events is the number of sensor and motor events in demonstrated data. Lib. size is the number of hypotheses in library after training. Avg. $|h|$ is the average hypothesis length after training.

Task	Training events	Library size	Avg. $\|h\|$
Task 1	3102	4049	9.81
Task 2	10419	30517	16
Task 3	5518	8797	11
Task 4	26476	38029	15

when necessary. S-Learning always learns and creates new sequences for all common events, while PSL only learns when prediction fails. However, it should be pointed out that even though S-Learning never stops to learn unless an explicit limit on sequence length is introduced, it quickly reduces the rate at which new sequences are created in domains where it already has extensive knowledge.

To evaluate the effect of these differences between PSL and S-Learning, prediction performance and library size were measured during training in three test cases. *Case 1* contained a demonstration of the loading procedure (*Task 1*) used in the LFD evaluation, Sect. 4.1. During the demonstration, the procedure was repeated seven times for a total of about 150 seconds (3000 sensor and motor events). *Case 2* encapsulated the whole composed behavior (*Task 4*) used in LFD evaluation. The behavior was demonstrated once for 120 seconds (2400 events). *Case 3* constituted 200 seconds of synthetic data, describing a 0.1 Hz sinus wave discretized with a temporal resolution of 20 Hz and an amplitude resolution of 0.1 (resulting in 20 discrete levels). The 4000 elements long data sequence created a clean repetitive pattern with minor fluctuations due to sampling variations.

In addition to PSL and S-Learning, a first order Markov model (*1MM*) was included in the tests. The Markov model can obviously not learn the pattern in any of the three test cases perfectly, since there is no direct mapping $e_t \Rightarrow e_{t+1}$ for many events. Hence, the performance of 1MM should be seen only as reference results.

The results from the three test cases can be seen in Fig. 3. The upper part of each plot show accumulated training error over the demonstration while lower parts show model growth (number of hypotheses in library). Since the Markov model does not have a library, the number of edges in the Markov graph is shown, which best corresponds to sequences or hypotheses in S-Learning and PSL, respectively.

5 Description

A novel robot learning algorithm called *Predictive Sequence Learning (PSL)* is presented and evaluated in an LFD setting. PSL is both parameter-free and model-free in the sense that no ontological information about the robot or conditions in the world is pre-defined in the system. Instead, PSL creates a state space (hypothesis library) in order to predict the demonstrated data optimally. This state space can thereafter be used to control the robot such that it repeats the demonstrated behavior.

In contrast to many other LFD algorithms, PSL does not build representations from invariants among several demonstrations that a human teacher considers to be

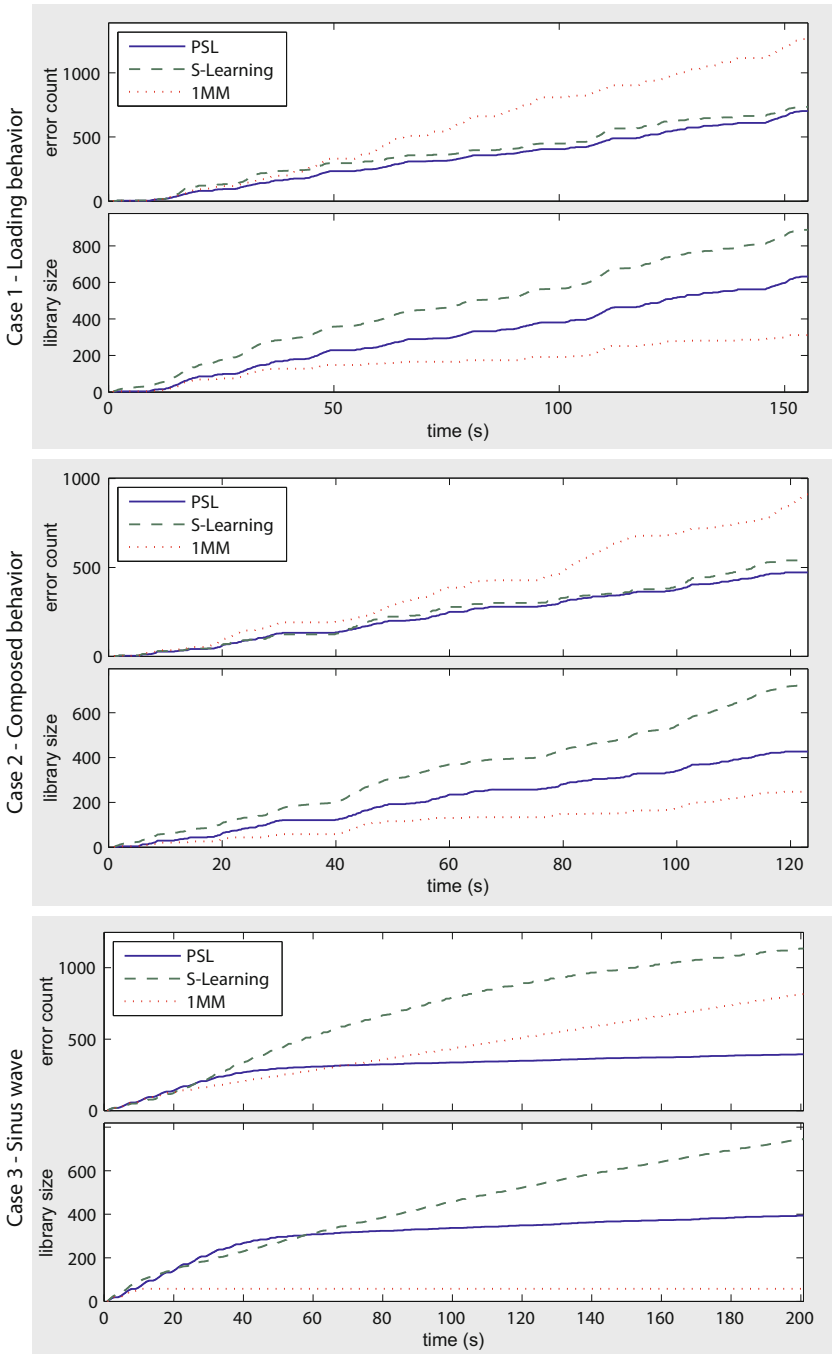

Fig. 3. Training results for all three test cases. See Sect. 4.2 for details.

"the same behavior". All knowledge, from one or several demonstrations, is stored as hypotheses in the library. PSL treats inconsistencies in these demonstrations by generating longer hypotheses that will allow it to make the correct predictions. In this way, the ambiguous definitions of *behavior* is avoided and control is seen purely as a prediction problem.

In the prediction performance comparison, PSL produces significantly smaller libraries than S-Learning on all three data sets. The difference is particularly large in Case 3 (Fig. 3), where both algorithms learn to predict the data almost perfectly. In this situation, S-Learning continues to create new sequences, while PSL does not.

In Case 3, PSL also shows the clearly fastest learning rates (least accumulated errors). The reason can be found in that PSL learns on each event where it fails to predict, while S-Learning learns based on sequence length. When the model grows, S-Learning decreases its learning rate even though the performance is still low. In contrast, the learning rate of PSL is always proportional to performance, which can also be seen in the plots for all three test cases (Fig. 3). However, even though PSL commits less accumulated errors than S-Learning in all three tests, the performance difference in Case 1 and 2 is small and how these results generalize to other kinds of data is still an open question.

In the demonstration-repetition evaluation, tasks 1, 2 and 3 were repeated correctly. Even though the robot made occasional mistakes, the imitation performance clearly increased with more demonstrations. However, in Task 4, which was a combination of the three first tasks, an opposite pattern could be observed. Despite the fact that PSL was still able to predict demonstrated data almost perfectly, knowledge from the three elementary tasks clearly interfered. The reason for this interference is that Task 4 requires much longer temporal dynamics than any of the elementary tasks did when learned separately.

One example of how this knowledge interference is manifested is the turning versus unloading. When the robot approaches the position marked as *turn* in Fig. 1, coming from the left and is supposed to take a right turn, it no longer sees the right wall behind it. Consequently, the situation looks identical to that of unloading. When the robot is to unload, it goes downward in Fig. 1 (position *unload*) but instead of turning it must wait for the cylinder to be placed to its right side. To make the right prediction, PSL has to base its decision on information relatively far back in the event history. Even though PSL has no problem to build a sufficiently large model from training data, the large temporal window produces a combinatorial explosion and the chance of the right patterns reappearing during repetition is small. As a result, PSL decreases the temporal window (i.e., uses shorter hypotheses), and the two situations become inseparable.

5.1 Conclusions and Future Work

The results show that the proposed algorithm is feasible for LFD problems up to a certain complexity. PSL implements very few assumptions of what is to be learned and is therefore likely to be applicable to a wide range of problems.

However, PSL also shows clear limitations when the learning problem increases and longer temporal dynamics is required. PSL is subject to combinatorial explosion and the amount of required training data increases exponentially with problem complexity.

In these situations, some higher-level coordination is clearly necessary. One possible solution is to place PSL as a module in a hierarchical system. PSL learns both to predict sensor data as a response to action (forward model) and to select actions based on the current state (inverse model). In the present work, PSL is viewed purely as a controller and the forward model is consequently not considered. However, in work parallel to this, we show that PSL can also be used as an algorithm for behavior recognition [11], i.e., as a predictor of sensor values. A big advantage of using PSL for both control and behavior recognition is that the forward and inverse computations are in fact based on the same model, i.e., the PSL library. This approach has several theoretical connections to the view of human perception and control as two heavily intertwined processes, as discussed in Section 2.1.

The present work should be seen as one step towards a hierarchical control architecture that can learn and coordinate itself, based on the PSL algorithm. The model-free design of PSL introduces very few assumptions into learning, and should constitute a good basis for many types of learning and control problems. Integrating PSL as both forward and inverse model to achieve a two-layer modular control system, is the next step in this process and will be part of our future work.

Acknowledgements

We would like to thank Brandon Rohrer at Sandia National Laboratories and Christian Balkenius at Lund University for valuable input to this work.

References

1. Arkin, R.C.: Behaviour-Based Robotics. MIT Press, Cambridge (1998)
2. Begleiter, R., Yona, G.: On prediction using variable order markov models. Journal of Artificial Intelligence Research 22, 385–421 (2004)
3. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: Robot programming by demonstration. In: Siciliano, B., Khatib, O. (eds.) Handbook of Robotics. Springer, Heidelberg (2008)
4. Billard, A., Epars, Y., Cheng, G., Schaal, S.: Discovering imitation strategies through categorization of multi-dimensional data. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2398–2403 (2003)
5. Billard, A., Mataric, M.J.: Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. Robotics and Autonomous Systems 37(2-3), 145–160 (2001)
6. Billing, E.A.: Representing behavior - distributed theories in a context of robotics. Technical report, UMINF 0725, Department of Computing Science, Ume University (2007)
7. Billing, E.A.: Cognition reversed (2009), `http://www.cognitionreversed.com`
8. Billing, E.A.: Cognition Reversed - Robot Learning from Demonstration. PhD thesis, Ume University, Department of Computing Science, Ume, Sweden (December 2009)
9. Billing, E.A., Hellström, T.: Behavior recognition for segmentation of demonstrated tasks. In: IEEE SMC International Conference on Distributed Human-Machine Systems, Athens, Greece, pp. 228–234 (March 2008)
10. Billing, E.A., Hellström, T.: A formalism for learning from demonstration. Paladyn: Journal of Behavioral Robotics 1(1), 1–13 (2010)

11. Billing, E.A., Hellström, T., Janlert, L.E.: Behavior recognition for learning from demonstration. In: Proceedings of IEEE International Conference on Robotics and Automation, Anchorage, Alaska (May 2010)
12. Brooks, R.A.: A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation RA-2 1, 14–23 (1986)
13. Brooks, R.A.: Elephants don't play chess. Robotics and Autonomous Systems 6, 3–15 (1990)
14. Brooks, R.A.: Intelligence without reason. In: Proceedings of 1991 Int. Joint Conf. on Artificial Intelligence, pp. 569–595 (1991)
15. Brooks, R.A.: New approaches to robotics. Science 253(13), 1227–1232 (1991)
16. 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)
17. Delson, N., West, H.: Robot programming by human demonstration: The use of human inconsistency in improving 3D robot trajectories. In: Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994. Advanced Robotic Systems and the Real World, IROS 1994, Munich, Germany, vol. 2, pp. 1248–1255 (September 1994)
18. Demiris, J., Hayes, G.R.: Imitation as a dual-route process featuring predictive and learning components: a biologically plausible computational model. In: Imitation in Animals and Artifacts, pp. 327–361. MIT Press, Cambridge (2002)
19. Demiris, Y., Johnson, M.: Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. Connection Science 15(4), 231–243 (2003)
20. Demiris, Y., Simmons, G.: Perceiving the unusual: Temporal properties of hierarchical motor representations for action perception. Neural Networks 19(3), 272–284 (2006)
21. Feder, M., Merhav, N.: Relations between entropy and error probability. IEEE Transactions on Information Theory 40(1), 259–266 (1994)
22. Friston, K.J.: Learning and inference in the brain. Neural Networks: The Official Journal of the International Neural Network Society 16(9), 1325–1352 (2003) PMID: 14622888
23. George, D.: How the Brain might work: A Hierarchical and Temporal Model for Learning and Recognition. PhD thesis, Stanford University, Department of Electrical Engineering (2008)
24. George, D., Hawkins, J.: A hierarchical bayesian model of invariant pattern recognition in the visual cortex. In: Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 2005), vol. 3, pp. 1812–1817 (2005)
25. 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)
26. Haruno, M., Wolpert, D.M., Kawato, M.: Hierarchical MOSAIC for movement generation. In: International Congress Series, vol. 1250, pp. 575–590. Elsevier Science B.V., Amsterdam (2003)
27. Haruno, M., Wolpert, D.M., Kawato, M.M.: MOSAIC model for sensorimotor learning and control. Neural Comput. 13(10), 2201–2220 (2001)
28. Hawkins, J., Blakeslee, S.: On Intelligence. Times Books (2002)
29. Jordan, M., Rumelhart, D.: Forward models: Supervised learning with a distal teacher. Cognitive Science: A Multidisciplinary Journal 16(3), 307–354 (1992)
30. K-Team. Khepera robot (2007), `http://www.k-team.com`
31. Kawato, M., Furukawa, K., Suzuki, R.: A hierarchical neural-network model for control and learning of voluntary movement. Biological Cybernetics 57(3), 169–185 (1987) PMID: 3676355
32. Lee, T.S., Mumford, D.: Hierarchical bayesian inference in the visual cortex. J. Opt. Soc. Am. A Opt. Image Sci. Vis. 20(7), 1434–1448 (2003)

33. Miall, R.C., Wolpert, D.M.: Forward models for physiological motor control. Neural Netw. 9(8), 1265–1279 (1996)
34. Nehaniv, C.L., Dautenhahn, K.: Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In: Demiris, J., Birk, A. (eds.) Learning Robots: An Interdisciplinary Approach, vol. 24, pp. 136–161. World Scientific Press, Singapore (2000)
35. Pfeifer, R., Scheier, C.: Sensory-motor coordination: the metaphor and beyond. Robotics and Autonomous Systems 20(2), 157–178 (1997)
36. Pfeifer, R., Scheier, C.: Understanding Intelligence. MIT Press, Cambridge (2001)
37. Poggio, T., Bizzi, E.: Generalization in vision and motor control. Nature 431(7010), 768–774 (2004)
38. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. Nature Neuroscience 2(11), 1019–1025 (1999) PMID: 10526343
39. Rohrer, B.: S-Learning: a biomimetic algorithm for learning, memory, and control in robots. In: CNE apos;07. 3rd International IEEE/EMBS Conference on Natural Engineering, Kohala Coast, Hawaii, pp. 148–151 (2007)
40. Rohrer, B.: S-learning: A model-free, case-based algorithm for robot learning and control. In: Eighth International Conference on Case-Based Reasoning, Seattle Washington (2009)
41. Rohrer, B., Bernard, M., Morrow, J.D., Rothganger, F., Xavier, P.: Model-free learning and control in a mobile robot. In: Fifth International Conference on Natural Computation, Tianjin, China (2009)
42. Rohrer, B., Hulet, S.: BECCA - a brain emulating cognition and control architecture. Technical report, Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA (2006)
43. Rohrer, B., Hulet, S.: A learning and control approach based on the human neuromotor system. In: Proceedings of Biomedical Robotics and Biomechatronics, BioRob (2006)
44. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philosophical Transactions of the Royal Society B: Biological Sciences 358(1431), 537–547 (2003) PMC1693137
45. Simon, H.A.: The Sciences of the Artificial. MIT Press, Cambridge (1969)
46. Wolpert, D.M.: A unifying computational framework for motor control and social interaction. Phil. Trans. R. Soc. Lond. B(358), 593–602 (2003)
47. Wolpert, D.M., Flanagan, J.R.: Motor prediction. Current Biology: CB 11(18), 729–732 (2001)
48. Wolpert, D.M., Ghahramani, Z.: Computational principles of movement neuroscience. Nature Neuroscience 3, 1212–1217 (2000)
49. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory 24(5), 530–536 (1978)

V

Paper V

Behavior Recognition for Learning from Demonstration*

Erik Billing, Thomas Hellström, and Lars-Erik Janlert

Dept. Computing Science, Umeå University, SE-901 87 Umeå, Sweden
billing@cs.umu.se, thomash@cs.umu.se, and lej@cs.umu.se
www.cs.umu.se/research/robotics

Abstract: Two methods for behavior recognition are presented and evaluated. Both methods are based on the dynamic temporal difference algorithm *Predictive Sequence Learning (PSL)* which has previously been proposed as a learning algorithm for robot control. One strength of the proposed recognition methods is that the model PSL builds to recognize behaviors is identical to that used for control, implying that the controller (inverse model) and the recognition algorithm (forward model) can be implemented as two aspects of the same model. The two proposed methods, *PSLE-Comparison* and *PSLH-Comparison*, are evaluated in a Learning from Demonstration setting, where each algorithm should recognize a known skill in a demonstration performed via teleoperation. PSLH-Comparison produced the smallest recognition error. The results indicate that PSLH-Comparison could be a suitable algorithm for integration in a hierarchical control system consistent with recent models of human perception and motor control.

Keywords: Learning and Adaptive Systems, Neurorobotics, Autonomous Agents

2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Behavior Recognition for Learning from Demonstration

Erik A. Billing
Department of Computing Science
Umeå University
Umeå, Sweden
Email: billing@cs.umu.se

Thomas Hellström
Department of Computing Science
Umeå University
Umeå, Sweden
Email: thomash@cs.umu.se

Lars-Erik Janlert
Department of Computing Science
Umeå University
Umeå, Sweden
Email: lej@cs.umu.se

***Abstract*—Two methods for behavior recognition are presented and evaluated. Both methods are based on the dynamic temporal difference algorithm *Predictive Sequence Learning (PSL)* which has previously been proposed as a learning algorithm for robot control. One strength of the proposed recognition methods is that the model PSL builds to recognize behaviors is identical to that used for control, implying that the controller (inverse model) and the recognition algorithm (forward model) can be implemented as two aspects of the same model. The two proposed methods, *PSLE-Comparison* and *PSLH-Comparison*, are evaluated in a Learning from Demonstration setting, where each algorithm should recognize a known skill in a demonstration performed via teleoperation. PSLH-Comparison produced the smallest recognition error. The results indicate that PSLH-Comparison could be a suitable algorithm for integration in a hierarchical control system consistent with recent models of human perception and motor control.**

***Index Terms*—Learning and Adaptive Systems, Neurorobotics, Autonomous Agents**

I. Introduction

In previous work [1], we present the dynamic temporal difference algorithm *Predictive Sequence Learning (PSL)* and apply it to a *Learning from Demonstration (LFD)* problem. In this application, PSL builds a model from a set of demonstrations, i.e., sequences of sensor and motor events recorded while a human teacher performs the desired task by teleoperating the robot. After training, PSL can be used to control the robot by continually predicting the next action based on the sequence of passed sensor and motor events.

PSL has many interesting properties seen as a learning algorithm for robots. It is model and parameter free, meaning that it introduces very few assumptions into learning and does not need any task specific configuration. Knowledge is stored in a *hypothesis library* H, where each *hypothesis* $h \in H$ describes a relation between a sequence of events $X = \left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)$ and a target event $Y = e_{t+1}$. $|h|$ denotes the length of h as the number of elements in X. PSL treats control as a prediction problem, and creates longer h when it fails to predict the next event. This corresponds to a dynamically growing state space similar to a *Variable order Markov Model (VMM)*. Hypotheses are only created when predictions fail, meaning that the learning rate is proportional to the prediction error and PSL will stop to learn in domains where it can predict future events perfectly.

Our evaluation of PSL indicates that the algorithm is suitable for learning problems up to a certain complexity. However, PSL is subject to combinatorial explosion and fails to reproduce more complex behavior properly. Specifically, PSL has problems capturing long-term variations within a behavior and some kind of higher level coordination is clearly necessary in these situations.

The design of PSL is inspired by several computational models of the human brain, MOSAIC [2], [3], [4], Predictive Coding [5], [6], [7], and Hierarchical Temporal Memory [8], [9]. These models propose a hierarchical organization of perception and control. Specifically, MOSAIC presents a modular view of central nervous system, where each module implements one *forward model* and one *inverse model*. The forward model predicts the sensory consequences as a result of a motor command, while the inverse model calculates the motor command that, in the current state, leads to the goal [4]. Each module works under a certain *context,* or bias, provided by higher ordinate modules in the hierarchy. One purpose of the forward model is to create a responsibility signal λ_β representing a measure of how well the present activity corresponds to the module's context. If the prediction error is small, the activity is familiar and λ_β is high. However, if the prediction error is large, the activity does not correspond to the module's context, and actions produced by the inverse model may be inappropriate. An overview of these approaches to intertwined control and perception for LFD is found in our previous work [10].

Placing PSL as a module in this kind of hierarchical structure could constitute one way to solve the problems with combinatorial explosion. In such an architecture, each PSL module would work under a certain context and only model the system variables that change quickly, while slower temporal dynamics are handled higher up in the hierarchy. However, this requires not only that PSL is useful as an inverse model, but also that it can constitute a forward model, able to compute λ_β. On the way to propose a fully developed hierarchical system based on the PSL algorithm, the present work proposes two ways of applying PSL as a forward model

and evaluates how well each approach is able to recognize a certain activity, a problem that within LFD is known as *behavior recognition*.

One of the few robot control architectures that employs this kind of organization is HAMMER [11], [12], [13], which focuses on direction of attention during action recognition. While HAMMER is in many respects further developed that the work presented here, it implements hard-coded forward models paired with inverse models. In the present work, both forward and inverse models are generated from demonstrated data in a model-free way.

The rest of this paper is organized as follows. Section II presents a short background to behavior recognition and introduces some of our earlier research relevant for the present work. Section III gives a detailed description of PSL, which is developed into the proposed methods for behavior recognition, presented in Section IV. Experimental setup and results from the conducted evaluation is presented in Section V. Finally, conclusions, limitations and future work are discussed in Section VI.

II. Behavior recognition for LFD

Recent work in LFD is often concerned with identification and selection of *behavior primitives,* or *skills,* which can be seen as simple controllers and correspond to larger parts of the demonstration [14]. Behavior primitives implement hard-coded or previously learned behaviors that the robot can execute. By matching these primitives with a demonstration, selected primitives can be compiled into a new, more complex, controller that will be able to repeat the demonstrated behavior under varying environmental conditions [15], [14].

This approach transforms the general LFD process into the three activities of *behavior segmentation*, *behavior recognition* and *behavior coordination* [16]. Behavior segmentation refers to the process of dividing the observed event sequence into segments which can be explained by a single primitive. Behavior recognition is the process of matching each segment with a primitive. Finally, behavior coordination involves identifying switching criteria that control when the robot should switch between different primitives. Identification of switching criteria corresponds to finding sub-goals in the demonstrated behavior.

The behavior recognition problem is closely related to the creation of a *metric of imitation performance* which is a common concept in the literature on LFD and imitation learning, e.g. [17], [18], [15]. The metric acts as a cost function for imitation of a skill and is in this sense very similar to the computation of a responsibility signal. Identification of a metric of imitation performance is often focused on finding the critical components of a skill by identifying invariants within a set of demonstrations. One promising approach to construct such a metric is to use the demonstrations to impose constraints in a dynamical system [19], [20]. However, we take an alternative approach: Using forward models to compute a measure of how well observed events correspond to respective controller. Both behavior recognition algorithms presented here compute λ_β as a direct or indirect function of prediction error. We believe that this approach has larger potential to provide a generalizable solution to the behavior recognition problem and it also has many interesting connections to neurological models. A longer discussion of these issues can be found in [21].

Another important aspect of the metric of imitation performance is to solve the correspondence problem, i.e., comparing actions when the body of the teacher is different from that of the student. This problem does not exist in LFD using teleoperation and is not considered in the present work.

There are many possible ways to demonstrate new behavior to a robot. Good overviews can be found in [15], [17]. In the present work, we focus on demonstrations performed by controlling the robot via teleoperation such that it performs the desired behavior. In this case, a demonstration is a sequence of sensor and motor events $\eta = (u_1, y_1, u_2, y_2 \ldots, u_\kappa, y_\kappa)$, where κ denotes the most recent stage. An *observation* $y_k \in Y$ is defined as the combination of all sensors readings and an *action* $u_k \in U$ is defined as the combination of the motor commands sent to the robot. The *observation space* Y and *action space* U constitute the complete set of possible events, known as an *event alphabet* $\sum = Y \cup U$. A *stage* (u_k, y_k) comprises one action and the directly following observation. In some situations we do not distinguish between observations and actions and define an event sequence $\eta = (e_1, e_2, \ldots, e_t)$ as a sequence of discrete events $e \in \sum$ up to the current time t.

In earlier work, we have developed and evaluated three methods for behavior recognition [1]. The focus of the work was to propose methods for constructing several interpretations from a single sequence of events, using the set of known skills. Seen as a pure classification problem, one skill would have to be selected as the one best representing that segment in η. However, in our methods, all recognition algorithms produces *activity level* λ_β for each skill β. The activity level is in this context identical to the notion of a *responsibility signal* λ_β, as discussed in the introduction.

While one of the evaluated recognition techniques showed clear limitations, the other two, known as *AANN-Comparison* and *S-Comparison*, showed promising results. AANN-Comparison is based on a set of Auto-Associative Neural Networks, one for each skill β. The network's reconstruction error for each stage (u_k, y_k) from the demonstration η is used as a measure of $\lambda_\beta(k)$.

S-Comparison is based on S-Learning, a prediction-based control algorithm inspired by the human neuro-motor system [22], [23]. S-Learning is a dynamic *temporal difference* (TD) algorithm able to extract temporal patterns ρ in presented data. S-Comparison computes a similarity measure $\delta_\rho(k)$ for each pattern ρ, and uses the highest similarity value $\delta_{max}(k)$ to compute $\lambda_\beta(k)$. Details of both S-Comparison and AANN-Comparison are found in [1].

III. Predictive Sequence Learning

PSL is trained on an *event sequence* $\eta = (e_1, e_2, \ldots, e_t)$, where each *event* e is a member of an alphabet $\sum$. η is

defined up to the current time t from where the next event e_{t+1} is to be predicted.

PSL stores its knowledge as a set of hypotheses, known as a *hypothesis library* H. A *hypothesis* $h \in H$ expresses a dependence between an event sequence $X = (e_{t-n}, e_{t-n+1}, \ldots, e_t)$ and a target event $I = e_{t+1}$:

$$h : X \Rightarrow I \tag{1}$$

X_h is referred to as the *body* of h and I_h denotes the *head*. Each h is associated with a *confidence* c reflecting the conditional probability $P(I|X)$. For a given η, c is defined as $c(X \Rightarrow I) = s(X, I)/s(X)$, where the *support* $s(X)$ is the proportion of transactions in η that contains X. (X, I) denotes the concatenation of X and I. A transaction is defined as a sub-sequence of the same size as X, occurring after the creation of h. The length of h, denoted $|h|$, is defined as the number of elements in X_h. Hypotheses are also referred to as *states*, since a hypothesis of length $|h|$ corresponds to VMM state of order $|h|$.

A. Detailed description of PSL

Let the library H be the empty set of hypotheses. During learning, described in Algorithm 1, PSL tries to predict the future event e_{t+1}, based on the observed event sequence η. If the prediction is wrong, a new hypothesis h_{new} is created and added to H. h_{new} is one element longer than the longest hypothesis $h_c \in H$ that would have produced a correct prediction (see Algorithm 1 for an exact description of how h_c is selected). In this way, PSL grows the library only when it produces incorrect predictions.

For example, consider the event sequence $\eta = ABCCABCC$. Let $t = 1$. PSL will search for a hypothesis with a body matching A. H is initially empty and consequently PSL will not be able to perform a prediction. Instead, PSL creates a new hypothesis $(A) \Rightarrow B$ which is added to H. The same procedure will be executed at $t = 2$ and $t = 3$ so that $H = \{(A) \Rightarrow B; (B) \Rightarrow C; (C) \Rightarrow C\}$. At $t = 4$, PSL will find a matching hypothesis $h_{max} : (C) \Rightarrow C$ producing the wrong prediction C. Consequently, a new hypothesis $(C) \Rightarrow A$ is added to H. The predictions at $t = 5$ and $t = 6$ will be successful while $h : (C) \Rightarrow A$ will be selected at $t = 7$ and produce the wrong prediction. As a consequence, PSL will create a new hypothesis $h_{new} : (B, C) \Rightarrow C$. PSL has now learned the pattern and will not add any more hypotheses to H until it observes another η containing elements that do not follow this pattern.

IV. Methods for behavior recognition

Two methods based on the PSL algorithm are here presented. The first method, *PSLE-Comparison*, is inspired by the HMOSAIC architecture [2] and computes the responsibility signal λ_β as an inverse function of the normalized prediction error, produced by PSL. The second method, *PSLH-Comparison*, is more closely built on the PSL algorithm. λ is in this method a function of hypothesis activation match.

Algorithm 1 Predictive Sequence Learning (PSL)

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_n)$

$t \leftarrow 1$
$H \leftarrow \emptyset$
$M \leftarrow \{h \in H \mid X_h = (e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t)\}$
if $M = \emptyset$ **then**
  let $h_{new} : (e_t) \Rightarrow e_{t+1}$
  add h_{new} to H
  goto 20
end if
$\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ for\ all\ h' \in M\}$
let $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ for\ all\ h' \in \hat{M}\right\}$
if $e_{t+1} \neq I_{h_{max}}$ **then**
  let h_c be the longest hypothesis $\{h \in M \mid I_h = e_{t+1}\}$
  if $h_c = null$ **then**
    let $h_{new} : (e_t) \Rightarrow e_{t+1}$
  else
    let $h_{new} : (e_{t-|h_c|}, e_{t-|h_c|+1}, \ldots, e_t) \Rightarrow e_{t+1}$
  end if
  add h_{new} to H
end if
update the confidence for h_{max} and $h_{correct}$ as described in Section III
$t \leftarrow t + 1$
if $t < n$ **then**
  goto 2
end if

A. PSLE-Comparison

The responsibility signal $\lambda_\beta(t)$ of skill β at time t is given by:

$$\lambda_\beta(t) = \sum_{i=t-\nu}^{t} \frac{1 - \Delta_i^\beta}{\nu} \tag{2}$$

where ν is a constant describing the temporal extension of the behavior, i.e., λ_β is defined as an average of prediction performance over ν time steps. The *prediction error* Δ_i^β is given by:

$$\Delta_i^\beta = \begin{cases} 0 & if\ e_i = \hat{e}_i^\beta \\ 1 & otherwise \end{cases} \tag{3}$$

where $\hat{e}_i^\beta$ is the output of the forward model at position i in η_β.

By training a PSL library H_β on each event sequence η_β, one forward model for each skill β is created. The precise training procedure is described in Algorithm 1. The event sequence η_β used for training is a demonstration of skill β. In practice, several demonstrations of each skill may of course be used, but for simplicity we here consider them to be a single sequence of events.

After training, H_β is used for prediction as described in Algorithm 2. In principle, any discrete prediction algorithm

Algorithm 2 Making predictions using PSL

Require: an event sequence $\eta = (e_1, e_2, \ldots, e_{t-1})$
Require: the trained library $H = (h_1, h_2, \ldots, h_{|H|})$

$M \leftarrow \{h \in H \mid X_h = (e_{t-|h|}, e_{t-|h|+1}, \ldots, e_{t-1})\}$
$\hat{M} \leftarrow \{h \in M \mid |h| \geq |h'| \ for\ all\ h' \in M\}$
let $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ for\ all\ h' \in \hat{M}\right\}$
return the prediction $\hat{e}_t = I_{h_{max}}$

could be used as forward model, but an advantage of PSL is that the same algorithm constitutes both forward and inverse model. As enforced by the MOSAIC framework [24], [3], the forward and inverse models should be paired, meaning that the forward model should be able to predict the consequences of actions produced by the inverse model. This pairing is built into PSL, since the prediction and control is actually performed by the same model.

B. PSLH-Comparison

One problem with comparison methods based directly on prediction error was observed during our previous investigation of methods for behavior recognition [1]. Prediction error can be seen as a measure of how consistent an event sequence η is with some skill β. However, the prediction error does not tell whether η demonstrates all aspects of β, or only a fraction of these aspects.

In an attempt to approach this problem, *PSLH-Comparison* was designed. PSL is here used to create a single library H from skill demonstrations η_β of all β. $\lambda_\beta(t)$ is defined as the intersection between the hypotheses activated during the demonstrations of β, and the hypotheses activated by η within the time span $t - \nu$ to t:

$$\lambda_\beta(t) = \frac{\sum_{h \in H} \min\left(a_h^{\eta_\beta}, a_h^{\eta_t}\right)}{\sum_{h \in H} a_h^t} \tag{4}$$

where $a_h^{\eta_\beta}$ = *hAct*(η_β, H, h_η, 1, $|\eta_\beta|$) and $a_h^{\eta_t}$ = *hAct*(η, H, h_η, $t-\nu$, t). Equation 4 is the Bayes P_e, the minimum error probability between the two hypothesis activation distributions [25]. The hypothesis activation function *hAct* is defined in Algorithm 3, calculating the prediction contribution for a specific hypothesis h_η, given a certain time interval in η. Similarly to PSLE-Comparison, ν is a constant describing the temporal extension of the skill. The minimum error probability gives reward for hypotheses that are activated in both behaviors (similar to inverted prediction error), but also gives penalty for hypotheses that are only activated by one of the event sequences η or η_β.

V. Evaluation

The two proposed methods for behavior recognition, PSLE-Comparison and PSLH-Comparison, are here tested in an LFD setting using a Khepera II miniature robot [26]. A *load-transport-unload* task is defined, consisting of three sub-behaviors or skills. *Skill 1* involves the robot moving

Algorithm 3 Hypothesis activation

function *hAct*(η, H, h_η, t_{start}, t_{stop})

$t \leftarrow t_{start}$
$a_h \leftarrow 0$
$M \leftarrow \{h \in H \mid X_h = (e_{t-|h_\eta|}, e_{t-|h_\eta|+1}, \ldots, e_{t-1})\}$
$M' \leftarrow \{h \in M \mid I_h = e_t\}$
$\hat{M} \leftarrow \{h \in M' \mid |h| \geq |h'| \ for\ all\ h' \in M'\}$
let $h_{max} \in \left\{h \in \hat{M} \mid c(h) \geq c(h') \ for\ all\ h' \in \hat{M}\right\}$
if $h_{max} = h$ **then**
  $a_h \leftarrow a_h + \frac{1}{t_{stop} - t_{start}}$
end if
$t \leftarrow t + 1$
if $t \leq t_{stop}$ **then**
  goto 3
end if

forward in a corridor approaching an object (cylindrical wood block). When the robot gets close to the object, it should stop and wait for the human teacher to "load" the object, i.e., place it upon the robot. After loading, the robot turns around and goes back along the corridor. *Skill 2* involves general corridor driving, taking turns in the right way without hitting the walls and so on. *Skill 3* constitutes the "unloading" procedure, where the robot stops in a corner and waits for the teacher to remove the object and place it to the right of the robot. Then the robot turns and pushes the cylinder straight forward for about 10 centimeters, backs away and turns to go for another object. The sequence of actions expected by the robot is illustrated in Figure 1 and the experimental setup can be seen in Figure 2. Even though the setup was roughly the same in all experiments, the starting positions and exact placement of the walls varied between demonstration and repetition.

The robot was given no previous knowledge about itself or its surroundings. The only obvious design bias is the thresholding of proximity sensors into three levels, *far*, *medium* and *close,* corresponding to distances of a few centimeters. This thresholding was introduced to decrease the size of the observation space Y, limiting the amount of training required. An *observation* $y \in Y$ is defined as the combination of the eight proximity sensors, producing a total of 3^8 possible observations. An *action* $u \in U$ is defined as the combination of the speed commands sent to the two motors.

To put the performance of PSLE-Comparison and PSLH-Comparison in a larger context, two of our previously proposed methods for behavior recognition, AANN-Comparison and S-Comparison [1], are included in the evaluation.

All four comparison methods are trained on the same set of demonstrations. Skill 1 is demonstrated seven times for a total of about 2.6 minutes. Skill 2 is demonstrated for about 8.7 minutes and Skill 3 is demonstrated nine times, in total 4.6 minutes. For Task 4, the demonstrations from all three partial tasks were used, plus a single 2 min demonstration of the entire task. Exactly the same set of demonstrations have previously been used for evaluating the PSL algorithm

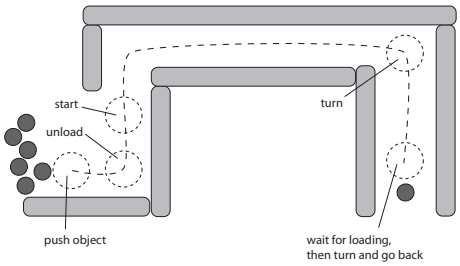

Figure 1. Schematic overview of the *load-transport-unload* task. Light gray rectangles mark walls, dark gray circles mark the objects and dashed circles mark a number of key positions for the robot. The robot starts by driving upwards in the figure, following the dashed line. until it reaches the object at the loading position. After loading, the robot turns around and follows the dashed line back until it reaches the unload position. When the cylinder has been unloaded (placed to the left of the robot), the robot turns and pushes the object. Finally, it backs away from the pile and awaits more instructions.

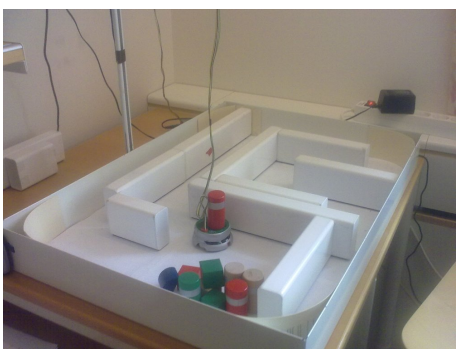

Figure 2. Experimental setup.

as a controller [21]. PSL is then able to repeat each of the three skills successfully, but unable to reproduce the complete *load-transport-unload* task. The reason PSL was unable to repeat the complete behavior is that knowledge from the three skills interfered. The algorithm is unable to separate the unloading activity from the turning, loading from pushing and so on. In these situations, some kind of higher level coordination is needed to prevent knowledge about the wrong activity from interfering. If PSL, when used as an algorithm for behavior recognition, is able to identify the present activity, it should constitute a good basis for building a coordination system separating the different activities into skills, preventing knowledge interference.

Ten demonstrations of the full *load-transport-unload* task are used for testing. A responsibility signal template is defined for each of the demonstrations, specifying which parts of the demonstration that corresponds to respective skill. See Figure 3 for an example template. The templates are manually constructed, based on time-synced video recordings of each demonstration. Parts of the templates contained overlapping skills, implying that these segments could have been produced by more than one skill. While manually constructed interpretations of the demonstrations may not constitute the ideal environment for an absolute performance measure, they should still constitute a good frame for comparing the behavior recognition algorithms.

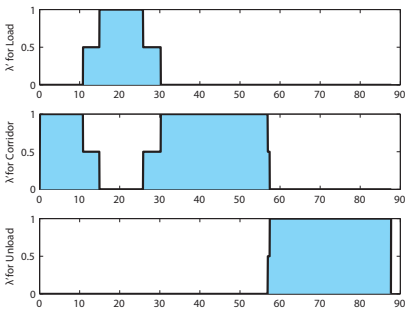

Figure 3. Example template for a single demonstration of the *load-transport-unload* task. The tick black line in top, middle and bottom plots indicates λ' for the Skill 1, 2 and 3, respectively. The green area below the line indicates the parts of the demonstration where respective skill should gain high responsibility. Overlapping periods are normalized such that the sum of activity levels for all skills equals 1.

Table I
AVERAGE RECOGNITION ERRORS AND λ VARIANCE ON THE *load-transport-unload* TASK.

Algorithm	$\tilde{\lambda}$	σ^2_λ	$\tilde{\lambda}_{Load}$	$\tilde{\lambda}_{Corr}$	$\tilde{\lambda}_{Unload}$
AANN-Comparison	0.405	0.027	0.350	0.423	0.442
S-Comparison	0.228	0.030	0.231	0.298	0.155
PSLE-Comparison	0.217	0.050	0.234	0.310	0.107
PSLH-Comparison	0.147	0.036	0.198	0.212	0.032

A. Results

Recognition errors for each of the four evaluated algorithms are presented in Table I. $\tilde{\lambda}_\beta(t) = \left|\lambda_\beta(t) - \lambda'_\beta(t)\right|$ is the recognition error at time t, where $\lambda_\beta(t)$ is the computed responsibility signal for skill β. $\lambda'_\beta(t)$ is the desired responsibility signal for β defined by the template. Both $\lambda_\beta(t)$ and $\lambda'_\beta(t)$ are normalized over all three skills.

In Table I, $\tilde{\lambda}$ is the total mean recognition error over all skills. σ^2_λ is the total variance over λ. $\tilde{\lambda}_{Load}$, $\tilde{\lambda}_{Corr}$ and $\tilde{\lambda}_{Unload}$ is the mean recognition error for each of the three skills. All values are normalized averages over 10 demonstrations. A standard t-test shows that both PSLE-Comparison and PSLH-Comparison have significantly smaller $\tilde{\lambda}$ than the other algorithms ($p < 0.005$) and that PSLH-Comparison is significantly better than PSLE-Comparison ($p < 0.005$).

Figure 4, 5 and 6 display the responsibility signals for skill 1, 2 and 3, respectively. Each figure shows both the desired responsibility signal λ', and the signals computed by each of the four recognition algorithms. Displayed values are from the same demonstration as the template signal in Figure 3 (and is consequently not an average over all ten

demonstrations, as opposed to values in Table I).

VI. Discussion

In the present work, two methods for behavior recognition are presented and evaluated. Both methods are based on the dynamic temporal difference algorithm Predictive Sequence Learning (PSL). PSL is both parameter-free and model-free in the sense that no ontological information about the robot or conditions in the world is pre-defined in the system. Instead, PSL creates a state space (hypothesis library) in order to predict the demonstrated data optimally.

The first method, *PSLE-Comparison*, takes inspiration from the MOSAIC architecture [2], [24] and computes the responsibility signal λ_β based on prediction error. The second method, *PSLH-Comparison*, is based on the minimum error probability between activation distributions over model H. PSLH-Comparison was designed to not only compute λ_β as a function of skill match (inverse prediction error) but also include a penalty for aspects of the skill not present in the demonstration.

The two algorithms are compared to two other methods for behavior recognition, *AANN-Comparison* and *S-Comparison* [1]. Performance is measured as the average recognition error. Both PSLE-Comparison and PSLH-Comparison shows significantly smaller recognition error than the other methods. However, the difference between PSLE-Comparison and S-Comparison is small. Overall, PSLH-Comparison is the winner with significantly smaller recognition errors than the other algorithms.

The present evaluation is based on ten demonstrations of a *load-transport-unload* task using a Khepera II robot [26]. This task is selected since the same data has previously been used to evaluate PSL as a controller [21]. In the previous evaluation, it was concluded that the PSL could learn each of the three skills (*load, corridor* and *unload*) but PSL was unable to repeat the overall task. The load-transport-unload task should consequently constitute a setting where some higher level coordination is necessary. Being able to identify each of the three skills in a demonstration of the whole behavior is one step towards creating such a coordination mechanism, allowing PSL to be placed within a hierarchical control architecture such as HMOSAIC [2].

The results show that the proposed recognition methods are significantly better than the benchmark methods used in the evaluation. An overall recognition error of less than 0.15 for PSLH-Comparison is in fact much better than expected. How well the algorithms would do in an on-line situation is however still an open question. The template signal $\lambda'_\beta(t)$ defined as the "correct" responsibility signal for skill β at time t merely reflects the teacher's high-level understanding of the demonstrated behavior, and is not necessarily the best way to separate the overall behavior into skills. It should also be mentioned that the algorithms are tested under conditions with relatively low noise levels. Even though S-Learning has been evaluated under noisy conditions with good results [23], it is expected that the PSL-based recognition methods is affected by noise in similar ways

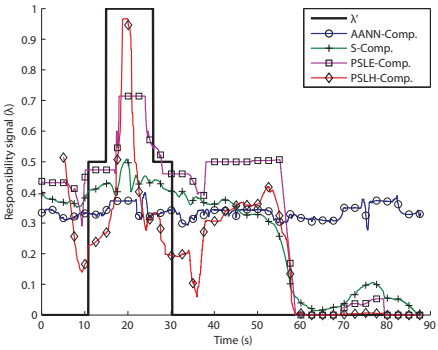

Figure 4. Responsibility signals for Skill 1 - Load. *AANN-Comp*, *S-Comp*, *PSLE-Comp* and *PSLH-Comp* indicates the responsibility signal computed with respective method.

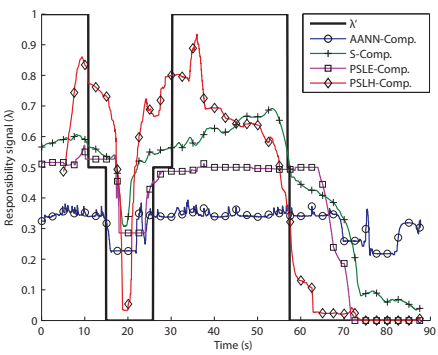

Figure 5. Responsibility signals for Skill 2 - Corridor. *AANN-Comp*, *S-Comp*, *PSLE-Comp* and *PSLH-Comp* indicates the responsibility signal computed with respective method.

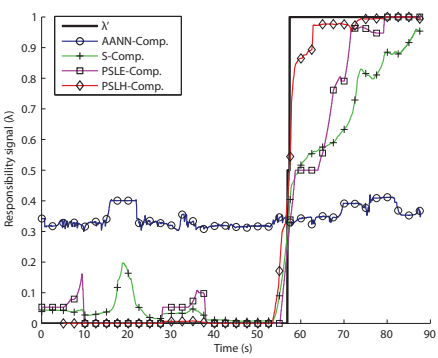

Figure 6. Responsibility signals for Skill 3 - Unload. *AANN-Comp*, *S-Comp*, *PSLE-Comp* and *PSLH-Comp* indicates the responsibility signal computed with respective method.

as PSL is subject to combinatorial explosion in large state spaces. Furthermore, both proposed algorithms compute the responsibility signal over a temporal extension ν, meaning that $\lambda_\beta(t)$ corresponds to how well β explains the events $(e_{t-\nu}, e_{t-\nu+1}, \ldots, e_t)$. I.e., we only know if the controller defined by β is the right choice for the present situation after these events have already occurred. Wolpert and co-workers [3] have also observed this problem, and introduce a *responsibility predictor* that estimates future responsibility signals. A corresponding mechanism is probably necessary when integrating the PSL based recognition methods with a controller.

A. Conclusions and future work

The results show that Bayes P_e (minimum error probability) over the activation pattern in the forward model (PSLH-Comparison) is a better method for behavior recognition than the prediction error (PSLE-Comparison), in the evaluated setting. While a more extensive study is necessary to draw any conclusions about the general performance of these algorithms, we find these results to be promising and intend to extend this evaluation to behavior recognition in other domains, and possibly to other types of data.

A big advantage of using PSL both for control (as described in [21]) and behavior recognition is that the forward and inverse computations are in fact based on the same model, i.e., the PSL library. This approach has several theoretical connections to the view of human perception and control as two heavily intertwined processes.

The present work should be seen as one step towards a hierarchical control architecture that can learn and coordinate itself, based on the PSL algorithm. The model-free design of PSL introduces very few assumptions into learning, and should constitute a good basis for many types of learning and control problems. Integrating PSLE-Comparison with a PSL-based control algorithm, to achieve a two-layer modular control system, is the next step in this process and will be part of our future work.

REFERENCES

[1] E. A. Billing and T. Hellström, "Behavior recognition for segmentation of demonstrated tasks," in *IEEE SMC International Conference on Distributed Human-Machine Systems*, Athens, Greece, March 2008, pp. 228 – 234.

[2] M. Haruno, D. M. Wolpert, and M. Kawato, "Hierarchical MOSAIC for movement generation," in *International Congress Series 1250*. Elsevier Science B.V., 2003, pp. 575– 590.

[3] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, no. 7–8, pp. 1317—1329, 1998.

[4] D. M. Wolpert, "A unifying computational framework for motor control and social interaction," *Phil. Trans. R. Soc. Lond.*, vol. B, no. 358, pp. 593–602, Mar. 2003.

[5] K. J. Friston, "Functional integration and inference in the brain," *Progress in Neurobiology*, vol. 68, no. 2, pp. 113–143, Oct. 2002.

[6] ——, "Learning and inference in the brain," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 16, no. 9, pp. 1325–52, 2003, PMID: 14622888.

[7] J. M. Kilner, K. J. Friston, and C. D. Frith, "Predictive coding: an account of the mirror neuron system," *Cogn Process*, vol. 8, pp. 159–166, 2007.

[8] D. George, "How the brain might work: A hierarchical and temporal model for learning and recognition," Ph.D. dissertation, Stanford University, Department of Electrical Engineering, 2008.

[9] D. George and J. Hawkins, "A hierarchical bayesian model of invariant pattern recognition in the visual cortex," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN'05)*, vol. 3, 2005, pp. 1812–1817 vol. 3.

[10] E. Billing, "Cognition reversed - robot learning from demonstration," Ph.D. dissertation, Umeå University, Department of Computing Science, Umeå, Sweden, December 2009.

[11] Y. Demiris and M. Johnson, "Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning," *Connection Science*, vol. 15, no. 4, pp. 231–243, 2003.

[12] Y. Demiris and A. Dearden, "From motor babbling to hierarchical learning by imitation: a robot developmental pathway," in *Proceedings of the 5th International Workshop on Epigenetic Robotics*, 2005, pp. 31—37.

[13] Y. Demiris and B. Khadhouri, "Hierarchical attentive multiple models for execution and recognition of actions," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 361–369, May 2006.

[14] M. Nicolescu, "A framework for learning from demonstration, generalization and practice in Human-Robot domains," Ph.D. dissertation, University of Southern California, 2003.

[15] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2008.

[16] E. A. Billing and T. Hellström, "A formalism for learning from demonstration," in *Cognition Reversed - Robot Learning from Demonstration*. Umeå, Sweden: Print & Media, Umeå University, 2009, pp. 73–102.

[17] C. L. Nehaniv and K. Dautenhahn, "Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications," in *Learning Robots: An Interdisciplinary Approach*, J. Demiris and A. Birk, Eds. World Scientific Press, 2000, vol. 24, pp. 136–161.

[18] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, "Action, state and effect metrics for robot imitation," in *15th IEEE International Symposium on Robot and Human Interactive Communication (ROMAN 2006)*, Hatfield, Sep. 2006, pp. 232–237.

[19] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *RSJ Advanced Robotics, Special Issue on Imitative Robots*, vol. 21, no. 13, pp. 1521–1544, 2007.

[20] S. Calinon, F. Guenter, and A. Billard, "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*, vol. 37, no. 2, pp. 286–298, 2007.

[21] E. A. Billing, T. Hellström, and L. E. Janlert, "Model free learning from demonstration," in *Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART)*, J. Filipe, A. Fred, and B. Sharp, Eds. Valencia, Spain: INSTICC, January 2010, pp. 62–71.

[22] B. Rohrer and S. Hulet, "BECCA - a brain emulating cognition and control architecture," Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA, Tech. Rep., 2006.

[23] ——, "A learning and control approach based on the human neuromotor system," in *Proceedings of Biomedical Robotics and Biomechatronics, BioRob*, 2006.

[24] M. Haruno, D. M. Wolpert, and M. M. Kawato, "MOSAIC model for sensorimotor learning and control," *Neural Comput.*, vol. 13, no. 10, pp. 2201–2220, 2001.

[25] S. Cha and S. N. Srihari, "On measuring the distance between histograms," *Pattern Recognition*, vol. 35, no. 6, pp. 1355–1370, Jun. 2002.

[26] K-Team, "Khepera robot," http://www.k-team.com, 2007. [Online]. Available: http://www.k-team.com

VI

Paper VI

Robot Learning from Demonstration using Predictive Sequence Learning*

Erik Billing, Thomas Hellström, and Lars-Erik Janlert

Dept. Computing Science, Umeå University, SE-901 87 Umeå, Sweden
billing@cs.umu.se, thomash@cs.umu.se, and lej@cs.umu.se
www.cs.umu.se/research/robotics

Abstract: In this chapter, the prediction algorithm *Predictive Sequence Learning (PSL)* is presented and evaluated in a robot *Learning from Demonstration (LFD)* setting. PSL generates hypotheses from a sequence of sensory-motor events. Generated hypotheses can be used as a semi-reactive controller for robots. PSL has previously been used as a method for LFD, but suffered from combinatorial explosion when applied to data with many dimensions, such as high dimensional sensor and motor data. A new version of PSL, referred to as *Fuzzy Predictive Sequence Learning (FPSL)*, is presented and evaluated in this chapter. FPSL is implemented as a Fuzzy Logic rule base and works on a continuous state space, in contrast to the discrete state space used in the original design of PSL. The evaluation of FPSL shows a significant performance improvement in comparison to the discrete version of the algorithm. Applied to an LFD task in a simulated apartment environment, the robot is able to learn to navigate to a specific location, starting from an unknown position in the apartment.

* To appear in A. Dutta (Eds.), *Robotic Systems - Applications, Control and Programming*. InTech.

To appear in A. Dutta (Eds.), *Robotic Systems - Applications, Control and Programming*. InTech.

Robot Learning from Demonstration using Predictive Sequence Learning

Erik Billing, Thomas Hellström, Lars-Erik Janlert
Department of Computing Science, Umeå University
Sweden

1. Introduction

In this chapter, the prediction algorithm Predictive Sequence Learning (PSL) is presented and evaluated in a robot *Learning from Demonstration* (LFD) setting. PSL generates hypotheses from a sequence of sensory-motor events. Generated hypotheses can be used as a semi-reactive controller for robots. PSL has previously been used as a method for LFD (Billing et al., 2010; 2011) but suffered from combinatorial explosion when applied to data with many dimensions, such as high dimensional sensor and motor data. A new version of PSL, referred to as *Fuzzy Predictive Sequence Learning* (FPSL), is presented and evaluated in this chapter. FPSL is implemented as a Fuzzy Logic rule base and works on a continuous state space, in contrast to the discrete state space used in the original design of PSL. The evaluation of FPSL shows a significant performance improvement in comparison to the discrete version of the algorithm. Applied to an LFD task in a simulated apartment environment, the robot is able to learn to navigate to a specific location, starting from an unknown position in the apartment.

Learning from Demonstration is a well-established technique for teaching robots new behaviors. One of the greatest challenges in LFD is to implement a learning algorithm that allows the robot pupil to generalize a sequence of actions demonstrated by the teacher such that the robot is able to perform the desired behavior in a dynamic environment. A behavior may be any complex sequence of actions executed in relation to sensor data (Billing & Hellström, 2010).

The LFD problem is often formulated as four questions, *what-to-imitate*, *how-to-imitate*, *when-to-imitate* and *who-to-imitate* which leads up to the larger problem of how to evaluate an imitation (Alissandrakis et al., 2002). Large parts of the literature approach the learning problem by trying to find the common features within a set of demonstrations of the same behavior. A skill is generalized by exploiting statistical regularities among the demonstrations (e.g. Calinon, 2009). This is reflected in the *what-to-imitate* question, originally introduced in a classical work by Nehaniv & Dautenhahn (2000) and is in a longer form described as:

> An action or sequence of actions is a successful component of imitation of a particular action if it achieves the same subgoal as that action. An entire sequence of actions is successful if it successively achieves each of a sequence of abstracted subgoals.

The problem is difficult since a certain behavior can be imitated on many different abstraction levels. Byrne & Russon (1998) identified two levels; the *action-level imitation* copying the surface of the behavior and a *program-level imitation* copying the structure of the behavior. A

third level, the *effect-level imitation*, was introduced by Nehaniv & Dautenhahn (2001) in order to better describe imitation between agents with dissimilar body structures. Demiris & Hayes (1997) proposed three slightly different levels: *1) basic imitation* with strong similarities to the notion of action-level imitation, 2) *functional imitation* that best corresponds to effect-level imitation and 3) *abstract imitation* that represents coordination based on the presumed internal state of the agent rather than the observed behavior. Demiris and Hayes give the example of making a sad face when someone is crying.

The necessity to consider the level of imitation in LFD becomes apparent when considering two demonstrations that look very different considered as sequences of data, but that we as humans still interpret as examples of the same behavior since they achieve similar results on an abstract level. This would correspond to a functional or program-level imitation. In these situations it is very difficult to find similarities between the demonstrations without providing high level knowledge about the behavior, often leading to specialized systems directed to LDF in limited domains.

A related problem is that two demonstrations of the same behavior may not have the same length. If one demonstration takes longer time than another, they can not be directly compared in order to find common features. Researchers have therefore used techniques to determine the temporal alignment of demonstrations. One common technique is *dynamic time warping* (Myers & Rabiner, 1981), that can be used to compensate for temporal differences in the data.

Behaviors can be demonstrated to a robot in many different ways. Argall et al. (2009) outline four types of demonstrations: A direct recording of sensor stimuli, joint angles, etc., is referred to as an *identity record mapping*. In this case, the robot is often used during the demonstration and controlled via teleoperation or by physically moving the robot's limbs (kinestetic teaching). An external observation, e.g. a video recording of the teacher, is called a *non-identity record mapping*. This type of demonstrations poses a difficult sensing problem of detecting how the teacher has moved, but also allows much more flexible demonstration setting. The teacher may have a body identical to that of the pupil (*identity embodiment*) or a body with a different structure (*non-identity embodiment*). In the latter case, the demonstration has to be transformed into corresponding actions using the body of the pupil, a difficult problem known as the *correspondence problem* (Nehaniv & Dautenhahn, 2001). In this work we focus on LFD via teleoperation. Sensor data and motor commands are in this setting recorded while a human teacher demonstrates the desired behavior by tele-operating the robot, producing demonstrations with identity in both record mapping and embodiment.

1.1 Metric of imitation

Successful imitation requires that relevant features of the demonstration are selected at a suitable imitation level and processed into a generalized representation of the behavior. The process is difficult to implement in a robot since it is often far from obvious which imitation level that is optimal in a specific situation, and the relevance of features may consequently vary significantly from one learning situation to another. This problem has been formalized as a *metric of imitation*, defined as a weighted sum over all strategy-dependent metrics on all imitation levels (Billard et al., 2003).

The metric of imitation was originally demonstrated on a manipulation task with a humanoid robot (Billard et al., 2003). With focus on the correspondence problem, Alissandrakis et al. (2005) propose a similar approach to imitation of manipulation tasks. The what-to-imitate problem is approached by maximizing trajectory agreements of manipulated objects, using several different metrics. Some metrics encoded absolute trajectories while other metrics

encoded relative object displacement and the relevant aspects of the behavior were in this way extracted as the common features in the demonstration set. Calinon et al. (2007) developed this approach by encoding the demonstration set using a mixture of Gaussian/Bernoulli distributions. The Gaussian Mixture Model approach is attractive since the behavior is divided into several distributions with different covariance, and different metrics can in this way be selected for different parts of the demonstrated behavior. More recently, similar encoding strategies have been evaluated for learning of a robot navigation task (de Rengervé et al., 2010).

1.2 Behavior primitives as a basis for imitation

Another common approach to LFD is to map the demonstration onto a set of pre-programmed or previously learned primitives controllers (Billing & Hellström, 2010). The approach has strong connections to *behavior-based architectures* (Arkin, 1998; Matarić, 1997; Matarić & Marjanovic, 1993) and earlier reactive approaches (e.g. Brooks, 1986; 1991). When introducing behavior primitives, the LFD process can be divided into three tasks (Billing & Hellström, 2010):

1. *Behavior segmentation* where a demonstration is divided into smaller segments.
2. *Behavior recognition* where each segment is associated with a primitive controller.
3. *Behavior coordination*, referring to identification of rules or switching conditions for how the primitives are to be combined.

Behavior segmentation and recognition can be seen as one way to approach the what-to-imitate problem, whereas behavior coordination is part of how-to-imitate. The approach represents one way of introducing good bias in learning and solve the generalization problem by relying on previous behavioral knowledge. While there are many domain specific solutions to these three subproblems, they appear very difficult to solve in the general case. Specifically, behavior recognition poses the problem of mapping a sequence of observations to a set of controllers to which the input is unknown. Again, the need to introduce a metric of imitation appears.

Part of the problem to find a general solution to these problems may lie in a vague definition of *behavior* (Matarić, 1997). The notion of behavior is strongly connected to the purpose of executed actions and a definition of goal. Nicolescu (2003) identified two major types of goals:

Maintenance goals: A specific condition has to be maintained for a time interval.

Achievement goals: A specific condition has to be reached.

The use of behavior primitives as a basis for imitation has many connections to biology (e.g. Matarić, 2002) and specifically the mirror system (Brass et al., 2000; Gallese et al., 1996; Rizzolatti et al., 1988; Rizzolatti & Craighero, 2004). While the role of the mirror system is still highly debated, several groups of researchers propose computational models where perception and action are tightly interweaved. Among the most prominent examples are the *HAMMER* architecture (Demiris & Hayes, 2002; Demiris, 1999; Demiris & Johnson, 2003) and the *MOSAIC* architecture (Haruno et al., 2001; Wolpert & Kawato, 1998). Both these architectures implement a set of modules, where each module is an inverse model (controller) paired with a forward model (predictor). The inverse and forward models are trained together such that the forward model can predict sensor data in response to the actions produced by the inverse model. The inverse model is tuned to execute a certain behavior when the forward model produces good predictions. The prediction error is used to compute a bottom-up

signal for each module. Based on the bottom-up signal, a top-down responsibility signal or confidence value is computed and propagated to each module. The output of the system is a combination of the actions produced by each inverse model, proportional to their current responsibility. The responsibility signal also controls the learning rate of each module, such that modules are only updated when their responsibility is high. In this way, modules are tuned to a specific behavior or parts of a behavior. Since the prediction error of the forward model is used as a measure of how well the specific module fits present circumstances, it can be seen as a metric of imitation that is learnt together with the controller. The architecture can be composed into a hierarchical system where modules are organized in layers, with the lowest layer interacting with sensors and actuators. The bottom-up signal constitutes sensor input for the layer above and actions produced by higher levels constitutes the top-down responsibility signal.

One motivation for this architecture lies in an efficient division of labor between different parts of the system. Each module can be said to operate with a specific temporal resolution. Modules at the bottom layer are given the highest resolution while modules higher up in the hierarchy have decreasing temporal resolution. State variables that change slowly compared to a specific module's resolution are ignored by that module and are instead handled by modules higher up in the hierarchy. Slowly changing states that lead to high responsibility for the module is referred to as the module's *context*. In a similar fashion, variables that change fast in comparison to the temporal resolution are handled lower in the hierarchy. This allows each module to implement a controller where the behavior depends on relatively recent states. Long temporal dependencies are modeled by switching between modules, which removes the requirement for each model to capture these dependencies. Furthermore, updates of a single behavior or parts of a behavior will only require updates of a few modules and will not propagate changes to other modules. See Billing (2009) for a longer discussion on these aspects of hierarchical architectures.

The HAMMER and MOSAIC architectures make few restrictions on what kind of controllers each module should implement. We argue however, that modules should be *semi-reactive*, meaning that action selection and predictions of sensor events should be based on recent sensor and motor events. Strictly reactive modules are not desirable since each module must be able to model any dependency shorter than the temporal resolution of modules in the layer directly above.

The division of behavior into modules is however also producing a number of drawbacks. The possibility for the system to share knowledge between behaviors is limited. Moreover, the system has to combine actions produced by different modules, which may be difficult in cases when more than one module receives high responsibility.

One architecture with similarities to HAMMER and MOSAIC able to share knowledge between different behaviors is *RNNPB* (Tani et al., 2004). *RNNPB* is a recurrent neural network with parametric bias (PB). Both input and output layer of the network contains sensor and motor nodes as well as nodes with recurrent connections. In addition, the input layer is given a set of extra nodes, representing the PB vector. The network is trained to minimize prediction error, both by back-propagation and by changing the PB vector. The PB vector is however updated slowly, such that it organizes into what could be seen as a context layer for the rest of the network. In addition to giving the network the ability to represent different behaviors that share knowledge, the PB vector can be used for behavior recognition.

Another architecture known as *Brain Emulating Cognition and Control Architecture* (BECCA) (Rohrer & Hulet, 2006) heavily influenced our early work on the PSL algorithm. The focus

of BECCA is to capture the discrete episodic nature of many types of human motor behavior, without introducing a priori knowledge into the system. BECCA was presented as a very general reinforcement learning system, applicable to many types of learning and control problems. One of the core elements of BECCA is the temporal difference (TD) algorithm *Sequence Learning* (SL) (Rohrer, 2007). SL builds sequences of passed events which is used to predict future events, and can in contrast to other TD algorithms base its predictions on many previous states.

Inspired by BECCA and specifically SL, we developed the PSL algorithm as a method for LFD (Billing et al., 2010; 2011). PSL has many interesting properties seen as a learning algorithm for robots. It is model free, meaning that it introduces very few assumptions into learning and does not need any task specific configuration. PSL can be seen as a variable-order Markov model. Starting out from a reactive (first order) model, PSL estimates transition probabilities between discrete sensor and motor states. For states that do not show Markov property, the order is increased and PSL models the transition probability based on several passed events. In this way, PSL will progressively gain memory for parts of the behavior that cannot be modeled in a reactive way. In theory, there is no limitation to the order of the state and hence the length of the memory, but PSL is in practice unable to capture long temporal dependencies due to combinatorial explosion.

PSL has been evaluated both as a controller (Billing et al., 2011) and as a method for behavior recognition (Billing et al., 2010). Even though the evaluation overall generated good results, PSL is subject to combinatorial explosion both when the number of sensors and actuators increase, and when the demonstrated behavior requires modeling of long temporal dependencies. PSL can however efficiently model short temporal dependencies in a semi-reactive way and should thus be a good platform for implementing a hierarchical system similar to the HAMMER and MOSAIC architectures.

In this chapter, we present and evaluate a new version of PSL based on Fuzzy Logic. While keeping the core idea of the original PSL algorithm, the new version can handle continuous and multi dimensional data in a better way. To distinguish between the two, the new fuzzy version of the algorithm is denoted FPSL, whereas the previous discrete version is denoted DPSL. A detailed description of FPSL is given in Section 2. An evaluation with comparisons between the two algorithms is presented in Section 3, followed by a discussion and conclusions in section 4.

2. Predictive Sequence Learning

FPSL builds fuzzy rules, referred to as *hypotheses* h, describing temporal dependencies between a sensory-motor event e_{t+1} and a sequence of passed events $\left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)$, defined up until current time t.

$$h : \left(\mathrm{Y}_{t-|h|+1} \; is \; E^h_{|h|-1} \wedge \mathrm{Y}_{t-|h|+2} \; is \; E^h_{|h|-2} \wedge \ldots \wedge \mathrm{Y}_t \; is \; E^h_0\right) \overset{C}{\Rightarrow} \mathrm{Y}_{t+1} \; is \; \bar{E}^h \tag{1}$$

Y_i is the event variable and $E^h(e)$ is a fuzzy membership function returning a membership value for a specific e. The right hand side $\bar{E}^h$ is a membership function comprising expected events at time $t+1$. $|h|$ denotes the length of h, i.e., the number of left-hand-side conditions of the rule. Both E and $\bar{E}$ are implemented as standard cone membership functions with base width ε (e.g. Klir & Yuan, 1995).

A set of hypotheses can be used to compute a prediction $\hat{e}_{t+1}$ given a sequence of passed sensory-motor events η, defined up to the current time t:

$$\eta = (e_1, e_2, \ldots, e_t) \tag{2}$$

The process of matching hypothesis to data is described in Section 2.1. The PSL learning process, where hypotheses are generated from a sequence of data, is described in Section 2.2. Finally, a discussion about parameters and configuration is found in Section 2.3.

2.1 Matching hypotheses

Given a sequence of sensory-motor events $\eta = (e_1, e_2, \ldots, e_t)$, a match $\alpha_t(h)$ of the rule is given by:

$$\alpha_t(h) : \bigwedge_{i=0}^{|h|-1} E_i^h(e_{t-i}) \tag{3}$$

where $\wedge$ is implemented as a *min*-function.
Hypotheses are grouped in fuzzy sets C whose membership value $C(h)$ describes the confidence of h at time t:

$$C(h) = \frac{\sum_{k=t^h}^{t} \alpha_k(h) \bar{E}^h(e_{k+1})}{\sum_{k=t^h}^{t} \alpha_k(h)} \tag{4}$$

.
t^h is the creation time of h or 1 if h existed prior to training. Each C represents a *context* and can be used to implement a specific behavior or part of a behavior. The *responsibility signal* $\lambda_t(C)$ is used to control which behavior that is active at a specific time. The combined confidence value $\tilde{C}_t(h)$ is a weighted sum over all C:

$$\tilde{C}_t(h) = \frac{\sum_C C(h) \lambda_t(C)}{\sum_C \lambda_t(C)} \tag{5}$$

.
$\tilde{C}_t$ can be seen as a fuzzy set representing the active context at time t. Hypotheses contribute to a prediction in proportion to their membership in $\tilde{C}$ and the *match set* $\hat{M}$. $\hat{M}$ is defined in three steps. First, the best matching hypotheses for each $\bar{E}$ is selected:

$$M = \left\{ h \mid \alpha(h) \geq \alpha(h') \ for\ all \ \left\{ h' \mid \bar{E}^{h'} = \bar{E}^h \right\} \right\} \tag{6}$$

.
The longest $h \in M$ for each RHS is selected:

$$\tilde{M} = \left\{ h \mid |h| \geq |h'| \ for\ all \ \left\{ h' \in M \mid \bar{E}^{h'} = \bar{E}^h \right\} \right\} \tag{7}$$

.
Finally, the match set $\hat{M}$ is defined as:

$$\hat{M}(h) = \begin{cases} \alpha(h) \tilde{C}(h) & h \in \tilde{M} \\ 0 & otherwise \end{cases} \tag{8}$$

.
The aggregated prediction $\hat{E}(e_{t+1})$ is computed using the Larsen method (e.g. Fullér, 1995):

$$\hat{E}\left(e_{t+1}\right) : \bigvee_{h} \bar{E}_h\left(e_{t+1}\right) \hat{M}\left(h\right) \tag{9}$$

$\hat{E}$ is converted to crisp values using a squared *center of sum* defuzzification:

$$\hat{e} = \frac{\sum_{e} e\hat{E}\left(e\right)^2}{\sum_{e} \hat{E}\left(e\right)^2} \tag{10}$$

The amount of entropy in $\hat{M}$ also bears information about how reliable a specific prediction is, referred to as the *trust* $\hat{c}$:

$$\hat{c}\left(\hat{M}\right) = \begin{cases} 0 & \hat{M} = \emptyset \\ \exp\left[\sum_{h} \hat{M}\left(h\right) \log_2\left(\hat{M}\left(h\right)\right)\right] & otherwise \end{cases} \tag{11}$$

The trust is important since it allows a controller to evaluate when to rely on PSL, and when to choose an alternate control method. The proportion of time steps in η for which $\hat{c} > 0$ and PSL is able to produce a prediction is referred to as the *coverage* $\phi\left(\eta\right)$:

$$\phi\left(\eta\right) = \frac{\sum_{i=1}^{t} \begin{cases} 1 & \hat{c}_i > 0 \\ 0 & otherwise \end{cases}}{t} \tag{12}$$

2.2 Generating hypotheses

Hypotheses can be generated from a sequence of sensory-motor events η. During training, PSL continuously makes predictions and creates new hypotheses when no matching hypothesis produces the correct prediction $\bar{E}$. The exact training procedure is described in Algorithm 0.1.

For example, consider the event sequence $\eta = abccabccabcc$. Let $t = 1$. PSL will search for a hypothesis with a body matching a. Initially, the context set C is empty and consequently PSL will create a new hypothesis $(a) \Rightarrow b$ which is added to C with confidence 1, denoted $C\left(a \Rightarrow b\right) = 1$. The same procedure will be executed at $t = 2$ and $t = 3$ such that $C\left((b) \Rightarrow c\right) = 1$ and $C\left((c) \Rightarrow c\right) = 1$. At $t = 4$, PSL will find a matching hypothesis $(c) \Rightarrow c$ producing the wrong prediction c. Consequently, a new hypothesis $(c) \Rightarrow a$ is created and confidences are updated such that $C\left((c) \Rightarrow c\right) = 0.5$ and $C\left((c) \Rightarrow a\right) = 1$. The new hypothesis receives a higher confidence since confidence values are calculated from the creation time of the hypothesis (Equation 4). The predictions at $t = 5$ and $t = 6$ will be correct and no new hypotheses are created. At $t = 7$, both $(c) \Rightarrow a$ and $(c) \Rightarrow c$ will contribute to the prediction $\hat{E}$. Since the confidence of $(c) \Rightarrow a$ is higher than that of $(c) \Rightarrow c$, $\hat{E}$ will defuzzify towards a, producing the wrong prediction (Equation 10). As a result, PSL creates a new hypothesis $(b, c) \Rightarrow c$. Similarly, $(c, c) \Rightarrow a$ will be created at $t = 8$. PSL is now able to predict all elements in the sequence perfectly and no new hypotheses are created.

Source code from the implementation used in the present work is available online (Billing, 2011).

Algorithm 0.1 Predictive Sequence Learning (PSL)

Require: $\psi = (e_1, e_2, \ldots, e_T)$ where T denotes the length of the training set
Require: $\hat{\alpha}$ as the precision constant, see text

1: **let** $t \leftarrow 1$
2: **let** $\eta = (e_1, e_2, \ldots, e_t)$
3: **let** $C \leftarrow \varnothing$
4: **let** $\hat{E}$ as Eq. 9
5: **if** $\hat{E}(e_{t+1}) < \hat{\alpha}$ **then**
6: **let** $h_{new} = CreateHypothesis(\eta, C)$ as defined by Algorithm 0.2
7: $C(h_{new}) \leftarrow 1$
8: **end if**
9: Update confidences $C(h)$ as defined by Equation 4
10: **set** $t = t + 1$
11: **if** t<T **then**
12: **goto** 2
13: **end if**

Algorithm 0.2 CreateHypothesis

Require: $\eta = (e_1, e_2, \ldots, e_t)$
Require: $C : h \rightarrow [0, 1]$
Require: α as defined by Eq. 3

1: **let** $\hat{M}(h)$ as Eq. 8
2: **let** $\bar{M} = \left\{ h \mid \bar{E}^h(e_{t+1}) \geq \hat{\alpha} \wedge \hat{M}(h) > 0 \right\}$ where $\hat{\alpha}$ is the precision constant, see Section 2.3
3: **if** $\bar{M} = \emptyset$ **then**
4: **let** E^* be a new membership function with center e_t and base ε
5: **return** $h_{new} : (\mathrm{Y}_t \ is \ E^*) \Rightarrow \mathrm{Y}_{t+1} \ is \ \bar{E}$
6: **else**
7: **let** $\bar{h} \in \bar{M}$
8: **if** $C(\bar{h}) = 1$ **then**
9: **return** *null*
10: **else**
11: **let** E^* be a new membership function with center $e_{t-|\bar{h}|}$ and base ε
12: **return** $h_{new} : \left(\mathrm{Y}_{t-|\bar{h}|} \ is \ E^*, \mathrm{Y}_{t-|\bar{h}|+1} \ is \ E^{\bar{h}}_{|\bar{h}|-1}, \ldots, \mathrm{Y}_t \ is \ E^{\bar{h}}_0 \right) \Rightarrow \mathrm{Y}_{t+1} \ is \ \bar{E}$
13: **end if**
14: **end if**

2.3 Parameters and task specific configuration

A clear description of parameters is important for any learning algorithm. Parameters always introduce the risk that the learning algorithm is tuned towards the evaluated task, producing better results than it would in the general case. We have therefore strived towards limiting the number of parameters of PSL. The original design of PSL was completely parameter free, with the exception that continuous data was discretized using some discretization method. The version of PSL proposed here can be seen as a generalization of the original algorithm (Billing et al., 2011) where the width ε of the membership function E determines the discretization resolution. In addition, a second parameter is introduced, referred to as the *precision constant* $\hat{\alpha}$. $\hat{\alpha}$ is in fuzzy logic terminology an *α-cut*, i.e., thresholds over the fuzzy membership function in the interval $[0, 1]$ (Klir & Yuan, 1995).

ε controls how generously FPSL matches hypotheses. A high ε makes the algorithm crisp but typically increases the precision of predictions when a match is found. Contrary, a low ε reduces the risk that FPSL reaches unobserved states at the cost of a decreased prediction performance. The high value of ε can be compared to a fine resolution data discretization for the previous version of PSL.

$\hat{\alpha}$ is only used during learning, controlling how exact a specific $\bar{E}$ has to be before a new hypothesis with a different $\bar{E}$ is created. A large $\hat{\alpha}$ reduces prediction error but typically results in more hypotheses being created during learning.

Both ε and $\hat{\alpha}$ controls the tolerance to random variations in the data and can be decided based on how exact we desire that FPSL should model the data. Small ε in combination with large $\hat{\alpha}$ will result in a model that closely fits the training data, typically producing small prediction errors but also a low coverage.

3. Evaluation

Two tests were performed to evaluate the performance of FPSL and compare it to the previous version. A simulated Robosoft Kompai robot (Robosoft, 2011) was used in the Microsoft RDS simulation environment (Microsoft, 2011). The 270 degree laser scanner of the Kompai was used as sensor data and the robot was controlled by setting linear and angular speeds.

Demonstrations were performed via tele-operation using a joypad, while sensor and motor data were recorded with a temporal resolution of 20 Hz. The dimensionality of the laser scanner was reduced to 20 dimensions using an average filter. Angular and linear speeds were however fed directly into PSL.

The first test (Section 3.1) was designed to compare FPSL and DPSL as prediction algorithms, using sensor data from the simulated robot. The second test (Section 3.2) demonstrates the use of FPSL as a method for LFD.

3.1 Sensor prediction

The two versions of PSL were compared using a series of tests of prediction performance. Even though DPSL and FPSL are similar in many ways, a comparison is not trivial since DPSL works on discrete data whereas FPSL uses continuous data. Prediction performance of DPSL will hence depend on how the data is discretized while the performance of FPSL depends on the parameters ε and $\hat{\alpha}$.

To capture the prediction performance of the two algorithms using different configurations, a series of tests were designed. 10 discretization levels were chosen, ranging from a fine resolution where DPSL could only produce predictions on a few samples in the test set, to a low resolution where DPSL rarely met unobserved states. Laser readings were discretized

Fig. 1. Simulation Environment (Microsoft, 2011) used for evaluations. Blue stars and yellow dots represent starting positions used for demonstrations and test runs, respectively. The green area marked with a *G* represents the target position. The white area under star 10 is the robot.

over 0.8 m for the finest resolution, up to 8 m for the lowest resolution. Motor data was discretized over 0.06m/s for the finest resolution up to 0.6 m/s for the lowest resolution. Similarly, 10 ε values were chosen, corresponding to a cone base ranging from 0.8 m to 8 m for laser data, and 0.06 m/s up to 0.6 m/s for motor data. $\hat{\alpha}$ was given a constant value of 0.9, corresponding to a error tolerance of 10% of ε.
10 data files were used, each containing a demonstration where the teacher directed the robot from a position in the apartment to the TV, see Figure 1. A rotating comparison was used, where PSL was tested on one demonstration at a time and the other nine demonstrations were used as training data. Prediction performance was measured in meters on laser range data.

3.1.1 Results

The results from the evaluation are illustrated in Figure 2. While the ε value of FPSL cannot directly be compared to the discretization level used for DPSL, the two parameters have similar effect on coverage. Prediction error is only calculated on the proportion of the data for which prediction are produced, and consequently, prediction error increases with coverage.

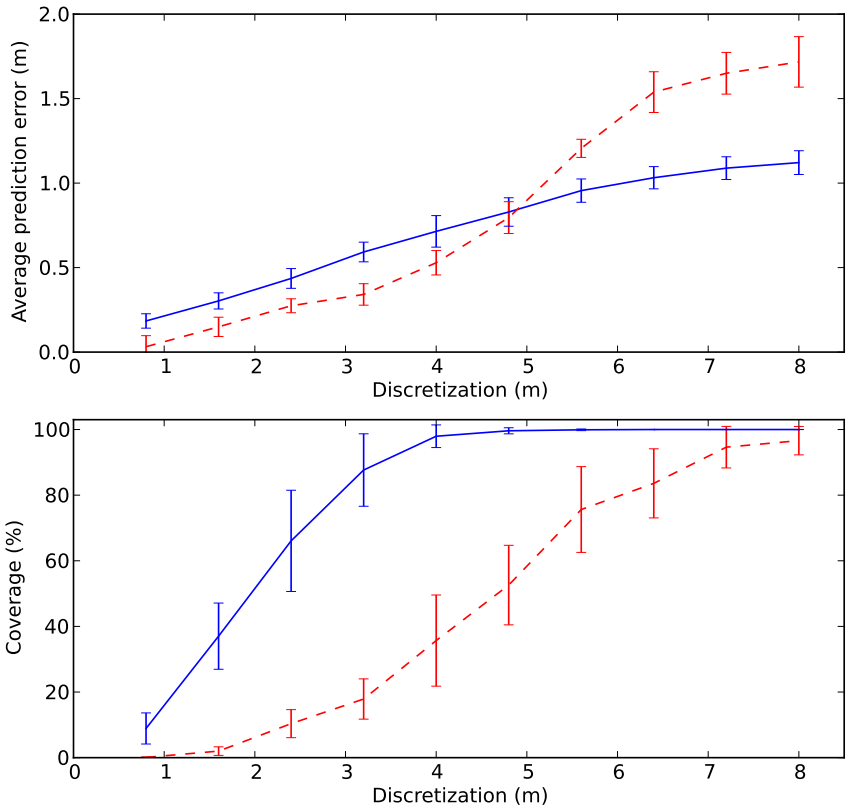

Fig. 2. Results from the prediction evaluation (Section 3.1). Upper plot shows prediction errors for FPSL (solid line) and DPSL (dashed line). Lower plot shows coverage , i.e. the proportion of samples for which the algorithm generated predictions, see Equation 12. Vertical bars represent standard deviation.

3.2 Target reaching

This evaluation can be seen as a continuation of previous tests with DPSL using a Khepera robot (Billing et al., 2011). The evaluation is here performed in a more complex environment, using a robot with much larger sensor dimensionality. Initial tests showed that DPSL has sever problems to handle the increased sensor dimensionality when used as a controller. A discretization resolution of about 2 m appeared necessary in order to produce satisfying discrimination ability. Even with this relatively low resolution, the 20 dimensional data produced a very large state space causing DPSL to frequently reach unrecognized states. DPSL could control the robot after intense training in parts of the test environment, but could

not compete with FPSL in a more realistic setting. We therefore chose to not make a direct controller comparison, but rather show the behavior of FPSL in an applied and reproducible setting.

FPSL was trained on 10 demonstrations showing how to get to the TV from various places in the apartment, see Figure 1. The performance of FPSL as a method for LFD was evaluated by testing how often the robot reached the target position in front of the TV starting out from 10 different positions than the ones used during training. FPSL controlled the robot by continuously predicting the next sensory-motor event based on the sequence of passed events. The motor part of the predicted element was sent to the robot controller. A standard reactive obstacle avoidance controller was used as fallback in cases where FPSL did not find any match with observed data. The task was considered successfully executed if the target position was reached without hitting any walls or obstacles. The experiment was repeated ten times, producing a total of 100 test runs.

3.2.1 Results

FPSL successfully reached the target position in front of the TV (the green area in Figure 1) in 79% of the test runs. In 68 runs, it stopped in front of the TV as demonstrated, but in 11 runs it failed to stop even though it reached the target position. The distribution over the 10 starting positions illustrated in Figure 3.

4. Discussion

Applied as a robot controller, PSL is a semi-reactive generative model that produces both actions and expected observations, based on recent sensory-motor events. We believe that this approach to robot learning has great potential since the behavior can be learnt progressively and previous knowledge contributes to the interpretation of new events. It is also general in the sense that very little domain specific knowledge is introduced. Memories are stored as sequences of sensory-motor events that in principle can represent any behavior. While PSL efficiently can represent behaviors with short temporal dependencies, it is subject to combinatorial explosion when the behavior requires representations over longer time spans. We argue that the gradually extending memory of PSL, from being purely reactive to containing representations over longer time when needed, provides a good bias in learning. It will however make learning of behaviors that do require long temporal dependencies slow. The fuzzy version of PSL presented in this work does not directly provide a solution to this problem, but is one step towards integrating PSL in a hierarchical structure as discussed in Section 1.2.

The seeds to FPSL came from the observation that a lot of training was required in order to cover the state space of DPSL with satisfying resolution. A better tradeoff between high precision in prediction and coverage would make PSL a more competitive alternative for real world LFD scenarios. Without sacrificing the strong attributes of the original PSL algorithm, such as the model free design, few parameters and progressively growing representations, FPSL was designed.

Expressing PSL with Fuzzy Logic is in many ways a natural extension and generalization of the discrete algorithm. By using a discrete uniform membership function E and a *max* operator for defuzzification, FPSL becomes very similar to DPSL. Even though the processing of continuous values does add significant processing requirements in comparison to DPSL, FPSL can still be efficiently implemented as a fuzzy rule controller.

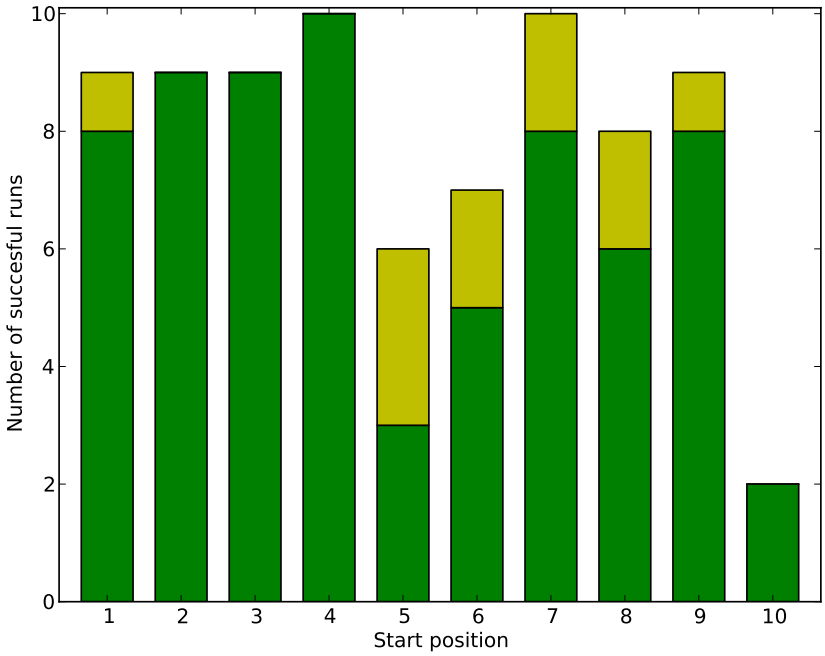

Fig. 3. Results from test runs in simulated environment (Section 3.2). Each bar corresponds to one starting position, see Figure 1. The green part of the bar represents number of successful runs where the robot reached and stopped at the target position in front of the TV. The yellow part represents runs when the robot successfully reached the target, but did not stop. The test was executed 10 times from each starting position.

The evaluation shows that FPSL produces significantly smaller prediction errors in relation to the coverage than DPSL (Section 3.1). This was expected since FPSL can be trained to produce small prediction errors by keeping a high precision constant $\hat{\alpha}$, while the coverage is still kept high by using a large ε. In contrast, when using DPSL, one must choose between a small prediction error with low coverage or a high coverage at the price of an increased prediction error. As can be seen in Figure 2, FPSL is also affected by the precision/coverage tradeoff, but not nearly as much as DPSL. Furthermore, the number of generated hypotheses will increase with $\hat{\alpha}$, which also has a positive effect on coverage for multidimensional data.
While FPSL performs much better than DPSL on large and multidimensional state spaces, it should not be seen as a general solution to the dimensionality problem. The increased number of hypotheses results in increased processing and memory requirements. Furthermore, FPSL is still not able to ignore uncorrelated dimensions in data, making it subject to the curse of dimensionality. One potential solution is to modify the size of membership functions in relation to the information content of the dimension. However, initial tests did not produce satisfying results and further experiments in this direction were postponed to future work.

We found the results from the controller evaluation very promising (Section 3.2). The application environment has been scaled up significantly in comparison to previous work (Billing et al., 2011) and we are now able to perform learning in a fairly realistic setting. When observing these results one should remember that PSL does not solve a spatial task. There is no position sensor or internal map of the environment and the robot is still able to navigate from almost any position in the environment, to a specific target location. The goal is better described as an attractor in a dynamic system, where the robot in interaction with the environment finally reaches a stable state in front of the TV (Figure 1).

An interesting observation is that it is often difficult to predict how well PSL will be able to handle a specific situation. For example, starting position 6 was not passed during any demonstration, but PSL still managed to control the robot such that it reached the target in 7 out of 10 test runs. On the other hand, position 5 and 10 produced worse results than expected. Even though these starting positions were spatially close to several positions passed during the demonstrations, the directions at which the robot reached these positions were different, producing different laser scans, and PSL could consequently not find a suitable match. In some of the cases, inappropriate matches were found and the robot turned in the wrong direction. In other cases, no match at all was found causing the robot to fall back on the reactive controller for a longer period and usually getting stuck in a corner.

The amount of training data used in this evaluation was fairly small. Only one demonstration from each starting position was performed. One reason why FPSL is able to solve the task despite the small amount of training is that all data potentially contribute to every action selection, independently of where in the demonstration it originally took place. Techniques that represent the whole behavior as a sequence with variations, typically require more training since information from the beginning of the demonstration does not contribute to action selection in other parts of the behavior. PSL does not relies on common features within a set of demonstrations and consequently does not require that demonstrations are compared or temporally aligned, see Section 1. In its current design, PSL is of course unable to perform a program-level imitation since it always rely on sensory-motor events, but it does not suffer from a large diversity in the demonstration set as long as the recent sensory-motor events bear necessary information to select a suitable action.

4.1 Conclusions and future work

In this chapter, we show that PSL can be used as a method for LFD, in a fairly realistic setting. The move from a discrete state space used in previous work to the continuous state space appears to have a positive effect on generalization ability and prediction performance, especially on multi-dimensional data. The next step is to conduct experiments with the physical Kompai robot (Robosoft, 2010) in an attempt to verify the results in the real world.

The fuzzy version of PSL proposed here, and specifically the introduction of context sets C, should be seen as one step towards integrating PSL in a hierarchical architecture. The higher level controller may be another instance of PSL working on a lower temporal resolution, or a completely different control system interacting with PSL by changing the responsibility $\lambda_t(C)$ for each context (Equation 5). For an actual interaction to take place, PSL also has to feed information upwards, to higher level controllers. In previous work on behavior recognition (Billing et al., 2010), we have shown that PSL can be used to compute a bottom-up signal providing information about how well each context corresponds to present circumstances. While this has not been the focus of this chapter, we intend to evaluate these aspects of PSL in future work.

5. References

Alissandrakis, A., Nehaniv, C. L. & Dautenhahn, K. (2002). Imitation With ALICE: Learning to Imitate Corresponding Actions Across Dissimilar Embodiments, *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 32: 482–496.

Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K. & Saunders, J. (2005). An Approach for Programming Robots by Demonstration: Generalization Across Different Initial Configurations of Manipulated Objects, *Proceedings of 2005 International Symposium on Computational Intelligence in Robotics and Automation*, Ieee, Espoo, Finland, pp. 61–66.

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

Arkin, R. C. (1998). *Behaviour-Based Robotics*, MIT Press.

Billard, A., Epars, Y., Cheng, G. & Schaal, S. (2003). Discovering imitation strategies through categorization of multi-dimensional data, *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Vol. 3, Las Vegas, Nevada, pp. 2398–2403 vol.3.

Billing, E. A. (2009). *Cognition Reversed - Robot Learning from Demonstration*, Lic. thesis, Umeå University, Department of Computing Science, Umeå, Sweden.

Billing, E. A. (2011). www.cognitionreversed.com.

Billing, E. A. & Hellström, T. (2010). A Formalism for Learning from Demonstration, *Paladyn: Journal of Behavioral Robotics* 1(1): 1–13.

Billing, E. A., Hellström, T. & Janlert, L. E. (2010). Behavior Recognition for Learning from Demonstration, *Proceedings of IEEE International Conference on Robotics and Automation*, Anchorage, Alaska.

Billing, E. A., Hellström, T. & Janlert, L. E. (2011). Predictive learning from demonstration, *in* J. Filipe, A. Fred & B. Sharp (eds), *Agents and Artificial Intelligence*, Springer Verlag, Berlin, pp. 186–200.

Brass, M., Bekkering, H., Wohlschläger, A. & Prinz, W. (2000). Compatibility between observed and executed finger movements: comparing symbolic, spatial, and imitative cues., *Brain and cognition* 44(2): 124–43.

Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot, *IEEE Journal of Robotics and Automation* 2(1): 14–23.

Brooks, R. A. (1991). New Approaches to Robotics, *Science* 253(13): 1227–1232.

Byrne, R. W. & Russon, A. E. (1998). Learning by Imitation: a Hierarchical Approach, *The Journal of Behavioral and Brain Sciences* 16(3).

Calinon, S. (2009). *Robot Programming by Demonstration - A Probabilistic Approach*, EFPL Press.

Calinon, S., Guenter, F. & Billard, A. (2007). 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.

de Rengervé, A., D'halluin, F., Andry, P., Gaussier, P. & Billard, A. (2010). A study of two complementary encoding strategies based on learning by demonstration for autonomous navigation task, *Proceedings of the Tenth International Conference on Epigenetic Robotics*, Lund, Sweden.

Demiris, J. & Hayes, G. (1997). Do robots ape?, *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents*, pp. 28–31.

Demiris, J. & Hayes, G. R. (2002). *Imitation as a dual-route process featuring predictive and learning components: a biologically plausible computational model*, MIT Press, pp. 327–361.

Demiris, Y. (1999). *Movement Imitation Mechanisms in Robots and Humans*, PhD thesis, University of Edinburgh.

Demiris, Y. & Johnson, M. (2003). Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning, *Connection Science* 15(4): 231–243.

Fullér, R. (1995). *Neural Fuzzy Systems*, Abo Akademi University.

Gallese, V., Fadiga, L., Fogassi, L. & Rizzolatti, G. (1996). Action recognition in the premotor cortex, *Brain* 119(2): 593–609.

Haruno, M., Wolpert, D. M. & Kawato, M. M. (2001). MOSAIC Model for Sensorimotor Learning and Control, *Neural Comput.* 13(10): 2201–2220.

Klir, G. J. & Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall.

Matarić, M. J. (1997). Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior, *Journal of Experimental and Theoretical Artificial Intelligence* 9(2-3): 323–336.

Matarić, M. J. (2002). *Sensory-motor primitives as a basis for imitation: linking perception to action and biology to robotics*, MIT Press, pp. 391–422.

Matarić, M. J. & Marjanovic, M. J. (1993). Synthesizing Complex Behaviors by Composing Simple Primitives, *Proceedings of the European Conference on Artificial Life*, Vol. 2, Brussels, Belgium, pp. 698–707.

Microsoft (2011). Microsoft Robotic Developer Studio.
URL: *http://www.microsoft.com/robotics/*

Myers, B. C. S. & Rabiner, L. R. (1981). A Comparative Study of Several Dynamic Time-Warping, *The Bell System Technical Journal* 60(7): 1389–1409.

Nehaniv, C. L. & Dautenhahn, K. (2000). *Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications*, Vol. 24, World Scientific Press, pp. 136–161.

Nehaniv, C. L. & Dautenhahn, K. (2001). Like Me? - Measures of Correspondence and Imitation, *Cybernetics and Systems* 32: 11–51.

Nicolescu, M. (2003). *A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains*, PhD thesis, University of Southern California.

Rizzolatti, G., Camarda, R., Fogassi, L., Gentilucci, M., Luppino, G. & Matelli, M. (1988). Functional organization of inferior area 6 in the macaque monkey. II. Area F5 and the control of distal movements., *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale* 71(3): 491–507.

Rizzolatti, G. & Craighero, L. (2004). The Mirror-Neuron System, *Annual Review of Neuroscience* 27: 169–192.

Robosoft (2010). www.robosoft.com.

Robosoft (2011). Kompai Robot.

Rohrer, B. (2007). S-Learning: A Biomimetic Algorithm for Learning, Memory, and Control in Robots, Kohala Coast, Hawaii, pp. 148 – 151.

Rohrer, B. & Hulet, S. (2006). BECCA - A Brain Emulating Cognition and Control Architecture, *Technical report*, Cybernetic Systems Integration Department, Univeristy of Sandria National Laboratories, Alberquerque, NM, USA.

Tani, J., Ito, M. & Sugita, Y. (2004). Self-Organization of Distributedly Represented Multiple Behavior Schemata in a Mirror System : Reviews of Robot Experiments Using RNNPB, *Neural Networks* 17: 1273–1289.

Wolpert, D. M. & Kawato, M. (1998). Multiple paired forward and inverse models for motor control.

VII

Paper VII

Simultaneous Control and Recognition of Demonstrated Behavior

Erik Billing, Thomas Hellström, and Lars-Erik Janlert

Dept. Computing Science, Umeå University, SE-901 87 Umeå, Sweden
billing@cs.umu.se, thomash@cs.umu.se, and lej@cs.umu.se
www.cs.umu.se/research/robotics

Abstract: A method for *Learning from Demonstration (LFD)* is presented and evaluated on a simulated Robosoft Kompai robot. The presented algorithm, called *Predictive Sequence Learning (PSL)*, builds fuzzy rules describing temporal relations between sensory-motor events recorded while a human operator is tele-operating the robot. The generated rule base can be used to control the robot and to predict expected sensor events in response to executed actions. The rule base can be trained under different contexts, represented as fuzzy sets. In the present work, contexts are used to represent different behaviors. Several behaviors can in this way be stored in the same rule base and partly share information. The context that best matches present circumstances can be identified using the predictive model and the robot can in this way automatically identify the most suitable behavior for precent circumstances. The performance of PSL as a method for LFD is evaluated with, and without, contextual information. The results indicate that PSL without contexts can learn and reproduce simple behaviors. The system also successfully identifies the most suitable context in almost all test cases. The robot's ability to reproduce more complex behaviors, with partly overlapping and conflicting information, significantly increases with the use of contexts. The results support a further development of PSL as a component of a dynamic hierarchical system performing control and predictions on several levels of abstraction.

Keywords: Behavior Recognition, Context Dependent, Fuzzy Logic, Learning and Adaptive Systems, Learning from Demonstration

Simultaneous Control and Recognition of Demonstrated Behavior

Erik Billing*, Thomas Hellström† and Lars-Erik Janlert‡
Department of Computing Science
Umeå University, Sweden

Abstract

A method for *Learning from Demonstration (LFD)* is presented and evaluated on a simulated Robosoft Kompai robot. The presented algorithm, called *Predictive Sequence Learning (PSL),* builds fuzzy rules describing temporal relations between sensory-motor events recorded while a human operator is tele-operating the robot. The generated rule base can be used to control the robot and to predict expected sensor events in response to executed actions. The rule base can be trained under different *contexts,* represented as fuzzy sets. In the present work, contexts are used to represent different behaviors. Several behaviors can in this way be stored in the same rule base and partly share information. The context that best matches present circumstances can be identified using the predictive model and the robot can in this way automatically identify the most suitable behavior for precent circumstances. The performance of PSL as a method for LFD is evaluated with, and without, contextual information. The results indicate that PSL without contexts can learn and reproduce simple behaviors. The system also successfully identifies the most suitable context in almost all test cases. The robot's ability to reproduce more complex behaviors, with partly overlapping and conflicting information, significantly increases with the use of contexts. The results support a further development of PSL as a component of a dynamic hierarchical system performing control and predictions on several levels of abstraction.

Index terms: Behavior Recognition, Context Dependent, Fuzzy Logic, Learning and Adaptive Systems, Learning from Demonstration

*Erik Billing (billing@cs.umu.se)
†Thomas Hellström (thomash@cs.umu.se)
‡Lars-Erik Janlert (lej@cs.umu.se)

1 Introduction

Learning from Demonstration (LFD) is a well-established technique for teaching robots new behaviors. One of the greatest challenges in LFD is to implement a learning algorithm that allows the robot pupil to generalize a sequence of actions demonstrated by the teacher such that the robot is able to perform the desired behavior under varying conditions. In earlier work (Billing et al., 2010b, 2011), we have developed and evaluated the algorithm *Predictive Sequence Learning (PSL)* as a method for LFD. PSL can be trained from demonstrations performed via tele-operation and used as a controller for robots. The algorithm treats control as a prediction problem, such that the next action is selected based on the sequence of recent sensory-motor events. In addition, PSL also produces predictions of expected sensor states. While these are not directly useful for control, predictions of sensor states appear to serve well as a method for behavior recognition (Billing et al., 2010a).

Here, we evaluate the possibility to use a context layer that interacts with the PSL algorithm, both during learning and reproduction of behaviors. The context layer activates relevant parts of the PSL knowledge base while inhibiting knowledge that could interfere with the current behavior, potentially allowing PSL to learn and reproduce more complex behaviors. The work can be seen as one step towards integrating PSL in a dynamic hierarchical learning system.

An introduction to LFD and hierarchical learning systems is presented in Section 2, followed by a more precise problem statement in Section 3. The PSL algorithm is presented in Section 4 and the problem of knowledge interference is described in Section 5. The experimental setup used for the present work is described in Section 6 and a formulation of expected results is given in Section 7. Finally, results are presented in Section 8 followed by a discussion in Section 9.

2 Background

One common approach to LFD is to map the demonstration onto a set of pre-programmed or previously learned primitives controllers (Billing & Hellström, 2010). The approach has strong connections to *behavior-based architectures* (Matarić & Marjanovic, 1993; Matarić, 1997; Arkin, 1998) and earlier reactive approaches (e.g. Brooks, 1986, 1991). When introducing behavior primitives, the LFD process can be divided into three tasks (Billing & Hellström, 2010):

1. *Behavior segmentation* where a demonstration is divided into smaller segments.

2. *Behavior recognition* where each segment is associated with a primitive controller.

3. *Behavior coordination*, referring to identification of rules or switching conditions for how the primitives are to be combined.

The approach represents one way of introducing good bias in learning and solve the generalization problem by relying on previously acquired behavioral knowledge. While there are many domain specific solutions to each one of these three subproblems, they appear very difficult to solve in the general case.

One argument for the use of behavior primitives in LFD is that known behaviors can constitute parts (i.e., primitives) of other, more complex, behaviors. If this process can be implemented in a general way, it would allow learnt behaviors to act as primitives in future learning sessions. The approach would potentially allow the robot to learn increasingly complex behaviors as its knowledge base grows, producing an hierarchical architecture of controllers (Byrne & Russon, 1998). While this approach appears to have great potential, it requires that not only pre-programmed primitives, but also controllers generated through learning, can be recognized as parts of a demonstrated behavior.

This approach has many connections to biology and specifically the mirror system (e.g. Rizzolatti et al., 1988; Gallese et al., 1996; Brass et al., 2000; Rizzolatti & Craighero, 2004). While the role of the mirror system is still highly debated, several groups of researchers propose computational models where perception and action are tightly interweaved. Among the most prominent examples are the work by Demiris & Khadhouri (2006) proposing an architecture called *Hierarchical Attentive Multiple Models for Execution and Recognition (HAMMER)*. A similar theoretical framework is presented by Haruno et al. (2003) under the name *Hierarchical Modular Selection and Identification for Control* (*HMOSAIC)*. Both these architectures implement a set of modules, where each module is an inverse model (controller) paired with a forward model (predictor). The inverse and forward models are trained together such that the forward model can predict sensor data in response to the actions produced by the inverse model. The inverse model is tuned to execute a certain behavior when the forward model produces good predictions. The prediction error is used to compute a bottom-up signal for each module. Based on the bottom-up signal, a top-down responsibility signal or confidence value is computed and propagated to each module. The output of the system is a combination of the actions produced by each inverse model, proportional to their current responsibility. The responsibility signal also controls the learning rate of each module, such that modules are only updated when their responsibility is high. In this way, modules are tuned to a specific behavior or parts of a behavior. Since the prediction error of the forward model is used as a measure of how well the specific module fits present circumstances, it can be seen as a metric of imitation (Billard et al., 2003) that is learnt together with the controller. The architecture can be composed into a hierarchical system where modules are organized in layers, with the lowest layer interacting with sensors and actuators. The bottom-up signal constitutes sensor input for the layer above and actions produced by higher levels constitute the top-down responsibility signal.

One motivation for this architecture lies in an efficient division of labor between different parts of the system. Each module can be said to operate at a specific temporal resolution. Modules at the bottom layer are given the highest temporal resolution while modules higher up in the hierarchy have decreasing

resolution, allowing these modules to express dependencies over longer periods of time. State variables that change slowly compared to a specific module's resolution are ignored by that module and are instead handled by modules higher up in the hierarchy. Slowly changing states that lead to high responsibility for the module is referred to as the module's *context*. In a similar fashion, variables that change fast in relation to the temporal resolution are handled lower in the hierarchy. This allows each module to implement a controller where the behavior depends on relatively recent states, at its level of temporal resolution. Long temporal dependencies are modeled by switching between modules, which removes the requirement for each model to capture these dependencies. Furthermore, updates of a single behavior or parts of a behavior will only require updates of a few modules and will not propagate changes to other modules. See Billing (2009) for a longer discussion on these aspects of hierarchical architectures.

The HAMMER and MOSAIC architectures make few restrictions on what kind of controllers each module should implement. We argue however, that modules should be *semi-reactive*, meaning that action selection and predictions of sensor events should be based on recent sensor and motor events. Strictly reactive modules are not desirable since each module must be able to model any dependency with a temporal resolution too high for modules at the layer above.

However, the division of behavior into modules also has a number of drawbacks. The possibility for the system to share knowledge between behaviors is limited. Moreover, the system has to combine actions produced by different modules, which may be difficult in cases when more than one module receives high responsibility.

One architecture with similarities to HAMMER and MOSAIC able to share knowledge between different behaviors is *Recurrent Neural Network with Parametric Bias (RNNPB)* (Tani et al., 2004). Both input and output layer of the network contain sensor and motor nodes as well as nodes with recurrent connections. In addition, the input layer is given a set of extra nodes, representing the PB vector. The network is trained to minimize prediction error, both by training the network using back-propagation and by changing the PB input vector. The PB vector is however updated slowly, such that it organizes into what could be seen as a context layer for the rest of the network. In addition to giving the network the ability to represent different behaviors that share knowledge, the PB vector can be used for behavior recognition.

All these architectures can be seen as examples of a larger body of work employing the motor system for perception and imitation (e.g. Atkeson & Schaal, 1997; Billard, 2001; Demiris & Hayes, 2002; Demiris & Johnson, 2003; Alissandrakis et al., 2002; George, 2008), with an emphases on being biologically plausible. While there are many important differences between these works, both in proposed architectures and the claims that they make, there is also a significant common ground. One attempt to describe this common ground was made by Billing (2009), proposing four criteria for general learning ability:

1. **Hierarchical structures**

   Knowledge gained from learning should be represented in hierarchies.

2. **Functional specificity**

   Knowledge gained from learning should be organized in functionally specialized modules.

3. **Forward and inverse**

   Prediction error reflects how well the state definition satisfies the Markov assumption, and by consequence a forward model can be used to improve knowledge representation when paired with an inverse model.

4. **Bottom-up and top-down**

   Both bottom-up and top-down signals must be propagated through the hierarchical structure. Bottom-up signals represent the state of modules, and the top-down signals specify context.

These criteria can be seen as typical properties of a system able to internalize a simulation of percepts, in response to actions. That should be understood as one way to give the robot an inner world, a simulation of the physical world that does not rely on a pre-defined physics simulator but is generated from interactions with the world. Such a simulation is inherently grounded in the robot's sensors and actuators and is therefore not subject to the symbol grounding problem (Harnad, 1990). A minimalistic implementation of this approach can be found in the work by Ziemke et al. (2005). This approach also has tight connections with the work by Barsalou and colleagues (e.g. Barsalou et al., 2003; Barsalou, 2009), describing the brain as a system simulating sensor percepts in relation to motor activity.

Rohrer & Hulet (2006) proposed an architecture called *Brain Emulating Cognition and Control Architecture* (BECCA). The focus of BECCA was to capture the discrete episodic nature of many types of human motor behavior, while limiting the use of task-specific prior knowledge. BECCA was presented as a very general reinforcement learning system, applicable to many types of learning and control problems. One of the core elements of BECCA is the temporal difference (TD) algorithm *Sequence Learning* (SL) (Rohrer, 2007; Rohrer et al., 2009). SL builds sequences of passed events which is used to predict future events, and can in contrast to other TD algorithms base its predictions on a sequence of previous states.

3 Problem statement

Inspired by BECCA (Rohrer & Hulet, 2006) and specifically SL (Rohrer, 2007; Rohrer et al., 2009), we developed the PSL algorithm as a method for LFD (Billing et al., 2010a,b). PSL has many interesting properties seen as a learning algorithm for robots. It is model free, meaning that it introduces very few assumptions into learning and does not need any task specific configuration. PSL can be seen as a variable-order Markov model. Starting out from a reactive (first-order) model, PSL estimates transition probabilities between discrete

sensory-motor states. For states that do not show Markov property, the order is increased and PSL models the transition probability based on several passed events. In this way, PSL will progressively gain memory for parts of the behavior that cannot be modeled in a reactive way.

While previous evaluations of PSL (Billing et al., 2010a,b, 2011) show that the algorithm can be used both for control and recognition of several different behaviors, PSL is subject to combinatorial explosion when the demonstrated behavior requires modeling of long temporal dependencies. PSL can however efficiently model short temporal dependencies in a semi-reactive way and is a good candidate for implementation of forward and inverse models in an architecture similar to those described above. The fact that PSL is not able to implement an arbitrary controller is here seen as an important bias and serves as a way to get around the "no free lunch" theorems (Wolpert & Macready, 1997; Ho & Pepyne, 2002). In the present work we combine PSL control with behavior recognition in order to reduce these limitations and take one step closer to a hierarchical learning systems satisfying all criteria for general learning ability (Section 2).

4 Predictive Sequence Learning

PSL builds fuzzy rules, referred to as *hypotheses* h, describing temporal dependencies between a sensory-motor event e_{t+1} and a sequence of passed events $\left(e_{t-|h|+1}, e_{t-|h|+2}, \ldots, e_t\right)$, defined up until current time t.

$$h : \left(\Upsilon_{t-|h|+1} \; is \; E^h_{|h|} \wedge \Upsilon_{t-|h|+2} \; is \; E^h_{|h|-1} \wedge \ldots \wedge \Upsilon_t \; is \; E^h_1\right) \Rightarrow \Upsilon_{t+1} \; is \; \bar{E}^h. \tag{1}$$

Υ_i is the event variable and $E^h(e)$ is a fuzzy membership function returning a membership value for a specific e. The right hand side $\bar{E}^h$ is a membership function comprising expected events at time $t+1$. $|h|$ denotes the length of h, i.e., the number of left-hand-side conditions of the rule. Both E and $\bar{E}$ are implemented as standard cone membership functions with base width ε (e.g. Klir & Yuan, 1995).

A set of hypotheses can be used to compute a prediction $\hat{e}_{t+1}$ given a sequence of passed sensory-motor events η, defined up to the current time t:

$$\eta = (e_1, e_2, \ldots, e_t). \tag{2}$$

The process of matching hypotheses to data is described in Section 4.1. The PSL learning process, where hypotheses are generated from a sequence of data, is described in Section 4.2 and interaction with the context layer is described in Section 4.3. The description of PSL given here is similar, but not identical, to Fuzzy PSL as described in our previous evaluation of this algorithm (Billing et al., 2011). A few changes to the algorithm was introduced as a result of optimizations made in order to allow on-line predictions with multiple contexts. These changes are further discussed in Section 4.4.

4.1 Matching hypotheses

Given a sequence of sensory-motor events η (Equation 2), a match $\alpha_t(h)$ of the rule is given by:

$$\alpha_t(h) = \bigwedge_{i=1}^{|h|-1} E_i^h(e_{t-i+1}) \tag{3}$$

where $\wedge$ is implemented as a *min*-function.

Hypotheses are grouped in fuzzy sets C whose membership value $C(h)$ describes the *confidence* of h at time t:

$$C(h) = \frac{\sum_{k=t^h}^{t} \alpha_k(h)\, \bar{E}^h(e_{k+1})}{\sum_{k=t^h}^{t} \alpha_k(h)} \tag{4}$$

where t^h is the creation time of h or 1 if h existed prior to training. I.e., $C(h)$ is a weighted average of how well the h predicts the event e_{k+1}, over all observation up to time t. Each set C represents a *context* and can be used to implement a specific behavior or part of a behavior. The *responsibility signal* $\lambda_t(C)$ is used to control which behavior is active at a specific time. The combined confidence value $\tilde{C}_t(h)$, for hypothesis h, is a weighted average over all C:

$$\tilde{C}_t(h) = \frac{\sum_C C(h)\, \lambda_t(C)}{\sum_C \lambda_t(C)}. \tag{5}$$

$\tilde{C}_t$ can be seen as a fuzzy set representing the active context at time t. Hypotheses contribute to a prediction in proportion to their membership in $\tilde{C}$ and the *match set* $\hat{M}$. $\hat{M}$ is defined in three steps. First, longest matching hypotheses are selected:

$$M = \{h \mid |h| \geq |h'| \ \mathit{for\ all}\ \{h' \mid \alpha(h') > 0\}\}. \tag{6}$$

The best matching $h \in M$ is selected:

$$\tilde{M} = \{h \mid \alpha(h) \geq \alpha(h') \ \mathit{for\ all}\ \{h' \in M\}\}. \tag{7}$$

Finally, the match set $\hat{M}$ is defined as:

$$\hat{M}(h) = \begin{cases} \tilde{C}(h) & h \in \tilde{M} \\ 0 & \mathit{otherwise} \end{cases}. \tag{8}$$

The aggregated prediction $\hat{E}(e_{t+1})$ is computed using the Larsen method (e.g. Fullér, 1999):

$$\hat{E}(e_{t+1}) = \bigvee_h \bar{E}_h(e_{t+1})\, \hat{M}(h). \tag{9}$$

$\hat{E}$ is converted to crisp values using a *center of max* defuzzification (e.g. Klir & Yuan, 1995, p. 337):

$$\hat{e} = \frac{\min\left\{e \mid \hat{E}(e) = \max\left(\hat{E}\right)\right\} + \max\left\{e \mid \hat{E}(e) = \max\left(\hat{E}\right)\right\}}{2}. \tag{10}$$

4.2 Generating hypotheses

Hypotheses can be generated from a sequence of sensory-motor events η. During training, PSL continuously makes predictions and creates new hypotheses when no matching hypothesis produces the correct prediction $\bar{E}$. The exact training procedure is described in Algorithm 1.

For example, consider the event sequence $\eta = abccabccabcc$. Let $t = 1$. PSL will search for a hypothesis with a left hand side matching a. Initially, the context set C is empty and PSL will create a new hypothesis $(a) \Rightarrow b$ which is added to C with confidence 1, denoted $C(a \Rightarrow b) = 1$. The same procedure will be executed at $t = 2$ and $t = 3$ such that $C((b) \Rightarrow c) = 1$ and $C((c) \Rightarrow c) = 1$. At $t = 4$, PSL will find a matching hypothesis $(c) \Rightarrow c$ producing the wrong prediction c. Consequently, a new hypothesis $(c) \Rightarrow a$ is created and confidences are updated such that $C((c) \Rightarrow c) = 0.5$ and $C((c) \Rightarrow a) = 1$. The new hypothesis receives a higher confidence since confidence values are calculated from the creation time of the hypothesis (Equation 4). The predictions at $t = 5$ and $t = 6$ will be correct and no new hypotheses are created. At $t = 7$, both $(c) \Rightarrow a$ and $(c) \Rightarrow c$ will contribute to the prediction $\hat{E}$. Since the confidence of $(c) \Rightarrow a$ is higher than that of $(c) \Rightarrow c$, $\hat{E}$ will defuzzify towards a, producing the wrong prediction (Equation 10). As a result, PSL creates a new hypothesis $(b, c) \Rightarrow c$. Similarly, $(c, c) \Rightarrow a$ will be created at $t = 8$. PSL is now able to predict all elements in the sequence perfectly and no new hypotheses are created.

Source code from the implementation used in the present work is available online (Billing, 2011).

4.3 Computing context responsibility

PSL is not only used as a controller but also as a method for behavior recognition. By letting PSL compute one prediction for each context, the responsibility of each context can be changed based on the size of respective prediction error. The method used here has strong similarities with the responsibility update mechanism used in the MOSAIC architecture (Haruno et al., 2001). Similar mechanisms can also be found in other learning and control frameworks with hierarchical structure (e.g. Demiris & Khadhouri, 2006).

One important difference between PSL and most other approaches is however that the context layer of PSL allows partial knowledge overlap between contexts. Furthermore, this overlap may be fuzzy in the sense that each hypothesis is a member of the context to a certain degree. This allows a much more flexible organization of knowledge compared to an architecture that requires each module

Algorithm 1 Predictive Sequence Learning (PSL)

Require: $\psi = (e_1, e_2, \ldots, e_T)$ where T denotes the length of the training set
Require: $\hat{\alpha}$ as the precision constant, see text

1: **let** $t \leftarrow 1$
2: **let** $\eta = (e_1, e_2, \ldots, e_t)$
3: **let** $C \leftarrow \varnothing$
4: **let** $\hat{E}$ as Equation 9
5: **if** $\hat{E}(e_{t+1}) < \hat{\alpha}$ **then**
6: **let** $h_{new} = CreateHypothesis(\eta, C)$ as defined by Algorithm 2
7: $C(h_{new}) \leftarrow 1$
8: **end if**
9: Update confidences $C(h)$ as defined by Equation 4
10: **set** $t = t + 1$
11: **if** t<T **then**
12: **goto** 2
13: **end if**

Algorithm 2 CreateHypothesis

Require: $\eta = (e_1, e_2, \ldots, e_t)$
Require: $C : h \rightarrow [0, 1]$
Require: α as defined by Equation 3

1: **let** $\hat{M}(h)$ as Equation 8
2: **let** $\bar{M} = \left\{ h \mid \bar{E}^h(e_{t+1}) \geq \hat{\alpha} \wedge \hat{M}(h) > 0 \right\}$ where $\hat{\alpha}$ is the precision constant, see Section 4.4
3: **if** $\bar{M} = \emptyset$ **then**
4: **let** E^* be a new membership function with center e_t and base ε
5: **return** $h_{new} : (\Upsilon_t \; is \; E^*) \Rightarrow \Upsilon_{t+1} \; is \; \bar{E}$
6: **else**
7: **let** $\bar{h} \in \bar{M}$
8: **if** $C(\bar{h}) = 1$ **then**
9: **return** *null*
10: **else**
11: **let** E^* be a new membership function with center $e_{t-|\bar{h}|}$ and base ε
12: **return** $h_{new} : \left(\Upsilon_{t-|\bar{h}|} \; is \; E^*, \Upsilon_{t-|\bar{h}|+1} \; is \; E^{\bar{h}}_{|\bar{h}|-1}, \ldots, \Upsilon_t \; is \; E^{\bar{h}}_0 \right) \Rightarrow \Upsilon_{t+1} \; is \; \bar{E}$
13: **end if**
14: **end if**

to be strictly separated from other modules. Several contexts may also be active simultaneously, without the need for a separate action coordination mechanism.

Let the $\hat{E}_t^C$ be the prediction for context C at time t, as given by Equation 9. The prediction for each context is calculated with the responsibility signal $\lambda_t(C) = 1$ and the responsibility of all other contexts equal to zero (see Equation 5). Based on these predictions, the responsibility signal for each context is updated using Bayes' rule:

$$\lambda_t(C) = \frac{\lambda_{t-1}(C) \exp\left(\frac{\left(E_t^C(e_t)-1\right)^2}{2\sigma^2}\right)}{\sum_{i=1}^{N}\left[\lambda_{t-1}(C_i) \exp\left(\frac{\left(E_t^{C_i}(e_t)-1\right)^2}{2\sigma^2}\right)\right]} \tag{11}$$

where e_t represents the observed sensory-motor event at time t. N is the number of contexts and σ^2, corresponding to the variance, is used as a scaling constant controlling the size of confidence changes in relation to prediction error size.

4.4 Parameters and implementation

A clear description of parameters is important for any learning algorithm. Parameters always introduce the risk that the learning algorithm is tuned towards the evaluated task, producing better results than it would in the general case. We have therefore strived towards limiting the number of parameters of PSL. The original design of PSL was completely parameter free, with the exception that continuous data was discretized using some discretization method. The version of PSL proposed here can be seen as a generalization of the original algorithm (Billing et al., 2010b,a) where the width ε of the membership function E determines the discretization resolution. In addition, a second parameter is introduced, referred to as the *precision constant* $\hat{\alpha}$. $\hat{\alpha}$ is with fuzzy logic terminology an α-*cut*, i.e., a threshold over the fuzzy membership function in the interval $[0, 1]$ (e.g., Klir & Yuan, 1995).

ε controls how generously PSL matches hypotheses. A high ε makes the algorithm crisp but typically increases the precision of predictions when a match is found. Conversely, a low ε reduces the risk that PSL reaches unobserved states at the cost of a decreased prediction performance. The high value of ε can be compared to a fine resolution data discretization for the previous version of PSL.

$\hat{\alpha}$ is only used during learning, controlling how exact a specific $\bar{E}$ has to be before a new hypothesis with a different $\bar{E}$ is created. A large $\hat{\alpha}$ reduces prediction error but typically results in more hypotheses being created during learning.

Both ε and $\hat{\alpha}$ control the tolerance to random variations in data and can be set based on how precise we want that PSL to model the data. Small ε in combination with large $\hat{\alpha}$ will result in a model that closely fits training data, typically producing small prediction errors but also requires more training data in order to cover the state space.

Updates of the responsibility signal (Equation 11) introduces a third parameter (the variance σ^2) scaling the prediction error and consequently controlling how quickly the responsibility signal changes. While this parameter could be estimated form actual data, it was manually set to a fixed value in the present work.

The original implementation of Fuzzy PSL (Billing et al., 2011) requires that the prediction for each context is computed separately, significantly increasing the computational load for each new context. For the present work, the algorithm was therefore reimplemented such that a majority of computations to be shared for all contexts, resulting in a minor extra load when multiple contexts are used. Even though no deeper study comparing the two versions of the algorithm has been made, our initial tests did not indicate any significant difference other than reduced computational requirements. In earlier work (Billing et al., 2010b,a), we used a discrete version of PSL that however differs from the present algorithm in several ways. See Billing et al. (2011) for details.

5 Knowledge interference

In early evaluations of PSL (Billing et al., 2010a,b), we noticed that increased training could affect the performance in both a positive and a negative way. On the positive side, more demonstrations provide more sensory-motor patterns that PSL can reuse in many situations. As long as the local sensory-motor history provides enough information to reliably separate between two situations, more training is always positive. However, when the recent sensory-motor history does not provide reliable information to select the right action, PSL produces longer hypotheses in order to increase prediction performance. While this in itself is not a large problem, it increases the risk for inappropriate action selection when PSL is used as a controller. If the current sensory-motor history does not match any long hypothesis, PSL falls back on shorter, less reliable, hypotheses. As a result, PSL sometimes selects an inappropriate action. We call this problem *knowledge interference*, since knowledge of one behavior or part of a behavior is interfering with the behavior currently being executed.

One potential solution to this problem is to separate behavioral knowledge into several contexts, and let a behavior recognition mechanism select one or several context that should be responsible for the present situation. Hypotheses that are strongly associated with the active contexts are prioritized over other hypotheses, and hypotheses that would have interfered with the current behavior can in this way be ignored. This can be seen as one way to achieve the criterion of Functional specificity presented in Section 2.

We have previously evaluated several techniques for behavior recognition (Billing & Hellström, 2008) and also shown that PSL can be used for behavior recognition (Billing et al., 2010a), based on the same model as used for control. We are however not aware of any previous work that connects the behavior recognition capabilities of PSL with a PSL based controller, such that the robot can continuously evaluate the responsibility of each context and in this way

reduce the problem of knowledge interference.

6 Experimental setup

In order to evaluate PSL, a simulated Robosoft Kompai robot (Robosoft, 2011) was used in the Microsoft RDS simulation environment (Microsoft, 2011). The 270 degree laser scanner of the Kompai was used as source for sensor data and the robot was controlled by setting linear and angular speeds. We used a similar setup in previous work (Billing et al., 2011) and here extend earlier tests to include the new features of PSL.

Demonstrations were performed via tele-operation using a joypad, while sensor and motor data was recorded with a temporal resolution of 20 Hz. The dimensionality of the laser scanner was reduced to 20 dimensions using an average filter. Angular and linear speeds were fed directly into PSL. ϵ was set to 0.8 m for laser data and 0.1 m/s for motor data. $\hat{\alpha} = 0.95$ was used for both sensor and motor data and σ^2 was set to 5.

In cases PSL does not find a match and is unable to produce a prediction, a reactive obstacle avoidance was used to control the robot. While PSL normally has full control over the robot, it can run into unknown states for a short periods of time, usually when close to walls and obstacles. The obstacle avoidance can in these cases prevent a collision. The robot may of course still have contact with objects as long as PSL is in control.

Four behaviors were used, each one demonstrated ten times with some variations. Each behavior is described below. Numbered areas are illustrated in Figure 1. The behaviors were intentionally designed to overlap, such that the robot would experience similar situations in parts of several behaviors.

ToTV Started from various locations in the apartment (area 2, 4, 5, and 6) and finished in front of the TV (area 1).

Wake Started close to the bed (area 3) and finished in the corridor (area 5).

ToKitchen Started in the hallway (area 7), made a left turn in the corridor (area 5) followed by a right turn towards the kitchen, finally turning around and stopping in the kitchen (area 2).

Serve Started in the kitchen (area 2), went clockwise around the table, slowly passing by each chair one by one, through area 8, and back to the kitchen. The behavior finished by turning around and stop.

6.1 Evaluation of simultaneous control and recognition

In order to test how well PSL could reproduce the demonstrated behaviors, and generalize, ten test cases were designed.

Case 1 PSL was trained on demonstrations of the ToTV behavior. Tests are made starting from area 3.

Figure 1: The simulated apartment environment used for evaluation. Numbered regions indicate critical areas used as reference for demonstrations and reproduced behaviors (see text).

Case 2 PSL was trained on demonstrations from the ToTV and Wake behaviors. Tests were made starting form area 3.

Case 3 PSL was trained on demonstrations from the ToKitchen behavior. Tests were made starting from area 7.

Case 4 PSL was trained on demonstrations from the Serve behavior. Tests were made starting from area 2.

Case 5, 6, and 7 PSL was trained with demonstrations of all four behaviors. The training data was not separated into different behaviors but trained and represented as a single PSL context. Tests were made starting from area 3 (case 5), area 7 (case 6) and area 2 (case 7).

Case 8, 9, and 10 PSL was trained on demonstrations from all four behaviors. The training data was separated into four different contexts, such that each context represented one behavior. Behavior recognition was used to continuously update the responsibility for each context. Tests were made starting from area 3 (case 8), area 7 (case 9) and area 10 (case 7).

Apart from the demonstrated data, the robot did not get any information of which behavior to execute at a certain time. When trained on more than one behavior, PSL had to recognize the present starting location and use this information to select the appropriate behavior.

6.2 Evaluation of behavior recognition during manual control

In addition to the evaluation described in the previous section, PSL was evaluated as a method for behavior recognition during manual control. This can be seen as an attempt to reproduce our previous results (Billing et al., 2010a) in a more realistic setting.

A single demonstration was made starting from area 7, moving out of the room and turning left towards area 6. After approximately 17 seconds, the robot reaches area 3, turns around and goes back towards area 6, out of the bedroom and reaches area 5 at t = 30 s. The robot continuous with the table to its right, passes area 8 at t = 37 s and makes a right turn around the table towards the kitchen. After 43 seconds of driving, the robot reaches the kitchen, turns around, leaves the kitchen at t = 48 s and makes a second lap around the table. When reaching area 8 for the second time (t = 56 s) the robot makes a left turn towards the TV and parks in front of the TV after a total of 67 seconds of driving.

7 Hypothesis

Since PSL represents behaviors as a semi-reactive controller, no specific coordination is required in order to merge two demonstrated behaviors. One example

when this is useful is when an existing behavior is to be extended to work in a partly new environment. In order to evaluate this aspect of PSL, test cases 1 and 2 were designed. Since no demonstrations in the ToTV behavior was made starting from the bed (area 3), test case 1 is expected to be a difficult task to reproduce. In test case 2, the original demonstration set is however combined with the demonstrated Wake behavior, showing how to exit the bedroom. The performance of test case 2 is therefore expected to be significantly higher than for case 1.

While the problem of knowledge interference (Section 5) may occur both within and between behaviors, the risk clearly increases when the robot is thought to act differently in several similar situations. Billing et al. (2010b) successfully taught a Khepera robot three partial behaviors, but when they were combined into a complete behavior, none of the partial behaviors could be reproduced correctly. In the present work, we aim to reproduce this scenario in a more realistic setting. Test cases 2, 3, and 4 represent three fairly simple behaviors that PSL should be able to reproduce when thought separately. However, since the three behaviors have significant overlaps, knowledge of one behavior may interfere with knowledge of another, and the performance of is therefore expected to decrease when all behaviors are trained together (case 5, 6, and 7).

Test cases 8, 9, and 10 were designed to evaluate the effect of behavior recognition during execution of the three different behaviors. If the behavior recognition system works as intended, the performance of case 8, 9, and 10 should be significantly higher than for cases 5, 6, and 7, respectively.

8 Results

Results for all ten test cases are summarized in Table 1. In test case 1, the robot were only able to exit the bedroom in 12 out of 20 runs, but reached the TV (area 1) every time it exited the bedroom. In test case 6, the robot reached the kitchen in 12 out of 20 runs, but took the right way only twice. During the other 10 runs, the robot went around the table as demonstrated in the Serve behavior, and in this way reached the kitchen. Similarly, in test case 9, the robot reached the kitchen in 15 out of 20 runs, and took the demonstrated path to the kitchen 13 times.

Figure 2 displays the responsibility signals from one execution of test case 8. The robot starts from area 3 (see Figure 1) with initially equal responsibilities for all four behaviors. The behavior recognition system quickly recognizes the present situation as a Wake behavior and the robot consequently starts to execute that behavior. The robot reaches area 5 after approximately 10 seconds. When continuing through the corridor, the Wake behavior no longer matches present circumstances causing a shift to the ToTV behavior. The robot also passed by the corridor during the ToKitchen behavior, making that behavior a possible candidate. Since the ToTV behavior had some demonstrations also from area 6, it receives increased confidence earlier than ToKitchen. As a result,

ToTV takes control of the robot as the responsibility of the Wake behavior decreases, quickly suppressing ToKitchen. After approximately 20 seconds, when the robot is passing by area 8, both ToTV and Serve have small prediction errors, causing slight fluctuations of the responsibilities. The high prior for ToTV will however cause the system to remain with that behavior. Finally, after approximately 30 seconds, the robot parks in front of the TV (area 1). Figure 3 displays the results from the evaluation of behavior recognition during manual control (Section 6.2).

9 Discussion

On the whole, results presented in Section 8 match our expectations (Section 7). Without the use of behavior recognition (case 5, 6 and 7), the robot successfully reproduces the demonstrated behavior only in 17 out of 60 trails. This is a clear case of what we call knowledge interference (Section 5) since the same set of demonstrations, when divided up in different behaviors (case 2, 3 and 4), resulted in successful reproductions in almost all trails. This problem is reduced when behavior recognition is active (case 8, 9 and 10), increasing the correctly reproduced trails to 45 out of 60.

The responsibility signals computed by the behavior recognition system (figures 2 and 3) is much more stable than previous evaluations have shown (Billing et al., 2010a). We believe that this can partly be explained by the large dimensionality of the laser scanner data, compared to the infrared proximity sensors of the Khepera robot used in previous experiments. Another reason may be that the fuzzy version of PSL used in this work produces smaller, and more reliable, prediction errors than the discrete version of PSL previously used. See Billing et al. (2011) for a comparison.

The results from behavior recognition during manual control (Figure 3) indicate that the responsibility signal serves as an extra memory for PSL. During t = 30 to 37s and t = 48 to 56s, the robot drives around the table in a similar way as during both ToTV and Serve behaviors (see Section 6.2 for details). The first episode is interpreted as a ToTV behavior while the Serve behavior receives the highest responsibility during the second episode. The reason for this difference is that the responsibility of each context is updated based on its prior responsibility (Equation 11), allowing the responsibilities to remain unchanged as long as the most active context does not produce significantly larger prediction errors than any other context. While this property may be a disadvantage in some situations, we argue that it serves as a powerful way to activate relevant parts of the PSL knowledge base based on information much further back in time than PSL can represent using hypotheses alone. At the same time, the reactive properties of the system remains. Even though the responsibilities for the ToTV and Serve behaviors remain stable during the manual demonstration, the system is able to reevaluate this interpretation as soon as the demonstrated behavior is diverging from its expected path (e.g., t = 37 and t = 56).

One of the weak parts of the present approach is that it still relies on a

Test case	Reached TV	Reached Kitchen	Fail
1: ToTV	12	0	8
2: ToTV + Wake	18	0	2
3: ToKitchen	0	19	1
4: Serve	0	19	1
5: All in one context	1	13	6
6: All in one context	5	2 (12)	3
7: All in one context	5	14	1
8: All in separate contexts	15	3	2
9: All in separate contexts	3	13 (15)	2
10: All in separate contexts	0	17	3

Table 1: Results for the ten test cases.

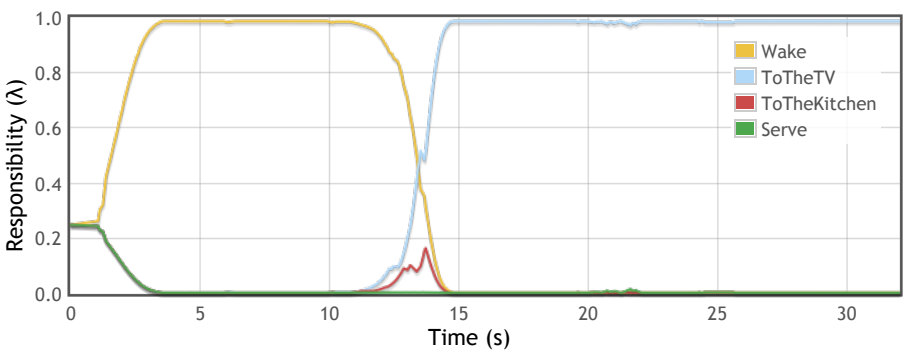

Figure 2: Typical responsibility signal from one execution of test case 8.

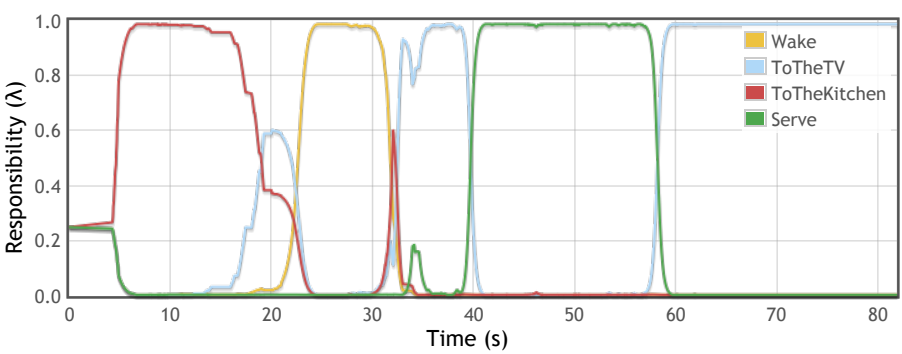

Figure 3: Responsibility signal during manual control of the robot (Section 6.2).

human to divide the demonstrated data in a way that reduces knowledge interference. We see no reason to believe that the human's interpretation of what is one behavior, and what is another, perfectly corresponds to the robot's need to categorize experiences. As a result, there may be other ways to divide the set of demonstrations used in this work (a total of 40 demonstrations) such that the results would be significantly better, or significantly worse. One way to get away from this problem may be to identify contexts (Equation 4) automatically, for example based on an entropy measure. While hypotheses frequently selected in sequence should go into the same context, each context should keep the amount of conflicting hypotheses low. This could possibly be formulated as an optimization problem where the total entropy over all contexts is to be minimized. Exploring this possibility to automatically identify contexts that potentially could be interpreted as behaviors from a human's point of view is part of future work.

Another interesting feature of the architecture presented here is that it provides a clean interface to higher level control. From an engineering point of view, PSL could be seen as a reactive layer, similar to the ones frequently used in hybrid systems, but with the ability to also feed information to the deliberating parts of the system. PSL could consequently constitute both a semi-reactive actuator layer and a sensor layer. Some research in this direction is already taking place, PSL have for example been considered for integration with a high level controller based on semantic networks (Fonooni et al., 2012).

The architecture could also be extended to a hierarchical structure with one instance of PSL running on each layer in the system. The lowest layer would interact with sensors and actuators of the robot. Layers higher up in the hierarchy would interact with the responsibility signals of the layer directly below and in this way affect the behavior of the system as a whole. The temporal resolution of PSL would decrease upwards in the hierarchy, allowing hypotheses on higher levels to extend over longer periods of time. Similarly to the hierarchical architectures discussed in Section 1, information in form of prediction errors would also be propagated as inputs to the layer directly above.

While a hierarchical version of PSL has many strong similarities with both HAMMER (Demiris & Khadhouri, 2006) and HMOSAIC (Haruno et al., 2003) there are also important differences. PSL contexts can share information in a straightforward way, not directly possible with the strictly separated modules proposed by HAMMER and HMOSAIC. We also make a emphasis on modules that are semi-reactive, following the division of labor between layers presented in Section 1. Both these properties are to large extent shared with RNNPB. While a recurrent neural network could be expected to handle dimensionality better than PSL, it may have drawbacks in the sense that it is more difficult to progressively extend the models size in the way PSL does. We hope to be able to conduct a comparison between PSL and RNNPB in future work.

References

Alissandrakis, A., Nehaniv, C. L., & Dautenhahn, K. (2002). Imitation With ALICE: Learning to Imitate Corresponding Actions Across Dissimilar Embodiments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 32, 482–496.

Arkin, R. C. (1998). *Behaviour-Based Robotics.* MIT Press.

Atkeson, C. G. & Schaal, S. (1997). Robot learning from demonstration. In D. H. Fisher Jr (Ed.), *Proceedings of the 14th International Conference on Machine Learning*, number 1994 (pp. 12–20). Nashville.

Barsalou, L. W. (2009). Simulation, situated conceptualization, and prediction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521), 1281–1289.

Barsalou, L. W., Simmons, K. W., Barbey, A. K., & Wilson, C. D. (2003). Grounding conceptual knowledge in modality-specific systems. *Trends in Cognitive Sciences*, 7(2), 84–91.

Billard, A. (2001). Learning motor skills by imitation: a biologically inspired robotic model. *Cybernetics and Systems*, 32, 155–193.

Billard, A., Epars, Y., Cheng, G., & Schaal, S. (2003). Discovering imitation strategies through categorization of multi-dimensional data. In *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, volume 3 (pp. 2398–2403 vol.3). Las Vegas, Nevada.

Billing, E. A. (2009). *Cognition Reversed - Robot Learning from Demonstration.* Licentiate thesis, Umeå University, Department of Computing Science, Umeå, Sweden.

Billing, E. A. (2011). www.cognitionreversed.com.

Billing, E. A. & Hellström, T. (2008). Behavior Recognition for Segmentation of Demonstrated Tasks. In *IEEE SMC International Conference on Distributed Human-Machine Systems* (pp. 228–234). Athens, Greece.

Billing, E. A. & Hellström, T. (2010). A Formalism for Learning from Demonstration. *Paladyn: Journal of Behavioral Robotics*, 1(1), 1–13.

Billing, E. A., Hellström, T., & Janlert, L. E. (2010a). Behavior Recognition for Learning from Demonstration. In *Proceedings of IEEE International Conference on Robotics and Automation* Anchorage, Alaska.

Billing, E. A., Hellström, T., & Janlert, L. E. (2010b). Model-free Learning from Demonstration. In J. Filipe, A. Fred, & B. Sharp (Eds.), *Proceedings of 2nd International Conference on Agents and Artificial Intelligence (ICAART)* (pp. 62–71). Valencia, Spain.

Billing, E. A., Hellström, T., & Janlert, L. E. (2011). Robot Learning from Demonstration using Predictive Sequence Learning. In A. Dutta (Ed.), *Robotic Systems - Applications, Control and Programming (to appear)*. In-Tech.

Brass, M., Bekkering, H., Wohlschläger, A., & Prinz, W. (2000). Compatibility between observed and executed finger movements: comparing symbolic, spatial, and imitative cues. *Brain and cognition*, 44(2), 124–43.

Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Journal of Robotics and Automation*, 2(1), 14–23.

Brooks, R. A. (1991). New Approaches to Robotics. *Science*, 253(13), 1227–1232.

Byrne, R. W. & Russon, A. E. (1998). Learning by Imitation: A Hierarchical Approach. *The Journal of Behavioral and Brain Sciences*, 16(3).

Demiris, J. & Hayes, G. R. (2002). Imitation as a dual-route process featuring predictive and learning components: A biologically plausible computational model. In K. Dautenhahn & C. L. Nehaniv (Eds.), *Imitation in animals and artifacts* (pp. 327–361). Cambridge, MA, USA: MIT Press.

Demiris, Y. & Johnson, M. (2003). Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. *Connection Science*, 15(4), 231–243.

Demiris, Y. & Khadhouri, B. (2006). Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54(5), 361–369.

Fonooni, B., Hellström, T., & Janlert, L. E. (2012). Learning High-Level Behaviors from Demonstration through Semantic Networks. In *Proceedings of the 4th International Conference on Agents and Artificial Intelligence (ICAART) (to appear)* Vilamoura, Algarve, Portugal.

Fullér, R. (1999). *Neural Fuzzy Systems*. Physica-Verlag GmbH & Co.

Gallese, V., Fadiga, L., Fogassi, L., & Rizzolatti, G. (1996). Action recognition in the premotor cortex. *Brain*, 119(2), 593–609.

George, D. (2008). *How the Brain might work: A Hierarchical and Temporal Model for Learning and Recognition*. Phd thesis, Stanford University.

Harnad, S. (1990). The Symbol Grounding Problem. *Physica*, D(42), 335–346.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10), 2201–2220.

Haruno, M., Wolpert, D. M., & Kawato, M. (2003). Hierarchical MOSAIC for movement generation. In *International Congress Series 1250* (pp. 575– 590).: Elsevier Science B.V.

Ho, Y. C. & Pepyne, D. L. (2002). Simple Explanation of the No-Free-Lunch Theorem and its Implications. *Journal of Optimization Theory and Applications*, 115(3), 549–570.

Klir, G. J. & Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall.

Matarić, M. J. (1997). Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3), 323–336.

Matarić, M. J. & Marjanovic, M. J. (1993). Synthesizing Complex Behaviors by Composing Simple Primitives. In *Proceedings of the European Conference on Artificial Life*, volume 2 (pp. 698–707). Brussels, Belgium.

Microsoft (2011). Microsoft Robotic Developer Studio, www.microsoft.com/robotics.

Rizzolatti, G., Camarda, R., Fogassi, L., Gentilucci, M., Luppino, G., & Matelli, M. (1988). Functional organization of inferior area 6 in the macaque monkey. II. Area F5 and the control of distal movements. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 71(3), 491–507.

Rizzolatti, G. & Craighero, L. (2004). The Mirror-Neuron System. *Annual Review of Neuroscience*, 27, 169–192.

Robosoft (2011). Kompai Robot, www.robosoft.com.

Rohrer, B. (2007). S-Learning: A Biomimetic Algorithm for Learning, Memory, and Control in Robots. In *Proceedings of the 3rd International IEEE/EMBS Conference on Neural Engineering* (pp. 148 – 151). Kohala Coast, Hawaii.

Rohrer, B., Bernard, M., Morrow, J. D., Rothganger, F., & Xavier, P. (2009). Model-free Learning and Control in a Mobile Robot. In *Proceedings of the Fifth International Conference on Natural Computation* (pp. 566–572). Tianjin, China.

Rohrer, B. & Hulet, S. (2006). *BECCA - A Brain Emulating Cognition and Control Architecture*. Technical report, Cybernetic Systems Integration Department, Sandria National Laboratories, Alberquerque, NM, USA.

Tani, J., Ito, M., & Sugita, Y. (2004). Self-Organization of Distributedly Represented Multiple Behavior Schemata in a Mirror System : Reviews of Robot Experiments Using RNNPB. *Neural Networks*, 17, 1273–1289.

Wolpert, D. H. & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.

Ziemke, T., Jirenhed, D. A., & Hesslow, G. (2005). Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68, 85–104.