
Modeling and Learning of Complex Motor Tasks: A Case Study with Robot Table Tennis

**Modellierung und Lernen von komplexen motorischen Aufgaben anhand von Fallstudien in
Roboter-Tischtennis**

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Dipl.-Inf. Katharina Mülling aus Prenzlau
Juni 2013 – Darmstadt – D 17



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Intelligente Autonome Systeme

Modeling and Learning of Complex Motor Tasks: A Case Study with Robot Table Tennis
Modellierung und Lernen von komplexen motorischen Aufgaben anhand von Fallstudien in
Roboter-Tischtennis

Genehmigte Dissertation von Dipl.-Inf. Katharina Mülling aus Prenzlau

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Tamim Asfour

Tag der Einreichung: 11. Juni 2013

Tag der Prüfung: 23. Juli 2013

Darmstadt – D 17

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-35576

URL: <http://tuprints.ulb.tu-darmstadt.de/35576>

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de



Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung – Keine kommerzielle Nutzung – Keine Bearbeitung 2.0 Deutschland

<http://creativecommons.org/licenses/by-nc-nd/2.0/de/>

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 11. Juni 2013

(K. Mülling)



Abstract

Most tasks that humans need to accomplish in their everyday life require certain motor skills. Although most motor skills seem to rely on the same elementary movements, humans are able to accomplish many different tasks. Robots, on the other hand, are still limited to a small number of skills and depend on well-defined environments. Modeling new motor behaviors is therefore an important research area in robotics. Computational models of human motor control are an essential step to construct robotic systems that are able to solve complex tasks in a human inhabited environment. These models can be the key for robust, efficient, and human-like movement plans. In turn, the reproduction of human-like behavior on a robotic system can be also beneficial for computational neuroscientists to verify their hypotheses. Although biomimetic models can be of great help in order to close the gap between human and robot motor abilities, these models are usually limited to the scenarios considered. However, one important property of human motor behavior is the ability to adapt skills to new situations and to learn new motor skills with relatively few trials. Domain-appropriate machine learning techniques, such as supervised and reinforcement learning, have a great potential to enable robotic systems to autonomously learn motor skills.

In this thesis, we attempt to model and subsequently learn a complex motor task. As a test case for a complex motor task, we chose robot table tennis throughout this thesis. Table tennis requires a series of time critical movements which have to be selected and adapted according to environmental stimuli as well as the desired targets. We first analyze how humans play table tennis and create a computational model that results in human-like hitting motions on a robot arm. Our focus lies on generating motor behavior capable of adapting to variations and uncertainties in the environmental conditions. We evaluate the resulting biomimetic model both in a physically realistic simulation and on a real anthropomorphic seven degrees of freedom Barrett WAM robot arm.

This biomimetic model based purely on analytical methods produces successful hitting motions, but does not feature the flexibility found in human motor behavior. We therefore suggest a new framework that allows a robot to learn cooperative table tennis from and with a human. Here, the robot first learns a set of elementary hitting movements from a human teacher by kinesthetic teach-in, which is compiled into a set of motor primitives. To generalize these movements to a wider range of situations we introduce the *mixture of motor primitives* algorithm. The resulting motor policy enables the robot to select appropriate motor primitives as well as to generalize between them. Furthermore, it also allows to adapt the selection process of the hitting movements based on the outcome of previous trials. The framework is evaluated both in simulation and on a real Barrett WAM robot. In consecutive experiments, we show that our approach allows the robot to return balls from a ball launcher and furthermore to play table tennis with a human partner.

Executing robot movements using a biomimetic or learned approach enables the robot to return balls successfully. However, in motor tasks with a competitive goal such as table tennis, the robot not only needs to return the balls successfully in order to accomplish the task, it also needs an adaptive strategy. Such a higher-level strategy cannot be programmed manually as it depends on the opponent and the abilities of the robot. We therefore make a first step towards the goal of acquiring such a strategy and investigate the possibility of inferring strategic information from observing humans playing table tennis. We model table tennis as a Markov decision problem, where the reward function captures the goal of the task as well as knowledge on effective elements of a basic strategy. We show how this reward function, and therefore the strategic information can be discovered with model-free inverse reinforcement learning from human table tennis matches. The approach is evaluated on data collected from players with different playing styles and skill levels. We show that the resulting reward functions

are able to capture expert-specific strategic information that allow to distinguish the expert among players with different playing skills as well as different playing styles.

To summarize, in this thesis, we have derived a computational model for table tennis that was successfully implemented on a Barrett WAM robot arm and that has proven to produce human-like hitting motions. We also introduced a framework for learning a complex motor task based on a library of demonstrated hitting primitives. To select and generalize these hitting movements we developed the mixture of motor primitives algorithm where the selection process can be adapted online based on the success of the synthesized hitting movements. The setup was tested on a real robot, which showed that the resulting robot table tennis player is able to play a cooperative game against a human opponent. Finally, we could show that it is possible to infer basic strategic information in table tennis from observing matches of human players using model-free inverse reinforcement learning.

Zusammenfassung

Menschen üben motorische Fähigkeiten, wie das Greifen einer Kaffeetasse oder das Fangen eines Gegenstandes mit großer Leichtigkeit aus. Sogar schwierigere und komplexe Aufgaben wie Fahrrad fahren oder Tischtennis spielen sind oft bis zu einem gewissen Grad schnell zu erlernen. Auch wenn viele dieser Fähigkeiten nur auf einer kleinen Anzahl elementarer Bewegungen beruhen, ist der Mensch dennoch in der Lage eine Vielzahl unterschiedlicher Aufgaben zu bewältigen. Roboter hingegen sind immer noch festgelegt auf eine bestimmte Anzahl motorischer Abläufe, die in wohl definierten Arbeitsumgebungen ausgeführt werden. Die Modellierung und das Lernen motorische Fähigkeiten ist daher ein wichtiger Aspekt in der Robotik. Mathematische Modelle der motorischen Kontrolle des Menschen können daher genutzt werden um Roboter zu entwickeln, die in der Lage sind komplexe Aufgaben in einem von Menschen bewohnten Umfeld zu bewältigen. Solche Modelle können der Schlüssel zu robusten, effizienten und menschenähnlichen Bewegungsabläufen sein. Im Gegenzug kann die Reproduktion von menschenähnlichen Bewegungsverhalten auf Robotern auch nützlich sein, um diese mathematischen Modelle zu verifizieren.

Auch wenn biomimetische Modelle eine große Hilfe sein können, um die Lücke zwischen Mensch und Roboter zu schließen, stellen sie dennoch einen fixen Plan dar, der auf eine bestimmte Anzahl von Szenarien begrenzt ist. Eine wichtige Eigenschaft des Menschen ist jedoch die Fähigkeit, motorische Abläufe an neue Gegebenheiten anzupassen und neue Bewegungen mit relativ wenigen Versuchen zu lernen. Domänenspezifische Verfahren des maschinellen Lernens, wie überwachtes Lernen und Reinforcement-Learning (Lernen durch Versuch und Fehlschlag), haben ein großes Potential um in der Robotik das autonome Lernen von motorischen Fähigkeiten zu ermöglichen.

Das Ziel dieser Doktorarbeit ist es eine komplexe motorische Aufgabe zu modellieren und anschließend zu lernen. Als Fallbeispiel verwenden wir Tischtennis. Im Tischtennis kommt es nicht nur darauf an, eine Bewegung bis zur Perfektion zu erlernen. Vielmehr besteht die Aufgabe aus mehreren zeitkritischen Bewegungen, die aufgrund spezifischer Reize der Umgebung ausgewählt, kombiniert und an neue Anforderungen angepasst werden müssen.

In dieser Arbeit analysieren wir zunächst Charakteristiken der menschlichen Bewegungskoordination im Tischtennis und erstellen anhand dessen ein mathematisches Modell, welches in der Lage ist menschenähnliche Schlagbewegungen auf einem Roboterarm zu erzeugen. Unser Fokus liegt dabei auf der Erzeugung von Bewegungsabläufen, die mit verschiedenen Variationen und Unsicherheiten der Umgebung umgehen können. Das resultierende biomimetische Modell wird sowohl in einer physikalisch realistischen Simulation, als auch auf einem realen antropomorphischen Barrett WAM Roboterarm mit sieben Freiheitsgraden getestet.

Das biomimetische Modell ist in der Lage menschenähnliche Schlagbewegungen zu produzieren, berücksichtigt jedoch nicht die Lernfähigkeit von Menschen. Um diese Anforderung zu erfüllen, zeigen wir in dieser Arbeit, dass die motorischen Fähigkeiten in einer so komplexen Aufgabe wie Tischtennis mittels Imitation und Reinforcement Learning gelernt werden können. Dafür verwenden wir die Erkenntnis, dass Menschen komplexe Bewegungsabläufe aus einer kleinen Anzahl einfacher generalisierbarer Bewegungsprimitive zusammensetzen. Dadurch können einzelne Schlagbewegungen dem Roboter demonstriert und mittels Imitationslernen reproduziert werden. Da sich die einzelnen Schlagbewegungen entsprechend der relativen Entfernung des zu schlagenden Balles vom Roboter, sowie vom Geschwindigkeitsprofil des Balles unterscheiden, muss der Roboter in der Lage sein aus einer kleinen Anzahl von Beispielen die Schlagbewegung zu generalisieren. Dafür entwickeln wir in dieser Arbeit einen neuen Algorithmus, genannt *Mixture of Motor Primitives*. Der Mixture of Motor Primitives Algorithmus ermöglicht es, basierend auf einer Bibliothek von Bewegungsprimitive die richtigen Schlagbewegungen, abhängig von der vorherrschenden Situation auszuwählen und zu generalisieren. Der

Selektionsprozess der Schlagbewegungen kann dabei selbstständig vom System mittels Reinforcement Learning erlernt werden. Das Framework wurde sowohl in Simulation als auch auf einem realen Barrett WAM Roboter getestet. Dafür lernt der Roboter zunächst eine kleine Anzahl von Schlagbewegungen von einem menschlichen Lehrer durch *kinesthetic teach-in*. Diese Bewegungen werden dann in eine Bibliothek von Bewegungsprimitiven übersetzt. Wir zeigen, dass diese Bewegungen mit Hilfe unseres Algorithmus zu einem breiteren Spektrum von Situationen verallgemeinert werden können. Unser Verfahren erlaubt dem Roboter dadurch, Bälle von einer Ballkanone erfolgreich zurück zu spielen sowie gegen einen menschlichen Gegner zu spielen und sein eigenes Verhalten dabei online zu verbessern.

Die Ausführung von Bewegungen auf dem Roboter mit Hilfe des biomimetischen und gelernten Ansatzes, ermöglicht dem Roboter zugespielte Bälle zurückzuspielen. Motorische Aufgaben mit kompetitiven Zielstellungen wie im Tischtennis erfordern jedoch zusätzlich eine Strategie, um das Spiel zu gewinnen. Solch eine Strategie kann nur schwer von Hand implementiert werden, da diese sowohl vom Gegner als auch von den Fähigkeiten des Roboters abhängig ist. In dieser Arbeit wird ein Grundstein gelegt um eine solche Strategie erlernen zu können. Im Detail wird die Möglichkeit strategische Informationen aus der Beobachtung von menschlichen Tischtennisspielen zu extrahieren diskutiert. Dafür modellieren wir Tischtennis als Markov-Entscheidungsproblem, in welchem die Belohnungsfunktion das Ziel sowie das Wissen um die elementaren strategischen Elemente enthalten sind. Wir zeigen wie diese Belohnungsfunktion und damit die strategischen Informationen unter Zuhilfenahme von modellfreien Inverse Reinforcement Learning Methoden aus Daten von Tischtennis spielenden Menschen extrahiert werden können. Diese Daten haben wir von Spielern mit unterschiedlichen Spielweisen und Spielfähigkeiten gesammelt. Wir zeigen, dass die resultierenden Belohnungsfunktionen in der Lage sind expertenspezifische strategische Informationen zu erfassen und zwischen den unterschiedlichen Spielweisen und Spielfähigkeiten der Versuchspersonen zu unterscheiden.

Acknowledgements

Many people crossed my way during the last years and most of them have contributed in one or the other way to this thesis. I especially thank Prof. Dr. Jan Peters who supervised me over the last five years. He was a wonderful supervisor, whose guidance, support and encouragement I highly appreciate. I am also indebted to Prof. Dr. Bernhard Schölkopf and Prof. Dr. Stefan Schaal for giving me the opportunity to work at such a great place as the Max-Planck-Institute in Tübingen and who created an inspiring research environment. I am grateful to my thesis referees, Prof. Dr. Jan Peters and Prof. Dr. Tamim Asfour for evaluating this thesis, Prof. Dr. Oskar von Stryk for heading the thesis committee, and Prof. Dr. Ulf Brefeld and Prof. Dr. Stefan Roth for participating in the defence. I also thank my coauthors Dr. Abdeslam Boularias, Dr. Jens Kober, Oliver Kroemer, and Dr. Betty Mohler for contributing to my publications. I can recommend you all as co-authors to anyone! I would also like to thank my lab colleagues Dr. Abdeslam Boularias, Tatjana Fomina, Dr. Jens Kober, Oliver Kroemer, Timm Meyer, Dr. Duy Nguyen-Tuong, and Zhikun Wang for all the discussions and feedback, but also for being my ball boy (or girl) and being my button monkeys ;). In particular I want to thank Timm Meyer for proofreading parts of this thesis and being my subject in order to test the various tapes for fixing the VICON markers to the skin! A big thank you goes also to Oliver Kroemer for proofreading this thesis and all the resulting publications. Another thanks goes to Ekaterina Volkova for her support with the calibration and advise for the motion suits and VICON system, as well as to Dr. Tobias Meilinger for his helpful comments on the psychological part of the human table tennis experiments. I am thankful to Finja Büchel for taking her time to proofread the German abstract of this thesis. I also appreciate the kind comments by the reviewers, whoever they are. Their comments and remarks were crucial to my work and I am thankful for the time they invested. Finally, I would like to thank Volker Grabe for his love, continuous encouragement, for the weekends and nights he spend with me in the office, helping me proofread my papers, and giving me a hand when two hands were not enough.



Contents

Abstract	iii
Zusammenfassung	v
Acknowledgments	vii
Abbreviations	xi
1 Introduction	1
1.1 Modeling Complex Behaviors	2
1.2 Learning Complex Behaviors with Robots	2
1.3 Table Tennis – An Example for a Complex Motor Task	3
1.4 Contributions	4
1.4.1 Modeling Robot Table Tennis	4
1.4.2 Learning Table Tennis	4
1.5 Organization of this Thesis	6
2 A Biomimetic Approach to Robot Table Tennis	9
2.1 Prologue	9
2.1.1 Related Work	10
2.1.2 Our Contributions	10
2.2 Modeling Human Striking Movements in Table Tennis	11
2.2.1 Striking Movement Generation	12
2.2.2 Initiation of Hitting Movements	13
2.2.3 Extracting Essential Context Parameters	14
2.2.4 Movement Stages of a Stroke	15
2.3 A Biologically-Inspired Trajectory Generator for Table Tennis Strokes	16
2.3.1 Overview of the Biomimetic Player	17
2.3.2 Dynamics Model of the Table Tennis Ball	18
2.3.3 Determining the Goal Parameters	19
2.3.4 Translating Virtual Hitting Points into Configurations	21
2.3.5 Movement Parameters	22
2.3.6 Movement Generation	22
2.4 Evaluations	24
2.4.1 Evaluation against a Ball Launcher	24
2.4.2 Accuracy of the Ball Dynamics Model	26
2.4.3 Comparison to Human Behavior and Performance	27
2.5 Discussion and Conclusion of Chapter 2	28
3 Learning to Select and Generalize Striking Movements for Robot Table Tennis	31
3.1 Prologue	31
3.2 Learning and Generalizing Motor Behaviors	33
3.2.1 Learning a Motor Task using the Mixture of Motor Primitives	34
3.2.2 Computation of the Augmented State	36

3.2.3	Representation of Behavior with Movement Primitives	37
3.3	Evaluation	42
3.3.1	Robot Table Tennis Setup	42
3.3.2	Computing the Meta-Parameters	42
3.3.3	Mixture of Motor Primitives	43
3.4	Discussion and Conclusion of Chapter 3	47
4	Learning Strategies in Table Tennis using Inverse Reinforcement Learning	49
4.1	Prologue	49
4.2	Modeling Human Strategies	51
4.2.1	Preliminaries	51
4.2.2	Learning the Reward Function	52
4.2.3	Computational Model for Representing Strategies in Table Tennis	55
4.3	Experiments and Evaluations	58
4.3.1	Experimental Setup and Data Collection	58
4.3.2	Results and Discussion	60
4.4	Conclusion of Chapter 4	66
5	Conclusion and Future Work	67
5.1	Summary of the Thesis	67
5.1.1	Modeling Complex Motor Tasks	67
5.1.2	Learning Complex Motor Tasks	67
5.2	Open Problems	69
5.2.1	Extension of the Table Tennis Player	69
5.2.2	Learning Complex Motor Skills	70
5.2.3	Learning Higher-Level Strategies	71
5.3	Publications	72
5.3.1	Journal Papers	72
5.3.2	Conference and Seminar Papers	72
	Bibliography	75
	Appendix	83
	List of Figures	87
	List of Algorithms	91
	List of Tables	93
	Curriculum Vitæ	95

Abbreviations and Glossary

In this thesis we use the following mathematical notation.

Notation	Description
$\mathbf{x} = [x_1, x_2, \dots, x_n]$	a vector
x_i	the i th component of the vector \mathbf{x}
\mathbf{x}^T	the transpose of vector
\mathbf{A}	a matrix
\mathbf{A}^T	the transpose of a matrix
\mathbf{A}^{-1}	matrix inverse
\mathbf{A}^\dagger	matrix pseudo-inverse
$Pr(x)$	probability density of \mathbf{x}
$\mathbb{E}(x)$	expectation of \mathbf{x}

The following symbols are used in this thesis.

Symbol	Description
t	time
Δt	time step
T	overall time
$\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}$	task space position, velocity and acceleration
$\theta, \dot{\theta}, \ddot{\theta}$	joint space position, velocity and acceleration
q	quaternion
\mathbf{s}	state
$\tilde{\mathbf{s}}$	augmented state
\mathbf{a}	action
r	reward
$k(s, s)$	kernel
π	policy
V^π	expected return of policy π

Throughout this thesis we use the following abbreviations.

Abbreviation	Description
CrKR	Cost regularized Kernel Regression
EKF	Extended Kalman filter
IRL	Inverse reinforcement learning
ITTF	International Table Tennis Federation
GMP	Generalized Motor Programs
GPU	Graphics Processing Unit
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
MDP	Markov Decision Problem
MM	Maximum Margin
MMG	Maximum Margin for Game Values

Abbreviation	Description
MMS	Maximum Margin for State Values
MoMP	Mixture of Motor Primitives
MP	Motor Primitives
DoF	Degrees of Freedom
DMP	Dynamical system Motor Primitive
PoMPD	Partially observable Markov Decision Problem
RE	Relative Entropy
RMSE	Root mean squared error
SD	Standart deviation

Here, we will give a short glossary for the terms in this thesis.

action \mathbf{a}	Vector of variables that change the state of the system. These actions can refer to motor commands, but also to higher-level movement parameters such as where and how to return the ball.
augmented state $\tilde{\mathbf{s}}$	Parameter vector consisting of the state \mathbf{s} and a set of parameters that describes the characteristics of the current task.
hitting manifold meta-parameters δ	The part of state-space that defines the ball-racket contacts. The movement goal, duration and amplitude of a movement described by a motor policy.
motor behavior	The study of all neuronal and muscular processes underlying observable motor responses due to internal and external environmental stimuli.
motor control	Study of the processes underlying the production of coordinated movements of the central nervous system.
motor policy $\pi(\tilde{\mathbf{s}})$	Function that maps the state \mathbf{s} of a movement system and the internal parameters \mathbf{w} to a control vector that defines the executable motion. Here, we use the augmented state $\tilde{\mathbf{s}}$ instead of the state of the system \mathbf{s} .
motor skill	Sequence of smooth movements executed in order to master a particular task.
motor task	An assignment that requires one or more motor skills in order to be concluded.
movement goal	The finite state of the desired movement.
movement library	Database of movement primitives stored as motor policies.
movement primitive	A sequence of motor commands executed to accomplish a given motor task.
policy π	Function that maps the state of the system to a control vector.
state of the system \mathbf{s}	Vector of variables necessary to describe the system.
strategy	A high-level plan of actions to accomplish a specific goal.
virtual hitting point	The point in time and space where the player plans to intercept the ball.

1 Introduction

The everyday life of an adult human is dominated by his ability to control his motor system in various ways: walking to work, grasping a cup of coffee, riding a bike, or playing a game of soccer. Humans are able to adapt their motor skills to new situations and learn new tasks based on only a small number of trials. These tasks are performed with apparently little effort and without active attention. Nevertheless, only little is understood of the mechanisms underlying these skills. It is still unknown how and why a particular movement plan is chosen, how perceptual information is integrated, movements are generalized or new skills can be acquired.

How little is known about human motor control is also reflected in the current abilities of robots. Robots are still limited to a small number of pre-defined manipulation tasks. These tasks are usually manually hard-coded after carefully analyzing the task and the environment. Thus, robots are still not able to adapt their movements to unforeseen situations or to autonomously learn new motor tasks. As a result, robots can usually only be used in well defined environments separated from humans.

Despite their limitations, humans still envision robots that are able to perform various tasks similar to human beings. Indeed, robots could be of great help for the society. Catastrophic events, such as the Fukushima Daiichi nuclear disaster or the Deepwater Horizon oil spill, revealed the need for robots that can perform complex tasks which would be too dangerous or even impossible to be performed by humans. Aside from those scenarios, robots could also be of great value in our everyday life. Especially in an ageing society, robots can act as helpers in various household tasks, be a companion, and alert medical services in case of an emergency. Furthermore, robots can be used in health care. It has been shown that the use of humanoid robots can help autistic children to develop social learning skills [Ricks and Colton, 2010, Dautenhahn and Werry, 2004]. Other studies are conducted in rehabilitation of stroke patients [Gomez-Rodriguez et al., 2011] or on the effect of long-term interaction of humanoid robots and children in a hospital that are engaged in a diabetes management course [The ALIZ-E project team, 2013].

Although Simon [1965] claimed already in the early sixties that machines will be able to perform tasks similar to humans within twenty years, today's robots are still far behind their fictional counterparts. There are many reasons for this gap between fiction and reality such as inappropriate mechanics and power supplies, computational load, and unsuitable sensing abilities of the environment. Even if these problems were solved, robots might still not be able to perform complex task such as to take out our trash. In order to create such skilled machinery, we need to know how robots can generate task specific motor behavior in an efficient and robust, but at the same time also flexible and safe manner. Computational models of human motor control can be used in order to get one step closer to this goal, as many properties of humans are essential for future robots working in human inhabited environments. Most of these computational models of human motor control are able to explain characteristics of motor behavior to a certain extent, but do not include how this behavior is learned. Here, domain-appropriate machine learning techniques could fill this gap. This thesis therefore focuses on both modeling *and* learning of motor behavior.

A motor behavior is always directed towards a certain goal one wants to achieve. This goal can be rather simple to accomplish such as an unconstrained reaching tasks towards a static object, but it could also be more complex such as striking a ball in a table tennis game. Complex motor tasks are characterized by different time critical movements that have to be sequenced and chosen due to environmental stimuli. Furthermore, they have to be executed in an accurate way and need to be adapted to unpredictable changes in the environment. The goal of this thesis is therefore to model and learn such a complex motor task. As a test case, we use table tennis throughout this thesis. We create a computational model of table tennis that can be implemented on a robotic system. The model

is inspired by hypotheses from the literature in human motor control. To account for the requirement of flexible systems that are able to adapt to unforeseen situations and learn new motor skills, we also investigate domain-appropriate machine learning approaches for robotic systems. Therefore, we present new algorithms in order to deal with the complexity of the task, and test existing algorithms on this new challenging task. We evaluate the resulting robot table tennis player in simulation as well as on a real Barrett WAM arm.

1.1 Modeling Complex Behaviors

The term *modeling complex behavior* refers in this thesis to the development of computational models based on findings in human motor control and sport science. Understanding how humans solve such complex tasks is essential when constructing robotic systems that exhibit natural movement characteristics. Human-like motor behavior can be the key to robust, efficient and human-like movement plans for anthropomorphic robots. Reproducing human behavior can also be beneficial for human-robot interaction, as it allows humans to identify with and, as a consequence, to accept and interact with the robot. On the other hand, using computational models of motor control and creating human-like motor behavior on robotic systems can also be beneficial for computational neuroscience or sport science to verify their hypotheses.

Numerous theories and hypotheses exist describing observed movement characteristics such as the speed-accuracy trade-off [Fitts, 1954], the ability to reach goals reliably and repeatedly while still exhibiting variability [Todorov and Jordan, 2002], or bio-mechanical redundancy [Bernstein, 1967]. These observations lead to many possible explanations which are often captured in the form of computational models allowing to predict and explain human motor behavior. For example, different cost functions were proposed in order to account for observed patterns in movement trajectories such as a bell-shaped velocity profile [Bryson and Ho, 1975, Hogan, 1984, Harris and Wolpert, 1998]. Besides those general models of movement generation, task specific studies from sport science can be used to derive the desired model. However, these task specific studies are usually not explicitly defined in the form of a general mathematical model and rather describe the observed differences between naive subjects and experts in a specific sport.

This thesis focuses on how humans generate fast and accurate movements, initiate these movements, select important visual information, and on how individual striking movements can be structured. Although using models based on human motor control has many advantages, it is also necessary to account for the differences between humans and robotic systems. Therefore, we have to adapt those models to the needs of robotic systems. We create such a computational model and evaluate it on a real robot arm.

1.2 Learning Complex Behaviors with Robots

The ability of humans to learn new and often complex motor skills while adapting their skills to new situations and tasks with a reasonable amount of trials can still not be explained by neuroscientists or reproduced by robots. Reinforcement learning [Sutton and Barto, 1998] has provided a mathematical formulation for learning control problems based on the concept of a reward function which also influenced computational neuroscience [Kawato and Samejima, 2007]. In reinforcement learning, the considered problem is learned by trial and error. It enables the system to autonomously self-improve its behavior by interacting with its environment. Reinforcement learning has been successfully applied to tasks like Backgammon [Tesauro, 1994] and Go [Chan et al., 1996]. However, when learning motor tasks directly with reinforcement learning on a real robot, it suffers from slow convergence rates due to the high dimensional problems and the real time constraints [Schaal, 1999, Kober et al., accepted].

Learning from demonstration can provide a good starting point for learning a behavior with reinforcement learning [Schaal et al., 2003a, Argall et al., 2009]. Here, the robot tries to reproduce the behavior of a teacher, similar to humans. Learning motor tasks with a combination of imitation and reinforcement learning allowed to learn tasks like Ball-in-a-cup [Kober et al., 2008], T-ball swings [Peters and Schaal, 2006], pancake-flipping [Kormushev et al., 2010], pouring water [Tamosiunaite et al., 2011] and drumming [Pongas et al., 2005]. As impressive as these examples of robot learning are, often only a single movement template is used. In complex tasks however, humans as well as robots have to acquire several movements that need to be chosen and executed depending on the current context.

Furthermore, if the reward signal used by the artificial system is unknown or wrong, the system will not be able to learn the correct behavior. The reward is crucial as it defines the goal and the knowledge to solve this task. Usually, it is assumed that the reward function is known. However, in complex tasks and especially when human behavior is involved or higher-level strategies should be modeled, it is hard to design the reward function manually beforehand. Instead of pre-defining the reward function, it can also be inferred from an expert demonstrating the task. This process is known as inverse reinforcement learning or inverse optimal control [Boyd et al., 1994, Ng and Russel, 2000]. Inverse reinforcement learning therefore offers a way to analyze human behavior, as the reward function describes the reasoning behind this behavior. This knowledge can then be transferred to an artificial system.

In this thesis, we discuss each of these aspects. We investigate how the different movements learned from demonstration can be chosen based on the current situation and how this selection process can be adapted using reinforcement learning. Subsequently, we also discuss how the reward function for a higher-level strategy can be inferred from observing a real game of human table tennis.

1.3 Table Tennis – An Example for a Complex Motor Task

Table tennis as a particularly difficult task has fascinated roboticists since 1983. A striking motion in table tennis does not only consist of just one specific movement that has to be perfected, but is composed of different phases, each using different movements: the preparation of the stroke by moving the arm backwards, the hitting of the ball at the desired point in time and space, and the movement of bringing the arm back to a neutral position [Ramanantsoa and Durey, 1994]. The hitting movement itself varies depending on the point of impact relative to the player, the time available, and the kind of stroke that should be performed. Furthermore, the involved movements are fast and need to be executed with high precision. Small inaccuracies in timing can lead to large deviations in the final bouncing point of the returned ball and result in an unsuccessful attempt to return the ball to the opponent's court.

In order to define the point and time of impact, the player has to predict the trajectory of the ball, decide on a suitable hitting point and estimate the time until impact. The movements must be timed such that the ball can be hit by the racket. Therefore, the hitting movement needs to be chosen and adapted due to uncertainties and changes in the ball trajectory. Furthermore, table tennis is a competitive game. That means, a player does not only has to return an incoming ball back, he also needs to decide where and how the ball should be returned to the opponent's court. This decision is defined by a higher-level strategy and does not only depend on the trajectory of the ball and the position of the player, but also on the opponent. The hitting movement should be chosen such that there is a high probability to successfully return the ball as well as to make the task harder for the opponent. Based on this information, a player is able to choose and generate suitable hitting movements.

1.4 Contributions

In this thesis, we introduce new approaches and algorithms for modeling and learning robot table tennis. The work presented in this thesis contributes to the fields of motor control and machine learning in robotics, but also gives further evidence for some hypotheses in human motor control. In the following, we discuss the contribution of each chapter individually. The contributions and their belonging to the different research fields are also summarized in Figure 1.2.

1.4.1 Modeling Robot Table Tennis

Research on robot table tennis started already in 1983 with the announcement of a robot ping pong competition [Billingsley, 1983]. Research continued until the end of this competition in 1993 and resulted in several publications [Andersson, 1988, Knight and Lowery, 1986, Hartley, 1987, Hashimoto et al., 1987, Fässler et al., 1990]. A few groups continued to work on this topic until today [Acosta et al., 2003, Miyazaki et al., 2006]. Most of these early approaches relied on hardware tailored for this specific task only and were used in a well defined environment optimized for reliable ball recognition. However, if one wants to create robotic systems that are able to perform various tasks in a human inhabited environment, one will have to move away from engineered hardware approaches that are applied in structured environments free of visual disturbances. Instead, one should focus on robust movement generation that is able to compensate for uncertainties in the environment.

The model of robot table tennis inferred and implemented in this thesis therefore differs in three ways from previous work: Firstly, instead of using hardware specifically designed for this task, we use a redundant anthropomorphic seven Degree of Freedom (DoF) robot arm. Secondly, instead of applying a vision system in an uncluttered environment, our vision system operates in a semi-structured human inhabited environment. Thirdly, we concentrate on generating smooth movements that properly distribute over the different DoF to reduce the stress on individual joints.

As humans still outperform robots that are specifically designed for this task, we study human motor control and derive a computational model for table tennis to yield a robust movement generation. We show that using different hypotheses of human motor control and sport science, such as the virtual hitting point hypothesis [Ramanantsoa and Durey, 1994], cost functions to resolve the redundancy [Cruse, 1986] or structuring of the hitting motion into four stages [Ramanantsoa and Durey, 1994] enable us to create a computational model that produces successful human-like hitting motions on a robot.

1.4.2 Learning Table Tennis

The goal of the second part of this thesis is to investigate domain-appropriate machine learning algorithms in order to enable the robot to learn complex motor tasks. Previous work on learning robot table tennis focused on predicting the hitting point with supervised learning techniques [Miyazaki et al., 2006, Lai and Tsay, 2011]. None of these approaches concentrated on learning robust hitting movements that are able to compensate for perturbations in the environment, inaccuracies in the impact point and time, or learning efficient strategies for winning the game.

Selecting and Generalizing Striking Movements

In order to ensure a movement representation which can be learned and optimized by the robot, we use Dynamical system Motor Primitives (DMP, Ijspeert et al. [2002]). DMPs allow to represent arbitrarily shaped smooth movements for many different tasks. The original formulation of Ijspeert's DMPs were adapted by Kober et al. [2010] to allow additionally for arbitrary target velocities in order to represent

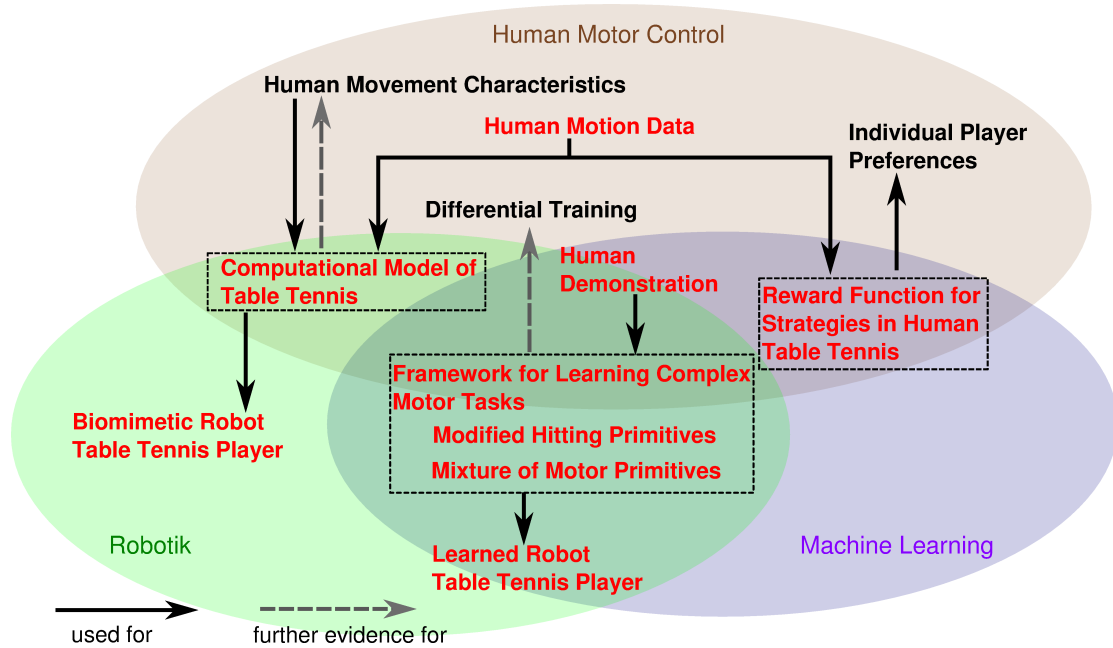


Figure 1.1: Overview of the thesis and its contributions. This figure shows the contributions of this thesis (red), how different aspects are connected with each other and their belonging to the different research fields.

hitting movements. We refine this formulation in this thesis to remove drawbacks such as infeasible acceleration profiles in Chapter 3. Most of the applications on learning and self-improving DMPs used only individual movement primitives to represent the whole motor skill at once. In complex motor tasks however, motor primitives not only need to be adapted to a constantly changing goal, these motor primitives should be chosen based on the environmental stimuli and their usage needs to be adapted to their performance. In Chapter 3, we therefore present a framework that enables the robot to learn basic cooperative table tennis from demonstration and interaction with a human player. We develop an algorithm called Mixture of Motor Primitives (MoMP) that selects and generalizes movements from a movement library based on environmental stimuli. The selection process can be autonomously adapted based on the performance of the hitting movement generated with the MoMP algorithm. We show (i) that the resulting framework is able to select movement primitives from a library based on the current task context instead of using only a single demonstration and (ii) that the adaptation of the selection process during a table tennis game improves the performance of the robot player. The setup is tested on a real robot in a match against a human opponent and with a ball launcher.

This work has been recognized by Dr. Muster, honorary member of the German table tennis coaches club (Verband deutscher Tischtennis Trainer, Muster [2013]). In his opinion the demonstration of multiple hitting motions to the robot and the subsequent association of situations and suited hitting motions is further support to the *differential training* method in table tennis [Schöllhorn, 2000, 2003].

Inferring Strategic Information

Even after acquiring the necessary motor skills for a complex motor tasks, a strategy is required to choose how the ball should be returned to the opponent's court in order to win the game. The data-driven identification of basic strategies in interactive tasks, such as table tennis, is a largely unexplored problem. Most existing work concentrates on identifying the frequencies and effectiveness of specific movement patterns. Those approaches usually require a large video library in which the strategic elements are labelled manually by experts. Furthermore, they do not allow to model the

Modeling and Learning of Complex Motor Tasks: A Case Study with Robot Table Tennis.




Chapter 1 Introduction		
Computational Model of Table Tennis	Learning Table Tennis	
<p style="text-align: center;">Chapter 2</p> <p>Biomimetic approach to robot table tennis</p>  <p style="text-align: center;">Analytical Robotics Methods</p> <p>Robot system that reproduces human striking behavior based on biological insights.</p>	<p style="text-align: center;">Chapter 3</p> <p>Learning to Select and Generalize Striking Movements</p>  <p style="text-align: center;">Imitation and Reinforcement Learning</p> <p>Maching Learning algorithm to select and generalize striking movements.</p>	<p style="text-align: center;">Chapter 4</p> <p>Learning Strategies in Table Tennis using Inverse Reinforcement Learning</p>  <p style="text-align: center;">Inverse Reinforcement Learning</p> <p>Extract expert knowledge on effective elements of basic strategy of human players.</p>
Chapter 5 Conclusion		

Figure 1.2: Outline of the thesis. The thesis is divided into two parts: Modeling complex motor behavior and learning complex motor behavior. Learning complex table tennis can be further divided into selecting and generalizing striking movements and learning higher level strategies.

reasoning behind those decisions and, as a consequence, to transfer the knowledge to artificial systems. In Chapter 4, we therefore model the decision process for choosing actions by players in a match of table tennis from a computational point of view. Thus, it is not only possible to use the learned model for artificial systems, such as table tennis robots, but also to yield a better insight into the reasons for a human player to choose an action in a certain state. We model the table tennis game as a Markov decision problem [Puterman, 1994] and extract expert knowledge on effective elements of a basic strategy in the form of a reward function using model-free inverse reinforcement learning. To accomplish this step, we collect data from humans playing table tennis under different conditions. The resulting reward function is evaluated on real table tennis data. We also show that our system is able to distinguish the expert among players with different skill levels and different playing styles in a leave-one-subject out testing scheme, as well as to capture expert-specific strategic information.

1.5 Organization of this Thesis

The individual chapters of this thesis can be read independently from each other. The outline of the thesis is illustrated in Figure 1.2. It is divided into two parts. The first part consists of Chapter 2, "A biomimetic approach to robot table tennis", and concentrates on modeling robot table tennis. In this chapter, we first discuss the aspects of skilled performance for fast interactive tasks, such as table tennis, and models of how a striking movement is organized. Based on these biological hypotheses, a computational model of robot table tennis is created and consequently implemented on an anthropomorphic seven DoF robot

arm. We evaluate the model in a physically realistic simulation as well on a real Barrett WAM arm. In this chapter, we also provide a review of existing robot table tennis approaches. This chapter is based on [Muelling et al., 2011].

The second part of the thesis includes Chapter 3 and Chapter 4 and discusses learning algorithms for motor behavior. In Chapter 3, "*Learning to Select and Generalize Striking Movements for Robot Table Tennis*", we present our framework to learn striking movements in robot table tennis based on imitation and reinforcement learning. In this chapter, we first show how suitable movement templates can be selected and mixed from a movement library using the MoMP algorithm, and how the selection process can be adapted during the table tennis match. Subsequently, we show in simulation and on a real robot that this system is able to learn striking movements from real demonstrations and that the framework is able to autonomously adapt the selection process in order to increase the performance in a game of table tennis. This chapter is based on [Muelling et al., 2013].

In Chapter 4, "*Learning Strategies in Table Tennis using Inverse Reinforcement Learning*", we show how strategic elements for winning a game can be inferred from observing humans playing table tennis using model-free inverse reinforcement learning. Therefore, we first present how to model table tennis as a Markov decision problem. Three model-free inverse reinforcement learning approaches are used to infer the reward function to capture the expert knowledge on effective elements of a basic strategy. We show how the data is collected from humans playing table tennis with different playing skills and styles. Finally, we present the results for evaluating all reward functions obtained by the different model-free inverse reinforcement learning techniques on real human data. This chapter is based on [Muelling et al., under review].

We summarize the presented contributions in Chapter 5 and give an overview over potential future work in continuation of this thesis.



2 A Biomimetic Approach to Robot Table Tennis

Robots are still limited in their motor abilities and far behind their fictional counterparts in books and movies. These limitations of robots lead to the question of how humans are able to produce robust and flexible motor behaviors. Studies on human motor control have not been able to solve this question yet, but resulted in several computational models that attempt to explain specific aspects of human motor behavior. These studies and computational models of human motor control are an inspiration for implementing robust motor behavior in robotics. The following chapter is based on [Muelling et al., 2011] and consists of two parts. First, we present a review of robot table tennis and the biological background on human motor control with a focus on table tennis. In the second part, we present the implementation of an analytical robot table tennis player based on the biological findings presented in the first part. Furthermore, we show that the resulting table tennis player setup produces human-like hitting motions on a real Barrett WAM robot arm.

2.1 Prologue

Humans perform complex tasks relying on little feedback with long latencies and have strong limits on their movement execution. Additionally, they suffer from inaccurate sensory information in largely unmodeled environments. Interception tasks, as table tennis, require fast and accurate movements that are precisely coordinated with the visual information of the ball and the opponent. However, the human central nervous system has little time to process feedback about the environment and has to rely largely on feed-forward components [Wolpert et al., 1998] such as accurate task models as well as predictions about the opponent and the ball. Nevertheless, human beings outperform table tennis and baseball robots, which are tailored for these tasks. This performance gap is to a significant part due to the way in which humans perform robust movements.

Computational models of motor control and learning that describe human motion generation can be useful for neuroscientists to verify hypotheses on human motor control. These models can also be useful in robotics to create robot systems that are able to perform a wide variety of movements robustly and adapt these movements to unexpected environmental conditions and new requirements.

Table tennis has long fascinated roboticists as a particularly difficult task. Significant research on robot table tennis started around 1983 [Billingsley], and was followed by a period of enthusiastic research until 1993 [Andersson, 1988, Knight and Lowery, 1986, Hartley, 1987, Hashimoto et al., 1987, Fässler et al., 1990]. A few groups continue to work on this problem [Miyazaki et al., 2006, Matsushima et al., 2005, Acosta et al., 2003], as detailed in Section 2.1.1. These early approaches used hardware engineering solutions to avoid inherent problems in high speed anthropomorphic movement generation and vision in a human inhabited environment.

In contrast to previous approaches, we use an anthropomorphic robot arm with seven degrees of freedom (DoFs) and concentrate on generating smooth movements that properly distribute the forces over the different DoFs. To cope with the resulting challenges of this approach, we present a biomimetic approach for trajectory generation and movement adaptation based on theories pertaining to human motor control in table tennis. Our goal is to generate human-like striking movements on an anthropomorphic robot arm with seven DoFs. We investigate the problem of determining joint angles and joint velocities for a redundant robot arm at the interception point. Furthermore, we study the planing of arm trajectories for returning an incoming ball towards a desired point on the opponent's court. The resulting trajectories can be adapted to new environmental conditions. The presented robot table tennis player is able to return balls served by a ball cannon on an International Table Tennis

Federation (ITTF) standard sized table. The system works well both in simulation and on a real Barrett WAM.

2.1.1 Related Work

Research on robot table tennis started with robot ping pong competitions initiated by Billingsley [1983]. Billingsley developed a special set of rules for the competition. In contrast to human table tennis, the table is only 0.5 m wide and 2 m long and the net has a height of 0.25 m. Wire frames were attached to each end of the table and the net. For a shot to be valid, the ball has to pass through all three frames. Thus, the maximum ball speed is limited to 10 m/s. Several early systems were presented by Knight and Lowery [1986], Hartley [1987], Hashimoto et al. [1987] and others. For this early work, the major bottleneck was the lack of fast real-time vision.

An important breakthrough was achieved when Andersson [1988] presented the first robot ping pong player capable of playing against humans and machines. Andersson and his team also employed the simplified robot table tennis rules suggested by Billingsley. He used a high-speed video system and a six DoF PUMA 260 arm with a 0.45 m long stick mounted between table tennis racket and robot. Andersson implemented an expert system-based controller that chooses the strategy as a response to the incoming ping pong ball. The system was later reproduced by Harrison et al. [2005].

In 1993, the last robot table tennis competition took place and was won by Fässler and his team of the Swiss Federal Institute of Technology [Fässler et al., 1990]. Although the competitions ceded to exist, the problem was by no means solved, but the current limits were met in terms of robot hardware, algorithms, and vision equipment. The focus in that period was mainly on designing better hardware rather than finding robust algorithms that can compensate for lower-quality hardware.

Nevertheless, interest in robot table tennis did not wane completely and groups continued to work on it. Acosta et al. [2003] constructed a low-cost robot showing that a setup with two paddles is sufficient for playing if the ball is just reflected at the correct angle by a stationary paddle. Zhang and Tau [2010] and Huang et al. [2011] concentrated on vision and trajectory prediction while Miyazaki et al. [2006] and Lai and Tsay [2011] concentrated on the prediction of the spacial and temporal prediction of the hitting point. Miyazaki et al. [2006] were able to show that a slow four DoF robot system consisting of two linear axes and a two DoF pan-tilt unit suffices if the right predictive mappings are learned. The group applied locally weighted regression to predict the impact time, ball position and velocity. Lai and Tsay [2011] proposed a setup to estimate the ball trajectory using fuzzy adaptive resonance theory network and learn the orientation angles of the racket using Self-organizing maps. See Table 2.1 for an overview of the major contributions in robot table tennis so far.

All systems were tailored for the table tennis task and relied heavily on high-gain feedback, over-powered motors (no saturation), linear axes (easy to control), and light-weight structures (no torque saturation, little moving inertia). They were engineered in such a way that they could execute any straight movement towards the ball at a rapid pace with the right approach angle. The important problems of generating smooth movements that properly distribute the forces over the arm's different DoFs was usually avoided.

2.1.2 Our Contributions

Instead of focusing on engineering a hardware solution specifically designed for robot table tennis, we aim to realize human striking behavior using an anthropomorphic robot arm. Therefore, our setup differs in three ways from the work existing up to now. First, instead of using a hardware approach that simplifies the motion generation, we use an anthropomorphic redundant seven DoF robot arm. The robot has revolute joints, and large inertia, for example, the wrist alone has a mass of 2.5 kg weight at the elbow. Thus, we have strong constraints on the joint velocities and accelerations. Second,

Author	Vision System	Robot System	Comments
Andersson [1988]	Four high speed cameras, 60 Hz	Puma 260, 6 degree of freedom (DoF), 45 cm stick between robot and racket.	First robot ping pong player able to play against humans and robots.
Fässler et al. [1990]	Two CCD cameras, 50 Hz.	Light weight aluminum structures, 6 DoF (3 linear, 3 rotation).	Won the last robot ping pong competition in 1993.
Acosta et al. [2003]	Single CCD video camera, 40 Hz.	Lightweight robot with two paddles, 5 DoF (2 prismatic, 3 revolution).	Engineered for table tennis, just reflecting the ball at the correct angle on a small table.
Angel et al. [2005]	Single Sony XC_HC 50 camera, placed on the end-effector.	Parallel robot, 4 DoF, maximum end-effector speed 4 m/s.	Focus on visual control of the robot.
Miyazaki et al. [2006]	Quick MAG System 3, 60 Hz.	Robot mounted on the table, 4 DoF (2 DoF linear + 2 DoF pan-tilt unit).	Learned input-output maps to predict the impact time, ball position and velocity.

Table 2.1: This table shows a few robot table tennis systems as examples. Note that most systems include linear axes to achieve the necessary speed or have an additional stick mounted between racket and the robot’s palm.

our vision system operates in a semi-structured and human-inhabited environment. The third aspect is the application of a biomimetic approach. Humans still outperform robots that are specifically designed for the specific task of robot table tennis. We must therefore investigate how humans control their movements and why their motions are more efficient. Hence, we derive a computational model for human motor control in table tennis and implement this model on a robot. We employ known hypotheses on the movement structure in table tennis, the identification of a virtual target, the timing of interception tasks and the resolution of redundancy.

In this chapter, we will proceed as follows. In Section 2.2, we present knowledge on modeling a table tennis stroke based on biological hypotheses such that we are able to obtain a trajectory of a table tennis stroke in Section 2.3. In Section 2.4, we present the evaluation of our system in simulation as well as on a real Barrett WAM showing that our setup is able to return incoming volleys to desired points on the opponent’s court. Finally, we summarize our work as well as our results.

2.2 Modeling Human Striking Movements in Table Tennis

To create an anthropomorphic table tennis playing robot, we rely on hypotheses on human motor control in table tennis. Hence, in this section, we present background information on modeling table tennis from a racket sports’ perspective. In particular, we focus on movement generation, movement initiation, the selection of important visual information, and the movement stages in table tennis. Finally, we will outline computational concepts that arise from the biological hypotheses.

2.2.1 Striking Movement Generation

To perform complex movements, the human brain has to determine a proper sequence of muscle activation, the integration of sensory information, and the generation of motor commands to produce coherent motor behavior. There exist many theories on how humans control coordinated movements, which describe characteristics observed in human motor control as the speed-accuracy trade-off [Woodworth, 1899, Fitts, 1954], the ability to reach goals reliably and repeatedly while still exhibiting variability in the movement of the individual degrees of freedom [Todorov and Jordan, 2002], goal-directed corrections [Elliott et al.], and biomechanical redundancy [Bernstein, 1967]. In order to find generative principles underlying movement generation, neuroscientists often employ concepts from optimal control and, thus, use cost functions to evaluate the performance of the system [Bryson and Ho, 1975, Hogan, 1984, Harris and Wolpert, 1998, Todorov and Jordan, 2002]. Another important theory is the so-called motor program, a memory-based representation of movement patterns [Henry and Rogers, 1960, Keele, 1968, Schmidt, 1975].

Cost Functions for Resolving the Redundancy in Striking Movements

The human body employs approximately 650 major muscles to control its joints [Scott and Fong, 2004]. However, although human beings have many DoF, only a few DoF are actually required for a movement. There exist an infinite number of configurations of joints and muscles in the arm that all accomplish the same task. Such biomechanical redundancy makes the system flexible, but also difficult to control.

One possible generative principle underlying movement generation is the concept of optimal control [Bryson and Ho, 1975]. Here, it is postulated that movement patterns for the task are chosen such that a specific cost J

$$\min_{\mathbf{w}} J = \frac{1}{2} \int_0^T c(\mathbf{w}) dt,$$

is minimized, where T is the movement duration and \mathbf{w} the parameters describing the movement. The cost function c measures the cost of the movement with respect to an undesirable feature, such as jerk or the amount of metabolic energy used for the movement.

Most studies on cost functions focus primarily on reaching and pointing movements [Uno et al., 1989, Hogan, 1984, Flash and Hogan, 1985, Harris and Wolpert, 1998] where one can observe a bell-shaped velocity curve [Morasso, 1981], as well as a clear relationship between movement duration and amplitude [Flash and Hogan, 1985, Roitman et al., 2004]. However, these relationships do not hold in striking sports as shown by the work of Bootsma and van Wieringen [1990]. Energy-optimality [Alexander, 1997, Kuo, 2005] does not appear to explain striking movements, as it would result in movements that are awkward both for human beings and robots. Cruse et al. [1986, 1990] suggested that the comfort of the posture may play a major role in arm movement generation and, hence, it may be part of the cost function of striking movements. According to Cruse [1986], the cost is induced by proximity to a fixed comfort posture in joint-space. This implies that the cost will be minimal if the joint angles are the same as for the comfort posture, and increases with the distance from this posture. Mathematically interpreted this cost function can be understood as minimizing $\sum_i^N c_i(\|\theta_i^* - \theta_i\|)$, where θ_i^* and θ_i are the optimal and current joint values respectively, $c_i(\cdot)$ is a sensitivity function that is attached to this joint and N is the number of DoF. We therefore employ this cost function and resolve the redundancy of the arm accordingly (see Section 2.3.3).

Motor Programs for Striking Movements

Humans are likely to use a set of pre-structured movement commands, often referred to as motor programs [Henry and Rogers, 1960, Keele, 1968, Schmidt, 1975, Schmidt and Wrisberg, 2000]. Motor programs determine the order and timing of muscle contractions and, hence, define the shape of the action. Sensory information can modify motor programs to generate rapid corrections for environmental changes, as found in table tennis by Bootsma and van Wieringen [1990].

Schmidt [1988], Schmidt and Wrisberg [2000], Schmidt [2003] proposed the concept of Generalized Motor Programs (GMP), which are stored in memory and contain elementary movements that may be used as actions. Such actions contain several movements that have similar invariant features such as relative timing and the sequence of the components. These invariant features are consistent when performing the action. A specific movement can be adapted to environmental conditions by altering the parameters of the GMP, such as movement time, amplitude and the goal position of the movement.

As a possible theory on how GMPs could generate coordinated movements, Schmidt [1975, 2003] suggested the motor schema theory. A schema is a set of rules that is developed by extracting information from related experience. The schema provides the parameters for the GMP and, thus, allows it to adapt to new situations and environmental context. According to the theory of Schmidt [1975, 2003], performing an action involves first selecting the appropriate motor program from the memory and, subsequently, setting the adjustable parameters according to the rules defined by the motor response schema.

The validity of this framework for explaining human striking behaviors in table tennis is supported by experimental evidence of Tyldesley and Whiting [1975]. Their experiments demonstrated consistent spatial and temporal movement patterns for expert ping pong players. Tyldesley and Whiting [1975] concluded that a professional player chooses a movement program for which the execution time is known from their repertoire, and then decides when to initiate the drive. This observation is known as *operational timing hypothesis*. For more information about the timing of movements see Section 2.2.2.

For the table tennis robot, we represent the movement program for each degree of freedom involved in striking a table tennis ball as 5th order splines.

2.2.2 Initiation of Hitting Movements

While spatial planning of the trajectory is necessary for any movement, striking a moving object relies critically on accurate and precise timing to achieve the desired interception. An important component of the visual information, which is used to control the timing in an interception task, is the *time-to-contact* [Hecht, 2004]. Time-to-contact is the time until the object reaches the observer or a point in space where it can be intercepted by the observer. Therefore, two essential questions need to be addressed. First, how to specify the time-to-contact and, second, how to coordinate the timing of the action accordingly. Several hypotheses have been suggested to address these questions and the most common ones are discussed below.

Tau Hypothesis

A possible time-to-contact estimation strategy employs the speed and distance of the approaching object. However, this approach cannot explain why humans are able to estimate the time to contact for unknown objects. Lee and Young [1985] hence suggested that the temporal information on the ball's arrival is defined by an optic variable τ . This variable is specified as the relative inverse rate of dilation of the object image on the retina.

The critical value that triggers a specific action is called the *tau margin*. Lee and Young [1985] suggested that humans direct their actions towards the tau margin rather than to the real time to contact.

Bootsma and van Wieringen [1988] presented experimental evidence that the initiation of the drive in table tennis may indeed be based on the tau-margin.

Stroke Timing Strategies

Hubbard and Seng [1954] concluded from their experiments with professional baseball players that the duration of the swing was constant and independent of the speed of the pitched ball. This consistency in swing duration indicates that the batters adjusted the initiation of their swings according to the tau-margin. Such behavior, hence, always results in the same timespan required for each swing, irrespective of the ball's speed.

Tyldesley and Whiting [1975] proposed the operational timing hypothesis independently of the work of Hubbard and Seng [1954]. Tildesley and Whiting conducted an experiment with amateur, intermediate and expert table tennis players. The subjects were asked to perform a forehand drive to a designated target area on the table. While expert players and, to a lesser extent, intermediate players achieved a high degree of spatial consistency, novice players failed to do so. The results provided support for the concept of consistent motor programs and lead to the assumption that these motor programs just need to be initiated at the right time. The operational timing hypothesis hence states that expert players reduce the interception task to the problem of determining when to initiate the action.

Bootsma and van Wieringen [1990] investigated the validity of the operational timing hypothesis using five expert table tennis players. As expected, they found that the subjects performed consistent movement patterns. However, in contrast to Tildesley and Whiting, they concluded that the small variations in the temporal movement pattern are functional, and not the result of noise. They measured the standard deviation in the timing of the stroke initiation and the ball-racket contact. The initial time accuracy was defined as the variability of the tau-margin. The contact timing accuracy was defined as the variability of the travel direction of the racket at the hitting point. According to the operational timing hypothesis, the variations at the stroke initiation should be less than those at the hitting point. Instead they found a standard deviation in timing of 2 – 5 ms for the moment of contact while having 5 – 21 ms at initiation. However, their experiments showed a decrease in variability. This finding, therefore, contradicts the operational timing hypothesis.

In later work, Bootsma and Wiering suggested a continuous perception-action model with a funnel-like type of control, in which the participant has to initiate his swing within a fixed spatio-temporal bandwidth [Bootsma and Peper, 1992, Williams and Starkes, 2002]. The drive can be aligned according to perceptual information available during the swing, which would cause an increasing accuracy during the moment of initiation and the moment of ball-racket contact.

In the robot table tennis setup, we will therefore initiate the hitting movement when the time-to-contact is below a threshold value.

2.2.3 Extracting Essential Context Parameters

To return an approaching ball successfully to the opponent's court, humans have to extract the necessary context parameters from observations of their environment. Knowledge of the gaze behavior can yield information about which parts of the ball flight are important for planning the striking movement.

Studies by Ripoll and Fleurance [1988] and Rodrigues et al. [2002] revealed that expert table tennis players track the ball only at the beginning of its trajectory. The duration of the ball tracking depends on the characteristics of the stroke. For example, balls moving to the player's body are tracked for a longer period than balls which moving lateral to the players body [Ripoll and Fleurance, 1988] due to the difference in measurement accuracy. In contrast to beginners, skilled players exhibit an earlier ball tracking onset [Rodrigues et al., 2002]. The final part of the ball trajectory starts with the last bounce of the ball, and ends when the ball and racket make contact. In this time period, head and eyes are

fixed on the estimated area of ball-racket contact [Ripoll and Fleurance, 1988]. Ripoll and Fleurance concluded from their studies that it is not necessary for expert players to track the ball throughout the entire trajectory. In contrast to the experts, beginners track the ball over the whole trajectory.

The observation of the ball during the first parts of the ball flight allows the player to obtain necessary information on the location of the bounce and for planning the striking motion. The eye-head stabilization in the period before ball-racket contact increases the sensitivity of the moving ball's peripheral image [Ripoll and Fleurance, 1988, Rodrigues et al., 2002]. This uncertainty reduction may be crucial for an accurate stroke generation.

Similar to the predictive eye-saccades towards a post-bounce position found in table tennis by Ripoll and Fleurance [1988] and Rodrigues et al. [2002], predictive eye-saccades towards the bouncing point was also found in cricket [Land and McLeod, 2000], squash [Hayhoe et al., 2012] and virtual racquetball [Diaz et al., 2013]. Moreover, Diaz et al. [2013] could show that these predictive eye movements do not only depend on visually available information of the pre-bounce ball trajectory such as the ball speed but also on non-visual information such as the elasticity of the ball. The authors argue further, that it is possible that player incorporate physical properties such as the coefficient of restitution in their predictions.

2.2.4 Movement Stages of a Stroke

Table tennis exhibits a regular, modular structure that has been studied by Ramanantsoa and Durey [1994]. They analyzed a top player and proposed four movement stages with respect to certain ball events, that is, bouncing, net crossing, and striking. According to their hypothesis, the following four stages can be distinguished during a match of expert player. For clarity, we have labeled them according to their functionality.

Awaiting Stage. The ball moves towards the opponent who returns it towards the net. The player's own racket is moving downwards. The player decides to use either a forehand or backhand stroke. At the end of this stage, the racket will be in a plane parallel to the table's surface.

Preparation Stage. The ball moves towards the player, has already passed the net, and will bounce off the table during this stage. The racket is moving backwards in order to prepare the stroke. For forehand strokes, the racket remains in the same plane as in the awaiting phase. The amplitude of the lateral displacements in the preparation stage depends on the ball's flight direction relative to the body's position when the ball crosses the net. The player chooses a point where he plans to intercept the ball. Ramanantsoa and Durey [1994] refer to this point as the virtual hitting point.

Hitting Stage. The ball moves towards the virtual hitting point where the player intercepts it. In a first substage, final adjustments are made. The precision of the stroke depends on the time available for the execution of this substage. During the second substage, the racket moves towards the virtual hitting point until it hits the ball. For expert players, the temporal duration of this phase appears to be constant and lasts approximately 80 ms.

Finishing Stage. After having been hit, the ball moves towards the opponent while the racket is moving upwards to a stopping position. This stage ends with the ball crossing the net and the velocity of the racket tending to zero.

We have verified the stages suggested by Ramanantsoa and Durey [1994] in a VICON motion capture setup with two naive players where each of the stages can be observed distinctly (see Figure 2.1). Color

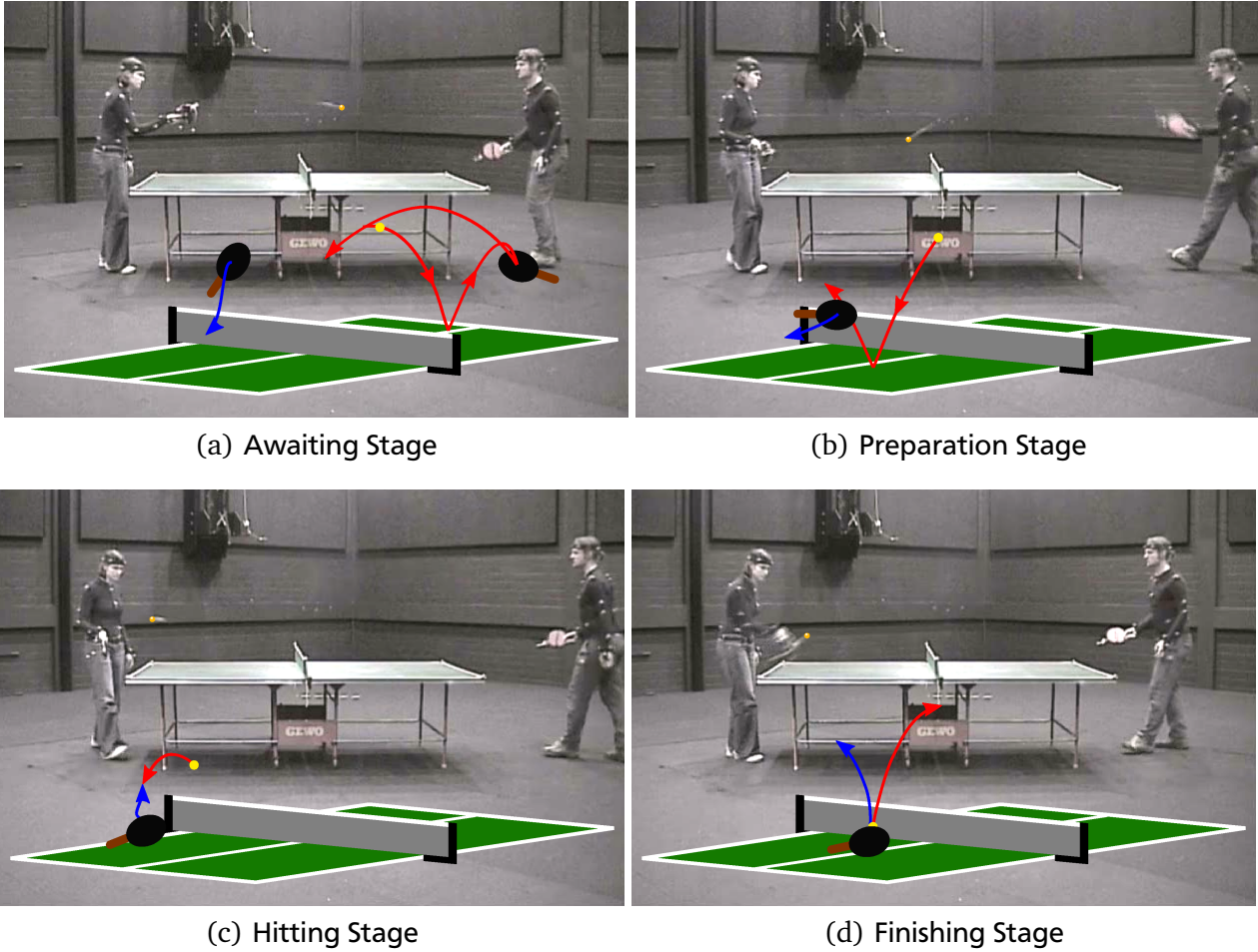


Figure 2.1: This figure illustrates the four movement stages of Ramanantsoa and Durey [1994] recorded in a VICON motion capture system where (a) shows the Awaiting Stage in which the opponent is observed, (b) the Preparation Stage in which the stroke is prepared, (c) the Hitting Stage in which the ball is intercepted, and (d) the Finishing Stage. The red and blue arrow show the movement of the ball and the racket, respectively, in each stage.

coded trajectories of the racket for one hitting motion starting with the awaiting stage can be seen in Figure 2.2. Note that for expert table tennis players the temporal and spacial timing would be more pronounced.

From a computational point of view, this model corresponds to a finite state automaton. By encoding such a sequence in the internal states of the policy, similar movements can be generated artificially.

2.3 A Biologically-Inspired Trajectory Generator for Table Tennis Strokes

To evaluate and use the observation-based model presented in Section 2.2, we have developed a computational realization that is suitable for real-time execution on a robot. We proceed as follows: First, we discuss the system's key elements in an overview. Subsequently, we explain how to determine the goal parameters of table tennis, that is, the time-to-contact and the virtual hitting point. Furthermore, we describe the realization of the movement generation for the striking motion, which includes the resolution of redundancy by minimizing the discomfort as described in Section 2.2.1.

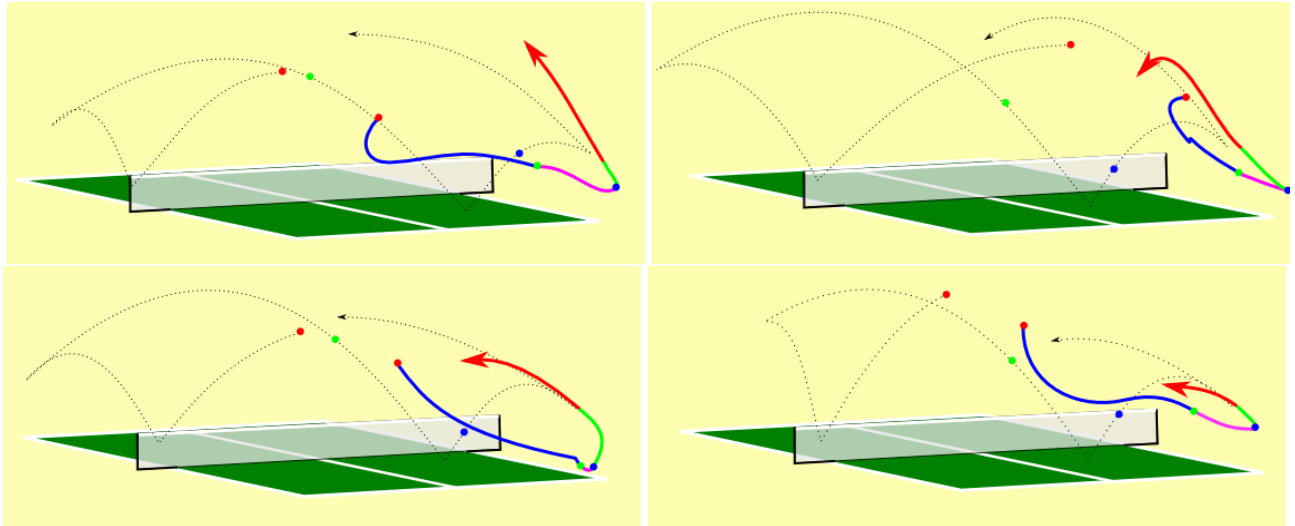


Figure 2.2: This figure shows different trajectories of intermediate table tennis players for one hitting motion. The trajectories are color coded according to the stages suggested by Ramanantsoa and Durey. The awaiting stage is colored in blue, the preparation stage in magenta, the hitting stage in green and the follow through stage in red. Colored circles show the corresponding position on the ball and arm trajectory respectively.

2.3.1 Overview of the Biomimetic Player

We adopt the movement stages of the model by Ramanantsoa and Durey [1994] as outlined in Section 2.2.4, and use a finite state automaton to represent this model. Subsequently, we have to plan the arm’s movement for each of the four stages.

We decided to plan the trajectories in the robot’s joint space. Planning in joint space has a variety of advantages over planning in task space. A joint-space trajectory can be directly controlled and does not require an intermediate inverse kinematics mapping. Thus, we avoid an additional nonlinear component in the control loop. In order to realize the movement program, we rely on a spline-based representation to encode the trajectory for each stage and each DoF. More details are given in Section 2.3.6.

To generate the arm’s trajectories, we have to determine constraints for the movements of each DoF. While desired final joint configurations suffice for the awaiting, preparation and finishing stages, the hitting stage requires a well-chosen movement goal. To determine the goal parameters, that is, the position, velocity and orientation of the end-effector, we rely on the virtual hitting point hypothesis [Ramanantsoa and Durey, 1994]. The goal parameters are chosen before the hitting motion begins. Therefore, the system has to first choose a point $\mathbf{x}_{\text{table}}$ on the opponent’s court where the ball will be returned to. The choice of $\mathbf{x}_{\text{table}}$ is part of a higher level strategy. Second, the system needs to determine the interception point of the ball and racket trajectories, which specifies the virtual hitting point \mathbf{x}_{hp} . The hitting point is the intersection point of the ball with the virtual hitting plane of the robot. The fixed virtual hitting plane used in our setup is illustrated in Figure 2.3. Given $\mathbf{x}_{\text{table}}$ and \mathbf{x}_{hp} , we can compute the batting position and velocity of the racket. Unfortunately, there are still infinitely many arm configurations that correspond to the desired racket position and velocity. To resolve this ambiguity problem, the system minimizes the distance to a comfort posture in joint space as proposed by Cruse et al. [1990] for human motor control. Doing so, we are able to find the orientation of the end-effector at the hitting point closest to the desired posture (see Section 2.3.3). The corresponding joint configuration for the virtual hitting point can then be computed using a quaternion-based inverse kinematics (see Section 2.3.4).

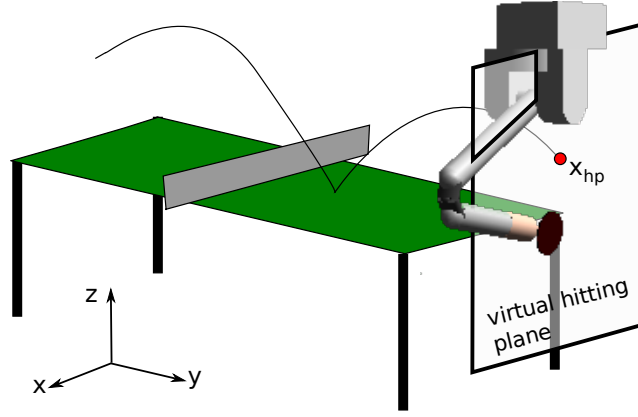


Figure 2.3: This figure illustrates the virtual hitting plane in the table tennis setup. The intersection point of the ball trajectory with the virtual hitting plane defines the virtual hitting point \mathbf{x}_{hp} . The x direction of the world coordinate system is parallel to the net, the y direction is parallel to the long side of the table and the z direction goes upwards.

The hitting movement is initiated when the time t_{hp} to the intersection of the ball with the virtual hitting plane is below a threshold. This step requires the system to predict when the ball is going to reach the virtual hitting plane. The expected hitting time can be estimated by predicting the trajectory of the ball using a model of the ball's dynamics, which is described in Section 2.3.2. Following the suggestion of Bootsma and van Wieringen [1988] that some online adaptation of the movement can be performed, the virtual hitting point is updated during the execution of the movement. As a result, the movement generation for the hitting and finishing stages is adapted to the new hitting point up to 100 ms before the predicted interception. This time period corresponds approximately to the human visuomotor delay [Abbs and Cole, 1987].

In order to realize these four stages, the system needs to detect the ball and determine its position \mathbf{x}_b . The vision system consists of two stereo cameras with Prosilica GE640C Gigabit Ethernet cameras and a GPU-based 60 Hz blob detection. Due to noise on the sensor level, the visual information is filtered using an extended Kalman filter (EKF) [Sorenson, 1985]. The system's equations used for the EKF are given in Equation (2.1). The resulting algorithm is described in Algorithm 1.

2.3.2 Dynamics Model of the Table Tennis Ball

To predict the position and velocity of the ball at time t_{j+1} based on the state at time t_j , we have to model the aerodynamics of the ball and the physics of the ball's bounces on the table. The ballistic flight model needs to incorporate air drag, gravity, and spin. As the latter is hard to observe from visual measurements, our model neglects spin. For table tennis balls, we can assume that the air drag is proportional to the square of the velocity of the ball. Using symplectic Euler integration, we can model the ball movement in discrete time form by

$$\ddot{\mathbf{x}}_b^{j+1} = \mathbf{g} - C \|\dot{\mathbf{x}}_b^j\| \dot{\mathbf{x}}_b^j, \quad \dot{\mathbf{x}}_b^{j+1} = \dot{\mathbf{x}}_b^j + \ddot{\mathbf{x}}_b^{j+1} \Delta t, \quad \mathbf{x}_b^{j+1} = \mathbf{x}_b^j + \dot{\mathbf{x}}_b^{j+1} \Delta t, \quad (2.1)$$

where $\ddot{\mathbf{x}}_b$, $\dot{\mathbf{x}}_b$, and \mathbf{x}_b denote the acceleration, velocity, and position vector of the ball respectively, \mathbf{g} the gravity vector and Δt the time difference. The air resistance scale factor $C = c_w \rho A / (2m)$ is determined by the drag coefficient c_w , the density of the air ρ , the size of the ball's cross-sectional area A and the mass of the table tennis ball m . For the bouncing behavior of the ball on the table, we assume velocity changes in all three directions. This change in velocity $\dot{\mathbf{x}}_b^{j+1} = \boldsymbol{\varepsilon}_T \dot{\mathbf{x}}_b^j$ is determined by the coefficient $\boldsymbol{\varepsilon}_T = [\varepsilon_{Tx}, \varepsilon_{Ty}, -\varepsilon_{Tz}]^T$ where ε_{Tx} is the coefficient of friction on the table and ε_{Tz} is the coefficient of restitution.

2.3.3 Determining the Goal Parameters

After estimating the virtual hitting point, we need to determine the orientation and velocity of the end-effector. Therefore, the system chooses the height z_{net} at which the returning ball should pass over the net, as well as the position $\mathbf{x}_{\text{table}}$ where the ball should bounce on the opponent's court. The choice of these variables belongs to the strategy of the system. A first step towards such a strategy can be found in Chapter 4. For the purpose of demonstration, we will chose z_{net} and $\mathbf{x}_{\text{table}}$ independently from sets of possible values according to a uniform distribution. To determine the goal parameters, we have to first compute the desired outgoing velocity \mathbf{o} of the ball that is, the velocity of the ball at the moment after the ball-racket impact. Directly from it, we can determine the required velocity and orientation of the racket.

Desired Outgoing Velocity

To compute the velocity \mathbf{o} of the ball after the ball-racket impact at \mathbf{x}_{hp} , we need to know which requirements the resulting trajectory has to fulfil. First, the ball has to pass the net at a certain height z_{net} . Second, the ball has to bounce at the desired target $\mathbf{x}_{\text{table}}$ on the opponent's court. These two constraints lead to the following nonlinear equations

$$\begin{aligned} x_{\text{net}} &= x_{\text{hp}} + o_x t_{\text{net}} - 0.5Cv o_x t_{\text{net}}^2 \\ z_{\text{net}} &= z_{\text{hp}} + o_z t_{\text{net}} - 0.5Cv o_z t_{\text{net}}^2 \\ \mathbf{x}_{\text{table}} &= \mathbf{x}_{\text{hp}} + \mathbf{o} t_{\text{tab}} - 0.5Cv \mathbf{o} t_{\text{tab}}^2 \end{aligned}$$

where $\mathbf{x}_{\text{net}} = [x_{\text{net}}, z_{\text{net}}]^T$, $\mathbf{x}_{\text{hp}} = [x_{\text{hp}}, y_{\text{hp}}, z_{\text{hp}}]^T$, $\mathbf{x}_{\text{table}} = [x_t, y_t, z_t]^T$, $\mathbf{o} = [o_x, o_y, o_z]^T$, x_{net} is the x-position of the net, z_t is the height of the table, and $v = \sqrt{o_x^2 + o_y^2 + o_z^2}$. The time variables t_{net} and t_{tab} correspond to the time at which the ball passes over the net and reaches the target of the opponent court respectively. Since these equations are nonlinear in the variables of interests, we have to solve the problem numerically. Therefore, we use a globally convergent solver for nonlinear equation systems, which combines the Newton-Raphson update with a modification for global convergence [Dennis and Schnabel, 1983].

Racket Orientation

The orientation of the end-effector is specified as a rotation that transforms the normal vector of the end-effector \mathbf{n}_e to the desired normal vector \mathbf{n}_{ed} . To define \mathbf{n}_{ed} , we have to compute the desired normal direction of the racket \mathbf{n}_{rd} that results in the desired outgoing vector \mathbf{o} of the ball given the velocity vector \mathbf{i} at the hitting point before impact (see Figure 2.4).

If we assume only a velocity change $\mathbf{o} - \mathbf{i}$ in the normal direction of the racket \mathbf{n}_{rd} , we obtain $\mathbf{o} - \mathbf{i} = \mathbf{n}_{\text{rd}}(o_{\parallel} - i_{\parallel})$, where o_{\parallel} and i_{\parallel} denotes the components of \mathbf{o} and \mathbf{i} , respectively, which are parallel to the desired racket normal. Note, that if we assume the absence of spin, all changes in velocity occur in this component. Hence, $\|\mathbf{o} - \mathbf{i}\| = o_{\parallel} - i_{\parallel}$. Thus, we can determine the desired racket normal by

$$\mathbf{n}_{\text{rd}} = \frac{\mathbf{o} - \mathbf{i}}{\|\mathbf{o} - \mathbf{i}\|}. \quad (2.2)$$

The rotation of \mathbf{n}_e to \mathbf{n}_{ed} , which defines the orientation, is represented in terms of quaternion $q_{\text{ed}} = q_{\text{rd}}q_{\text{re}}$ where q_{re} is the quaternion that describes the rotation from the racket to the end-effector and $q_{\text{rd}} = (\cos(\gamma/2), \mathbf{u} \sin(\gamma/2))$, with $\gamma = \mathbf{n}_e^T \mathbf{n}_{\text{rd}} / (\|\mathbf{n}_e\| \|\mathbf{n}_{\text{rd}}\|)$ and $\mathbf{u} = \mathbf{n}_e \times \mathbf{n}_{\text{rd}} / \|\mathbf{n}_e \times \mathbf{n}_{\text{rd}}\|$, defines

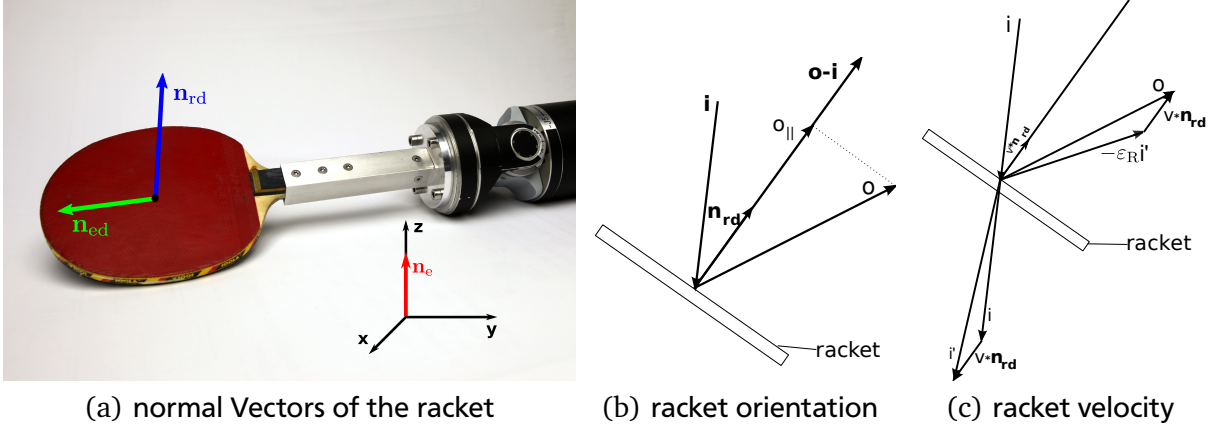


Figure 2.4: Computation of the desired normal vector \mathbf{n}_{ed} of the end-effector and the velocity based on the orientation of the racket \mathbf{n}_{rd} , the velocity \mathbf{o} and \mathbf{i} of the ball after and before the impact respectively. Figure (a) illustrates the normal vector \mathbf{n}_e , the desired orientation of the racket \mathbf{n}_{rd} and the resulting desired orientation \mathbf{n}_{ed} . The normal of the end-effector \mathbf{n}_{ed} is perpendicular to \mathbf{n}_{rd} . Figure (b) shows the relationship between \mathbf{n}_{rd} , \mathbf{o} and \mathbf{i} . Assuming the absence of spin and a speed change only in the $\mathbf{o} - \mathbf{i}$ direction, \mathbf{n}_{rd} is given by the normalized difference vector of \mathbf{o} and \mathbf{i} . Figure (c) illustrates the computation of the velocity \mathbf{v} of the racket based on the relation of \mathbf{v} , \mathbf{n}_{rd} , \mathbf{o} and \mathbf{i} .

the transformation of the normal of the end-effector \mathbf{n}_e to the desired racket normal \mathbf{n}_{rd} . See Figure 2.4 for an illustration of the computation of the racket normal used to define the orientation of the end-effector.

As there exist infinitely many racket orientations that have the same racket normal, we need to determine the final orientation depending on a preferred joint configuration. The resulting quaternion of the end-effector q_{ed} is determined by a rotation about \mathbf{n}_{rd} . The corresponding joint values and velocities are then computed using inverse kinematics (see Section 2.3.4). The orientation whose corresponding joint configuration θ_{hp} yields the minimum distance to the comfort posture θ_{com} is used as the desired racket orientation. Therefore, we weight each DoF according to its contribution to the cost function. The comfort position θ_{com} is a fixed joint configuration for each of the different striking motions (see Section 2.3.5). We choose this configuration such that the distance to the joint limits is as large as possible.

Required Racket Velocity

After computing the orientation of the end-effector, we have to calculate the velocity of the end-effector at the time of the ball's interception. We can describe the relation between the components of the incoming and outgoing velocity parallel to the racket norm as $o_{||} - v = \epsilon_R(-i_{||} + v)$, where ϵ_R denotes the coefficient of restitution of the racket, v the speed of the racket along its normal. This equation can be solved for v which yields the desired racket velocity

$$\mathbf{v} = \frac{o_{||} + \epsilon_R i_{||}}{1 + \epsilon_R}. \quad (2.3)$$

Given the velocity and orientation of the racket, we now have all necessary information to specify the virtual hitting point.

2.3.4 Translating Virtual Hitting Points into Configurations

From the virtual hitting point, we have computed the position, orientation and velocity of the end-effector in task space (i.e., Cartesian positions and rotations in the form of quaternions). As we plan the motions in joint space (as discussed in Section 2.3.1), we have to compute the corresponding joint state $\boldsymbol{\theta}$ consisting of the joint angles θ_i , joint velocities $\dot{\theta}_i$ and joint accelerations $\ddot{\theta}_i$ for each joint i of the arm. The transformation of joint values into Cartesian positions and velocities \mathbf{x} can be determined using the forward kinematics $\mathbf{x} = f(\boldsymbol{\theta})$. Hence, to realize the transformation from Cartesian space into the joint space, we require the inverse kinematics, that is, $\boldsymbol{\theta} = f^{-1}(\mathbf{x})$. Solving the inverse kinematics problem by analytically inverting the forward kinematics is not feasible, as there may exist infinitely many solutions due to the redundancy. To yield human-like motions, we follow the suggestion of Cruse et al. [1990] for resolving the biomechanical redundancy in humans. Hence, the inverse kinematics problem for the redundant DoFs can be solved numerically by minimizing the distance to the comfort posture in joint space, while finding a racket position and orientation that coincides with the virtual hitting point \mathbf{x}_{hp} . The cost function of the inverse kinematics problem is given by

$$\min_{\Delta\boldsymbol{\theta}_{0:T}} F = \sum_{t=0}^T \frac{1}{2} (\Delta\boldsymbol{\theta}_t + \boldsymbol{\theta}_t - \boldsymbol{\theta}_R)^T \mathbf{W} (\Delta\boldsymbol{\theta}_t + \boldsymbol{\theta}_t - \boldsymbol{\theta}_R) \quad (2.4)$$

$$\text{s.t. } \Delta\mathbf{x}_t = \mathbf{J}(\boldsymbol{\theta}_t) \Delta\boldsymbol{\theta}_t, \quad (2.5)$$

where $\boldsymbol{\theta}_R$ is the optimization posture, the change in the joint state $\Delta\boldsymbol{\theta} = \dot{\boldsymbol{\theta}} \delta t$, $\boldsymbol{\theta}_t$ is the current joint configuration of the robot and \mathbf{W} is a symmetric, positive definite weight matrix. Thus, we have the Lagrangian

$$L = \sum_{t=0}^T \frac{1}{2} (\Delta\boldsymbol{\theta}_t + \boldsymbol{\theta}_t - \boldsymbol{\theta}_R)^T \mathbf{W} (\Delta\boldsymbol{\theta}_t + \boldsymbol{\theta}_t - \boldsymbol{\theta}_R) + \boldsymbol{\lambda}_t^T (\Delta\mathbf{x} - \mathbf{J}(\boldsymbol{\theta}_t) (\Delta\boldsymbol{\theta}_t + \boldsymbol{\theta}_t - \boldsymbol{\theta}_R)). \quad (2.6)$$

Minimizing L with respect to $\boldsymbol{\lambda}$ and $\Delta\boldsymbol{\theta}$ yields

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \mathbf{J}_w^\dagger \Delta\mathbf{x}_t + (\mathbf{I} - \mathbf{J}_w^\dagger \mathbf{J}) (\boldsymbol{\theta}_t - \boldsymbol{\theta}_R), \quad (2.7)$$

where \mathbf{I} denotes the identity matrix, $\mathbf{J}_w^\dagger = \mathbf{J}_w^T (\mathbf{J}_w^T \mathbf{J}_w)^{-1}$ denotes a weighted pseudo inverse, and $\mathbf{J}_w = \mathbf{J} \mathbf{W}^{-1}$ denotes the weighted Jacobian. The final joint configuration for the desired hitting point can be computed by iteratively running the algorithm until the error is below a certain error threshold. Note that this method may have numerical problems at singularities where the rank reduction makes the Jacobian inversion impossible and an additional ridge term is needed. Nevertheless, this method enables a fast computation of the inverse kinematic function that provides control over the arm configurations and is used for the transformation of the Cartesian position and orientation into joint space.

The step length in task space $\Delta\mathbf{x} = [\Delta\mathbf{x}_p; \Delta\mathbf{x}_o]$ determines the accuracy of the cost and, hence, is essential for computing a short path. We define the position part of the Cartesian difference vector $\Delta\mathbf{x}$ as

$$\Delta\mathbf{x}_p = g_p (\mathbf{x}_{\text{hp}} - \mathbf{x}_e), \quad (2.8)$$

where g_p is a scaling factor and \mathbf{x}_e and \mathbf{x}_{hp} are the actual and desired Cartesian states of the end-effector, respectively. For orientation control, we use the quaternion error q^E between the actual and desired orientation for the second part of the step length

$$q^E = q_{\text{ed}} q_e^*, \quad (2.9)$$

$$\Delta\mathbf{x}_o = g_o [q_1^E, q_2^E, q_3^E]^T, \quad (2.10)$$

where q_{ed} is the quaternion describing the desired end-effector orientation, q_e^* denotes the complex conjugate of the actual quaternion of the end-effector orientation and g_o defines a scaling factor. The scaling factors g_p and g_o determine the convergence speed and robustness of the method; large values result in a higher convergence speed, but the algorithm may never find a solution if the values are set too high.

When the algorithm cannot find an appropriate joint configuration, the system decides not to return the incoming ball.

2.3.5 Movement Parameters

To return an incoming ball, the movement is generated for each of the four stages. We determine the start and end position, velocity and acceleration for each of the four stages as described in Section 2.3.1. The start and end positions for all stages are fixed except the hitting point which defines the end configuration of the hitting stage and the start configuration of the finishing stage. The fixed start and end positions are chosen to produce hitting movements similar to those exhibited by humans. The corresponding joint velocities and accelerations are set to zero. The configuration at the hitting point is computed for each stroke individually as described above. The duration of each stage (i.e., t_{as} , t_{ps} , t_{hs} , t_{fs}) is chosen such that the robot is able to execute the movement. The durations of the hitting stage t_{hs} is equal to the estimated time to hit t_{hp} but is at least 250 ms and at most 350 ms. If the estimated time to hit is less than 250 ms and the hitting stage is not yet initiated, the system decides not to execute the stroke, since this would require infeasible accelerations for the robot's joints.

Similar to humans who distinguish between several striking movements in table tennis, we have to implement different strokes to adapt to different hitting points in the workspace of the robot. Therefore, we divide the virtual hitting plane into three areas. A stroke movement is assigned to each of these areas, resulting in two forehand and one backhand strokes. Each stroke is defined by an individual start and end position for the individual stages and joints. When a ball moving towards the robot is detected, the virtual hitting point is estimated. Based on the area in which the predicted hitting point is located, the corresponding stroke type is chosen.

2.3.6 Movement Generation

The trajectory needs to be planned in joint space where high velocity movements can be executed more reliably. For the execution of the movements, we need a representation to obtain position $\theta(t)$, velocity $\dot{\theta}(t)$ and accelerations $\ddot{\theta}(t)$ of the joints of the manipulator at each point in time t such that it can be executed with an inverse dynamics-based controller [Spong et al., 2006]. Fifth order polynomials represent the trajectory at all stages (see Appendix A). Such polynomials are the minimal sufficient representation to generate smooth trajectories, and can be evaluated quickly [Craig, 1989]. Mimicking the four stage model of Ramanantsoa and Durey [1994], we select four different splines that interpolate between the initial and final configurations. As the trajectory of the hitting and finishing stages depends on the hitting point, trajectories have to be calculated jointly at the beginning of the hitting stage and have to be adjusted every time the virtual hitting point is updated.

The coefficients and final acceleration of the hitting stage are chosen such that the maximal acceleration during this stage is minimized (see Appendix). In addition to the minimization of the acceleration, this solution results in a decreased position overshoot before the hitting point. Thus, it reduces the risk of running into joint limits.

The pseudo-code summarizing the table tennis setup described in this section is shown in Algorithm 1.

Algorithm 1 Table Tennis Algorithm

Initialize: switch to *AwaitingStage*

repeat

 Extract ball position \mathbf{x}_b

 EK-Filter: $\mathbf{x}_b \rightarrow \mathbf{x}_t, \dot{\mathbf{x}}_t$

 EK-Prediction: $\mathbf{x}_t, \dot{\mathbf{x}}_t \rightarrow t_{\text{hp}}, \mathbf{x}_{\text{hp}}, \dot{\mathbf{x}}_{\text{hp}}$

 —————Switch Stage—————

if *FinishingStage* **and** *MovementEnds* **then**

 Switch to *AwaitingStage*

 Compute $\alpha_k = \mathbf{M}^{-1}(0, t_{\text{as}})\mathbf{b}_k^{\text{as}}$ for each DoF k

else if *AwaitingStage* **and** $t_{\text{hp}} \leq t_{\text{as}} + t_{\text{hs}}$ **then**

 Switch to *PreparationStage*

 Compute $\alpha_k = \mathbf{M}^{-1}(0, t_{\text{ps}})\mathbf{b}_k^{\text{ps}}$ for each DoF k

else if *PreparationStage* **and** $t_{\text{hp}} \leq t_{\text{hs}}$ **then**

 Switch to *HittingStage*

else if *HittingStage* **and** *BallHit* **then**

 Switch to *FinishingStage*

 Compute $\alpha_k = \mathbf{M}^{-1}(0, t_{\text{fs}})\mathbf{b}_k^{\text{fs}}$ for each DoF k

end if

 —————Update Striking Motion—————

if *HittingStage* **then**

 Solve with Newton-Raphson for \mathbf{o} using

$$f(\mathbf{o}, \mathbf{x}_{\text{hp}}, t_{\text{net}}) = \mathbf{x}_{\text{net}}$$

$$f(\mathbf{o}, \mathbf{x}_{\text{hp}}, t_{\text{table}}) = \mathbf{x}_{\text{table}}$$

 Determine joint configuration at hitting point

$$\mathbf{v} = \mathbf{o}_{\parallel} + \varepsilon_{\text{R}} \dot{\mathbf{i}}_{\parallel} / (1 + \varepsilon_{\text{R}})$$

$$\mathbf{n}_{\text{rd}} = \mathbf{o} - \mathbf{i} / (\|\mathbf{o} - \mathbf{i}\|)$$

$$q_{\text{ed}} = q_{\text{rd}} q_{\text{yrot}}$$

 Determine optimal rotation about \mathbf{n}_{rd} by

$$\boldsymbol{\theta}_{\text{opt}} = \arg \min_{\boldsymbol{\theta}_{\text{hp}}} \|\boldsymbol{\theta}_{\text{com}} - \boldsymbol{\theta}_{\text{hp}}\|$$

 with Inverse Kinematics: $\mathbf{x}_{\text{hp}}, q_{\text{ed}}, \mathbf{v} \rightarrow \boldsymbol{\theta}_{\text{hp}}$

 Compute $\alpha_k = \mathbf{M}^{-1}(0, t_{\text{hs}})\mathbf{b}_k^{\text{hs}}$ for each DoF k

end if

 —————Executing Movement—————

for each DoF k **do**

$$\theta_k = \sum_{l=1}^5 \alpha_{kl} t^l$$

end for

 Execute $(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$ with Inverse Dynamics Control.

until user stops program

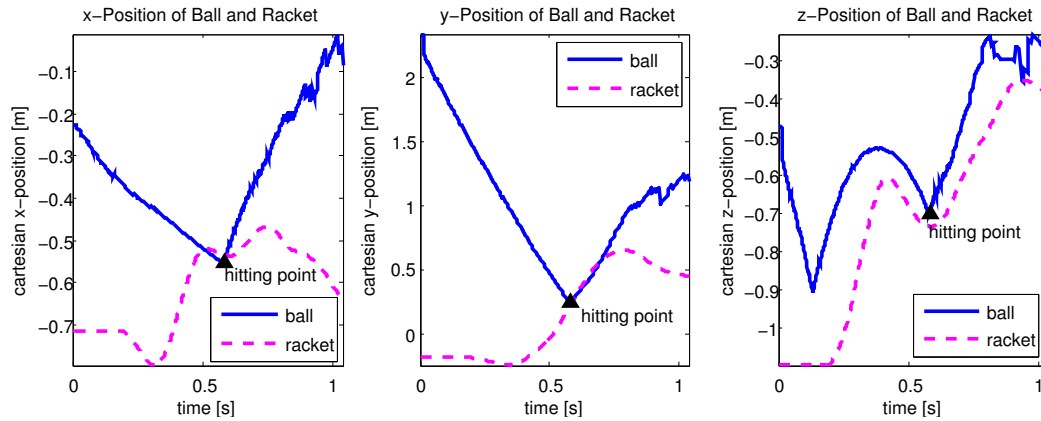


Figure 2.5: This figure shows the movement of the racket and the ball on the real Barrett WAM for a successful striking movement. The ball (solid blue line) moves towards the robot until it is hit by the racket (dashed magenta line) at the virtual hitting point (black triangle). The y-axis is aligned with the long side of the table tennis table, the x-axis is aligned with the width of the table and the z-axis goes upwards.

2.4 Evaluations

In this section, we demonstrate that robot table tennis is feasible with the proposed biomimetic player. To evaluate the performance of the player, we examined the accuracy in striking an incoming ball. We also analyzed the accuracy of the prediction of the virtual hitting point, the time of flight, and the accuracy of returning the ball to a desired position on the opponent's court. Furthermore, we analyzed the performance of the dynamics model of the ball and compared the biomimetic player's movement generation to a human player. For the experimental evaluations, we used a Barrett WAM arm, as well as a physically realistic simulated version of the complete setup.

2.4.1 Evaluation against a Ball Launcher

We assume a setup consisting of a Barrett WAM, a table, a racket, four cameras and a ball launcher. The Barrett WAM is an anthropomorphic seven DoF arm that is capable of high speed motion (e.g., for the end-effector we measured velocities of 6.5 m/s and accelerations of 234.6 m/s²). The robot is mounted in a hanging position from the ceiling. A standard racket, with a 16 cm diameter, is attached to the end-effector. The setup includes a standard sized table and a table tennis ball according to the ITTF rules. The ball is served randomly by a ball launcher to the work space of the robot. As a result, the ball passes the robot's end of the table in an area of approximately 2 m × 0.75 m. This area serves as the virtual hitting plane. The target point on the opponent's court is chosen randomly over the whole area.

Evaluation in a Simulated Environment

The table tennis system is capable of returning an incoming volley to the opponent's court which is served by a ping pong ball launcher at random times and to arbitrarily chosen positions in the workspace of the robot. In simulation, when the ball was served 1,000 times to random positions in the workspace of the robot, the system was able to return 95.5% of the balls. In 85% of the trials the ball was returned successfully to the opponent's court. In 4.5% of the trials the system refused to perform a hitting motion due to joint and acceleration limits. The mean deviation from the target point of the opponent's court was approximately 13 cm. The mean deviation of the position of the racket's mid-point to the ball at

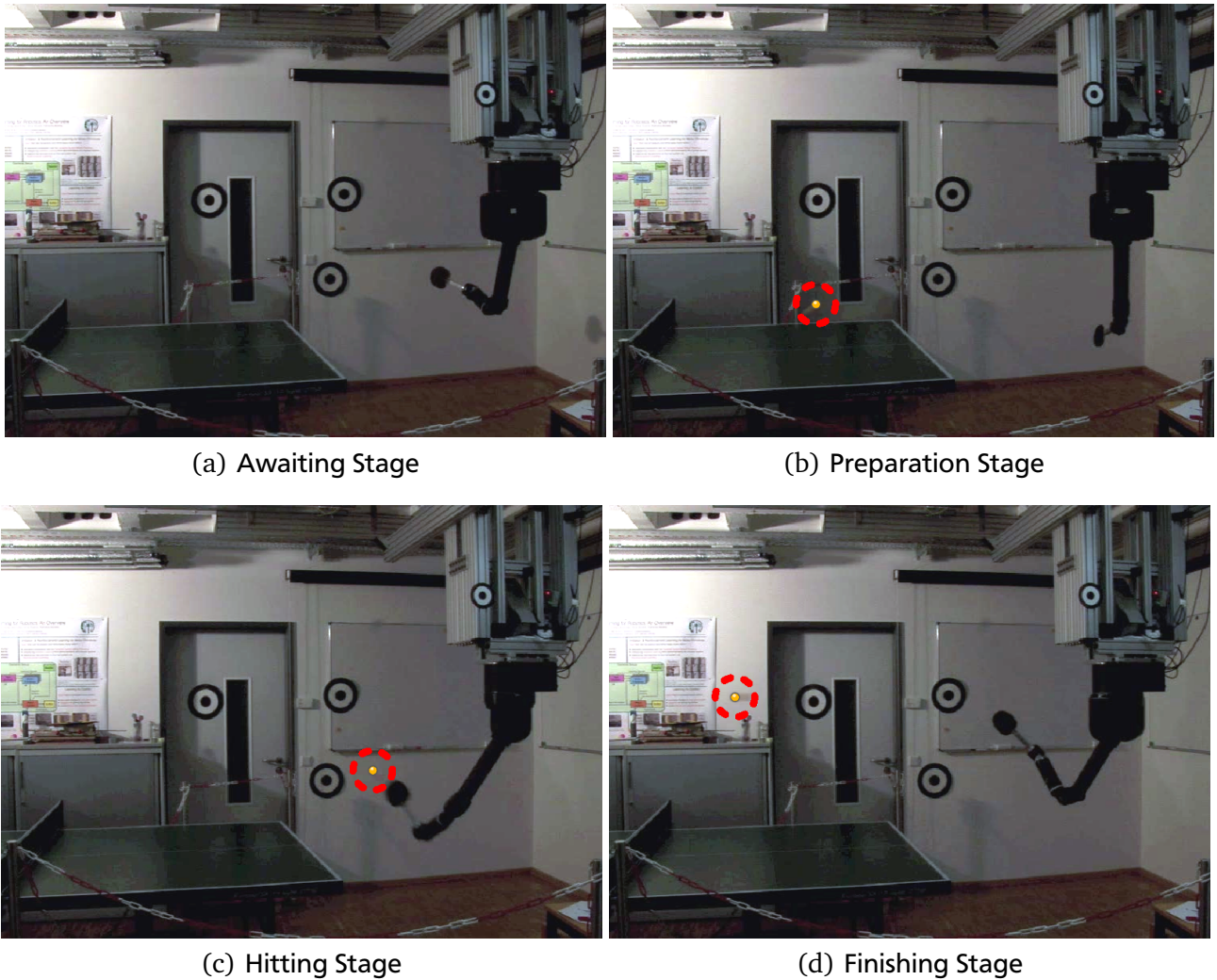


Figure 2.6: The figure shows the different stages, matching those in Figure 2.1, but performed by the real robot. At the beginning the robot is in the rest posture waiting for an incoming ball (Subfigure a). As a ball moves towards the robot, the arm performs a backswing motion to prepare the striking movement (Subfigure b). Based on the prediction of the ball the system chooses a hitting point. When the estimated time to contact reaches a critical value the arm moves towards the hitting point and hits the ball back to the opponent (Subfigure c). The robot arm moves upwards with decreasing velocity until the velocity is zero (Subfigure d).

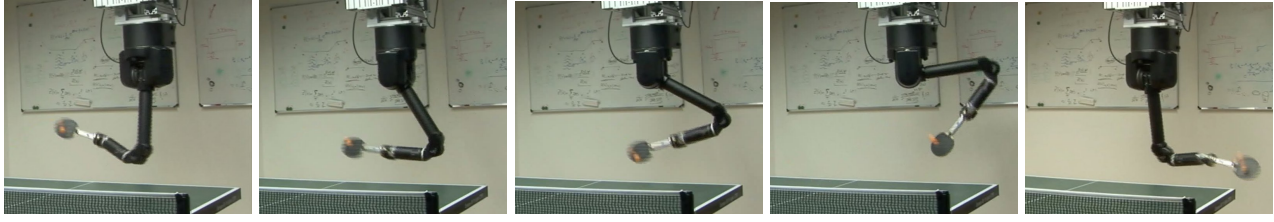


Figure 2.7: Various target of the biomimetic table tennis setup. The plane of possible hitting points has a length of 1.8 m.

the moment of contact was 3 cm. The estimated hitting point was corrected by 2.5 cm from the initial estimate. The estimated hitting time changed by approximately 20 ms. When the system gets the true position and velocity of the ball, the ball was returned to the opponent’s court in 96% of the trials. The simulation of the Barrett WAM robot arm was created using the SL framework [Schaal, 2009]. To build the table tennis environment, we used a model of the flight and bouncing behavior of the ball as discussed in Section 2.3.2. The coefficients of restitution of both racket-ball and ball-table interactions were determined experimentally ($\epsilon_R = 0.78$, $\epsilon_{T_x} = 0.73$, $\epsilon_{T_z} = 0.92$). As the parameters for the air drag, we estimated $c_w = 0.47$, $m = 0,0027$ kg, $\rho = 1.293$ kg/m³ and $A = 0.0013$ from real table tennis trajectories.

Evaluation on the Real System

Subsequently, we successfully transferred the setup onto a real Barrett WAM robot equipped with two partially overlapping stereo camera pairs. We used the same biomimetic player as in the simulated setup. The extended Kalman filter, based on the ballistic flight of a point mass with estimated restitution factors, tracks the ball well (for an extensive discussion see Section 2.4.2). However, the prediction of the virtual hitting point and time is less accurate in the real world than in simulation due to the neglected spin and the inaccuracies of the vision system. As ground truth data is not available, this can only be assessed qualitatively. Furthermore, all balls served by the ball launcher possess a large amount of topspin. Hence, these predictions needed to be updated more frequently and the trajectory generation adapts accordingly. A ball launcher served 150 balls to the fore- and backhand area of the robot. 99 % of these balls were returned by the system, approximately 62 % of those were returned successfully to the desired target on the opponent’s court. See Figure 2.6 for snapshots of a stroke and Figure 2.5 for trajectories of the racket and the ball of the real system. In the evaluations the robot covered a hitting plane with a length of 1.8 m illustrated in Figure 2.7. A video showing the performance of the system can be found at <http://www.youtube.com/watch?=BcJ4S4L1n78>.

2.4.2 Accuracy of the Ball Dynamics Model

We analyzed the accuracy of the ball’s dynamics model (see Section 2.3.2) on various ball trajectories. To assess the model’s quality, we recorded ball paths from a competitive and a cooperative human table tennis game. Furthermore, we recorded trajectories with extreme top-, side- and underspin to study the robustness of the proposed biomimetic player in returning these balls.

In order to investigate how well the dynamics model corresponds to the real system, we computed the deviation of the internal dynamics model from the trajectories captured by the vision system. Therefore, we first estimated the velocity of the ball at the bouncing point using the EKF. The trajectory of the ball was then precomputed for 200 ms using the dynamics model. The mean deviation of the ball position 200 ms after the bounce was estimated by computing the Root Mean Squared Error (RMSE) between

	Error in cm		
	x	y	z
Human game (cooperative)	03.43 ± 2.05	03.59 ± 2.40	3.19 ± 1.79
Human game (competitive)	04.03 ± 2.85	10.54 ± 3.51	2.58 ± 2.44
extreme Topspin	02.01 ± 1.57	38.58 ± 4.77	6.22 ± 3.06
extreme Sidespin	17.79 ± 1.46	04.63 ± 2.61	2.79 ± 1.44
extreme Underspin	01.17 ± 0.77	09.84 ± 3.53	4.90 ± 1.97

Table 2.2: Root Square Mean Error in centimeters of the deviation of the applied dynamics model of the ball and the vision information of the ball 200 ms after the bounce. The velocity of the model was set to the estimate of the EKF before bouncing. The y direction corresponds to the side direction of the table, x direction corresponds to the long side of the table and z to the height (see Figure 2.3).

both trajectories. In a cooperative game of two naive players (including the ball trajectories of 200 volleys) the average amount of deviation was $5.9 \text{ cm} \pm 2.6 \text{ cm}$. The deviation in each direction was smaller than $3.6 \text{ cm} \pm 2.4 \text{ cm}$. In the competitive game the player used different kinds of spin as well as smash strokes such that it was difficult for the opponent to return the ball. The mean deviation of the used dynamics model from the vision information was $11.59 \text{ cm} \pm 6.8 \text{ cm}$. The main difference to the cooperative game was the average amount of topspin and therefore the deviation in the y direction which increased from $3.6 \text{ cm} \pm 2.4$ to $10.54 \text{ cm} \pm 7.6 \text{ cm}$.

Unlike a human being, a ball launcher can create trajectories with only top-, side- or underspin. Hence, we analyzed the biomimetic player using trajectories produced by a ball launcher with regard to sensitivity to the different types of spin. Despite that the mean deviation of the topspin trajectory was $39.16 \text{ cm} \pm 4.5 \text{ cm}$ (with a deviation of $38.58 \text{ cm} \pm 4.8 \text{ cm}$ in y direction), the biomimetic player was still able to adapt to these changes and returned 61% of the balls successfully to the opponent's court. The underspin trajectories had a mean deviation of $11.05 \text{ cm} \pm 3 \text{ cm}$. Here, the biomimetic player was able to return 85% of the served balls successfully to the opponent's court. However, its performance decreased when facing sidespin as the biomimetic player was not able to adapt to these trajectories. The mean deviation from the observed trajectories was $18.6 \text{ cm} \pm 1.1 \text{ cm}$. Most prediction error was due to the differences in the x direction, that is, the direction along the net. Table 2.2 shows the deviations of the different recordings from the vision information 200 ms after the bounce for all three Cartesian directions. Note that the spin produced by the ball launcher was much higher than the spin that we found in games of naive human players.

2.4.3 Comparison to Human Behavior and Performance

The robot is able to return balls over the whole width of the table. However, compared to a human player, the robot lacks the acceleration abilities. Therefore, the human can adapt to unexpected ball trajectories after the ball bounces of the table much better than the robot. While the human can perform the hitting motion in 80 ms the robot needs 350 ms due to lower accelerations. Furthermore, the human is able to switch from for- to backhand while observing the ball moving towards him. The Barrett WAM may needs up to 500 ms for this action to avoid violating its acceleration and torque limits. The robot is not just limited by its acceleration abilities but also by the missing degrees of freedom; as it has no floating base, it cannot move sideways as well as back and forwards. Thus, to compare the performance of our biomimetic player to a human, we have to compare it to a human that is standing at a fixed

location. When the ball was served to the small forehand area of a stationary human, the performance of returning a ball successfully to an certain area of the table decreased to 50%. Without a given target on the table, the human was able to return 97 % of the balls successfully to the opponent's court.

We also compared the hitting movements of the robot qualitatively to the strikes of a naive human player. Similar to humans, the robot hits the ball in a circular hitting movement in an upward motion. The impact point of the racket and the ball is on an axis parallel to the table going through the base of the human and robot respectively.

2.5 Discussion and Conclusion of Chapter 2

In this section, we discuss the inclusion of strategy, optimization possibilities for the movement generation and spin estimation. Finally, we summarize the contributions of this chapter.

Including a Strategy

The presented evaluation is based on a strategy where the player does not attempt to beat the opponent. Thus, the player does not consider the game history and chooses the target on the opponent's court randomly. This randomized policy enables the player to return the ball successfully. However, this policy would not suffice to win against an intermediate player. Therefore, we have to choose the goal parameters according to the position and actions of the opponent. To realize this aim, a model of the opponent's strategy is necessary. From the observed game play, the robot player may be able to learn opponent's preferences and how to prepare for the opponent's return. See Chapter 4 for an detailed description of strategies in table tennis.

Optimizing the Movement Generation

Since the robot is limited in its accelerations abilities, switching between fore- and backhand while playing on the real system is often not possible. Predicting the expected hitting area based on the movements of the opponent would give the real system additional time to adjust its motion generation and to switch from fore- to backhand. Here, Wang et al. [2013] was able to show in a prototype setup that it is possible to infer whether to play a fore- or a backhand before the ball was hit by the opponent.

Estimating Spin

As discussed in the analysis of accuracy of the ball dynamics model, the inclusion of a spin estimate (especially sidespin) could improve the performance of the robot. However, spin cannot be determined from the few pixels in which the ball is observed by the camera. Even worse, the number of frames is insufficient to estimate spin based on the ball's trajectory. Hence, using only vision information of the ball would not allow us to receive a good model of the present spin. Opponent modeling could enable the system to observe the opponent's arm and racket movements and to use this data to predict the expected spin.

Conclusion

In this chapter, we have presented a novel table tennis setup which is based on biological assumptions on human movement generation. Our setup, with an anthropomorphic seven DoF robot arm and a human-inhabited environment, is significantly more challenging than the tailored ones of previous robot table tennis players. The movement structure and initiation is based on a hypothesis of the movement structure of expert table tennis players by Ramanantsoa and Durey [1994]. The resulting biomimetic player structures the table tennis stroke into four stages and uses a virtual hitting point and pre-shaping of the orientation to parametrize the goal. The redundancy of the arm is solved by minimizing the distance to a defined comfortable hitting posture. This concept has proven to be successful in operation and to produce human-like hitting motions. We demonstrated that the biomimetic player can be used as

an explicit policy for returning incoming table tennis balls to a desired point on the opponent's court in a physically realistic simulation as well as on the real Barrett WAM robot. A video can be found at <http://www.youtube.com/watch?v=BcJ4S4L1n78>.



3 Learning to Select and Generalize Striking Movements for Robot Table Tennis

The analytical model of robot table tennis presented in Chapter 2 has proven to produce successful hitting movements that are able to return balls to the opponent's court. However, such an approach also leads to rigid movement plans that require a programmer to implement this highly specific approach. Future autonomous robots need to be able to *execute* more than one specific task and, they need to be able to *learn* such complex motor tasks and to adapt them to new situations. In this chapter, we demonstrate how a complex motor task, such as table tennis, can be learned and how the elementary movements involved can be chosen. Therefore, we create a library of elementary hitting movements demonstrated by kinesthetic teach-in. We employ Dynamic system Motor Primitives (DMPs, [Ijspeert et al., 2002]) and present a refinement of the modification of the DMPs for hitting movements presented by Kober et al. [2010]. In order to enable the robot to associate the right motor primitives with different situations and to adapt those to a wider range of situations, we propose the *mixture of motor primitives* algorithm. We test the approach in a physically realistic simulation as well as on a real Barrett WAM arm. The following chapter is based on work, which was originally published in the Journal of Robotics Research [Muelling et al., 2013].

3.1 Prologue

Humans perform complex motor tasks in uncertain and changing environments with little apparent effort. They are able to generalize and, as a consequence, to adapt to new tasks based on their motor abilities. In contrast, current robots rely strongly on well-modeled environments with accurately estimated parameters. Therefore, changes in the environment or task require an expert to program new rules of motor behaviors. To alleviate this problem, robots need to autonomously learn new motor skills and improve their abilities. While this problem has been recognized by the robotics community [Schaal et al., 2002], it is far from being solved. Instead, learning new motor tasks autonomously and adapting motor skills online while interacting with the environment has become an important goal in robotics as well as machine learning. In recent years, it has become well accepted that for coping with the complexity involved in motor skill learning for robots, we need to rely on the insight that humans decompose motor tasks into smaller subtasks. These subtasks can be solved using a small number of generalizable movement pattern generators, also called movement primitives [Giszter et al., 2000, Billard et al., 2008, Flash and Hogan, 1985]. Movement primitives are sequences of motor commands executed in order to accomplish a given motor task. Efficient learning of movement primitives is crucial as the state space can be high dimensional and the number of scenarios that may need to be explored grows exponentially with the number of dimensions and time-steps [Schaal, 1999]. Here, learning from demonstrations can provide a good starting point for motor skill learning as it allows the efficient acquisition of single movement primitives [Guenter et al., 2007, Peters and Schaal, 2008a, Kober et al., 2008, Peters and Schaal, 2008b, Bitzer et al., 2010]. Most approaches for robot imitation learning are either based on physical demonstrations of a motion sequence to the robot by kinesthetic teach-in [Guenter et al., 2007, Peters and Schaal, 2008a,b, Bitzer et al., 2010] or through the use of a motion capture system such as a VICON setup [Kober et al., 2008]. We refer to [Schaal et al., 2003a, Billard et al., 2008, Argall et al., 2009] for a review of imitation learning methods.

To represent movement primitives such that they can adapt trajectories obtained by imitation learning to the requirements of the current task, several methods have been suggested. Several approaches make use of via-point based formulations using splines to interpolate between the via-points [Miyamoto et al., 1996], Hidden Markov Models (HMMs, [Williams et al., 2008]) or Gaussian Mixture Models (GMMs, [Calinon et al., 2007]). While these approaches are favorable in many situations, spline-based

via-point models are difficult to use in new situations while HMMs and GMMs are difficult to train for high-dimensional systems. Hence, an alternative approach based on dynamical systems was suggested by Ijspeert et al. [2002] and called DMPs (Dynamical system Motor Primitives). DMPs are robust against perturbations, allow changing the final state, speed, and duration of the motion without altering the overall shape of the movement. Furthermore, they are straightforward to learn by imitation learning [Ijspeert et al., 2002] and well suited for reward driven self-improvement [Kober and Peters, 2009]. DMPs have been successfully used to learn a variety of motor skills in robotics, including planar biped walking [Schaal et al., 2003b, Nakanishi et al., 2004], tennis-like swings to a static end-point [Ijspeert et al., 2002], T-ball batting [Peters and Schaal, 2006], constrained reaching tasks [Guenter et al., 2007], and Ball-in-a-cup [Kober et al., 2008]. However, up to now, most applications of learning and self-improving Ijspeert’s DMPs use only individual movement primitives to represent the whole motor skill. An exception is the work of Ude et al. [2010], in which the internal parameters of the DMPs are recomputed from a library of movement primitives in each trial using locally weighted regression. However, complex motor tasks require several movement primitives which are used in response to an environmental stimulus and the usage needs to be adapted according to the performance.

In this chapter of the thesis, we attempt to create such a framework based on the idea that complex motor tasks can frequently be solved using a relatively small number of movement primitives [Flash and Hogan, 1985] and do not require a complex monolithic approach. The goal of this part of the thesis is to acquire a library of movement primitives from demonstration (to which we will refer as movement library), to improve the performance of the stored movements with respect to the current situation as well as to select and generalize among these movement primitives to adapt to new situations. Each movement primitive stored in the library is associated with a set of parameters to which we refer as the augmented state that describes the situation present during demonstration. The primitives in the movement library are used as components in our Mixture of Motor Primitives (MoMP) algorithm. The MoMP algorithm activates components (i.e., single movement primitives) using a gating network based on the augmented state and generates a new movement using the activated components. Our approach is validated using robot table tennis as a benchmark task. In table tennis, the robot has to interact in real-time with a human partner. Thus, it has to adapt to the humans variable behavior. As already pointed out in Chapter 1 and 2, a typical movement in table tennis consists of the preparation of the stroke by moving backwards, hitting the ball at a desired position, with the right orientation and velocity and moving the arm back to a rest posture. The hitting movement itself may vary depending on the point of impact relative to the base of the robot, the time available until impact or the kind of stroke that should be performed. Furthermore, small inaccuracies in timing can lead to large deviations in the final bouncing point of the returned ball that can result in unsuccessful attempts to return the ball to the opponent’s court. The goal of this chapter is to learn autonomously from and with a human to return a table tennis ball to the opponent’s court and to adapt its movements accordingly. Therefore, the robot first learns a set of striking movements from a human teacher from physical human robot guiding, known as kinesthetic teach-in. Thus, the system recorded the demonstrated movement of the arm and the position of the ball. From this stream of data, the movement primitives were extracted. Secondly, the learning system needs to identify the augmented state that includes where, when and how the ball should be hit. Subsequently, the system generalizes these movements to a wider range of situations using the proposed MoMP algorithm. Here, generalizing refers to the ability to generate striking movements for an arbitrary incoming ball that was not seen during the demonstrations. The resulting system is able to return balls served by a ball launcher as well as to play in a match against a human.

In the remainder of the chapter, we will proceed as follows. In Section 3.2, we present our general framework for learning complex motor skills. We show (i) how to initialize the library using imitation learning, (ii) how to derive the augmented state from the state of the system and (iii) how to adapt the setup to new situations using the MoMP algorithm. We evaluate the MoMP approach in a robot

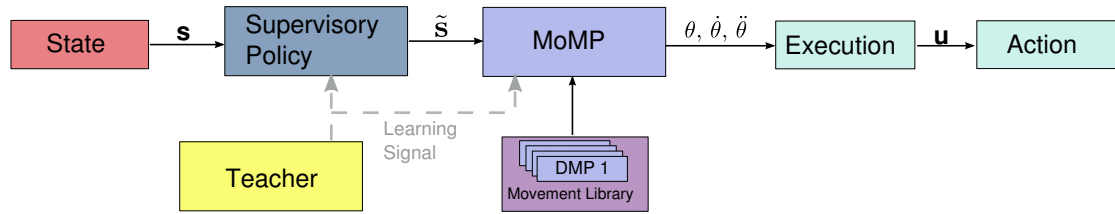


Figure 3.1: General setup for learning a motor task using the Mixture of Motor Primitives (MoMP). A supervisory level creates the augmented state \tilde{s} containing the relevant information of the task based on the state of the system. MoMP selects and generalizes among the movement templates in a library according to the augmented state \tilde{s} which is provided by the supervisory level. As a result we obtain a new motor policy that can be executed. A teacher provides learning signals to the supervisory level as well as the movement generation level such that the system is able to adapt both the generation of the task relevant information and the generation of the movement.

table tennis scenario in Section 3.3. Here, we will use all components of the presented motor skill learning framework to achieve this task. In Section 3.4, we present our results and summarize our approach. A glossary with the definition for the technical terms used in this chapter can be found in the Section Abbreviations and Glossary. A review of robot table tennis is presented in Chapter 2. The overall performance of our approach is illustrated in a video that can be found at <http://www.youtube.com/watch?v=SH3bADiB7uQ>.

3.2 Learning and Generalizing Motor Behaviors

In a complex motor task such as table tennis, we need to coordinate different movements which highly depend on a changing context. Unlike in many classical examples [Peters and Schaal, 2008b, Pongas et al., 2005, Nakanishi et al., 2004], single movement primitives which were demonstrated to work well in a certain situation do not necessarily perform equally well in other situations that might occur throughout the task. In table tennis, the movement profile depends strongly on where, when and how the ball has to be hit, as well as the velocity of the incoming ball and the desired target on the opponent’s court. As it is not feasible to demonstrate all possibly required movements to the robot, the system needs to generalize from a small number of movement primitives. Hence, we propose an algorithm called Mixture of Motor Primitives (MoMP) which generates a generalized movement for a given augmented state that can be executed by a robot. Therefore, a set of movement primitives and their corresponding augmented states are extracted from a set of demonstrations and stored in a library.

To generate a movement for a *new* augmented state \tilde{s} (that was not presented during demonstration), the system selects movement primitives from the library. Therefore, a parametrized gating network is used in the MoMP algorithm to activate movement primitives based on the presented augmented state. In some cases, the augmented state might be directly available as part of the state of the system. However, in table tennis, additionally parameters δ need to be estimated from the state by a supervisory system. The state of the system s consists of all variables necessary to model the system, e.g., in our table tennis task, the position and velocity of the ball moving towards the robot and the current joint configuration of the robot itself. The additional parameters δ of the augmented state \tilde{s} are given by the point of impact, the velocity and orientation of the racket at the hitting point, and the time until hitting the ball.

The supervisory system which generates the augmented state, as well as the gating network of the MoMP algorithm that selects and mixes the movement primitives according to the augmented state need to be adapted to improve the performance of the system. Figure 3.1 illustrates the general setup

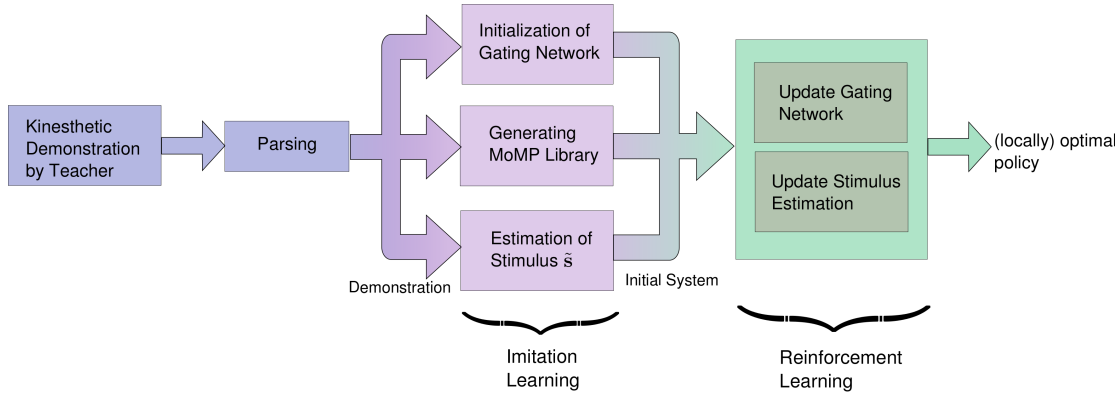


Figure 3.2: The learning process in the presented MoMP framework. Given a set of demonstrations recorded by kinesthetic teach-in, the system generates the movement library, initializes the gating network as well as the estimation of the augmented states used in the MoMP. The resulting initial system can be used to perform a given motor task. During the execution, the gating network, the augmented states estimation and the primitives are further improved using reinforcement learning.

for executing a motor task based on the current state of the system and the relation between the state, the augmented state and the movement generation. The learning structures involved in the MoMP algorithm are shown in Figure 3.2.

In the remainder of this section, we will first present the MoMP framework (Section 3.2.1). Subsequently, we explain how to compute the augmented state (Section 3.2.2). Finally, we show how to use and initialize DMPs as elementary motor policies in the MoMP framework (Section 3.2.3).

3.2.1 Learning a Motor Task using the Mixture of Motor Primitives

A movement performed by an artificial or biological system can be formalized as a policy

$$\mathbf{u} = \pi(\tilde{\mathbf{s}}, \mathbf{w})$$

that maps the state of the system, described by vector $\tilde{\mathbf{s}} = [\mathbf{s}, \boldsymbol{\delta}]$, to a control vector \mathbf{u} . The vector \mathbf{w} contains task-specific adjustable parameters. In the following, we will refer to such a policy as motor policy $\pi(\tilde{\mathbf{s}})$. The state of the system $\tilde{\mathbf{s}}$ corresponds here to the augmented state.

We generate the motor policy $\pi(\tilde{\mathbf{s}})$ based on a library consisting of L movement primitives. Each movement primitive $i \in \{1, \dots, L\}$ in the library is stored in a motor policy π_i . Additionally, for each motor policy π_i , the augmented state $\tilde{\mathbf{s}}_i$ associated with this movement primitive is stored. The movement library can be initialized using movements demonstrated in different situations of the task (more details on the acquisition of single primitives will follow in Section 3.2.3).

The MoMP generates a new movement for the current situation, represented by the augmented state $\tilde{\mathbf{s}}$ by computing the weighted average of all movement primitives π_i (see Figure 3.3). The resulting motor policy generated by MoMP is given by

$$\pi(\tilde{\mathbf{s}}) = \frac{\sum_{i=1}^L \gamma_i(\boldsymbol{\delta}) \pi_i(\tilde{\mathbf{s}})}{\sum_{j=1}^L \gamma_j(\boldsymbol{\delta})}, \quad (3.1)$$

where the function $\gamma_i(\boldsymbol{\delta})$ generates the weight of $\pi_i(\tilde{\mathbf{s}})$ given the augmented state $\tilde{\mathbf{s}} = [\mathbf{s}, \boldsymbol{\delta}]$. All weights γ_i together form the *gating network* of the MoMP similar to a gating network in a mixture of experts [Jacobs et al., 1991].

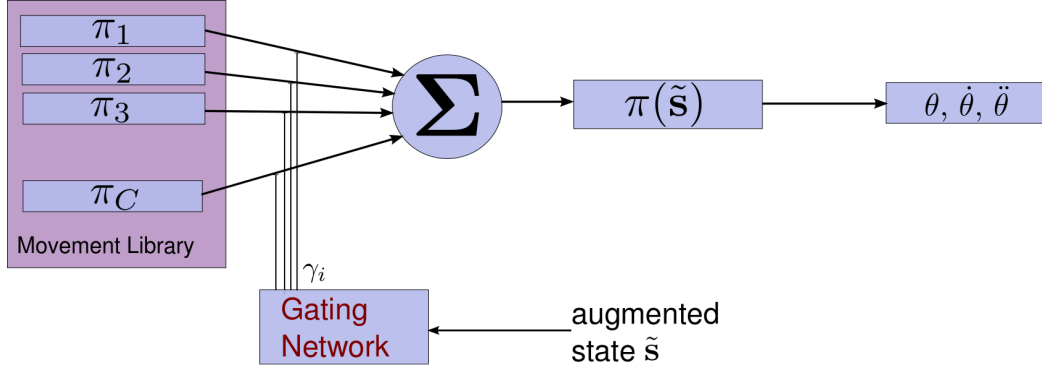


Figure 3.3: An illustration of the mixture of motor primitive framework. The gating network weights the single movement templates stored in a movement library based on an augmented state. The weighted sum of these primitives defines the new motor policy which produces the joint positions, velocities and accelerations for one degree of freedom. The resulting movement is then executed using a control law for execution which generates the required motor torques.

The weights of the gating network ensure that only movement primitives that are well-suited for the current situation contribute to the resulting behavior. It appears reasonable to assume that movement primitives associated with augmented states similar to the currently observed one are more likely to produce successful movements than movement primitives whose augmented states differ significantly from the observation. However, any large set of demonstrations will include rather poor attempts and, thus, some demonstrations are more suitable for generalization than others. Therefore, the gating network has to weight the movement primitives based on their expected performance within the current context. Such weights can be modeled by an exponential family distribution

$$\gamma_i(\tilde{\mathbf{s}}) = \xi \exp\{\boldsymbol{\lambda}_i^T \boldsymbol{\phi}_i(\tilde{\mathbf{s}})\}, \quad (3.2)$$

where $\boldsymbol{\phi}_i(\tilde{\mathbf{s}})$ denotes the feature vector of $\tilde{\mathbf{s}}$, $\boldsymbol{\lambda}_i$ is a vector containing internal parameters, and ξ is a normalization constant. The weight γ_i corresponds to the probability that the motor policy π_i is the right policy in the context described by $\tilde{\mathbf{s}}$, i.e., $\gamma_i(\tilde{\mathbf{s}}) = p(\pi_i|\tilde{\mathbf{s}})$. In an ideal world, there would be just one motor policy in the library that is perfectly suited for the current context. However, in practice, usually several motor policies correspond to this context imperfectly. Therefore, the MoMP needs to generalize among these motor policies, i.e., it mixes these movement primitives according to their weights γ_i in order to yield a motor policy that can be used in a broader context.

The choice of $\boldsymbol{\phi}_i(\tilde{\mathbf{s}})$ depends on the task. In our experiments however, a Gaussian basis function where the center is given by the augmented state $\tilde{\mathbf{s}}$ proved to be a good choice. The parameters $\boldsymbol{\lambda}_i$ of the probability distribution $\gamma_i(\tilde{\mathbf{s}})$ are unknown and have to be determined. If good features $\boldsymbol{\phi}(\tilde{\mathbf{s}})$ are known, linear regression methods are well suited to learn the parameters $\boldsymbol{\lambda}_i$ given examples of γ_i and the corresponding $\boldsymbol{\phi}(\tilde{\mathbf{s}})$. Hence, we use linear Bayesian regression [Bishop, 2006] to update the mean and variance of the parameters of the distribution online for each motor policy in the library (see Appendix C for a short review of linear Bayesian regression). The parameters are updated during the execution of the task based on the performance of the system. The performance can be measured by the reward r which is provided by a teacher to the system and corresponds in table tennis to the distance of the returned ball to the desired goal on the opponent's court. As a result, the system is able to adapt the choice of the used motor policies.

Altogether, the mixture of motor primitives selects and generalizes between movement primitives in the library based on the current augmented state $\tilde{\mathbf{s}}$. The resulting motor policy $\pi(\tilde{\mathbf{s}})$ is composed of *several* primitives weighted by their suitability in the given context of the task. The weights are determined by a gating network and adapted to the task based on the outcome of previous trials.

Algorithm 2 Generalizing Movements

Initialize:

Initiate movement library from demonstrations
Determine initial states \mathbf{S}^0 , costs \mathbf{C}^0 and meta-parameters \mathbf{D}^0 from demonstrations
Choose a kernel k , \mathbf{k} , \mathbf{K} and scaling parameter φ
Choose basis function $\phi(\tilde{\mathbf{s}})$, reward function r

For each trial

Determine current state \mathbf{s}^{N+1}
Choose $\delta^{N+1} \sim \mathcal{N}(\delta | \bar{\delta}(\mathbf{s}), \Sigma(\mathbf{s}))$ with CrKR, where
 $\bar{\delta}(\mathbf{s}) = \mathbf{k}(\mathbf{s})^T (\mathbf{K} + \varphi \mathbf{C})^{-1} \mathbf{D}$
 $\Sigma(\mathbf{s}) = k(\mathbf{s}, \mathbf{s}) + \varphi - \mathbf{k}(\mathbf{s})^T (\mathbf{K} + \varphi \mathbf{C})^{-1} \mathbf{k}(\mathbf{s})$
Set $\tilde{\mathbf{s}}^{N+1} = [\mathbf{s}^{N+1}, \delta^{N+1}]$

Calculate motor policy with MoMP

$$\pi(\tilde{\mathbf{s}}) = \frac{\sum_{i=1}^L \gamma_i(\delta) \pi_i(\tilde{\mathbf{s}})}{\sum_{j=1}^L \gamma_j(\delta)}$$

Execute motor policy $\pi(\tilde{\mathbf{s}})$

Determine reward r for executing $\pi(\tilde{\mathbf{s}})$

Update \mathbf{S} , \mathbf{D} and \mathbf{C} according to \mathbf{s} , δ and r

For all π_i in library update λ_i with LBR

$$c_i = \gamma_i / \sum_j \gamma_j$$

$$\mathbf{v}_i^{N+1} = \left((\mathbf{v}_i^N)^{-1} + \beta \phi(\tilde{\mathbf{s}}^{N+1})^T c_i \phi(\tilde{\mathbf{s}}^{N+1}) \right)^{-1}$$

$$\lambda_i^{N+1} = \mathbf{v}_i^{N+1} \left((\mathbf{v}_i^N)^{-1} \lambda_i^N + \beta \phi(\tilde{\mathbf{s}}^{N+1})^T c_i r \right)$$

end

3.2.2 Computation of the Augmented State

Some parameters required for the task are not part of the state and need to be computed. In table tennis, these parameters include the temporal and spacial interception point of the ball and the racket, as well as the velocity and orientation of the racket while hitting the ball. When planning in joint space, this corresponds to finding the position and velocity at the interception point for each joint. These parameters are an essential part of the generation of a desired movement with the motor policy $\pi(\tilde{\mathbf{s}})$ as they define the final state and duration of the movement. We refer to these additional parameters as the meta-parameter δ . The augmented state $\tilde{\mathbf{s}}$ is given by $\tilde{\mathbf{s}} = [\mathbf{s}, \delta]$.

One possibility of computing the meta-parameters is to predict the trajectory of the ball using an extended Kalman predictor starting with the current state of the system. Subsequently, we can determine a well suited hitting point on the trajectory and compute the desired velocity and orientation of the racket given a target on the opponent's court. Using inverse kinematics is one way to compute the corresponding joint configuration (see Chapter 2 for a detailed description).

An alternative possibility is to use an episodic reinforcement learning approach to acquire the mapping from \mathbf{s} to the meta-parameter δ directly. Here, Cost-regularized Kernel Regression (CrKR), see [Kober et al., 2012b], has also proven to be suitable for learning meta-parameters for motor policies as used in robot table tennis. CrKR uses a stochastic policy from which it draws the meta-parameters. The initial policy is obtained by training the algorithm with the demonstrations. The cost is used to weight the training data points down based on the reward r . See Appendix B for a short description of the CrKR algorithm.

In Algorithm 2, we show the complete approach of computing the meta-parameters δ using CrKR, the generation of a generalized motor policy using the MoMP, and the corresponding update according to the system's performance.

3.2.3 Representation of Behavior with Movement Primitives

To represent a single motor policy π_i used in the MoMP, we employ motor primitives represented by dynamical systems (DMPs). DMPs, as suggested in [Ijspeert et al., 2002, Schaal et al., 2007], are a particular kind of dynamical systems that is well-suited for imitation and reinforcement learning. It can be understood as a set of two differential equations that are referred to as the *canonical* and the *transformed* system. The canonical system h determines the phase z of the movement generated by

$$\dot{z} = h(z). \quad (3.3)$$

Intuitively, one could state that the canonical systems drives the transformed system similar to a clock. The transformed system

$$\mathbf{u} = \pi(\mathfrak{s}) = d(y, g_f, z, \mathbf{w}), \quad (3.4)$$

generates the desired movement for a single degree of freedom (DoF). It is a function of the current position y of the system, the final goal position g_f , the phase variable z and the internal parameter vector \mathbf{w} . The movement can be specified in joint or task space. The desired movement is specified by a dynamical system for each DoF, linked together by one shared canonical system.

DMPs allow us to represent arbitrarily shaped smooth movements by the parameter vector \mathbf{w} , which can be learned from demonstration by locally weighted regression (see Section 3.2.3). Furthermore, it is straightforward to adapt the DMPs with respect to the final position, movement amplitude and duration of the movement without changing the overall shape. In contrast to non-autonomous movement representations such as splines, DMPs are robust against perturbations in the environment, because time is not used explicitly. Furthermore, the DMP representation allows us to mimic a presented movement shape and, to represent arbitrary movements instead of only one special kind of movement. As a consequence, we can adapt the movement during execution to new movement goals and time constraints without re-planning the whole movement. However, the original formulation cannot be used for striking movements in a straightforward manner as the formulation does not account for non-zero end-velocities or via-points without changing the shape of the movement. Thus, the use of movement primitives for the whole striking movement with zero end-velocity is not suitable. The generated movements from such an approach will fail to return the ball successfully, since the trajectory is not guaranteed to cross the hitting point with the desired end-effector velocity at the right point in time. As a result, the success-rate of returning a ball to the opponent's court would decrease drastically. Hence, we need movement primitives that allow for different movement stages where the stages are switched based on features of the state. To achieve this goal, we introduce a type of two-stage movement primitive suited for hitting movements and use the feature of ball-racket contact to allow the system to switch the stage. Using the ball-based features, it becomes straightforward to compute the hitting point and therefore, the point where we have to switch between the two movement stages. While Kober et al. [2010] augmented Ijspeert's approach to make it possible to strike moving objects with an arbitrary velocity at the hitting point, we will show in the following the shortcomings of their approach and present an improved alternative form.

Movement Primitives for Striking Motions

For discrete movements (i.e., movements between fixed start and end positions such as reaching, pointing, grasping and striking movements) the canonical system is defined as

$$\tau \dot{z} = -\alpha_z z, \quad (3.5)$$

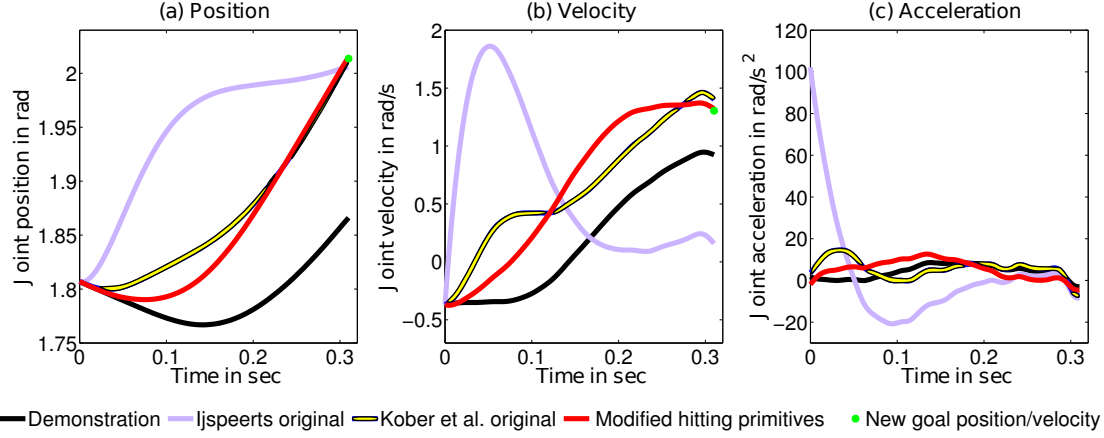


Figure 3.4: Changing the goal position and velocity is essential for adapting the demonstration to new situations. This figure illustrates how the different versions of the dynamical system based movement primitives are modulated by changing the goal position and velocity of the movement as they frequently occur in striking sports. The demonstration of a striking movement was obtained by kinesthetic teach-in in table tennis. After learning, all movement primitive formulations presented were able to reproduce the demonstration to a certain extend. However, the three formulations are differently robust against changes in the desired goal position and velocity. We changed the position by 0.15 m and the velocity by 0.4 m/s. The original formulation of Ijspeert is not able to reach the desired velocity as the system ends with zero velocity. The formulation of Kober et al. [2010] is able to adapt to a new final velocity. However, the accuracy of the adapted movement does not suffice for practical problems. The reformulation presented here reduces this inaccuracy drastically and stays closer to the desired movement shape.

where τ is a temporal scaling factor and α_z is a pre-defined constant which is chosen such that the behavior is stable [Ijspeert et al., 2002, Schaal et al., 2007]. Initially z is set to 1 and converges to zero at the end of the movement. Please note that as long as the time parameter of the movement does not change, we can integrate with respect to the time directly and use the solution $z(t) = \exp(-\alpha_z/\tau t)$ instead. However, implementing a change of the hitting time, slowing down or speeding up is not straight forward anymore. Therefore, when such parameter changes are essential (as in our setup), the phase should be computed as described in Equation 3.5.

For hitting movements, Kober et al. [2010] proposed the transformed system

$$\begin{aligned} \tau \dot{v} &= (1-z)\alpha_y \left(\beta_y(g-y) + \dot{g}_f \tau - v \right) + \eta f(z), \\ \tau \dot{y} &= v, \quad g = g^0 - \dot{g}_f \tau \frac{\ln(z)}{\alpha_z}, \end{aligned} \quad (3.6)$$

where y and v are the desired position and velocity generated by the policy, $\eta = (g_f - y_0)$ defines the amplitude of the movement, y_0 is the start position, g_f is the desired final position, \dot{g}_f the desired final velocity of the system, g is the current goal position defined by a moving target, $g^0 = g_f - \tau \dot{g}_f$ is the initial position of the moving target and $\tau \ln(z)/\alpha_z$ corresponds to the time. At the end of the movement, g is equal to the desired final goal g_f . For each new movement g and \dot{g} are computed based

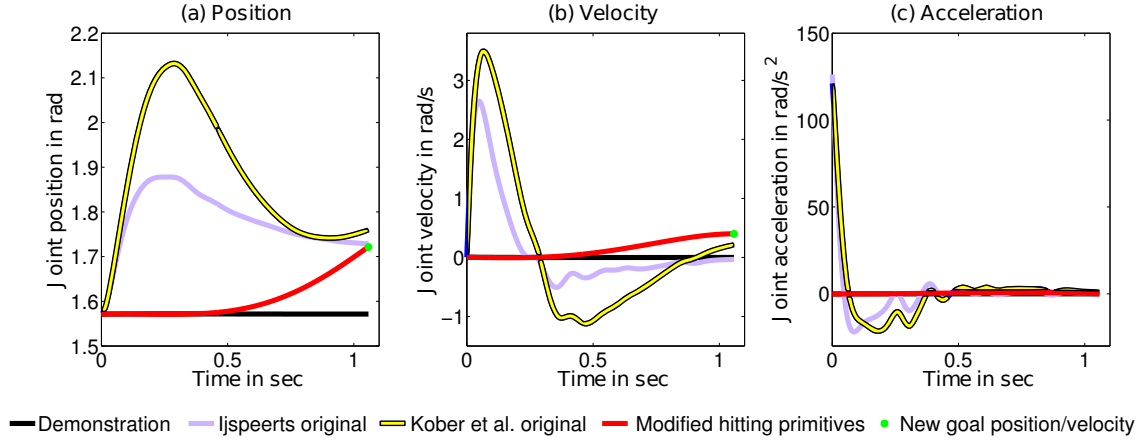


Figure 3.5: A key problem of the previous movement primitive formulation is the highly uneven distributed accelerations with a peak at the beginning of the movement. Such jumps can affect the position and velocity drastically as shown in this figure. They may result in the attempt to generate infeasible trajectories for real robots. Kober et al. reduced this effect by gradually activating the attractor dynamics. However, if the initial movement amplitude is close to zero in a demonstration jumps will occur when the goal position is changed. By introducing a new scaling term for f , we can avoid jumps at the beginning of the movement and reduce overshooting in the position and velocity profile. The demonstration was obtained by kinesthetic teach-in of a striking movement. Both position and velocity were then changed to a new goal. Note, that the acceleration of the modified hitting primitives is so much smaller that it appears to be zero when compared to Kober’s and Ijspeert’s original versions.

on the new desired goal position g_f and the desired final velocity \dot{g}_f . The pre-defined spring-damper constants α_y and β_y are chosen such that the system is critically damped. The transformation function

$$f = \frac{\sum_{j=1}^N w_j \psi_j(z) z}{\sum_{j=1}^N \psi_j(z)}, \quad (3.7)$$

employs Gaussian basis functions $\psi_j(z) = \exp(-\rho_j(z - \mu_j)^2)$ characterized by a center μ_j and a bandwidth ρ_j . N is the number of adjustable parameters w_j . The transformation function f alters the output of the spring damper model and thereby allows the generation of arbitrarily shaped movements. As z converges to zero at the end of the movement, the influence of the non-linear function f will vanish. The augmented form of the DMP enables us to pull the DoF simultaneously to a desired goal position and an arbitrary end velocity without changing the overall shape of the movement or its duration. If \dot{g} is set to zero, the formulation corresponds to the original formulation of Ijspeert except for the first term $(1 - z)$ which prevents large jumps in the acceleration at the beginning of the movement. A proof of the stability of the system can be found in Appendix D.

Modified Hitting Primitives

The formulation of Kober et al. [2010] with a linear velocity allows to directly incorporate the desired velocity such that the desired final velocity can be adapted (in addition to the desired initial and final position in Ijspeert’s formulation). However, this formulation has two drawbacks. First, fast movement changes of the final position and velocity can lead to inaccuracies in the final velocity (see Figure 3.4).

Algorithm 3 Mixture of Motor Primitives (MoMP)

Input: augmented state $\tilde{\mathbf{s}}$
for $t = 1$ to T **do**
 for $i = 1$ to L **do**
 Compute the phase variable z
 $\dot{z} = -\alpha_z z$
 and the transformed system
 $g = \sum_{j=0}^5 b_j \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^j$
 $\dot{g} = \sum_{j=1}^5 j b_j \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{j-1}$
 $\ddot{g} = \sum_{j=2}^5 (j^2 - j) b_j \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{j-2}$
 $\eta = \frac{(1+g_f - y_0)^2}{(1+a)^2}$
 $\tau \dot{v} = \alpha_y (\beta_y (g - y) + \dot{g} \tau - v) + \ddot{g} \tau^2 + \eta f(z)$
 $\tau \dot{y} = \dot{v}$
 end for
 Mixture of motor primitives output
 $\pi(\tilde{\mathbf{s}}) = \frac{\sum_{i=1}^L \gamma_i(\tilde{\mathbf{s}}) \pi_i(\tilde{\mathbf{s}})}{\sum_{i=1}^L \gamma_i(\tilde{\mathbf{s}})}$
 end for

In table tennis, such errors at the hitting point can cause the returned ball to miss the opponent's court. Second, if $(g_f - y_0)$ is close to zero, the scaling of f with $\eta = (g_f - y_0)$ can cause huge accelerations when g_f is changed (see Figure 3.5). Using a polynomial target velocity, we modified the transformed system to

$$\begin{aligned} \tau \dot{v} &= \alpha_y (\beta_y (g - y) + \dot{g} \tau - v) + \ddot{g} \tau^2 + \eta f(z), \\ \tau \dot{y} &= v, \quad \eta = \frac{(1 + g_f - y_0)^2}{(1 + a)^2}, \\ g &= \sum_{i=0}^5 b_i \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^i, \quad \dot{g} = \sum_{i=1}^5 i b_i \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{i-1}, \\ \ddot{g} &= \sum_{i=2}^5 (i^2 - i) b_i \left(-\tau \frac{\ln(z)}{\alpha_z}\right)^{i-2}, \end{aligned} \quad (3.8)$$

where g , \dot{g} and \ddot{g} are the current position, velocity and acceleration defined by the moving target and a is a reference amplitude. If the internal parameters \mathbf{w} are estimated by imitation learning as in the experiments performed in this chapter, a will correspond to the amplitude of the demonstrated movement. The parameters b_j are computed by applying the bounding conditions, i.e., the condition that the fifth order polynomial starts with the initial position, velocity and acceleration and ends at the desired goal g_f with the desired velocity \dot{g}_f and zero acceleration. Thus, using a fifth order polynomial allows us to control the initial and final position, velocity and acceleration more accurately. The new scaling term $\eta = (1 + g_f - y_0)^2 / (1 + a)^2$ ensures that f does not cause infeasible acceleration. To avoid infeasible acceleration profiles the velocity and acceleration is set constant outside the interval $[0, T_f]$. The term $(1 - z)$ is not necessary anymore, as jumps in the acceleration are avoided by incorporating the polynomial position profile starting at y_0 . Note that Ning et al. [2011] also used splines in a DMP setting. However, while Ning et al. [2011] always try to follow the reference trajectory and adapt only the beginning and ending of the trajectory, we attempt to adapt the whole trajectory. A proof of the

Algorithm 4 Imitation Learning of one DMP for MoMP

Input: $\Gamma_i = [\theta_t, \dot{\theta}_t, \ddot{\theta}_t]$, $t \in \{1, \dots, T_f\}$
For all DoF i and each parameter w_n **do**
 Set constant parameters $\alpha_z, \alpha_y, \beta_y, p_n$ and μ_n
 Set $g_f = \theta_\tau$ and $\dot{g}_f = \dot{\theta}_{T_f}$
 Calculate g, \dot{g} and \ddot{g}
 Calculate z_t by integrating $\tau \dot{z} = \alpha_z z$ for all t
 Calculate $\psi_t^n = \exp(-\rho_n(z_t - \mu_n)^2)$
 Calculate reference value f^{ref} from demonstration
 $f_t^{\text{ref}} = \tau^2 \ddot{\theta}_t - \alpha_y(\beta_y(g - \theta_t) + \dot{g} - \tau \dot{\theta}_t) - \ddot{g} \tau^2$
 Create matrices $\mathbf{z} = [z_1, \dots, z_{T_f}]^T$, $\Psi = \text{diag}(\psi_1^n, \dots, \psi_{T_f}^n)$ and $\mathbf{f}^{\text{ref}} = [f_1^{\text{ref}}, \dots, f_{T_f}^{\text{ref}}]^T$
 Compute weights via locally weighted regression
 $w_n = (\mathbf{z}^T \Psi \mathbf{z})^{-1} \mathbf{z}^T \Psi \mathbf{f}^{\text{ref}}$
end

stability of the system can be found in Appendix D. The complete algorithm for generalizing DMPs using MoMP is given in Algorithm 3.

Initialization of the Behaviors

A library of movement primitives can be built using demonstrated motor behaviors. Imitation learning for DMPs, as suggested in [Ijspeert et al., 2002, Schaal et al., 2003b], enables us to learn initial DMPs from observed trajectories and to reproduce these movements. The observed trajectories are captured using kinesthetic teach-in. To learn the DMP from a demonstration, we assume that the policy that produced the observed trajectory can be represented by a DMP as described in Equation (3.8). Consequently, the problem reduces to inferring the set of internal parameters \mathbf{w} such that the error between the reproduced and demonstrated behavior is minimized. Therefore, the problem can be solved straightforwardly by a regression algorithm.

In the following, we will assume that each DoF is independent and that we can learn each primitive separately. Redundant primitives can be eliminated [Chiappa et al., 2008]. For a recorded trajectory $\Gamma = [\theta_t, \dot{\theta}_t, \ddot{\theta}_t]$ over a time interval $t \in \{1, \dots, T_f\}$, we can compute the reference signal of a striking movement based on Equation (3.8) by

$$f_t^{\text{ref}} = \tau^2 \ddot{\theta}_t - \alpha_y(\beta_y(g - \theta_t) + \tau \dot{g} - \tau \dot{\theta}_t) - \ddot{g} \tau^2. \quad (3.9)$$

The appropriate cost function to be minimized for each parameter w_n is the weighted squared error

$$e_n^2 = \sum_{t=1}^{T_f} \psi_t^n (f_t^{\text{ref}} - z_t w_n)^2, \quad (3.10)$$

where $\psi_t^n = \exp(-\rho_n(z_t - \mu_n)^2)$ while z_t can be determined by integrating Equation (3.5). We can rewrite this error in matrix form by

$$e_n^2 = (\mathbf{f}^{\text{ref}} - \mathbf{z} w_n)^T \Psi_n (\mathbf{f}^{\text{ref}} - \mathbf{z} w_n), \quad (3.11)$$

where \mathbf{f}^{ref} is the vector consisting of f_t^{ref} for each time step t , $\Psi_n = \text{diag}\{\psi_1^n, \dots, \psi_{T_f}^n\}$ and $\mathbf{z} = [z_1, \dots, z_{T_f}]^T$. The resulting regression problem can be solved straightforwardly using locally weighted regression as originally suggested in [Ijspeert et al., 2002, Schaal et al., 2003b]. Thus, the optimal w_n are given by

$$w_n = (\mathbf{z}^T \Psi_n \mathbf{z})^{-1} \mathbf{z}^T \Psi_n \mathbf{f}^{\text{ref}}. \quad (3.12)$$

The detailed method for learning movement primitives with imitation learning for all DoFs is shown in Algorithm 4.

3.3 Evaluation

In Section 3.2, we have described a framework for selecting and generalizing movements based on augmented states as well as for computing the meta-parameters which are part of the augmented states from the state information of the environment. Here, we will show that we can use these methods to learn robot table tennis from and with a human being. Therefore, we will first give a short overview of the table tennis task and then evaluate the methods in simulation as well as on a real robot.

3.3.1 Robot Table Tennis Setup

For the robot table tennis task, we developed a system that includes a Barrett WAM arm with seven DoFs capable of high speed motion for hitting the ball and a vision system with four Prosilica Gigabit GE640C cameras for tracking the ball [Lampert and Peters, 2012]. The robot is mounted in a hanging position from the ceiling. A standard racket, with a 16 cm diameter, is attached to the end-effector. The setup incorporates a standard sized table tennis table and a table tennis ball in accordance with the International Table Tennis Federation (ITTF) rules [International Table Tennis Federation, 2011]. The ball is served either by a ball launcher to the forehand of the robot with a randomly chosen velocity or served by a human. The area covered is approximately 1 m^2 . The ball is visually tracked with a sampling rate of 60 Hz and the vision information is filtered using an extended Kalman filter (EKF). For the internal model of the EKF, we assume a simplified model of the flight and bouncing behavior of the ball, i.e., we consider gravity and air drag, but neglect the spin acting on the ball due to its limited observability. The world frame is fixed at the base of the robot, with the negative y -axis pointing towards the opponent and the z -axis pointing upwards.

If the visual system detects a table tennis ball that moves towards the robot, the system needs to compute the relevant movement information, i.e., where, when and how the ball has to be hit (see Section 3.3.2). The target location on the opponent's court was fixed to the center of the court to make the results comparable.

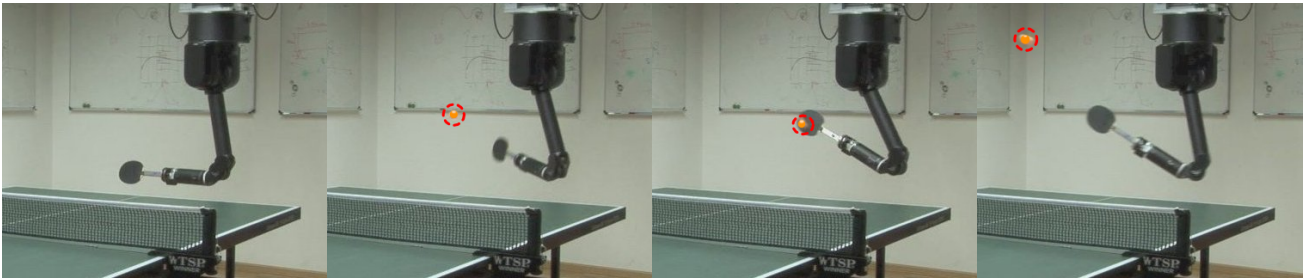
Special attention was given to prevent physical damage to the setup during movement execution. As a result, the system evaluates at each time point the position of the end-effector. If the end-effector was too close to the table, the movement would be stopped and the arm moves back to its rest posture. During the real robot evaluation, however, the table was 80 cm away from the real robot. Thus, the robot was not able to hit the table. However, even in simulation, where the table was at a distance of 30 cm from the robot, collisions with the table were rarely a problem. Balls which cannot be reached by the robot are ignored. To avoid joint limitations, the solutions for the joint configuration predicted for the hitting point were evaluated before performing the motion. If the joint configuration violated the joint limits, the solution would be rejected. Furthermore, the desired joint configuration for the next time step, was tested. If the joint limitations were violated, the movement would stop and not continue the planned movement. However, if additional object or joint limit avoidance is necessary, the movement could be adapted by adding a repellent force. In this case, a potential field centered around the obstacle (as described by Park et al. [2008] and Kroemer et al. [2010]) could be used. Luckily, these security precautions were never required in our experiments in practice as the MoMP generates movements within the convex combination of demonstrations. Hence, the system will not encounter joint limits or hit the table. Similarly, singularities were avoided by additionally restricting the hitting area of the robot.

3.3.2 Computing the Meta-Parameters

To use DMPs as motor policies in a table tennis game, the system needs to identify the hitting position, velocity and orientation of the racket, as well as the time until impact. These parameters need to be



(a) Physical human robot interaction: kinesthetic teach-in of a striking motion in table tennis.



(b) Reproduced hitting motion by imitation learning.

Figure 3.6: Sequence of a hitting motion in table tennis demonstrated by a human teacher and reproduced with a Barrett WAM arm with seven DoF. From the left to the right the single pictures represent the system at the end of the awaiting, preparing, hitting and follow through stage respectively.

estimated for each incoming ball in a match. When planning in joint space, the hitting position, velocity and orientation of the racket are defined by the joint configuration of the robot at this point in time. Altogether, 15 parameters need to be determined, which include the joint position and velocity of the seven DoFs of the Barrett WAM and the timing parameter that determines the initiation of the hitting movement. These values depend on the impact point of the racket and the ball, the velocity of the ball and the desired goal on the court of the opponent. As the goal on the opponent's court is kept constant, we can neglect this parameter.

We can learn the 15 task parameters using CrKR as described in Section 3.2.2 based on the position and velocity of the ball above the net, which is used as the state of the system. The Euclidean distance of the ball and the racket at the estimated hitting time is used as the cost function. At the beginning the robot misses the ball in 95% of the trials. At the end the trained robot hits almost all balls. A more detailed description and evaluation of this approach can be found in [Kober et al., 2012b].

Besides learning the meta-parameters directly, the position, velocity and orientation of the racket can be computed analytically based on the state of the system and the target on the opponent's court. These task space parameters can also be converted into joint space parameters using inverse kinematics as described in Chapter 2.

3.3.3 Mixture of Motor Primitives

To analyze the performance of our system described in Section 3.2, we analyzed the MoMP framework in simulation as well as on a real Barrett WAM in a physical table tennis setup. The system learns a set of basic hitting movements from demonstration and, subsequently, generalizes these demonstrations to a wider range of situations.

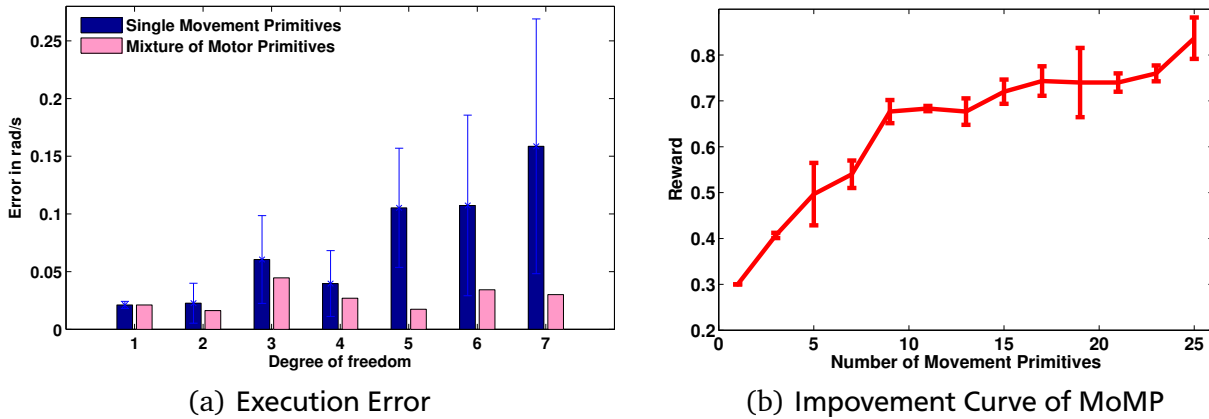
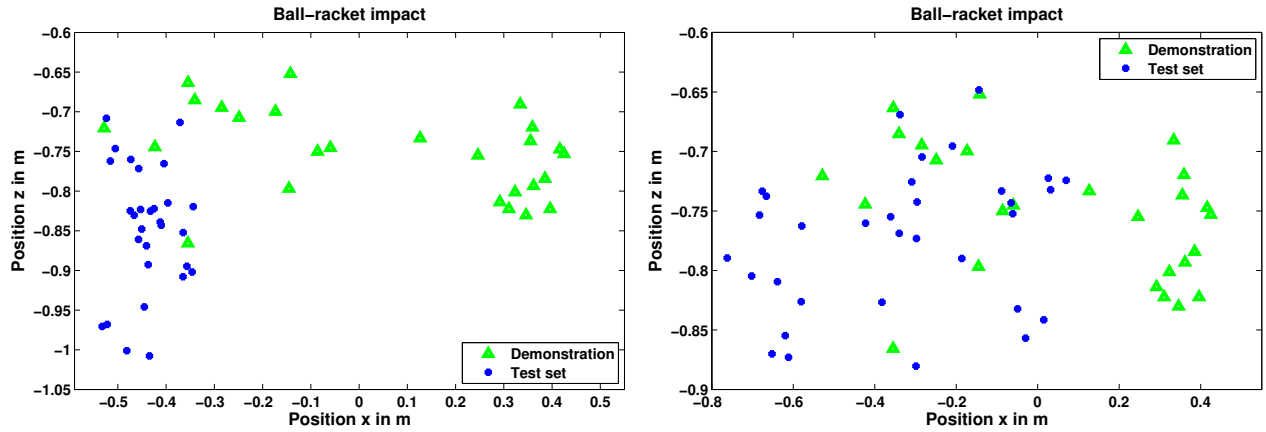


Figure 3.7: (a) Velocity error at the movement goal of the single movement primitives and the MoMP. The execution error of each movement primitive in the library was determined. The first bar of each DoF shows the mean error of the individual movement primitives and the corresponding standard deviations. The second bar of each DoF shows the mean error of the motor policies generated by the MoMP. (b) Improvement of the MoMP system with a growing number of movement primitives in the library.

Evaluation in Simulation

In order to evaluate our setup, as described in Section 3.2, in a variety of different situations under near-perfect conditions, we first evaluate it in simulation. The parameters of the hitting movement depend on the interception point of the racket and the ball and vary in their overall shape. To generate a movement that is able to cope with the varying conditions, we use our MoMP framework with the estimated hitting point and velocities as augmented state. The movement library was built using 300 movement templates sampled from successful strokes of the analytical robot table tennis player described in Chapter 2. For the simulation of the ball, we neglected air drag in this evaluation. The system received the true position and velocity of the ball which reduces the sources of potential errors. Thus, the success rate of returning the ball back to the desired goal on the opponent’s court reflects the performance of the algorithm more accurately. We collected arm, racket and ball trajectories and extracted the duration of the submovements and the Cartesian ball positions and velocities at the hitting point. The parameters w for all movement primitives were learned offline by imitation learning as described in Algorithm 4. All DoFs were modeled independently in the transformed system but are synchronized such that they all start at the same time, have the same duration and are driven by the same canonical system. The Cartesian position and velocity of the expected hitting point were used as meta-parameters of the augmented state. The balls were served equally distributed on an area of 1.1 m x 0.3 m.

First, we evaluated the single mixture components (i.e., the movement primitives learned by imitation learning) independently. Testing randomly selected movement primitives individually, we observed a success rate of 23 % to 89 %, where success was defined as the ability to return the ball to the opponents court. The combination of these components in an untrained mixture of movement primitives algorithm resulted into a player which achieved a success rate of 67 %. Learning of the weight parameters γ_i over 1000 trials improved the performance to a 94 % success rate. Analyzing the weight parameters for all movement primitives allowed the learning system to reduce the size of the library as all primitives π_i where γ_i converged to zero were effectively removed. We observed that about 27 % of the primitives were removed and that these primitives had an average performance of only a 30 % success rate.



(a) Fixed ball launcher.

(b) Oscillating ball launcher.

Figure 3.8: The locations of ball-racket impacts during demonstration and evaluation on the real robot. (a) The ball-racket impacts during the evaluation with the fixed ball launcher. (b) The ball-racket impacts of the evaluation of the oscillating ball launcher.

Real Robot Table Tennis

We evaluated our setup on a Barrett WAM arm. Kinesthetic teach-in was used to record 25 striking motions which all started at the same initial position (see Figure 3.6). Gravity compensation of the robot was enabled, but no additional movement commands were sent to the robot. As a result, the robot could be moved freely by the teacher. To ensure the safety of the teacher, reduced limits for velocity and acceleration were enforced. A second person manned the emergency shut-off. For the demonstrations, the ball was served by a ball launcher covering the forehand area of the robot. The recorded arm movements were divided into submovements according to the following events: contact with the ball, zero velocity and change in movement direction. As a result, each demonstration could be divided into four submovements: preparing the hitting by moving the racket backwards, hitting the ball in a circular forward movement, follow throw until the velocity goes to zero and moving back to the default position. We created the movement library of DMPs for each movement recorded from the human teacher using imitation learning. As augmented state, we used the estimated hitting position and velocity.

We evaluated the performance of the created movement library intensively in simulation first. Here, we used the exponential of the negative distance between the desired joint velocity and the actual joint velocity at the interception point of the ball and the racket. The balls were served to an area of 1.1 m x 0.3 m. The system was trained using 100 trials and evaluated over 200 additional trials. First, we evaluated the performance of the individual movement primitives in our library. Analyzing the ability of each DMP to generalize to new situations (i.e., the ability to perform equally well in new situations), we found that not all DMPs were able to adapt to the changing velocity profiles as required. In the last three DoFs towards the racket, we found mean errors ranging from 0.02 rad/s to 0.4 rad/s from the desired joint velocities. Using our MoMP system reduced the mean errors to 0.04 rad/s (see Figure 3.7(a)). Using just the average performance of the last three DoF as a reward signal, the system was able to reduce the average error to 0.02 rad/s where 8 movement primitives were mixed in average. The average performance of the system using MoMP without updating the gating network was 0.08 rad/s. Updating the gating network and just choosing the movement primitive with the best expected performance, we had an error of 0.05 rad/s and mixing the movement primitives yielded an error of 0.02 rad/s. An overview of the improvement of the system with increasing number of movement primitives used is shown in Figure 3.7(b).

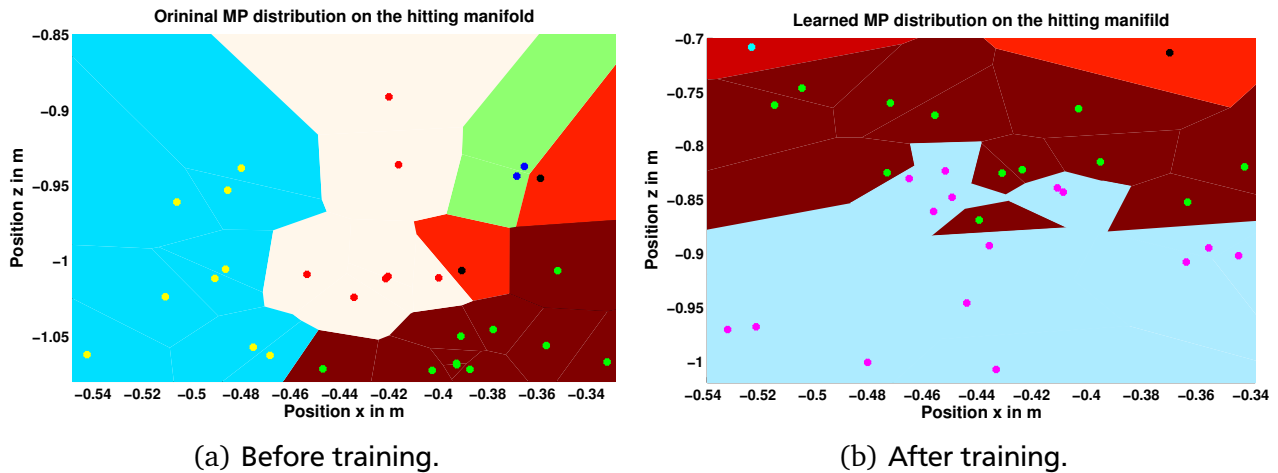


Figure 3.9: Distribution on the hitting manifold (i.e., the part of the state-space that defines the ball-racket contacts) of the most dominant movement primitives before (a) and after (b) training. Each colored region corresponds to one movement primitive in the movement library. Each point corresponds to the point of impact during evaluation. The usage of the movement primitives changed after training the system with a ball launcher. While two movement primitives were relocated, three were avoided and replaced by two better suited primitives. Please note that we have displayed only a small excerpt of the served forehand area here in order to visualize the re-organization of a few primitives.

Performing the movement on a real system, we used the exponential of the negative distance between the desired goal on the opponent’s court and the actual bouncing point as reward signal. If the ball missed the table, the distance would be set to 5 m and the reward was set close to zero. We evaluated the algorithm in three experiments. In the first two experiments we used the ball launcher to serve the ball to the robot to ensure similar conditions for both initial and final test phases. In the first experiment, the ball launcher was in an oscillating mode. The area served by the ball launcher measured 0.7 m x 0.4 m. The striking movement would be considered to be successful, if the ball was returned to the opponent’s court. The setup was tested with 30 trials. Before training, the performance of the system was 69%. The distance between the bouncing points on the opponent’s court and the desired target on the table had an average distance of 1.9 m to the target on the table. After training the system in 150 trials, the system returned 97% of the balls successfully to the opponent’s court. The mean distance between the bouncing point and the target on the table was 0.6 m. Evaluating only the successful trials, the mean distance to the target was 0.46 m. The point of impact of the ball and the racket during demonstration and evaluation is shown in Figure 3.8.

In the second experiment, we chose to fix the ball launcher in a position where the system was not able to return the ball using the initial policy generated by MoMP. The area served by the ball launcher was 0.25 m x 0.25 m. Initially, the system was not able to return any of these balls. After training for 60 trials, the system was able to return 79% of the balls successfully to the opponent’s court. The mean distance between the bouncing point of the returned balls and the desired target on the table was 0.31 m. During the training, the usage of the applied movement primitives changed drastically. While some of the movement primitives were relocated, other movement primitives were avoided completely and replaced by others used instead (see Figure 3.9).

In a third experiment, a human played against the robot. The human served balls on an area of 0.8 m x 0.6 m. The robot hit back up to 9 balls in a row in a match against the human opponent. Initially the robot was able to return 74.4 % of the balls. After playing one hour against the human, the robot was able to return 88 % of the balls successfully, i.e., to the opponent’s court.

	$\min \dot{x}$	$\max \dot{x}$	$\min \dot{y}$	$\max \dot{y}$	$\min \dot{z}$	$\max \dot{z}$	v_{\max}
Simulation	-0.92 m/s	0.34 m/s	1.88 m/s	4.03 m/s	-2.11 m/s	1.23 m/s	4.64 m/s
Experiment 1	-0.66 m/s	0.00 m/s	2.8 m/s	3.27 m/s	-2.75 m/s	-1.03 m/s	4.33 m/s
Experiment 2	-0.49 m/s	-0.22 m/s	2.59 m/s	2.99 m/s	-3.04 m/s	-1.55 m/s	4.29 m/s
Experiment 3	-0.92 m/s	0.57 m/s	2.03 m/s	4.04 m/s	-2.02 m/s	0.29 m/s	4.61 m/s

Table 3.1: Variance of the velocity of the ball before impact in the different experiments of the evaluation. The maximal velocity v_{\max} of the ball before impact is defined by $v_{\max} = \sqrt{|\dot{x}|^2 + |\dot{y}|^2 + |\dot{z}|^2}$. All values are given in meters per second [m/s].

Setup	Goal Perturbation				Resulting Error			
	δg_f [rad]		$\delta \dot{g}_f$ [rad/s]		e_{g_f} [rad]		$e_{\dot{g}_f}$ [rad/s]	
	average	max	average	max	mean	std	mean	std
real robot evaluation	0.44	1.93	1.54	3.01	0.0017	0.0013	0.0431	0.0375
	min	max	min	max				
simulation analysis (5 configurations)	-2.00	2.00	0.00	0.00	0.0054	0.0056	0.1695	0.2323
	-1.00	1.00	0.00	0.00	0.0035	0.0030	0.0929	0.1083
	0.00	0.00	-2.00	2.00	0.0031	0.0024	0.0597	0.0701
	0.00	0.00	-1.00	1.00	0.0028	0.0024	0.0566	0.0706
	-1.00	1.00	-2.00	2.00	0.0036	0.0031	0.0950	0.0944

Table 3.2: Analysis of the accuracy of the modified hitting primitives after both imitation and reinforcement learning. The right part of the table shows the changes in the desired final position and velocity compared to the values in the demonstration. On the left side of the table the corresponding mean errors (averaged over all DoFs) in the desired position and velocity are displayed at time T_f . The simulation analysis consisted of several configurations in which the motor policies yield through demonstrations were perturbed. For each configuration, either the final position, final velocity or both were modified using uniform value shifts between [-2,2] and [-1,1]. Please note that these position shifts exceed those which were observed on the real system.

The information about the velocity of the ball before impact, as well as on the variance of the incoming ball’s direction are summarized for all experiments in Table 3.1. The mean error of the desired final position and velocity at time T_f during evaluation on the real robot was 0.0017 rad and 0.0431 rad/s, respectively. A systematic study of the errors in simulation showed that changes in the position affects the final error more than changes in the velocity. The results from this analysis are depicted in Table 3.2.

3.4 Discussion and Conclusion of Chapter 3

In this chapter, we presented a new framework that enables a robot to learn basic cooperative table tennis from demonstration and interaction with a human player. To achieve this goal, we created an initial movement library from kinesthetic teach-in and imitation learning. The movements stored in the movement library can be selected and generalized using the proposed mixture of movement primitives algorithm. As a result, we obtain a task policy that is composed of several movement primitives weighted by their ability to generate successful movements in the given task context. These weights are computed by a gating network and can be updated autonomously.

The setup was evaluated successfully in a simulated and real table tennis environment. Our experiments showed that both (i) selecting movement primitives from a library based on the current task

context instead of using only a single demonstration and (ii) the adaptation of the selection process during a table tennis game improved the performance of the table tennis player. As a result, the system was able to return balls served to the forehand area of the robot successfully to the opponent's court. We managed to perform and improve the movements involved in table tennis with a reasonable amount of training data for real robot evaluation.

The presented method of generating motor policies from a library of demonstrated movements is not limited to the demonstrated show-case of robot table tennis. In particular, the approach is suited for many tasks where a set of movements has to be adapted to new, different but similar situations as commonly found in striking sports. Additionally, it could potentially be adapted to other goal-oriented movements (e.g., simple manipulation tasks, foot-step generation for walking over rough terrain, etc.) where our approach would be a good starting point.

Our approach generalizes well if the situations are within the convex combination of demonstrations. For extrapolating, more additional reinforcement learning will be needed as exploration is required. In our approach, only local reinforcement learning is employed for self improvement. As a result, the solution can be extrapolated to new similar situations and movements but not totally novel ones. Discovering a movement that differs from all existing ones (as e.g., Dick Fosbury did in high-jumping with the Fosbury flop [Wikipedia, 2012]) is clearly a limit of this data-driven framework.

4 Learning Strategies in Table Tennis using Inverse Reinforcement Learning

Performing a motor task does not only depend on the perfect execution of robust movements. Even after acquiring the necessary motor skills as shown in Chapter 2 and 3, a higher-level strategy is required to choose where and how to return the ball to the opponent's court in order to win the game. Defining a successful strategy manually is difficult as such a strategy depends strongly on the abilities of the player and the opponent. Therefore, it would be highly desirable to infer strategic information from observing humans playing table tennis. This strategic information can be used by the robot as a foundation for finding an optimal policy. In this chapter, we want to come one step closer towards this goal. Thus, we suggest a computational model for representing and inferring strategies, based on a Markov Decision Problem (MDP), where the reward function models the goal of the task as well as all strategic information. We demonstrate how this reward function can be inferred from demonstrations of real table tennis games using model-free inverse reinforcement learning. Therefore, we collect data from table tennis players with different skill levels and styles. We show that the resulting reward functions are able to distinguish between the different playing skills and styles on real human table tennis data.

4.1 Prologue

Understanding the complex interplay between learning, decision making and motion generation is crucial both for creating versatile, intelligent robot systems as well as for understanding human motor control. In complex competitive and cooperative motor tasks, mastering the task is not merely a matter of perfect execution of a specific movement pattern. In table tennis, a player usually cannot win the game by always returning the ball safely to the same position. Instead, players need a good strategy that defines where and how to return the ball to the opponent's court. An action should always be chosen to have a high probability of successfully returning the ball as well as to make the task of the opponent harder, i.e., it should improve the chance of winning the game. In this chapter of the thesis, we want to model and understand the decision processes underlying this behavior. To accomplish this goal, we investigate how basic strategic information can be inferred from the observation of a human table tennis game.

In racket science, researcher identified so called *winning patterns* in tennis video sequences in order to help trainers analyze their game [Wang et al., 2004, Wang and Parameswaran, 2005, Vis et al., 2010]. Here, specific repetitive movement patterns of both the players and the ball were turned into tactical templates. In table tennis, Hohmann et al. [2004] determined the transition probabilities of different stroke positions, directions and types individually. Such transition probabilities allow identifying the components that can be used most efficiently. Similar to Diaz et al. [2013], where the authors showed that memory-based information is used for predictive eye movements in racquetball, Seve et al. [2004] showed that such memory-based information is also used for strategies in table tennis. Seve et al. [2004] conducted interviews with professional table tennis players to analyze the actions that they chose in a match. They concluded that the players selected their actions not only based on the current situation, but also based on the knowledge of sequences that have proven to be effective in the past in similar situations. Rather than identifying the frequencies and effectiveness of specific movement patterns, we want to model the decision process for choosing actions by players in a match of table tennis from a computational point of view. Thus, we are not only able to use the learned model for artificial systems, but also yield a better insight into the reasons for choosing a given action in a specific state. Therefore, we only consider basic features available to the player such as the bouncing point and, the direction and velocity of the ball.

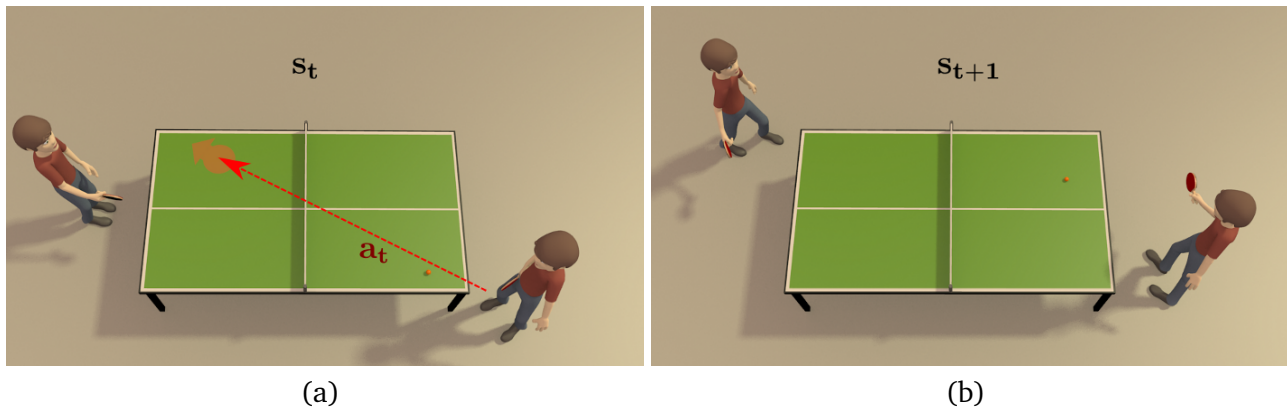


Figure 4.1: Considered scenario: A table tennis player (agent) plays a game of table tennis. At time point t , he has to decide how to return the approaching ball to the opponents court such that the chance of winning the point will increase. Returning the ball to a specific goal on the opponent's court (with a specific orientation and velocity) corresponds to an action a_t executed by the agent. The player chooses this action based on his current state s_t (Figure a). Due to this action, the system will transfer to the state s_{t+1} defining a new situation for the player (Figure b).

A common way to model decision processes in artificial systems is to use a Markov Decision Problem (MDP, Puterman [1994]). Here, an agent interacts with a dynamic environment. It chooses and executes an action that will change the state of the player and its environment (see Figure 4.1). The agent can observe this state change and may receive a reward for its action. A strategy defines the general plan of choosing actions in specific states in order to achieve a goal. A strategy in the MDP framework is usually called a *policy* and is denoted by π . Given a MDP model, one can find an optimal policy using optimal control techniques [Sutton and Barto, 1998, Powell, 2011]. The goal is to find a policy that maximizes the expected reward. The reward thus, encodes the goal of the task. Having encountered states, actions, rewards and policies in Chapter 3, we will address these concepts in a more detailed way in this chapter.

While it is possible to learn a policy directly from demonstrations using supervised learning [Schaal, 1999, Argall et al., 2009], such behavioral cloning approaches usually have limited generalization abilities since they are restricted to the demonstrated scenarios. As they do not consider the underlying dynamics, they cannot be applied in a task with altered or constantly changing dynamics. In table tennis, the dynamics of the environment changes as the opponent changes. The player may also encounter new states, and hence need to learn new strategic elements while his experience increases with training. Therefore, blindly following the strategy of an observed expert will not lead to a successful strategy. Rather than mimicking a specific policy, we want to learn which information the expert uses in order to win the game. This knowledge can sometimes be succinctly captured in the reward function that defines the reward the agent will receive in a specific situation when executing an action. Based on this reward function, we can compute and adapt the policy when new state-action pairs are encountered. As a result, the transfer of strategic information from humans to artificial systems such as robots will be possible. When transferring human policies to robots we have to account for differences in their respective dynamics. The reward function is independent from the dynamics. It can therefore be used to obtain a policy with reinforcement learning.

Given an exact model, simple reward functions that only specify an immediate positive reward for winning, a negative one for losing a rally, and zero reward for non-terminal actions may be sufficient. However, such simplified rewards will cause slow convergence rates for behavior generation as the system will need to pass through several state-action pairs before receiving a reward. Clearly, such

a simplified reward cannot suffice for explaining human playing behavior in table tennis. Instead of pre-defining the reward function, we seek to identify it from human game-play. Such an approach will also allow us to reveal individual preferences of table tennis players. The process of determining the reward function from an expert demonstration is referred to as *Inverse Reinforcement Learning* (IRL) or inverse optimal control [Boyd et al., 1994, Ng and Russel, 2000]. IRL has been applied to many problems such as helicopter control [Abbeel et al., 2010], parking lot navigation [Abbeel et al., 2008], navigating a quadruped robot across different terrains [Kolter and Ng, 2011], routing preferences of drivers [Ziebart et al., 2008], user simulation in spoken dialog management systems [Chandramohan et al., 2011] and, modeling goal directed trajectories of pedestrians [Ziebart et al., 2009]. In most of these approaches, the underlying dynamics of the system is assumed to be known. However, the dynamics of human behavior is usually difficult to model. We avoid modeling these complex dynamics by learning the strategies directly from human demonstration. Thus, the dynamics model underlying the task is implicitly encoded in the observed data. To collect demonstrations, we asked skilled and naive table tennis players to compete in several matches. We recorded the ball trajectories as well as the Cartesian position and orientation of the upper body joints for all players with a VICON motion capture system.

During the course of this chapter, we will answer the following questions: (1) Can we infer a reward function that captures expert-specific information using model-free inverse reinforcement learning? (2) Using this reward function, can we distinguish players with different playing styles and skill levels? (3) Which parts of the sensory information are the key elements for selecting the movement parameters?

In the remainder of this chapter, we will proceed as follows. In Section 4.2, we present the theoretical background for modeling decision processes, including MDPs and IRL algorithms. We present the experimental setup and evaluations in Section 4.3. In Section 4.4, we summarize our approach and the results.

4.2 Modeling Human Strategies

As discussed in the prologue, we use model-free Inverse Reinforcement Learning (IRL) to learn the strategic components. Here, we will first introduce the notation and basic elements necessary for the table tennis model. Subsequently, we will discuss different model-free IRL approaches and show how the states, actions and reward features in the table tennis task can be represented.

4.2.1 Preliminaries

To employ IRL, the problem at hand needs to be modeled as a Markov Decision Problem (MDP). Formally, a MDP is a tuple $(S, A, \mathcal{T}, r, d_0, \gamma)$, where S is the state space, A is the action space, and \mathcal{T} is a transition function

$$\mathcal{T}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) = Pr(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t),$$

with states $\mathbf{s}_t, \mathbf{s}_{t+1} \in S$ and actions $\mathbf{a}_t \in A$. The function $r(\mathbf{s}, \mathbf{a})$ defines the reward for executing action \mathbf{a} in state \mathbf{s} , the initial state distribution $d_0(\mathbf{s})$ models the start conditions, and the discount factor $\gamma \in [0, 1)$ determines the effective planning horizon.

A deterministic policy π is a mapping: $S \mapsto A$ and defines which action is chosen in a state $\mathbf{s} \in S$. A stochastic policy is a probability distribution over actions in a given state \mathbf{s} and is defined as $\pi(\mathbf{a} | \mathbf{s}) = Pr(\mathbf{a} | \mathbf{s})$. The performance of a policy is measured with the so-called *value function* $V^\pi(\mathbf{s})$. The value function of a policy π evaluated at state \mathbf{s} is given by

$$V^\pi(\mathbf{s}) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \mid \pi, \mathcal{T}, \mathbf{s}_0 = \mathbf{s} \right],$$

and corresponds to the expected reward following policy π starting from state \mathbf{s} . The optimal value function is defined by $V^*(\mathbf{s}) = \max_{\pi} V^{\pi}(\mathbf{s}) \forall \mathbf{s} \in S$. The goal of an agent in a MDP is to find the optimal policy π^* , i.e., a policy that maximizes the value for every $\mathbf{s} \in S$.

We assume that the reward function r is given by a linear combination of m feature functions f_i with weights w_i . The reward function is therefore defined by

$$r(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^m w_i f_i(\mathbf{s}, \mathbf{a}) = \mathbf{w}^T \mathbf{f}(\mathbf{s}, \mathbf{a}),$$

where $\mathbf{w} \in \mathbb{R}^m$ and $\mathbf{f}(\mathbf{s}, \mathbf{a}) \in \mathbb{R}^m$. The features f_i are fixed, known, bounded basis functions mapping from $S \times A$ into \mathbb{R} . For a given trajectory $\tau = s_1 a_1, \dots, s_T a_T$ the feature counts are given by $f_i^{\tau} = \sum_{t=1}^H \gamma^t f_i(\mathbf{s}_t, \mathbf{a}_t)$. Similarly to the value function, we can define the feature count f_i^{π} under policy π by

$$f_i^{\pi}(\mathbf{s}) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t f_i(s_t, a_t) \middle| \pi, \mathcal{T}, \mathbf{s}_0 = \mathbf{s} \right]$$

as the expected features observed when following policy π . Since the reward function can be represented as a linear combination of features f_i , the expected return of policy π can be written as

$$V_{\mathbf{w}}^{\pi}(\mathbf{s}) = \sum_{i=1}^m w_i f_i^{\pi}(\mathbf{s}) = \mathbf{w}^T \mathbf{f}^{\pi}(\mathbf{s}),$$

where $\mathbf{f}^{\pi} \in \mathbb{R}^m$ is a vector containing the single feature counts $f_i^{\pi}(\mathbf{s})$ as entries.

4.2.2 Learning the Reward Function

The reward function is a crucial part of the MDP as it defines the goal of the task and shapes the policy optimization process. Usually, it is assumed that the reward function is given. However, it is hard to specify the reward function for solving a complex task beforehand and the learned behavior is sensitive to the provided reward function. This problem is especially evident when the task requires modeling the dynamics of human actions. The problem of designing the right reward function led to the development of IRL methods. Given the actions of an agent that is assumed to behave in an optimal manner, the available sensory information about the environment and, if possible, a model of the environment, the goal of IRL is to determine a reward function that can (mostly) justify the demonstrated behavior.

The IRL problem was originally formulated within the MDP framework by Ng and Russel [2000]. Many researches provided further refinements in order to improve the original algorithms suggested by Ng and Russel [2000] and Abbeel and Ng [2004]. For example, Ratliff et al. [2006] suggested a maximum margin planning approach. Ziebart et al. [2008] suggested an algorithm where the principle of maximum entropy was exploited. Ramachandran and Amir [2007] modeled the uncertainties involved as probabilities where the demonstrations are treated as evidence of the unknown reward function. A recent review of IRL algorithms can be found in [Zhifei and Joo, 2012].

However, most IRL approaches rely on a given model of the environment \mathcal{T} or assume that it can be accurately learned from the demonstrations. The reward function is found by first computing a policy that optimizes a reward function for an initial weight vector \mathbf{w} . Subsequently, the expected feature count of the new policy \mathbf{f}^{π} can be computed. Based on this feature count, a new weight vector that separates the values of the expert feature \mathbf{f}^{π_E} and the features of the current policy \mathbf{f}^{π} can be computed. These steps are repeated until the weight vector converges. This general algorithm is displayed in Algorithm 5.

In this chapter of the thesis, we want to estimate the underlying reward function for playing table tennis based on demonstrations *without having to model the correct dynamics model*. Only few model-free

Algorithm 5 General IRL Algorithm

Input: $D^E = \{\tau\}_{p=1}^P$ expert demonstrations
Initialize: reward feature weights \mathbf{w}^0 , $j = 1$
expert feature counts $\mathbf{f}^{\pi_E} = \frac{1}{P} \sum_{\tau \in D^E} \mathbf{f}^\tau$
repeat
 Optimize π_j based on \mathbf{w}^{j-1}
 Estimate \mathbf{f}^{π_j}
 Update \mathbf{w}^j such that $\mathbf{w}^j f^{\pi_j} < \mathbf{w}^j \mathbf{f}^{\pi_E}$
 $j \leftarrow j + 1$
until $\|\mathbf{w}^j - \mathbf{w}^{j-1}\|_2 < \varepsilon$

IRL methods have been suggested: Boularias et al. [2011] derived a relative entropy approach which was evaluated on a ball-in-a-cup scenario. Mori et al. [2011] used Least Squares Policy Iteration and Least Squares Temporal Difference learning and applied their algorithm on human impedance control.

To avoid modeling the dynamics in the table tennis task, we collect demonstrations from an expert as well as demonstrations from less skilled players in order to find the reward function of the underlying behavior. This approach stands in contrast to previous methods, where a model of the dynamics is used to iteratively generate optimal trajectories under different reward functions until the generated trajectories match the ones provided by the expert. We use both expert and non-expert data to compute the weight vector \mathbf{w}^* that maximizes the differences between the non-expert and the expert reward values. To compute the reward weights, we compared three different methods, where the results can be found in Section 4.3.2. The first two evaluated methods are based on the max-margin method of Abbeel and Ng [2004], while the third algorithm is the model-free relative entropy approach of Boularias et al. [2011]. In the following sections, we assume that we are given a set of expert demonstrations $D^E = \{\tau_p\}_{p=1}^P$, where $\tau_p = \mathbf{s}_1^p \mathbf{a}_1^p, \dots, \mathbf{s}_{T_p}^p \mathbf{a}_{T_p}^p$ corresponds to one rally (i.e., state-action trajectory), as well as a set of sub-optimal demonstrations $D^N = \{\tau_l\}_{l=1}^L$. Here, T_p is the number of volleys (i.e., state-action pairs) in the observed rally τ_p .

Model Free Max-Margin for Game Values

The max-margin method of Abbeel and Ng [2004] aims at finding a policy π that has an expected return close to that of the expert, i.e., $\max_{\mathbf{w}} |V_{\mathbf{w}}^{\pi}(\mathbf{s}) - V_{\mathbf{w}}^{\pi_E}(\mathbf{s})| \leq \epsilon$, where $\|\mathbf{w}\|_2 \leq 1$. As the value is a linear function of the reward, it suffices to find an optimal policy π that has feature counts close to the ones of the expert's trajectories, i.e., $\|\mathbf{f}^{\pi} - \mathbf{f}^{\pi_E}\|_2 \leq \epsilon$. The policy π needs to be chosen from the set of previously recorded sub-optimal policies due to the lack of a model for generating policies. We use the projection algorithm of Abbeel and Ng [2004] to solve the following optimization problem

$$\max_{\xi, \mathbf{w}} \xi \quad s.t. \quad \mathbf{w}^T \mathbf{f}^{\pi_E} \geq \mathbf{w}^T \mathbf{f}^{\pi_j} + \xi, \quad \|\mathbf{w}\| \leq 2,$$

where ξ is the difference of the value of the expert and the value of the non-expert, and π_j are the policies of non-expert players. $f_i^{\pi_j}$ therefore corresponds to the average feature count for all rallies demonstrated by a player in one game. The corresponding algorithm is displayed in Algorithm 6. In the following, we will refer to this algorithm as MMG (Maximum Margin for Game values).

Model Free Maximum Margin of States Values

Using the max-margin method of Abbeel and Ng [2004] in a model-free setup as described above has one drawback. We assume that the initial state of the rally largely defines all following state-actions

Algorithm 6 Maximum Margin for Game Values

Input: $D^E = \{\tau\}_{p=1}^P$ expert demonstrations
 $D^N = \{\tau\}_{l=1}^L$ sub-optimal demonstrations
Initialize: $\mathbf{f}^{\pi_E} = \frac{1}{P} \sum_{\tau \in D^E} \mathbf{f}^\tau$
 $\mathbf{f}^{\pi_i} = \frac{1}{L} \sum_{\tau \in D^{N_i}} \mathbf{f}^\tau$ with $D^{N_i} \subset D^N$
 $\mathbf{w}^0 = \mathbf{0}, j = 1$

repeat

$$i = \min_i \mathbf{w}^{j-1} (\mathbf{f}^{\pi_E} - \mathbf{f}^{\pi_i})$$

$$\mathbf{f}^{j-1} = \mathbf{f}^{\pi_i}$$

Compute $\tilde{\mathbf{f}}^{j-1}$, the projection of \mathbf{f}^{π_E} on $(\tilde{\mathbf{f}}^{j-2}, \mathbf{f}^{j-1})$

$$\mathbf{w}^j = \mathbf{f}^{\pi_E} - \tilde{\mathbf{f}}^{j-1}$$

$$\Delta f = \|\mathbf{f}^{\pi_E} - \tilde{\mathbf{f}}^{j-1}\|_2$$

$$j \leftarrow j + 1$$

until $\Delta f < \varepsilon$

pairs. However, in table tennis, it is unlikely that any player plans the strokes for more than only a few steps ahead. Computing the value function based on only a few state-action pairs after the initial serve would cause the agent to lose important information that led to winning or losing the rally. To avoid this information loss, we need to compare the values of the expert in every state in the recorded trajectories to the ones of the non-experts in the same state. As the states are continuous, it is unlikely that exactly the same state is encountered in both the expert and sub-optimal trajectories. Nevertheless, we can find the weight vector \mathbf{w} by solving the quadratic optimization problem

$$\max_{\mathbf{w}} \sum_{p=1}^P \sum_{t=0}^{T_p} (V_{\mathbf{w}}^{\pi_E}(\mathbf{s}_t^p) - \hat{V}_{\mathbf{w}}^{\pi_N}(\mathbf{s}_t^p)) - \lambda \|\mathbf{w}\|_2, \quad (4.1)$$

where $\hat{V}_{\mathbf{w}}^{\pi_N}(\mathbf{s}_t^p)$ is an estimated value of the non-expert players in the current state \mathbf{s}_t^p of the expert. Estimating the value \hat{V}^{π_N} in a given state \mathbf{s} is a regression problem that we propose to solve by using the k -nearest neighbors method,

$$\hat{V}_{\mathbf{w}}^{\pi_N}(\mathbf{s}) = \frac{1}{k} \sum_{\mathbf{s}' \in \mathcal{N}_k(\mathbf{s})} V_{\mathbf{w}}^{\pi_N}(\mathbf{s}'),$$

where $\mathcal{N}_k(\mathbf{s})$ is the set of k -nearest neighbors of \mathbf{s} among all the states that have been observed in the sub-optimal trajectories. We use a Gaussian kernel $K(\mathbf{s}, \mathbf{s}') = \exp(-(\mathbf{s} - \mathbf{s}')^T \Sigma^{-1} (\mathbf{s} - \mathbf{s}')^T)$ to define a similarity measure between states. The diagonal matrix Σ contains weight parameters. Note that one can also use other non-parametric methods, such as kernel regression.

The value functions V^{π_E} and V^{π_N} of the expert's policy π_E and sub-optimal policies π_N are computed as

$$V_{\mathbf{w}}^{\pi}(\mathbf{s}_t^p) = \frac{1}{H_t^p - t + 1} \sum_{i=t}^{H_t^p} \mathbf{w}^T \mathbf{f}^{\pi}(\mathbf{s}_i^p, \mathbf{a}_i^p),$$

where $H_t^p = \min\{t + H - 1, T_p\}$ and H is the planning horizon, i.e., the number of steps we look into the future. The corresponding algorithm is displayed in Algorithm 7. In the following, we will refer to this algorithm as MMS (Maximum Margin of State values).

Algorithm 7 Maximum Margin of States

Input: $D^E = \{\tau_p\}_{p=1}^P$ expert demonstrations
 $D^N = \{\tau_l\}_{l=1}^L$ sub-optimal demonstrations
Initialize: $n = 1$
for all $p \in D^E$ **do**
 for all $s_t^p \in \tau_p$ **do**
 $[\mathbf{F}^{\pi_E}]_n = \sum_{i=t}^{H_t^p} \mathbf{f}(s_i^p, \mathbf{a}_i^p)$
 Compute k -nearest neighbors $\mathcal{N}_k(s_t^p)$
 $[\mathbf{F}^{\pi_N}]_n = \frac{1}{k} \sum_{s_i^l \in \mathcal{N}_k(s_t^p)} \sum_{i=t}^{H_t^l} \mathbf{f}(s_i^l, \mathbf{a}_i^l)$
 $n \leftarrow n + 1$
 end for
end for
 $\mathbf{w} = \max_{\mathbf{w}} \mathbf{w}(\mathbf{F}^{\pi_E} - \mathbf{F}^{\pi_N}) - \lambda \|\mathbf{w}\|_2$

Relative Entropy Method

The relative entropy IRL method [Boularias et al., 2011] finds a distribution \mathcal{P} over trajectories that minimizes the KL-divergence to a reference distribution Q , while ensuring that the feature counts under \mathcal{P} are similar to the feature counts in the expert trajectories. The reference distribution Q encodes prior preferences and constraints of the learned behavior, which makes this method well-suited for transferring the expert’s policy to a robot. The solution to this problem takes the following form

$$\mathcal{P}(\tau|\mathbf{w}) = \frac{1}{Z(\mathbf{w})} Q(\tau) \exp\left(\mathbf{w}^T f_i^\tau\right),$$

where $Z(\mathbf{w}) = \sum_{\tau} Q(\tau) \exp\left(\mathbf{w}^T f_i^\tau\right)$. The reward weight vector \mathbf{w} is found by solving the optimization problem

$$\max_{\mathbf{w}} \mathbf{w}^T \mathbf{f}^{\pi_E} - \ln Z(\mathbf{w}) - \lambda \|\mathbf{w}\|_1. \quad (4.2)$$

The gradient of this objective function is calculated by re-using the expert and sub-optimal trajectories with importance sampling. For our experiments, we choose the reference distribution Q to be uniform, as we are mainly interested in extracting the most informative reward function and not in transferring the expert’s policy. The corresponding algorithm is displayed in Algorithm 8. In the following, we will refer to this algorithm as RE (Relative Entropy).

4.2.3 Computational Model for Representing Strategies in Table Tennis

In the previous sections, we have given a general description of how the decision processes in table tennis can be modeled as a MDP. We also showed several approaches for obtaining the reward function from the table tennis player’s demonstrations. As a next step, we need to specify the states, actions and reward features of the table tennis task.

States

The state of the system consist of all sensory information experienced by the agent. However, learning in such high-dimensional continuous state domains is likely to be intractable. Hence, we need to remove redundant or irrelevant information. Therefore, we assume that the player has to

Algorithm 8 Relative Entropy IRL Algorithm

Input: $D^E = \{\tau_p\}_{p=1}^P$ expert demonstration
 $D^N = \{\tau_l\}_{l=1}^L$ sub-optimal demonstration
Initialize: $\mathbf{f}^{\pi_E} = \frac{1}{P} \sum_{\tau \in D^E} \mathbf{f}^\tau$
 $\mathbf{w}^0 = \mathbf{0}, j = 1$

repeat

$$\text{Compute } \mathcal{P}(\tau | \mathbf{w}^{j-1}) = \frac{Q(\tau) \exp\left(\sum_{i=1}^m w_i^{j-1} f_i\right)}{\sum_{\tau \in D^N} Q(\tau) \exp\left(\sum_{i=1}^m w_i^{j-1} f_i\right)}$$

for all $\tau \in D^N$

for all features f_i **do**

$$\frac{\partial}{\partial w_i} g(\mathbf{w}) = f_i^{\pi_E} - \sum_{\tau \in D^N} \mathcal{P}(\tau | \mathbf{w}^{j-1}) f_i(\tau) - \beta_i \lambda_i$$

$$w_i^j = w_i^{j-1} + \frac{\partial}{\partial w_i} g(\mathbf{w})$$

end for

$$\Delta w = \|\mathbf{w}^{j-1} - \mathbf{w}^j\|_2$$

$j \leftarrow j + 1$

until $\Delta w < \varepsilon$

decide where and how to hit the ball when the hitting movement is initiated. Furthermore, we assume that the decision depends on the following information: the planar Cartesian position of the agent $\mathbf{d}_s = [d_{sx}, d_{sy}]$, the opponent's position $\mathbf{d}_o = [d_{ox}, d_{oy}]$ and velocity \mathbf{v}_o , the state of the rally $\mathbf{g} \in \{\text{player serve, opponent serve, not served}\}$ as well as the ball position $\mathbf{d}_b = [d_{bx}, d_{by}]$, velocity $|\mathbf{v}_b|$ and direction given by the bouncing angles α_{py} and α_{pz} (see Figure 4.2).

Thus, the state can be represented by the parameters $\mathbf{s}_i = [\mathbf{d}_b, |\mathbf{v}_b|, \alpha_{py}, \alpha_{pz}, \mathbf{d}_s, \mathbf{d}_o, \mathbf{g}]$. The variables α_{py} and α_{pz} are defined as the horizontal and vertical bouncing angles of the ball at the moment of impact on the player's side of the table, respectively. α_{pz} defines the bouncing angle in the xz -plane and therefore corresponds to how flat the ball was played. α_{py} defines the bouncing angle in the xy -plane (see Figure 4.3). Playing the ball diagonal to the backhand area of the opponent results in a smaller negative angle for α_{py} , while playing the ball diagonal to the forehand area results in an increased angle. Playing the ball straight corresponds to an angle of zero. Additionally, we define a set of terminal states $s_T \in \{W, L\}$. A rally will end when either the subject won the rally ($s_T = W$), or the subject lost the rally ($s_T = L$).

Actions

To perform a hitting movement, the system needs the following information: (i) where and when to hit the ball, (ii) the velocity of the racket and (iii) the orientation of the racket at impact. While the first may directly result from the current state of the system, the second and third points are determined by how the player decides to return the ball to the opponent's court. This decision includes the desired bouncing point \mathbf{p}_b of the ball on the opponent's court, the corresponding bouncing angles α_{oy} and α_{oz} , the overall velocity of the ball $\|\mathbf{v}_b\|$ and the spin of the ball. Since the different kinds of spin are hard to capture without an expert classifying the sampled data, we discard the spin and use only basic strategic elements. Therefore, an action can be defined as $\mathbf{a} = [\mathbf{p}_b, \|\mathbf{v}_b\|, \alpha_{oy}, \alpha_{oz}]$. We do not distinguish between serves and non-serves for the actions, as the first bounce of the serve will be fully described by the second bounce.

Reward Features

The reward features $f_i(\mathbf{s}, \mathbf{a})$ for each state-action pair are defined by the following attributes.

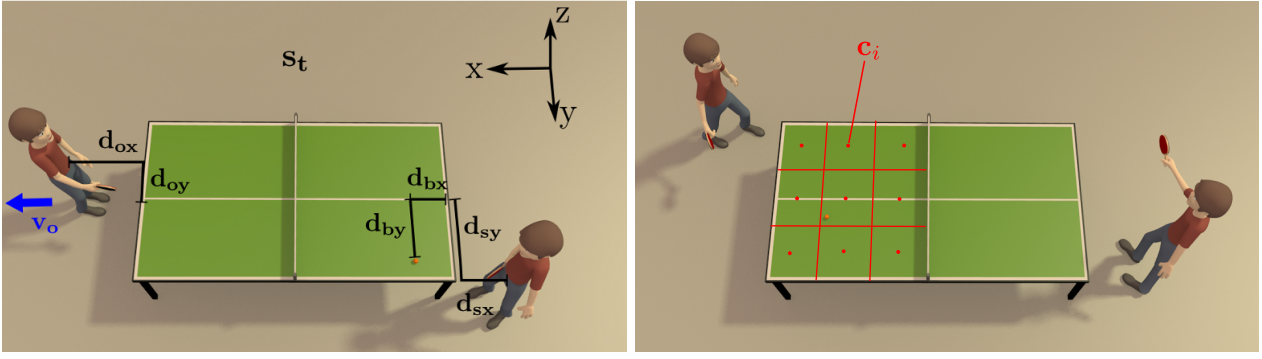


Figure 4.2: Figure(a): The state of the system is defined by the relative position of the agent (d_{sx}, d_{sy}) and the the relative position (d_{ox}, d_{oy}) and velocity (v_o) of the opponent towards the table, as well as the the position (d_{bx}, d_{by}) and velocity (v_b) of the ball when bouncing on the table. Figure(b): In order to compute the table preferences on the opponent's court the table was divided into nine cells. Each cell was assigned a center (red points) c_i .

Position on the table. We divided the table into nine cells. Each cell was specified by a center c_i (see Figure 4.2b). Nine features were considered

$$Pr(c_i | p_b) = \frac{\exp(-\varepsilon \|p_b - c_i\|_2^2)}{\sum_j \exp(-\varepsilon \|p_b - c_j\|_2^2)},$$

each defining the probability that the player was aiming for the specific cell. This computation is based on the euclidean distance between the desired bouncing point p_b of the ball (which is part of state s) and the cell center c_i , where ε is a shape parameter.

Distance to the edges of the table. We provided two features defining the proximity to the edge e_t , one for the x-direction $\delta_{tx} = \exp(-1.5|e_{tx} - p_{bx}|)$ and one for the y-direction $\delta_{ty} = \exp(-1.5|e_{ty} - p_{by}|)$.

Distance to the opponent. Two features defined the distance of the bouncing point of the ball on the opponent's court and the right hand of the opponent. One of the features is defined by the distance in x-direction $\delta_{ox} = |p_{ox} - p_{bx}|$, while the second is defined by the distance in y-direction $\delta_{oy} = |p_{oy} - p_{by}|$.

Elbow. One feature is the closeness of the ball to the elbow and, therefore, it measures if the ball was played to the elbow of the opponent e_o . It is defined by $\delta_{elbow} = \exp(-|e_{oy} - p_{by} + \tan(\alpha_y)(e_{ox} - p_{bx})|)$, where $\tan(\alpha_y)(e_{ox} - p_{bx})$ is an extrapolation of the ball position. This feature also provides a measurement of how close the ball bounces relative to the opponent.

Velocity of the ball. The velocity of the ball $\|v_b\|$ in meters per second was used as another feature.

Movement direction of the opponent. One feature was derived in order to define the velocity of the opponent and the ball in y-direction. It is defined by $v_o = 10(p_{oy} - p_{by})v_{oy}$.

Bouncing angles. We computed two bouncing angles α_{oz} and α_{oy} which define the direction of the ball when bouncing on the opponent's side of the court.

Smash. One of the features defined whether the ball was a smash. When the ball velocity was higher than 10 m/s, this feature was set to one, otherwise this feature was set to zero. The velocity of 10 m/s was defined empirically.

Winning and Loosing. One binary feature was used to assign a reward to the terminal states (i.e., winning and losing). For all non-terminal states, this feature was set to zero. For the terminal states, a value of one was assigned to the feature for $s_T = W$ and a value of -1 for $s_T = L$.

All features are scaled to lie in an interval of $[0, 1]$, except for the direction sensitive features α_{oy} and v_o which lie in an interval of $[-1, 1]$.

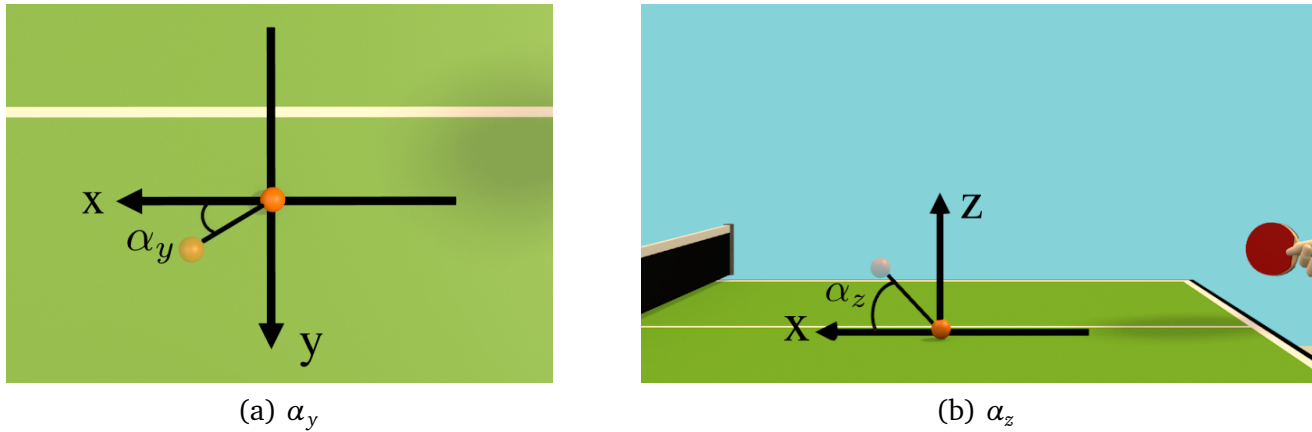


Figure 4.3: The bouncing angles α_y and α_z in the xy - and xz -surface define the orientation of the ball. While α_z corresponds to the horizontal bouncing angle, α_y corresponds to the direction of the ball and thereby defines if the ball is played cross to the left, cross to the right or straight.

4.3 Experiments and Evaluations

To validate the suitability of using IRL algorithms in order to extract basic strategic elements, we recorded table tennis players with various skill levels. The subjects played under three different conditions. This data was used to compute the reward feature weights and to validate the potential reward functions.

In the following, we will first describe the experiment and the data processing procedure. Subsequently, we will present the results.

4.3.1 Experimental Setup and Data Collection

The purpose of the experiment was to investigate basic strategic elements in table tennis (excluding all types of spin which are difficult to capture), using inverse reinforcement learning techniques. Therefore, a data set with expert demonstrations, and a data set with different sub-optimal policies were collected. In this study, there were both participants serving as subjects who rarely played table tennis, as well as subjects who played on a regular basis in a table tennis club.

Participants

Eight healthy right-handed subjects of all genders (seven males, one female) participated in this study. The mean age of the participants was 26.25 years (standard deviation (SD) 3.38 years). All subjects had normal or corrected-to-normal eye sight. All participants gave their consent prior to the experiment and completed a form about their playing skills according to which they were grouped in one of two classes: 1) naive players and 2) skilled players.

The group of naive players consisted of five subjects (four males and one female) with a mean age of 28.4 years (SD 1.14 years). The subjects were recruited from the Max Planck Campus in Tübingen and the University of Tübingen. All naive players fulfilled the following requirements: (i) never played in a table tennis club, (ii) did not train on a regular basis (weekly or daily) in the last five years, (iii) did not participate in table tennis tournaments and (iv) did not play any other racket sports on a regular basis. The group of skilled players consisted of three subjects (all male) with a mean age of 22.67 years (SD 2.08 years). The subjects were recruited from a local table tennis club and fulfilled the following requirements: (i) played for at least eight years in a table tennis club, (ii) trained on a weekly basis (at least twice a week) and (iii) participated regularly in table tennis competitions.

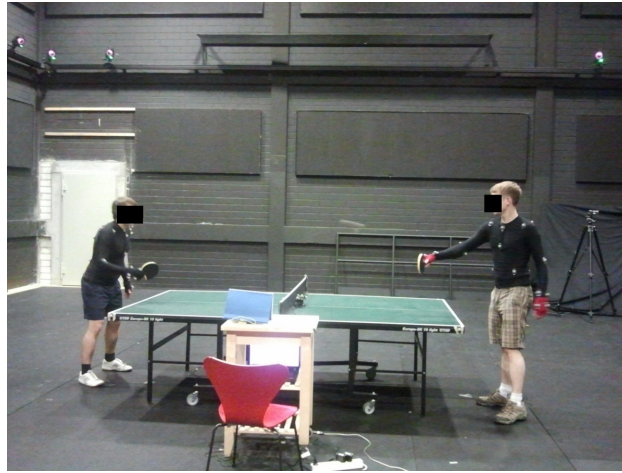


Figure 4.4: Experimental setup. A naive player (right side) plays against a skilled opponent (left side). The upper body of both players, as well as the ball are tracked by a motion capture system.

One of the skilled players were used as a permanent fixed *opponent* and, therefore, was not considered part of the subject set. Furthermore, only one of the skilled subjects was used for the expert demonstrations. All other subjects were used as sub-optimal demonstrations.

Apparatus

In order to collect information about the position of the participants, the table and the ball during the game, we used a VICON motion capture system (16 VICON MX-13 cameras with the VICON IQ 2.5 software, 120 frames per second). Therefore, 25 VICON infrared reflecting markers were attached to the hands, wrists, elbows, shoulders, hips and the back and front of the participants. With this setup and a 3D kinematic model of the upper body of each individual, we could capture their whole body movement during the game. To identify the table and the net, we placed four markers at each corner of the table and one marker on one of the edges of the net. A standard table tennis table (length 2.74 m, width 1.53 m and height 0.76 m) and rackets conform with the rules of the International Table Tennis Federation [2011] were used. The surfaces of the rackets were chosen such that they did not allow for spin on both sides. The table tennis ball was covered with a gray-green infrared reflecting powder in order to detect it with the VICON system. As a result the ball had an additional weight of 2 grams. This coating slightly changed its physical properties (e.g., it additionally reduced the spin during the game). Additionally, the subjects were recorded with two video cameras. The experimental setup is also shown in Figure 4.4.

Procedure

The participants were asked to play a game of table tennis under three different conditions.

Condition 1. The subjects played a cooperative game of table tennis. The goal for the subjects is to maximize the number of returns in a rally for a ten minute period.

Condition 2. The subject was told to perform a competitive game of table tennis, while the opponent was instructed to return the ball “nicely” (i.e., the opponent was instructed to play towards the subject when possible in a cooperative way).

Condition 3. Both the subject and the opponent were instructed to play a competitive game of table tennis.

Each of the seven subjects played against the opponent one game under each of the three conditions. The participants were required to play table tennis according to the standard table tennis rules defined

	Method	Naive 1	Naive 2	Naive 3	Naive 4	Naive 5	Skilled 1	Coop.
Average reward difference	MMG	1.01	0.28	0.90	1.16	0.69	0.49	0.55
	MMS	1.16	0.07	1.24	0.86	0.71	0.33	0.50
	RE	0.70	0.11	0.60	0.80	0.42	0.31	0.55
Scores Condition 2		5:33	12:33	2:33	5:33	2:33	21:34	
Scores Condition 3		13:33	17:33	10:33	5:33	17:33	20:33	

Table 4.1: Summary of the results of the evaluations for the different methods. The differences in the average rewards with respect to the expert, define the differences between the reward of the expert and the spared test subject of the non-expert data set. The feature of winning and loosing the game were not included. MMG corresponds to the model-free maximum-margin of game values, MMS corresponds to the model-free maximum margin of states values with an horizon of three and RE corresponds to the relative entropy method (see Section 4.2.2).

by the International Table Tennis Federation [2011] with the following exceptions: (i) The players did not switch sides after a game, (ii) the expedite system¹ did not apply during the game, and (iii) the first serve of the match was always executed by the subject (never by the opponent). A game consisted of the best of five matches, i.e., the game was won by the player who first won three matches. Before the experiment started, the subjects played a friendly game with the opponent for 10 minutes in order to get used to the slightly altered bouncing properties of the table tennis ball (due to the coating with reflective powder). Each subject was required to read the rules before the experiment. The current score of the game in Condition 2 and 3 were displayed on a scoreboard visible for both of the two players. In each game, a referee ensured that the game was conducted in accordance with the rules. The score was protocolled by two of the experimenters independently and reconciled afterwards.

Data Processing

The captured motion was post-processed using the VICON IQ 2.5 software. The marker labels were automatically assigned to each marker using the VICON IQ 2.5 trajectory labeler. Errors that occurred during this automatic labeling process were manually corrected afterwards. The ball had to be labeled manually as it was tracked as a single VICON marker. The VICON IQ 2.5 kinematic fitting function computed the 3D kinematic information of the subjects automatically. Bouncing and hitting events for all data were then automatically labeled during another MATLAB post-processing step and manually reassigned if necessary. For each point, the score was automatically computed based on this information and reconciled with the score information recorded by the experimenters. Finally, for each time where the ball was hit by the subject, the corresponding state and reward features were extracted and saved in a MATLAB file.

4.3.2 Results and Discussion

Only one of the subjects was able to win against the opponent in the competitive game under Condition 3. All other games were won by the skilled opponent. The scoring results of the subjects that lost the game can be found in Table 4.1. The skilled player who won the game in Condition 3 was able to win 41 out of 75 rallies. Based on these results, the data was divided into two subsets: (1) a non-expert data set and (2) an expert data set. The non-expert data set included all games of the subjects who lost against the fixed opponent, i.e., all naive subjects and one of the skilled players, as well as all cooperative games.

¹ Expedite system: additional rules to discourage slow play in a table tennis match. It is used after 10 minutes of play or if requested by both players.

We will refer to the players that lost as Naive 1 to 5 and Skilled 1. The expert data set consisted of all rallies in the competitive game (Condition 3) played by the skilled player that won against the opponent. We will refer to this player as Expert. When asked which player performed worst, the opponent stated that Naive 3 was the worst.

We tested all three IRL methods as described in Section 4.2.2. To evaluate the potential reward functions, we performed a leave-one-subject-out testing scheme. We computed the reward feature weights for each of the three methods seven times. Every time leaving out all rallies (i.e., state-action trajectories) of one of the subjects that lost or the rallies of the cooperative game of the Expert respectively. We also excluded 20 rallies of the Expert for the validations. The obtained reward functions were tested for the different skill levels of the subjects using the excluded rallies demonstrated in the game under Condition 3 only and the different styles using the cooperative game of the Expert.

All resulting reward functions yielded the highest rewards for the feature of the terminal state for losing or winning the rally. Winning the rally was therefore highly desirable for the agent while losing should be avoided. For the evaluations, we did not consider this feature in order to see how well we can distinguish the subjects based on the other strategic elements.

In the following, we will first present the overall results of the three methods showing that we were able to distinguish between different playing skills and styles. Subsequently, we will discuss the influence of the horizon for the MMS algorithm. Finally, we discuss the results for all features separately.

Classifying the Skill Levels of the Players

We computed the differences in the average reward for a state-action pair of the spared expert and non-expert data for the reward functions obtained from the three methods described in Section 4.2.2 abbreviated as before as MMG, MMS, and RE. The results in terms of the differences in the average reward between expert and non-expert are displayed in Table 4.1. All three reward functions were able to distinguish between the non-expert games and the expert game, as well as between the different playing styles of the expert (competitive vs cooperative). In general the average reward for each player reflected the skill level of the players with the exception of Naive 2. For all naive players except Naive 2, the differences were high, while the difference between Skilled 1 and the Expert was moderate. These differences were more distinctive for the MMS algorithm.

All three reward functions obtained in the evaluation resulted in a very small difference in the average reward of the Expert and Naive 2, followed by Skilled 1 and Naive 5. Furthermore, all three methods showed relatively large differences between the Expert and the players Naive 1, Naive 3 and Naive 4. However, they disagree in the ranking of these three players. While the reward function obtained by the MMG and RE algorithm show the highest difference for the Expert and Naive 4, the reward function obtained by the MMS algorithm yield the highest difference between the Expert and Naive 3. Naive 4 being the worst player is in compliance with the scoring results of Experiment 3, while Naive 3 being the worst player is in compliance with the statement of the permanent opponent.

Analyzing player Naive 2, we can conclude that the player chooses his actions based on the same principles as the Expert. This player lost against the opponent due to his inaccurate movement execution. In comparison, the player Skilled 1 has a very good movement execution due to his long training and experience. However, he was not able to win against the opponent, although this player had the most experience in terms of years. This suggests that Skilled 1 was a very good player in terms of playing the ball successfully back to the opponent, but was not efficient in choosing his actions without the strategic element of spin.

Evaluating the differences of each feature individually yielded different performances for the three reward functions. While the reward function obtained by maximum margin of state values was able to distinguish 91 % of the dominant features of expert and non-expert, the reward function of relative entropy was able to distinguish 83 % and the reward function obtained by maximum margin of game values only 71 %.

	H	Naive 1	Naive 2	Naive 3	Naive 4	Naive 5	Skilled 1	Coop.
Average reward difference	1	1.30	0.04	1.17	0.91	0.74	0.30	0.43
	2	1.20	0.07	1.22	0.87	0.72	0.33	0.47
	3	1.16	0.07	1.24	0.86	0.71	0.33	0.50
Average reward diff. before terminal state	2	0.91	-0.21	0.92	0.57	0.38	-0.12	0.23
	3	1.12	0.04	1.23	0.89	0.76	0.24	0.53

Table 4.2: Summary of the results for the different horizons with the MMS algorithm. The differences in the average reward with respect to the expert trained with the different horizons H . The differences in the average reward directly before the terminal, define the differences of the reward of the expert and the spared test subject for the state before the terminal or the average reward of the two states before the terminal for the horizons 2 and 3 respectively.

Influence of the Planning Horizon

For the MMS algorithm, we evaluated the setup with three different horizons. We chose the horizons of $H = 1$, $H = 2$ and $H = 3$. The horizon of one considers only one state-action pair. The horizon of two also considers the state-action pair presented directly after the current one. A horizon of three means that we consider up to two state-action pairs following the current one.

The results of the average reward differences of the sub-optimal policies and the expert for the whole game and the states directly before the terminal are displayed in Table 4.2. In general, the average reward difference was reduced slightly with increasing horizon, while the average reward difference for the last $H - 1$ states before the terminal state increases with growing planning horizon, reaching its maximum with a horizon of three. Horizons larger than three did not improve the differences in the reward.

Individual Reward Features

Analyzing the reward weights individually, the different methods showed similar weights for the most important features (i.e., the features with the highest weights). The largest influence resulted from the bouncing angles α_y and α_z , the table preferences and the distance between the desired bouncing point and the racket of the opponent. For simplicity, we will only discuss the parameter values for the individual features of the reward functions obtained by the MMS and RE algorithm (as MMG had the worst performance in terms of individual feature classification).

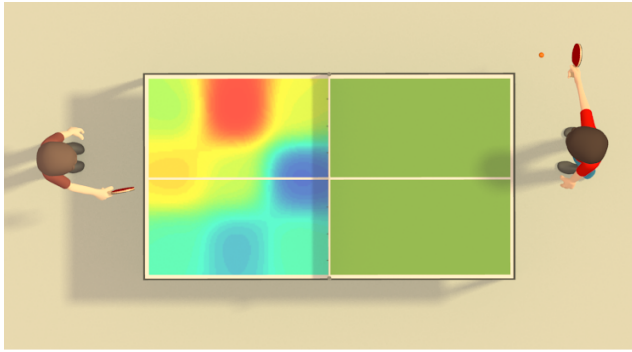
The reward weights for the individual features are displayed in Figure 4.5a and b. We also showed the average reward differences for the spared test data sets for each feature individually in Figure 4.5c and for the different time steps in Figure 4.5d. Figure 4.6 shows the various characteristics of the features for each subjects individually. We will discuss all features in the next sections.

Goal Preferences on the Table

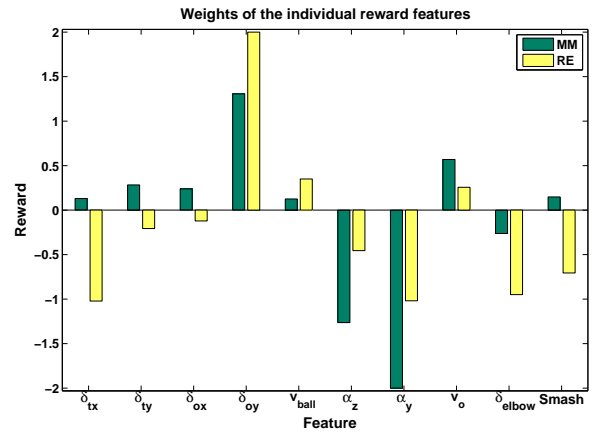
The preferences of the locations on the table are independent from the state information of the opponent, but they do reflect parts of the strategy that will also be covered by other features. The resulting reward functions of the different algorithms showed a preference for the areas where the opponent would have to return the ball using the backhand, while the areas that are suited for returning the ball with the forehand and the areas directly after the net are often rather avoided (see Figure 4.5a).

Distance to the Edges of the Table

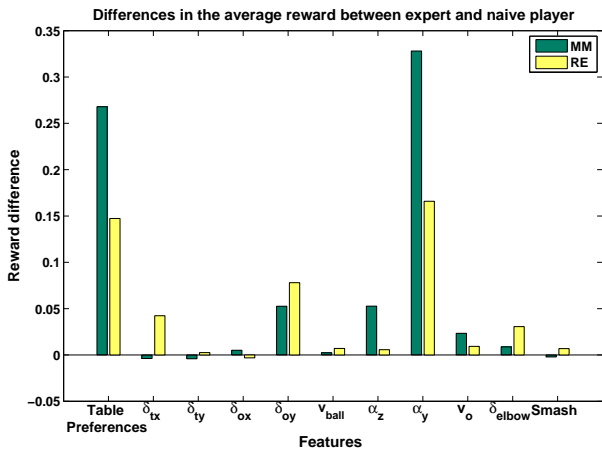
The distance of the bouncing point of the ball to the edges of the table had only a small positive influence in the reward function yielded by the max-margin algorithm. The reward function yield by the RE algorithm assigned a little negative reward for playing the ball close to the edge in the y-direction



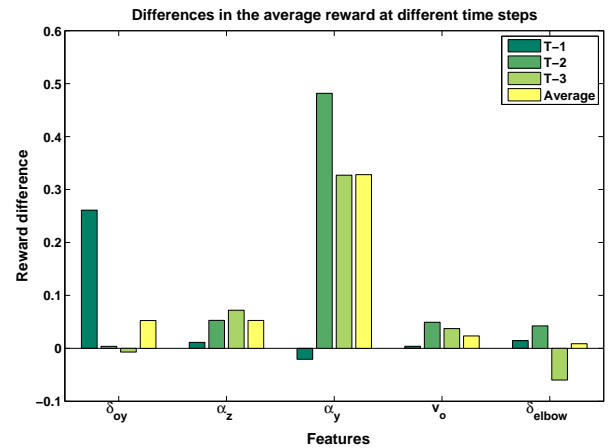
(a) Reward function for table preferences



(b) Reward feature weights



(c) Average reward differences



(d) Reward differences features at different time steps

Figure 4.5: Resulting parameter values for the individual features. Figure a shows the resulting reward function of the table preferences for Algorithm 7 (MM). Figure b shows the weights of all other features for Algorithm 7 (MM) and Algorithm 8 (RE), respectively. Figure c shows the differences of the average reward of the expert and the naive player for each feature separately using the reward function of the max-margin algorithm (green) and the relative entropy algorithm (yellow). Figure d shows the differences of the average rewards for the most important features at different time steps before the terminal state (win or loss) for the reward function yield with the max-margin algorithm.

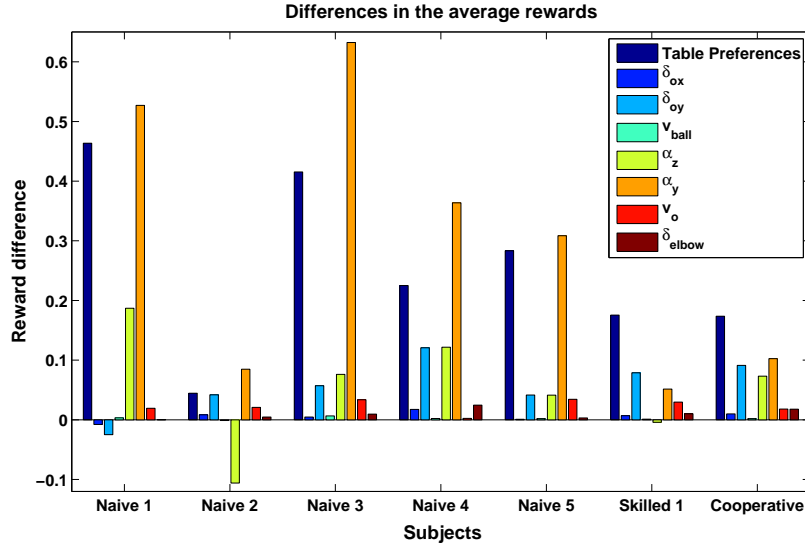


Figure 4.6: Histogram of the average reward differences between the expert and sub-optimal players for each player and each feature individually. The reward function was received by the MMS algorithm with a horizon of three.

(i.e., along the width of the table) and a relatively high negative reward for playing the ball close to the edge in the x-direction (direction towards the player). The average reward differences in the evaluations indicate that the reward assigned by the reward function of the RE method is to be favored (see Figure 4.5b).

Distance to the Opponent

Maximizing the distance between the position of the bouncing point and the position of the opponent in the x-direction (i.e., direction towards the opponent) received only a small reward (Figure 4.5a) and also had only a small effect in the evaluations (Figure 4.5b). While the reward function of the max-margin algorithm assigned a slightly positive reward for maximizing this distance, the reward function yielded by the relative entropy algorithm assigned a slightly negative reward. The evaluations on the spared test data were in favor for the positive reward weights.

The distance in y-direction (i.e., along the width of the table) between the bouncing point and the racket of the opponent resulted in a high reward in both reward functions. This feature also influenced the differences in the reward yield by the naive and expert table tennis player.

The overall performance on average only increased by $\sim [0.05|0.08]^2$. The differences in the average reward for the features before a terminal state, increased dramatically by $\sim [0.26|0.40]$ and became a dominant factor in the reward function (see Figure 4.5d). The differences between the average reward two states before the terminal was below average. This observation suggests that the chance of winning a point increases with an increasing distance between the bouncing point and the racket between the player.

² In the following, the first value will correspond to the reward differences obtained by MMS algorithm and the second value will correspond to the reward differences obtained by the RE algorithm

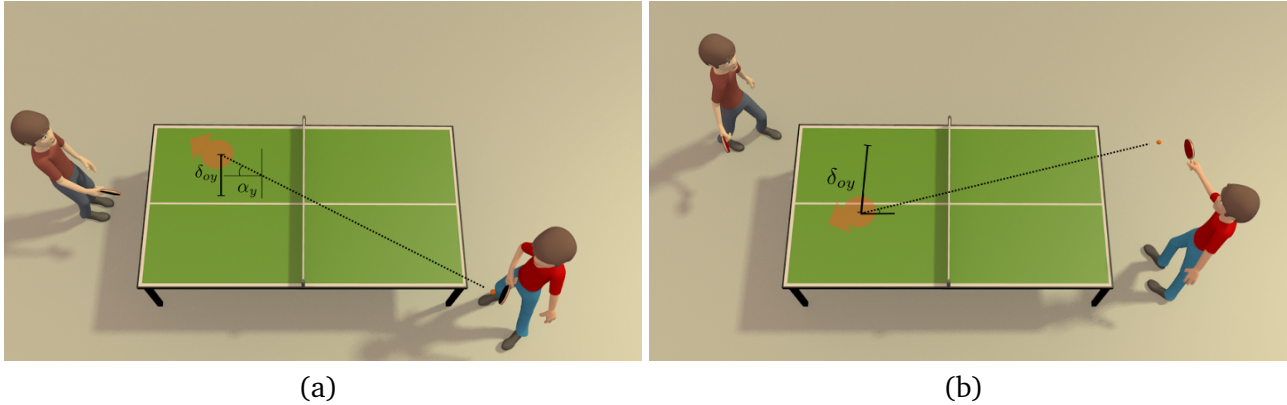


Figure 4.7: Possible strategy that distinguished the expert player that won the game, from the non-expert players that lost the game against the opponent. If the expert had the chance, he would play the ball very cross to the backhand area (Figure a). As a result the opponent was forced to move more into the left corner. The expert could then play the ball to the forehand area in order to increase the distance between the ball and the opponent (Figure b).

Proximity to the Elbow

Playing towards the elbow of the opponent had a negative effect. The weights for the elbow features were negative and increased the differences in the average reward between non-expert players and the expert player (see Figure 4.5b).

Velocity of the Ball and Opponent

The feature for the velocity of the ball had only a small positive weight and almost no influence on the difference between the players (see Figure 4.5a and b) in the evaluations.

The movement direction of the opponent relative to the ball had a moderate positive weight (see Figure 4.5a), but only a small influence in the evaluations on the differences between the non-expert and expert data set. This observation indicates that this feature was used by the Expert but did not dominate his behavior.

Bouncing Angles

We evaluated the direction of the ball by means of two angles: α_z and α_y . The horizontal angle α_z had a high negative reward value, i.e., smaller angles were preferred. The overall difference in the performance between the Expert and the naive players did increase the overall reward difference only slightly. Hence, the ball was in general played in a slightly flatter manner by the expert.

The angle α_y also had a high negative weight, i.e., playing the ball cross to the backhand area was preferred opposed to playing the ball cross towards the forehand area. These results are conform with the table preferences as displayed in Figure 4.5a. This feature was one of the dominating factors in the reward function and in the evaluations of the excluded subjects. The average difference between expert and naive players for the state right before the terminal state was only decreased by $\sim [0.02|0.01]$. The average reward two states before the terminal state on the other side were much higher than the overall average reward ($\sim [0.48|0.25]$).

This observation together with the results of the distance of the bouncing point and the racket, suggests the following strategy successfully applied by the Expert. When playing the ball very cross to the outer backhand area of the opponent, the opponent was forced to move to his left. The expert used this opportunity to play the ball to the other side of the table in order to increase the distance between the ball and the opponent (see Figure 4.7).

The observation that the overall difference in the reward between the Expert and Naive 2 and the Expert and Skilled 1 are not high, indicates that these two players use similar techniques in terms of playing the ball cross to the backhand area. However, when comparing the results in the last hits before the terminal state, we notice that i) the expert usually plays the ball more cross in the backhand area, forcing the opponent to move further in this direction and ii) the two players did not play the ball into the other direction afterwards in order to increase the distance.

4.4 Conclusion of Chapter 4

In this chapter, we modeled table tennis games as a Markov decision problem. We have shown that it is possible to automatically extract expert knowledge on effective elements of basic strategy in the form of a reward function using model-free Inverse Reinforcement Learning (IRL). To accomplish this step, we collected data from humans playing table tennis using a motion capture system. Participants with different skill levels played in both a competitive and a cooperative game during this study. Based on their performance, we divided the data into an expert and a sub-optimal data set. These data sets have been used to infer and evaluate the reward functions.

We have tested three different model-free inverse reinforcement learning methods. Two were derived from the model-based IRL method of Abbeel and Ng [2004]. The third algorithm was the model-free relative entropy method of Boularias et al. [2011]. The resulting reward functions were evaluated successfully in a leave-one-subject-out testing scheme. All learned reward functions were able to distinguish strategic information of players with different playing skills and styles.

The presented approach used information about the position of the player and the opponent as well as the ball position, velocity and orientation. However, spin was not included in this setup. In order to include spin, a system that infers the spin of the ball from its trajectory would be necessary. The reward function was able to capture the goal of the task, in terms of winning the rally while avoiding to lose it. The key elements revealed by the model were (i) playing cross to the backhand area of the opponent, (ii) maximizing the distance of the bouncing point of the ball and the opponent, and (iii) playing the ball in a flat manner. Other elements as playing against the moving direction and the velocity of the ball were also positively correlated.

The presented approach is not limited to analyze individual preferences of players and successful strategic components against a specific opponent. Rather, the learned reward function can also be used within the MDP framework for artificial systems such as table tennis robots or virtual reality-based table tennis games. Thus, the robot can learn a strategy against a human opponent. The described method allows an artificial system to analyze the strategy of the opponent and as a result, the system will be able to anticipate the actions of its opponent. Such anticipation can allow artificial systems to adapt their own strategies to improve their chances.

5 Conclusion and Future Work

This thesis was dedicated to the goal of modeling and learning a complex motor task using robot table tennis as a test case. We developed a model of robot table tennis based on human motor control in order to achieve robust, efficient, and human-like hitting motions. We also showed that such a complex task can be learned from demonstrations and interactions with humans. Both setups, the biomimetic and the learned approach, were successfully tested on a real Barrett WAM robot arm. Furthermore, we showed that the reward function which captures basic strategic knowledge can be inferred from human table tennis data using model-free inverse reinforcement learning. In the following, we give a detailed summary of the thesis and its contributions. We also discuss possible extensions to the work presented and list the publications which resulted from this thesis.

5.1 Summary of the Thesis

This thesis has contributed to the fields of learning and robotics, and has further supported the different hypotheses in human motor control. The results and contributions have been organized into three different chapters. Following this grouping, we summarize in this section the individual chapters and their contributions individually.

5.1.1 Modeling Complex Motor Tasks

In Chapter 2, we presented a novel model for robot table tennis. Previous work on robot table tennis concentrated on efficient vision algorithms in well defined environments and often used hardware which was developed for this specific purpose. In contrast to these contributions, we used an anthropomorphic seven DoF Barrett WAM arm in a semi-structured environment and concentrated on smooth movement generation using a biomimetic approach. In order to model table tennis, we relied on hypotheses and observations from human motor control and sport science. The resulting player structures the hitting movement into four stages following the observation of Ramanantsoa and Durey [1994] who reported such a movement organization in human striking motions. The trajectories in the individual stages were represented by 5th order splines. We used the hypothesis of a virtual hitting point [Ramanantsoa and Durey, 1994] to define the interception parameters as opposed to a reactive approach as suggested for catching tasks [Gigerenzer, 2005, Yeo et al., 2012]. The redundancy of the arm was solved by minimizing the distance to a defined comfortable hitting posture following the suggestion of Cruse [1986]. The resulting robot table tennis player was able to successfully return balls played towards the robot in simulation and on a real robot while producing human-like hitting motions. A video showing the performance of the system, including a small match against its creator, can be found at <http://www.youtube.com/watch?v=BcJ4S4L1n78>.

The positive results of this study showed that the used features of human motor control in our model resulted in human-like hitting motions on a robot. This result gives further evidence that these hypotheses might also be used in human motor control.

5.1.2 Learning Complex Motor Tasks

While previous approaches on learning robot table tennis concentrated on learning the interception parameters for the hitting tasks [Miyazaki et al., 2006, Lai and Tsay, 2011, Kober et al., 2012b], we concentrated on learning robust hitting movements and how these can be selected and generalized in a

table tennis game. Furthermore, we also used machine learning approaches in order to infer a reward function to model a higher-level strategy and therefore to win the game.

Selecting and Generalizing Striking Movements

In Chapter 3, we presented a framework for learning cooperative table tennis based on Dynamical system Motor Primitives (DMPs, [Ijspeert et al., 2002]), imitation and reinforcement learning. In contrast to most approaches of learning motor tasks using DMPs so far, complex tasks cannot be learned using only one primitive. In order to account for the complexity involved, several movement primitives need to be chosen and adapted in response to the current environmental stimuli. To meet this requirement, we developed an algorithm called Mixture of Motor Primitives (MoMP) that selects and generalizes movement primitives in a library based on the suitability for the current task context. Instead of selecting and generalizing among the motor primitives purely based on a manually pre-defined similarity to the current context, we weighted the primitives with their predicted performance for the given situation. These weights can be adapted autonomously based on the performance of the generated hitting movements using reinforcement learning. As a result, we can suppress unsuited motor primitives from the library and use only those that qualify for the specific situation to create the hitting motion. The framework was successfully evaluated for the table tennis task on a real Barrett WAM robot arm. The initial movement library was created from 25 striking movements demonstrated by a teacher using imitation learning. Using the MoMP algorithm, the system was able to select the appropriate primitives and to generalize these to new situations. As a result, balls played towards the robot can be returned successfully to the opponent's court. We were able to show that the selection process of movement primitives can be adapted while playing table tennis and that due to the adapted selection process by MoMP the performance increases significantly. A video showing the performance of the robotic system can be found at <http://www.youtube.com/watch?v=SH3bADiB7uQ>.

The presented framework is not limited to the show-case of table tennis. The approach is suited in general for tasks where the movements need to be adapted to new, but still similar, situations as commonly found in striking sports. Additionally, it could also be used for other goal-oriented movements such as foot-step generation for walking over rough terrain.

This work was also recognized by Dr. Muster, honorary member of the German table tennis coaches club (Verband deutscher Tischtennis Trainer, Muster [2013]). In his opinion the demonstration of multiple hitting motions to the robot and the subsequent learning of the association of situations and suited hitting motions is further support to the *differential training* method [Schöllhorn, 2000, 2003]. Instead of repeating the same hitting movement every time, the theory of differential training recommends the player to train under varying conditions allowing to learn a different movement plan for the each situation.

Inferring Strategical Information

Even after acquiring the necessary motor skills as demonstrated in Chapters 2 and 3, competitive games still require a higher-level strategy in order to win the game. The strategy defines for each particular situation where and how the ball should be returned to the opponent's court. Defining such a higher-level strategy manually seems to be infeasible. Therefore, the strategy needs to be learned and adapted towards a certain opponent. To get one step closer to this goal, we showed in Chapter 4 that it is possible to automatically extract expert knowledge on effective elements of a basic strategy captured in the form of a reward function of a Markov decision problem. In order to capture these strategic elements, we collected optimal and sub-optimal data from human table tennis players with different playing skills and styles using a motion capture system. This data was used to infer the reward function using model-free inverse reinforcement learning. We tested three different inverse reinforcement learning methods. Two were derived from the model-based inverse reinforcement learning method of Abbeel and Ng [2004]. The third algorithm used was the model-free relative entropy method of Boularias

et al. [2011]. The resulting reward functions were evaluated in a leave-one-subject-out testing scheme on the collected human table tennis data. The reward functions were able to distinguish the strategic information of the different playing styles and skills. It was possible to capture the goal of the task in terms of winning the game. Furthermore, the methods were able to reveal key elements used to achieve this goal. Playing the ball to the backhand area of the opponent, maximizing the distance of the bouncing point of the ball and the opponent as well as playing the ball in a flat manner were the most prominent features that distinguished expert and sub-optimal demonstrations.

Using such a reward function, an artificial agent can learn an effective strategy against a human using reinforcement learning as well as to analyze the strategy of the opponent. Furthermore, this approach can be used to analyze individual preferences of players and successful strategic components against a specific opponent which would be helpful in situation-driven table tennis training.

5.2 Open Problems

This thesis presented encouraging results towards the goal of modeling and learning complex motor tasks. On top of this work, there are several possible extensions that will be discussed in the following.

5.2.1 Extension of the Table Tennis Player

Although the created robot table tennis systems were able to return balls successfully, the presented work can be extended to a mobile platform and to opponent modeling in order to further improve the performance of the system. In this section, we will discuss these possible extensions of the robot table tennis player.

Exporting the System to a Mobile Platform

While we were able to achieve very good results with both the biomimetic and the learned player, the system is still limited to a small working space due to the fixed base of the robot. Exporting the system to a mobile platform would enable the system to move its *body* to a position such that it can perform an optimal stroke movement. Such an optimal hitting execution would increase its probability to succeed. Furthermore, a mobile platform would also be crucial in order to implement and learn different strategies against a human opponent and is an essential step to obtain a robot player that gets a performance close to a human player. One possibility to achieve this goal would be to use a moving platform consisting of linear axes. Another solution would be to implement the framework on a humanoid robot. This approach would increase the overall complexity involved in this task, but would also be more flexible to use in a broader range of situations and allows to study whole body coordination in complex tasks. As a first step towards this goal, we moved the robotic framework described in Chapter 2 successfully to a setup consisting of a BioRob mounted on a mobile platform with three linear axes in simulation. However, real robot evaluations would be necessary to judge the suitability of these approaches.

Intention Inference

The table tennis frameworks presented in Chapter 2 and 3 rely on visual position tracking of the ball only. This information alone is already sufficient to return a ball in a friendly game of table tennis. However, in a competitive game, the information given by only the ball trajectory would be insufficient. Therefore, it could be beneficial to use additional information given by the movement patterns of the opponent in order to predict his future actions. Such additional information can help the robot to foresee the intentions of the opponent. We could already show that it is possible to predict the direction of the returned ball and as a consequence whether the robot has to play a fore- or backhand even before

the ball was hit by the opponent [Wang et al., 2013]. Furthermore, it might be possible able to predict the spin of the ball based on the observed hitting movement of the opponent.

5.2.2 Learning Complex Motor Skills

In Chapter 3 and 4, we demonstrated how a complex task can be learned using imitation, reinforcement, and inverse reinforcement learning. Therefore, the teaching input was carried out in the form of kinesthetic teach-in or an observation of the environment. This form of teaching is very effective, but can still be extended and improved by using additional forms of communication and instruction. In the following, we will shortly discuss the integration of natural language and gesture recognition for this propose. Furthermore, we will also discuss the problem of modeling the dynamics of unknown objects that the robot needs to interact with and the validation of the presented methods in other domains.

Natural Language and Gestures as Teaching Input

The framework for learning a hitting movement in Chapter 3 only uses physical demonstrations and the observation of the environment together with a well defined reward function to teach the robot a skill. This form of interaction is very useful when teaching a motor skill, but should not be the only interaction between human and robot. Often a teacher expresses his feedback additionally in a verbal manner after physical demonstrations: "Try to rather roll over the ball instead of hitting it directly!" or "You are holding the racket the wrong way!". Also, the reward can be expressed in a verbal way: "Yes, very good" or "Not there yet". Therefore, natural language should be used additionally as an interface between human and robot. The use of spoken language should thereby be bidirectional. That means, it should not only be used to express a description or feedback about the execution of a behavior towards the robot, it should also be used by the robot in order to ask for feedback or for an additional demonstration in order to accelerate the learning. To communicate this way, the spoken words need to be recorded, recognized, and the semantic as well as the discourse needs to be interpreted. Natural language processing is an active research area and not solved yet. The difficulties encountered include the different ways of parsing one sentence as well as word ambiguities [Manning and Schuetze, 1999, Jurafsky and Martin, 2009].

Hardware developments such as the Kinect camera have enabled a relatively simple form of human posture real-time tracking. Therefore, gesture recognition can be used alongside with natural language processing to resolve ambiguities, infer the intentions and to complete unfinished sentences [Perzanowski et al., 1998]. Although computer systems are far away from understanding spoken language as well as humans, it seems a promising direction to guide the motor learning process additionally with spoken language and gestures.

Learning the Dynamics of an Unknown Object

Until now, most research on predicting the hitting point in striking sports concentrates on using supervised or reinforcement learning to predict the hitting point directly from a part of the ball trajectory [Miyazaki et al., 2006, Lai and Tsay, 2011, Kober et al., 2012b]. Each time an update of the prediction is necessary, a new model needs to be trained. These predictions usually depend on specific parts of the trajectory observed by the agent and require the system to store huge amounts of data points to enable accurate predictions. Humans on the other hand are not just able to predict the hitting point very reliable and independent from the observed part of the trajectory, they are also able to predicted the whole trajectory of a ball. Therefore, humans appear to have an inner model of the dynamics of the object they interact with. Hence, it should be investigated how such an inner model of the dynamics of an object can be learned and how the interception point can be predicted based on the predicted ball trajectory. Such an approach might also help to develop movement plans that minimize the uncertainty of intercepting an object successfully.

Testing of the Proposed Algorithms in Other Domains

The algorithms and frameworks presented in Chapter 3 and 4 were only tested on table tennis so far. Therefore, the potential of these methods should also be evaluated on other tasks such as tennis, catching, pointing tasks, badminton or foot placing in different terrains. Here, we already could show that the hitting movements acquired by the system to play table tennis can be generalized to catching skills [Kober et al., 2012a].

5.2.3 Learning Higher-Level Strategies

In Chapter 4, we discussed and showed how strategic elements for winning a match in table tennis can be inferred from observing humans playing the game. This information was captured in the form of a reward function using inverse reinforcement learning. Here, we would like to point out additional future research direction and extensions to this work.

Learning the Feature Basis Functions

The foundation of most inverse reinforcement learning approaches are feature basis functions which can be observed by the agent. These features define the reward function and are the bottleneck of this approach as they are usually defined manually. Therefore, it is necessary that all relevant features are known by the programmer. To create a fully autonomous robot system, it is necessary to construct these features automatically. This problem was also recognized by [Abbeel and Ng, 2004]. Levine et al. [2010] suggested that the features can be constructed from a logical combinations of components that are the most relevant to the task. Nevertheless, this approach also requires the definition of the most relevant components of the state space beforehand. Even if it would be possible to consider the whole state space as components, some features might be the result of a non-trivial combination of these elements. Other feature combinations might be redundant and could dominate the behavior due to their multiple occurrences. Therefore, it should be investigated how these relevant feature components and their relation to each other can be extracted from observations. Here, the study of human attention processing [Begum and Karray, 2011] might be helpful to achieve this goal.

Partially Observable Markov Decision Problems

In Chapter 4, we modeled table tennis as a Markov Decision Problem (MDP), assuming the task consists of one agent that has perfect knowledge about its environment. This approach is a good starting point, but might be an overly strong assumption. As in many other complex tasks, table tennis includes multiple agents. In the current model, we did not account for the opponent's personal weaknesses and strategy as well as the possibility of imperfect sensory information. Here, Partially Observable Markov Decision Problems (POMDPs, [Monahan, 1982]) could be useful. In contrast to modeling the task using a MDP, POMDPs assume that the agent cannot completely observe its environment. POMDPs model uncertainty of the state the agent is currently in such that we are able to include beliefs about the intentions of the opponent. Here, it should be investigated whether it is possible to extend the model-free methods presented in Chapter 4 to POMDPs.

RGBD Cameras versus VICON

In Chapter 4, we captured the motion data using a VICON tracking system based on infrared cameras and infrared reflecting markers to collect the data of humans playing table tennis. Such a system is effective but also constrained to a facility providing the necessary equipment. The development of RGBD cameras such as the Kinect made it possible to use a small mobile device to track human postures in real time. Unfortunately, RGBD cameras have only a limited working area and their accuracy is not as high as the one of a calibrated VICON system. However, as RGBD cameras can be used more flexible

and allow for real time tracking of the upper body of the player, it should be investigated whether these cameras can be used in this context.

5.3 Publications

The research presented in this thesis has been partly published in or has lead to the following publications.

5.3.1 Journal Papers

Muelling, K.; Boularias, A.; Schoelkopf, B.; Peters, J. (under review). Learning Strategies in Table Tennis using Inverse Reinforcement Learning. *Biological Cybernetics*.

Muelling, K.; Kober, J.; Kroemer, O.; Peters, J. (2013). *Learning to Select and Generalize Striking Movements in Robot Table Tennis*, *International Journal of Robotics Research (IJRR)* vol. 32, 3.

Wang, Z.; **Muelling, K.;** Deisenroth, M. P.; Ben Amor, H.; Vogt, D.; Schoelkopf, B.; Peters, J. (2013). Probabilistic Movement Modeling for Intention Inference in Human-Robot Interaction, *International Journal of Robotics Research (IJRR)* vol. 32,7

Muelling, K.; Kober, J.; Peters, J. (2011). *A Biomimetic Approach to Robot Table Tennis*, *Adaptive Behavior Journal* vol. 19, 5.

5.3.2 Conference and Seminar Papers

Muelling, K.; Kober, J.; Kroemer, O.; Peters, J. (2012). *Learning to Select and Generalize Striking Movements in Robot Table Tennis*, Proceedings of the AAAI 2012 Fall Symposium on Robotics that Learn Interactively from Human Teachers

Peters, J.; Kober, J.; **Muelling, K.;** Nguyen-Tuong, D.; Kroemer, O. (2012). *Robot Skill Learning*, Proceedings of the European Conference on Artificial Intelligence (ECAI)

Kober, J.; **Muelling, K.;** Peters, J. (2012). *Learning Throwing and Catching Skills*, Proceedings of the International Conference of Robot Systems (IROS), Video Track

Wang, Z.; Lampert, C.H.; **Muelling, K.;** Peters, J. (2011). *Learning Anticipation Policies for Robot Table Tennis*, Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)

Wang, Z.; Boularias, A.; **Muelling, K.;** Peters, J. (2011). *Balancing Safety and Exploitability in Opponent Modeling*, Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence (AAAI).

Muelling, K.; Kober, J.; Peters, J. (2010). *Learning Table Tennis with a Mixture of Motor Primitives*, Proceedings of the 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2010)

Peters, J.; **Muelling, K.;** Kober, J. (2010). *Experiments with Motor Primitives to learn Table Tennis*, Proceedings of the 12th International Symposium on Experimental Robotics (ISER 2010), Springer, Berlin, Germany

Muelling, K.; Kober, J.; Peters, J. (2010). *A Biomimetic Approach to Robot Table Tennis*, Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)

Muelling, K.; Kober, J.; Peters, J. (2010). *Simulating Human Table Tennis with a Biomimetic Robot Setup*, From Animals to Animats 11: Eleventh International Conference on the Simulation of Adaptive Behavior

(SAB 2010),

Peters, J.; **Muelling**, K.; Altun, Y. (2010). *Relative Entropy Policy Search*, Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence,

Peters, J.; **Muelling**, K.; Kober, J.; Nguyen-Tuong, D.; Kroemer, O. (2009). *Towards Motor Skill Learning for Robotics*, Proceedings of the International Symposium on Robotics Research (ISRR),

Muelling, K., and Peters, J. (2009). *A computational model of human table tennis for robot application*, Proceedings of Autonome Mobile Systeme (AMS 2009)



Bibliography

- P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference of Machine Learning (ICML)*, 2004.
- P. Abbeel, D. Dolgov, A. Ng, and S. Thrun. Apprenticeship learning for motion planning with application to parking lot navigation. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 1083 – 1090, 2008.
- P. Abbeel, A. Coates, and A. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29:1608 – 1679, 2010.
- J. Abbs and K. Cole. Neural mechanisms of motor equivalence and goal achievement. In S. Wise, editor, *Higher Brain Functions: Recent Explorations of the Brain's Emergent Properties*, pages 15–43. John Wiley & Sons, Hoboken, NY, 1987.
- L. Acosta, J. Rodrigo, J. Mendez, G. Marchial, and M. Sigut. Ping-pong player prototype. *IEEE Robotics and Automation Magazine*, 10(4):44–52, 2003.
- R. Alexander. A minimum energy cost hypothesis for human arm trajectories. *Biological Cybernetics*, 76(2):97–105, 1997.
- R. Andersson. *A robot ping-pong player: experiment in real-time intelligent control*. MIT Press, Cambridge, MA, USA, 1988.
- L. Angel, J. Sebastian, R. Saltaren, R. Aracil, and R. Gutierrez. Robotenis: Design, dynamic modeling and preliminary control. In *Proceedings of the International Conference on Advanced Intelligent Mechatronics*, pages 747–752, 2005.
- B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous System*, 57(5):469 – 483, 2009.
- M. Begum and F. Karray. Visual attention for robotic cognition: A survey. *IEEE Transactions of Autonomous Mental Development*, 3:92 – 105, 2011.
- N. Bernstein. *The Coordination and Regulation of Movements*. Pergamon, Oxford, 1967.
- A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 1371–1394. Springer Berlin Heidelberg, 2008.
- J. Billingsley. Machineroe joins new title fight. *Practical Computing*, May/June:14–16, 1983.
- C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006.
- C. Bishop and M. Tipping. Bayesian regression and classification. In J. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Models and Applications*, volume 190, pages 267 – 289. IOS Press, 2003.
- S. Bitzer, M. Howard, and S. Vijayakumar. Using dimensionality reduction to exploit constraints in reinforcement learning. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 3219 – 3225, 2010.

- R. Bootsma and C. Peper. Predictive visual information sources for the regulation of action with special emphasis on catching and hitting. In L. Proteau and D. Elliott, editors, *Vision and Motor Control*, pages 285 – 314. North-Holland, 1992.
- R. Bootsma and P. van Wieringen. Visual control of an attacking forehand drive in table tennis. In O. Meijer and K. Roth, editors, *Complex Movement Behaviour: The Motor-Action Controversy*, pages 189–199. Amsterdam: North-Holland, 1988.
- R. Bootsma and P. van Wieringen. Timing an attacking forehand drive in table tennis. *Journal of Experimental Psychology: Human Perception and Performance*, 16(1):21–29, 1990.
- A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Artificial Intelligences and Statistics (AISTATS)*, pages 20 – 27, 2011.
- S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.
- A. Bryson and Y. Ho. *Applied Optimal Control*. Wiley, New York, 1975.
- 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*, 32(2):286–298, 2007.
- H. Chan, I. King, and J. Lui. Performance analysis of a new updating rule for $td(\gamma)$ learning in feedforward networks for position evaluation in go game. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1716 – 1720, 1996.
- S. Chandramohan, M. Geist, F. Lefevre, and O. Pietquin. User simulation in dialogue systems using inverse reinforcement learning. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association*, pages 1025 – 1028, 2011.
- S. Chiappa, J. Kober, and J. Peters. Using bayesian dynamical systems for motion template libraries. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 297 – 304, 2008.
- J. Craig. *Introduction to Robotics. Mechanisms and Control*. Addison-Wesley Publishing Company, Inc, Reading, Massachusetts, 1989.
- H. Cruse. Constraints for joint angle control of the human arm. *Biological Cybernetics*, 54(2):125–132, 1986.
- H. Cruse, M. Brüwer, P. Brockfeld, and A. Dress. On the cost functions for the control of the human arm movement. *Biological Cybernetics*, 62:519–528, 1990.
- K. Dautenhahn and I. Werry. Towards interactive robots in autism therapy: background, motivation and challenges. *Pragmatics and Cognition*, 12(1):1–35, 2004.
- J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.
- G. Diaz, J. Cooper, C. Rothkopf, and M. Hayhoe. Saccades to futur ball location revial memory-based prediction in a virtual-reality interception task. *Journal of Vision*, 2013.
- D. Elliott, G. Binsted, and M. Heath. The control of goal-directed limb movements: Correcting errors in the trajectory. *Human Movement Science*, 18(2).
- H. Fässler, H. A. Beyer, and J. T. Wen. A robot ping pong player: optimized mechanics, high performance 3d vision, and intelligent sensor control. *Robotersysteme*, 6(3):161–170, 1990.

-
- P. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.
- T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of Neurosciences*, 5(7):1688–1703, 1985.
- G. Gigerenzer. I think, therefore i err. *Social Research*, 72(1):195 – 218, 2005.
- S. Giszter, K. Moxon, I. Rybak, and C. J. Neurobiological and neurobotic approaches to control architectures for a humanoid motor system. *Intelligent Systems and their Applications*, 15:64 – 69, 2000.
- M. Gomez-Rodriguez, J. Peters, J. Hill, B. Schoelkopf, A. Gharabaghi, and M. Grose-Wentrup. Closing the sensorimotor loop: haptic feedback helps decoding of motor imagery. *Journal of Natural Engineering*, 8(3), 2011.
- F. Guenter, M. Hersch, S. Calinon, and A. Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics, Special Issue on Imitative Robots*, 21(13):1521 – 1544, 2007.
- C. Harris and D. Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394:780–784, 1998.
- R. Harrison, J. Mackenzie, B. Morris, and J. Springett. Design and build of a robotic ping pong player. Technical report, The University of Adelaide, 2005.
- J. Hartley. Toshiba progress towards sensory control in real time. *The Industrial Robot*, 14(1):50–52, 1987.
- H. Hashimoto, F. Ozaki, K. Asano, and K. Osuka. Development of a ping pong robot system using 7 degrees of freedom direct drive. In *Proceedings of the IEEE International Conference on Industrial Electronics, Control and Instruments (IECON)*, pages 608–615, 1987.
- M. Hayhoe, T. McKinney, K. Chajka, and J. Pelz. Predictive eye movements in natural vision. *Experimental Brain Research*, 217:125 – 138, 2012.
- H. Hecht. *Time-To-Contact*. Elsevier Science, Amsterdam, The Netherlands, 2004.
- F. Henry and D. Rogers. Increased response latency for complicated movements and the memory drum theory of neuromotor reaction. *Research Quarterly*, 31:448–458, 1960.
- N. Hogan. An organizing principle for a class of voluntary movement. *Journal of Neuroscience*, 4(11): 2745–2754, 1984.
- A. Hohmann, H. Zhang, and A. Koth. Performance diagnosis through mathematical simulation in table tennis. In A. Lees, J.-F. Kahn, and I. Maynard, editors, *Science and Racket Sports III*, pages 220 – 226. Routledge, London, 2004.
- Y. Huang, D. Xu, M. Tan, and H. Su. Trajectory prediction of spinning ball for ping-pong robot. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 3434 – 3439, 2011.
- A. Hubbard and C. Seng. Visual movements of batters. *Research Quarterly*, 25:42–57, 1954.
- A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1398 – 1403, 2002.

-
- International Table Tennis Federation. Table tennis rules, October 2011. URL http://www.ittf.com/ittf_handbook/hb.asp?s_number=2.
- R. Jacobs, M. Jordan, S. Nowlan, and G. Hilton. Adaptive mixtures of local experts. *Neural Computation*, 3:79 – 87, 1991.
- D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition and Computer Linguistics*. Prentice-Hall, 2009.
- M. Kawato and K. Samejima. Efficient reinforcement learning: computational theories, neuroscience and robotics. *Current Opinion in Neurobiology*, 17:205–212, 2007.
- S. Keele. Movement control in skilled motor performance. *Psychological Bulletin*, 70(6):387–403, 1968.
- J. Knight and D. Lowery. Pingpong-playing robot controlled by a microcomputer. *Microprocessors and Microsystems*, 10(6):332–335, 1986.
- J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems 21 (NIPS)*, pages 849 – 856, 2009.
- J. Kober, B. Mohler, and J. Peters. Learning perceptual coupling for motor primitives. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 834 – 839, 2008.
- J. Kober, K. Muelling, O. Kroemer, C. Lampert, B. Schölkopf, and J. Peters. Movement templates for learning of hitting and batting. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 688 – 694, 2010.
- J. Kober, K. Muelling, and K. Peters. Learning throwing and catching skills. In *Proceedings of the International Conference on Robot Systems (IROS), Video Track*, pages 5167 – 5168, 2012a.
- J. Kober, A. Wilhelm, E. Oztop, and J. Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361 – 379, 2012b.
- J. Kober, D. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. accepted.
- Z. Kolter and A. Ng. The Stanford LittleDog: A learning and rapid replanning approach to quadruped locomotion. *The International Journal of Robotics Research*, 30(2):150 – 174, 2011.
- P. Kormushev, S. Calinon, and D. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *Proceedings of the IEEE/RSJ Intl Conference on Intelligent Robots and Systems (IROS)*, pages 3232 – 3237, 2010.
- O. Kroemer, R. Detry, J. Piater, and J. Peters. Grasping with vision descriptors and motor primitives. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 47 – 54, 2010.
- A. Kuo. Harvesting energy by improving the economy of human walking. *Science*, 309(5741):1686–1687, 2005.
- C. Lai and J. Tsay. Self-learning for a humanoid robot ping-pong player. *Advanced Robotics*, 25:1183 – 1208, 2011.
- C. Lampert and J. Peters. Real-time detection of colored objects in multiple camera streams with off-the-shelf hardware components. *Journal of Real-Time Image Processing*, 7(1):31–41, 2012.
- M. Land and P. McLeod. From eye movements to actions: How batsmen hit the ball. *Nature Neuroscience*, 3(12):1231 – 1239, 2000.

-
- D. Lee and D. Young. Visual timing of interceptive action. In D. Ingle, M. Jeannerod, and D. Lee, editors, *Brain mechanisms and spatial vision*, pages 1–30. Dordrecht, Netherlands: Martinus Nijhoff, 1985.
- S. Levine, Z. Popovic, and V. Koltun. Feature construction for inverse reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.
- C. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999.
- M. Matsushima, T. Hashimoto, M. Takeuchi, and F. Miyazaki. A learning approach to robotic table tennis. *IEEE Transactions on Robotics*, 21:767–771, 2005.
- H. Miyamoto, S. Schaal, F. Galdolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato. A kendama learning robot based on bi-directional theory. *Neural Networks*, 9:1281–1302, 1996.
- F. Miyazaki, M. Matsushima, and M. Takeuchi. Learning to dynamically manipulate: A table tennis robot controls a ball and rallies with a human being. In S. Kawamura and M. Svinin, editors, *Advances in Robot Control*, pages 3137–341. Springer Berlin Heidelberg, 2006.
- G. Monahan. A survey of partially observable markov decision processes: Theory, models and algorithms. *Management Science*, 28:1 – 16, 1982.
- P. Morasso. Spatial control of arm movements. *Experimental Brain Research*, 42(2):223–227, 1981.
- T. Mori, M. Howard, and S. Vijayakumar. Model-free apprenticeship learning for transfer of human impedance behaviour. In *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS)*, pages 239 – 246, 2011.
- K. Muelling. Motor control and learning in table tennis. Master’s thesis, Eberhard-Karls University, 2009.
- K. Muelling, J. Kober, and J. Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359 – 376, 2011. URL <http://adb.sagepub.com/content/19/5/359.refs>.
- K. Muelling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263 – 279, 2013. URL <http://ijr.sagepub.com/content/32/3/263.refs>. SAGE Publications Ltd, All rights reserved.
- K. Muelling, A. Boularias, B. Mohler, B. Schoelkopf, and J. Peters. Learning strategies in table tennis using inverse reinforcement learning. *Biological Cybernetics*, under review.
- M. Muster. Lernende roboter: Wie ein roboter tischtennis lernen kann. *Trainerbrief des Verband Deutscher Tischtennistainer (VDTT)*, 2:8 – 15, 2013.
- J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaption of biped locomotion. *Robotics and Autonomous Systems (RAS)*, 47(2-3):79 – 91, 2004.
- A. Ng and X. Russel. Algorithms for inverse reinforcement learning. In *Proceedings of the 17th International Conference of Machine Learning*, pages 663 – 670, 2000.
- K. Ning, T. Kulvicius, M. Tamosiunaite, and F. Wörgötter. Accurate position and velocity control for trajectories based on dynamic movement primitives. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 5006–5011, 2011.
- D. Park, H. Hoffmann, P. Pastor, and S. Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *Proceedings of the 8th International Conference on Humanoid Robots (Humanoids)*, pages 91 –98, 2008.

-
- D. Perzanowski, A. Schultz, and W. Adams. Integrating natural language and gesture in a robotics domain. In *International Symposium of Intelligent Control*, pages 247 – 252, 1998.
- J. Peters and S. Schaal. Policy gradient methods for robotics. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 2219 – 2225, 2006.
- J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 05 2008a.
- J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 03 2008b.
- D. Pongas, A. Billard, and S. Schaal. Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In *Proceedings of the International Conference Intelligent Robots and Systems (IROS)*, pages 2911–2916, 2005.
- W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley& Sons, Inc, New York, NY, USA, 1 edition, 2011.
- W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical recipes: The Art of Scientific Computing*. Cambridge University Press, 3th edition, 2007.
- M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.
- D. Ramachandran and E. Amir. Bayesian inverse reinforcement learning. In *Proceedings of the 20th International Joint Conference of Artificial Intelligence (IJCAI)*, pages 2586 – 2591, 2007.
- M. Ramanantsoa and A. Durey. Towards a stroke construction model. *Int. Journal of Table Tennis Science*, 2(2):97–114, 1994.
- C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, USA, 2006.
- N. Ratliff, J. Bagnell, and M. Zinkevich. Maximum margin planning. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 729 – 736, 2006.
- D. Ricks and M. Colton. Trends and considerations in robot-assisted autism therapy. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 4354 – 4359, 2010.
- H. Ripoll and P. Fleurance. What does keeping one’s eye on the ball mean? *Ergonomics*, 31:1647–1654, 1988.
- S. Rodrigues, J. Vickers, and M. Williams. Head, eye and arm coordination in table tennis. *Journal of Sport Sciences*, 20:187–200, 2002.
- A. Roitman, S. G. Massaquoi, K. Takahashi, and T. Ebner. Kinematic analysis of manual tracking in monkeys: characterization of movement intermittencies during a circular tracking task. *Journal of Neurophysiology*, 91(2):901–911, 2004.
- S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233 – 242, 1999.
- S. Schaal. The SL simulation and real-time control software package. Technical report, University of Southern California, 2009.
- S. Schaal, C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparameteric statistics for real-time robot learning. *Applied Intelligence*, 17(1):49 – 60, 2002.

-
- S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, 358:537 – 547, 2003a.
- S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning motor primitives. In *International Symposium on Robotics Research (ISRR)*, pages 561 – 572, 2003b.
- S. Schaal, P. Mohajjerian, and A. Ijspeert. Dynamics systems vs. optimal control – a unifying view. *Progress in Brain Research*, 165(1):425 – 445, 2007.
- R. Schmidt. A schema theory of discrete motor skill learning theory. *Psychological Review*, 82(4): 225–260, 1975.
- R. Schmidt. The motor-action controversy. In O. Meijer and K. Roth, editors, *Complex movement behavior*, pages 3 – 44. Amsterdam: Elsevier, 1988.
- R. Schmidt. Motor schema theory after 27 years: Reflections and implications for a new theory. *Research Quarterly for Exercise and Sport*, 74(4):366–379, 2003.
- R. Schmidt and C. Wrisberg. *Motor Learning and Performance*. Human Kinetics, second edition, 2000.
- W. Schöllhorn. Practical consequences of systems dynamic approach to technique and strength training. *Acta Academiae Olympique Estonia*, 8:25 – 37, 2000.
- W. Schöllhorn. Coordination dynamics and its consequences on sport and exercise. *International Journal of Computer Science in Sports*, 2:40 – 46, 2003.
- A. Scott and E. Fong. *Body structures and functions*. Delmar Learning, Clifton Park, NY, 10th edition, 2004.
- C. Seve, J. Saury, S. Leblanc, and M. Durand. Course-of-action theory in table tennis: a qualitative analysis of the knowledge used by three elite players during matches. *Revue europeen de psychologie appliquee*, 55:145 –155, 2004.
- H. Simon. *The Shape of Automation: for Men and Management*. Harper & Row, New York, Evanston and London, 1965.
- H. W. Sorenson. *Kalman filtering: theory and application*. IEEE Press, Los Alamitos, CA, 1985.
- M. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, Inc., Hoboken, NY, 2006.
- R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, USA, 1998.
- M. Tamosiunaite, B. Nemeč, A. Ude, and F. Wörgötter. Learning to pour with a robot arm; combining goal and shape learning for dynamic movement primitives. *Robotics and Autonomous Systems*, 59: 910–922, 2011.
- G. Tesauro. Td-gammon, a self-teaching backgammon program achieves master-level play. *Neural Computation*, 6(2):215 – 219, 1994.
- The ALIZ-E project team. Adaptive strategies for sustainable long-term social interaction, April 2013. URL <http://www.aliz-e.org>.
- E. Todorov and M. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, 2002.

-
- D. Tyldesley and H. Whiting. Operational timing. *Journal of Human Movement Studies*, 1(4):172–177, 1975.
- A. Ude, A. Gams, T. Asfour, and J. Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800 – 815, 2010.
- Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement – minimum torque-change model. *Biological Cybernetics*, 61(2):89–101, 1989.
- J. Vis, W. Kusters, and A. Terroba. Tennis patterns: Player, match and beyond. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence*, 2010.
- J. Wang and N. Parameswaran. Analyzing tennis tactics from broadcasting tennis video clips. In *Proceedings of the 11th International Multimedia Modelling Conference*, pages 102 – 106, 2005.
- P. Wang, R. Cai, and S. Yang. A tennis video indexing approach through pattern discovery in interactive process. *Advances in Multimedia Information Processing*, 3331:59 – 56, 2004.
- Z. Wang, K. Muelling, M. Deisenroth, H. Ben-Amor, D. Vogt, B. Schoelkopf, and J. Peters. Probabilistic movement modeling for intention inference in human-robot interaction. *International Journal of Robotics Research (IJRR)*, 32(7):841 – 858, 2013.
- Wikipedia. Fosbury flop, August 2012. URL http://en.wikipedia.org/wiki/Fosbury_Flop.
- A. Williams and J. Starkes. Cognitive expertise and performance in interceptive actions. In *Interceptive Actions in Sport: Information and Movement*. Routledge Chapman & Hall, New York, NY, 2002.
- B. Williams, M. Toussaint, and A. Storkey. Modelling motion primitives and their timing in biologically executed movements. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1609–1616, 2008.
- D. Wolpert, C. Miall, and M. Kawato. Internal models in the cerebellum. *Trends in Cognitive Science*, 2: 338–347, 1998.
- R. Woodworth. The accuracy of voluntary movement. *Psychological Review*, 3(13):1–106, 1899.
- S. Yeo, M. Lesmana, N. D.R., and P. D. Eyecatch: Simulating visuomotor coordination for object interception. *ACM Transactions on Graphics (TOG)*, 31(4):42, 2012.
- D. Zhang, Z. an Xu and M. Tau. Visual measurements and prediction of ball trajectory for table tennis robot. *IEEE Transactions on Instrumentation and Measurements*, 59(12):3195–3205, 2010.
- S. Zhifei and E. Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293 – 311, 2012.
- B. Ziebart, A. Maas, A. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23th National Conference of Artificial Intelligence (AAAI)*, pages 1433 – 1438, 2008.
- B. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, A. Bagnell, M. Herbert, and S. Srinivasa. Planning based prediction for pedestrians. In *Proceedings of the International Conference on Intelligent Robotics and Systems (IROS)*, pages 3931 – 3936, 2009.

Appendix

Appendix A – Fifth order Polynomials

A fifth order polynomial is given by

$$\theta_k = \sum_{l=0}^5 \alpha_{kl} t^l, \quad (5.1)$$

where $\alpha_k = [\alpha_{k0}, \alpha_{k1}, \alpha_{k2}, \alpha_{k3}, \alpha_{k4}, \alpha_{k5}]^T$ are adjustable parameters and k denotes the DoF. The boundary conditions for the joint positions, velocities, and accelerations at the time points t_i and t_f are given by

$$\theta_k(t_i) = p_i, \quad \dot{\theta}_k(t_i) = v_i, \quad \ddot{\theta}_k(t_i) = a_i, \quad (5.2)$$

$$\theta_k(t_f) = p_f, \quad \dot{\theta}_k(t_f) = v_f, \quad \ddot{\theta}_k(t_f) = a_f, \quad (5.3)$$

where p_i, v_i, a_i, p_f, v_f and a_f are the joint angles, velocities, and accelerations at the time points t_i and t_f , respectively. With the linear equation system $\mathbf{M}\alpha = \mathbf{b}$ given by

$$\underbrace{\begin{bmatrix} 1 & t_i & t_i^2 & t_i^3 & t_i^4 & t_i^5 \\ 0 & 1 & 2t_i & 3t_i^2 & 4t_i^3 & 5t_i^4 \\ 0 & 0 & 2 & 6t_i & 12t_i^2 & 20t_i^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}}_{\mathbf{M}(t_i, t_f)} \underbrace{\begin{bmatrix} \alpha_{k0} \\ \alpha_{k1} \\ \alpha_{k2} \\ \alpha_{k3} \\ \alpha_{k4} \\ \alpha_{k5} \end{bmatrix}}_{\alpha_k} = \underbrace{\begin{bmatrix} p_i \\ v_i \\ a_i \\ p_f \\ v_f \\ a_f \end{bmatrix}}_{\mathbf{b}}, \quad (5.4)$$

we can efficiently solve for α using Gauss-Seidel elimination [Press et al., 2007].

Acceleration-minimizing Fifth Order Polynomials

For table tennis, the trajectories resulting from the 5th order polynomials can result in high accelerations that exceed the acceleration limits of the robot. Hence, we would like to find a final acceleration value a_f for the hitting stage such that the maximal acceleration is minimized. Therefore, we apply the following constraints to the trajectory

$$\theta_k(t_i) = p_i, \quad \dot{\theta}_k(t_i) = v_i, \quad \ddot{\theta}_k(t_i) = a_i, \quad (5.5)$$

$$\theta_k(t_f) = p_f, \quad \dot{\theta}_k(t_f) = v_f, \quad \ddot{\theta}_k(t_j) = 0, \quad (5.6)$$

where $\ddot{\theta}_k(t_j)$ is the jerk and t_j is an arbitrary time point with $t_i < t_j < t_f$ at which the acceleration is maximal. Given these constraints we can compute the coefficients α_{k0} to α_{k5} . Evaluating $\partial \ddot{\theta}(t) / \partial t_j = 0$, yields the solutions

$$t_j^1 = -\frac{(\sqrt{6}-4)t_i + (\sqrt{6}-6)t_f}{10}, \quad (5.7)$$

$$t_j^2 = \frac{(\sqrt{6}+4)t_i + (6-\sqrt{6})t_f}{10}. \quad (5.8)$$

The expression for t_j corresponds to the maximum near to the goal while t_j^2 corresponds to the maximum near the start position. The first solution for t_j is the one of interest, since updating the trajectory at the end of the stage can cause high acceleration peaks at this point. The resulting acceleration that minimizes the acceleration at $t_j = -0.1((\sqrt{6} - 4)t_i + (\sqrt{6} - 6)t_f)$ is given by

$$a_f = \frac{-(2\sqrt{6} - 3)(t_i - t_f)v_i + ((4\sqrt{6} - 1)(t_i - t_f)v_f + (4 - 6^{\frac{3}{2}}))(p_i - p_f)}{(\sqrt{6} + 1)(t_i - t_f)^2}. \quad (5.9)$$

This final acceleration does not just minimize the maximal acceleration but also the position overshoot at the end of the trajectory.

Appendix B - Cost regularized Kernel Regression

Cost-regularized Kernel Regression (CrKR), as suggested in [Kober et al., 2012b], is a reinforcement learning approach based on a kernelized version of reward-weighted regression. The goal is to find a mapping between the meta-parameter δ of a motor policy and a situation described by the state vector \mathbf{s} . Therefore, we are looking for a stochastic policy $\mu(\delta|\mathbf{s})$ that maximizes the expected reward

$$J(\mu) = \mathbb{E} \{r(\mathbf{s}, \delta)|\mu\}, \quad (5.10)$$

where $r(\mathbf{s}, \delta)$ denotes the reward following the selection of δ in state \mathbf{s} . The CrKR is based on the use of the Gaussian distribution with mean $\bar{\delta}$ and variance Σ for the policy μ , i.e., $\mu(\delta|\mathbf{s}) = \mathcal{N}(\delta|\bar{\delta}, \Sigma)$. The mean and the variance are computed by

$$\bar{\delta} = \mathbf{k}(\mathbf{s})^T (\mathbf{K} + \varphi \mathbf{C})^{-1} \mathbf{D} \quad (5.11)$$

$$\Sigma = k(\mathbf{s}, \mathbf{s}) + \varphi - \mathbf{k}(\mathbf{s})^T (\mathbf{K} + \varphi \mathbf{C})^{-1} \mathbf{k}(\mathbf{s}) \quad (5.12)$$

where \mathbf{D} is a matrix that contains the training examples δ_n^T , $\mathbf{C} = \text{diag}\{r_1^{-1}(\mathbf{x}_1, \mathbf{s}_1), \dots, r_N^{-1}(\mathbf{x}_N, \mathbf{s}_N)\}$ is the cost matrix, φ is a ridge factor, $\mathbf{k}(\mathbf{s})$ is a vector that measures the distance between the current state and the state of the training example using a Gaussian kernel and \mathbf{K} is a matrix that contains the pairwise distances between all combinations of training points. A detailed derivation of this policy can be found in [Kober et al., 2012b].

In this policy μ , the variance corresponds to the uncertainty over output δ and a high cost results in a high uncertainty. If we have a training example with a high cost, the policy will explore new actions and the training point will only have a strongly reduced influence. On the other hand, when the cost is low, we can assume that we are close to the optimal policy and the variance shrinks to a small region around the observed example.

Appendix C - Linear Bayesian Regression

Linear Bayesian Regression is a linear regression approach in the context of Bayesian inference. The following summary of the approach is based on Bishop [2006]. For more details we refer the reader to Chapter 2 of Rasmussen and Williams [2006] or Chapter 3 of Bishop [2006].

Assume we are interested in the target variable t which can be modeled by a function $y(\mathbf{x}, \boldsymbol{\lambda})$ with additive Gaussian noise such that

$$\begin{aligned} t &= y(\mathbf{x}, \boldsymbol{\lambda}) + \epsilon \\ &= \boldsymbol{\lambda}^T \boldsymbol{\phi}(\mathbf{x}) + \epsilon, \end{aligned}$$

where $\phi(\mathbf{x})$ is the vector of basis functions, $\boldsymbol{\lambda}$ is the vector of weights and ϵ is a zero mean Gaussian random variable with precision β . The conditional distribution of the target variable \mathbf{t} given \mathbf{x} , $\boldsymbol{\lambda}$ and β is given by the Gaussian distribution $p(\mathbf{t}|\mathbf{x}, \boldsymbol{\lambda}, \beta) = \mathcal{N}(\mathbf{t}|\boldsymbol{\lambda}^T \boldsymbol{\phi}(\mathbf{x}), \beta^{-1})$ [Bishop and Tipping, 2003]. Using the conjugate prior $p(\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\lambda}|\mathbf{m}_0, \mathbf{V}_0)$, the posterior distribution of the unknown parameter vector $\boldsymbol{\lambda}$ is again a Gaussian distribution and given by $p(\boldsymbol{\lambda}|\mathbf{t}) = \mathcal{N}(\boldsymbol{\lambda}|\mathbf{m}_N, \mathbf{V}_N)$, where

$$\begin{aligned}\mathbf{V}_N &= (\mathbf{V}_0^{-1} + \beta \boldsymbol{\phi}^T \boldsymbol{\phi})^{-1}, \\ \mathbf{m}_N &= \mathbf{V}_N (\mathbf{V}_0^{-1} \mathbf{m}_0 + \beta \boldsymbol{\Phi}^T \mathbf{t}),\end{aligned}$$

and \mathbf{m}_0 and \mathbf{V}_0 are the mean and covariance of the prior of $\boldsymbol{\lambda}$ [Bishop, 2006].

We can weight each observation similarly to a weighted least squares regression by assigning bad observations a higher variance than others. Therefore, we obtain $p(\mathbf{t}|\mathbf{x}, \boldsymbol{\lambda}, \beta) = \mathcal{N}(\mathbf{t}|\boldsymbol{\lambda}^T \boldsymbol{\phi}(\mathbf{x}), (\beta w)^{-1})$. Following the derivation in Bishop [2006], mean and covariance of the posterior distribution of $\boldsymbol{\lambda}$ are defined by

$$\begin{aligned}\mathbf{V}_N &= (\mathbf{V}_0^{-1} + \beta \boldsymbol{\phi}^T \mathbf{W} \boldsymbol{\phi})^{-1}, \\ \mathbf{m}_N &= \mathbf{V}_N (\mathbf{V}_0^{-1} \mathbf{m}_0 + \beta \boldsymbol{\Phi}^T \mathbf{W} \mathbf{t}),\end{aligned}$$

where \mathbf{W} is a diagonal matrix whose diagonal entries define the uncertainty of the observation. When considering the special case of a Gaussian prior with zero mean $p(\boldsymbol{\lambda}) = \mathcal{N}(\boldsymbol{\lambda}|\mathbf{0}, \alpha^{-1}\mathbf{I})$ [Bishop, 2006], the corresponding mean and covariance of the posterior distribution are given by

$$\begin{aligned}\mathbf{V}_N &= (\alpha \mathbf{I} + \beta \boldsymbol{\Phi}^T \mathbf{W} \boldsymbol{\Phi})^{-1}, \\ \mathbf{m}_N &= \beta \mathbf{V}_N \boldsymbol{\Phi}^T \mathbf{W} \mathbf{t}.\end{aligned}$$

In this case, the mean \mathbf{m}_N corresponds to the solution of the Weighted Ridge Regression.

Appendix D - Stability of Hitting DMPs

Here, we provide the mathematical proof of the stability and convergence of the extension of the DMPs presented in Kober et al. [2010] and in this paper. Please note that for the ease of reading the variables used in this section are not related to those with the same name elsewhere in this manuscript. The block diagrams for the second-order-system for the system is illustrated in Figure 5.1.

Stability of Hitting DMPs with Linear Velocity

If we assume a linear velocity of the target object as in the setup of Kober et al. [2010], the forward channel of the second-order-system expressed in Laplace space is given by

$$G(s) = \frac{Y(s)}{E(s)},$$

where $Y(s) = (AE(s) + BsE(s))/s^2$ is the state of our system, $U(s)$ is our desired position of the moving target, $E(s) = U(s) - Y(s)$ and $s \in \mathbb{C}$ is the (complex) frequency variable. The transfer function is given by

$$T(s) = \frac{Y(s)}{U(s)} = \frac{G(s)}{1 + G(s)} = \frac{A + Bs}{s^2 + Bs + A}. \quad (5.13)$$

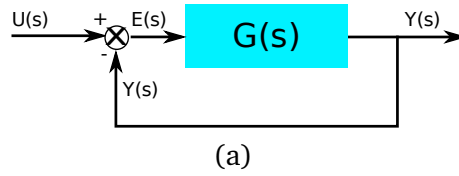


Figure 5.1: Diagrams of the employed second-order-system for the hitting DMPs in Laplace space.

Using $A = \omega^2$ and $B = 2\omega$ the system has a double pole at $-\omega$ which results in a critically damped response. The steady-state error can be calculated by

$$e_{ss} = \lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} sU(s)(1 - T(s)). \quad (5.14)$$

For a linear moving object $U(s) = V/s^2 + C/s$ and therefore we obtain

$$e_{ss} = 0. \quad (5.15)$$

The error's change rate is given by

$$\dot{e}_{ss} = 0. \quad (5.16)$$

Stability of Hitting DMPs with a Polynomial Velocity Profile

For the extension of the DMPs used and presented in this paper, the feed-forward term $\tau^2 \ddot{g}$ cancels all higher order terms, hence the hitting DMPs with polynomial velocity profile has the same stability properties as the hitting DMPs with linear velocity.

List of Figures

1.1	Overview of the thesis and its contributions. This figure shows the contributions of this thesis (red), how different aspects are connected with each other and their belonging to the different research fields.	5
1.2	Outline of the thesis. The thesis is divided into two parts: Modeling complex motor behavior and learning complex motor behavior. Learning complex table tennis can be further divided into selecting and generalizing striking movements and learning higher level strategies.	6
2.1	This figure illustrates the four movement stages of Ramanantsoa and Durey [1994] recorded in a VICON motion capture system where (a) shows the Awaiting Stage in which the opponent is observed, (b) the Preparation Stage in which the stroke is prepared, (c) the Hitting Stage in which the ball is intercepted, and (d) the Finishing Stage. The red and blue arrow show the movement of the ball and the racket, respectively, in each stage.	16
2.2	This figure shows different trajectories of intermediate table tennis players for one hitting motion. The trajectories are color coded according to the stages suggested by Ramanantsoa and Durey. The awaiting stage is colored in blue, the preparation stage in magenta, the hitting stage in green and the follow through stage in red. Colored circles show the corresponding position on the ball and arm trajectory respectively.	17
2.3	This figure illustrates the virtual hitting plane in the table tennis setup. The intersection point of the ball trajectory with the virtual hitting plane defines the virtual hitting point \mathbf{x}_{hp} . The x direction of the world coordinate system is parallel to the net, the y direction is parallel to the long side of the table and the z direction goes upwards.	18
2.4	Computation of the desired normal vector \mathbf{n}_{ed} of the end-effector and the velocity based on the orientation of the racket \mathbf{n}_{rd} , the velocity \mathbf{o} and \mathbf{i} of the ball after and before the impact respectively. Figure (a) illustrates the normal vector \mathbf{n}_e , the desired orientation of the racket \mathbf{n}_{rd} and the resulting desired orientation \mathbf{n}_{ed} . The normal of the end-effector \mathbf{n}_{ed} is perpendicular to \mathbf{n}_{rd} . Figure (b) shows the relationship between \mathbf{n}_{rd} , \mathbf{o} and \mathbf{i} . Assuming the absence of spin and a speed change only in the $\mathbf{o} - \mathbf{i}$ direction, \mathbf{n}_{rd} is given by the normalized difference vector of \mathbf{o} and \mathbf{i} . Figure (c) illustrates the computation of the velocity \mathbf{v} of the racket based on the relation of \mathbf{v} , \mathbf{n}_{rd} , \mathbf{o} and \mathbf{i}	20
2.5	This figure shows the movement of the racket and the ball on the real Barrett WAM for a successful striking movement. The ball (solid blue line) moves towards the robot until it is hit by the racket (dashed magenta line) at the virtual hitting point (black triangle). The y-axis is aligned with the long side of the table tennis table, the x-axis is aligned with the width of the table and the z-axis goes upwards.	24
2.6	The figure shows the different stages, matching those in Figure 2.1, but performed by the real robot. At the beginning the robot is in the rest posture waiting for an incoming ball (Subfigure a). As a ball moves towards the robot, the arm performs a backswing motion to prepare the striking movement (Subfigure b). Based on the prediction of the ball the system chooses a hitting point. When the estimated time to contact reaches a critical value the arm moves towards the hitting point and hits the ball back to the opponent (Subfigure c). The robot arm moves upwards with decreasing velocity until the velocity is zero (Subfigure d).	25
2.7	Various target of the biomimetic table tennis setup. The plane of possible hitting points has a length of 1.8 m.	26

3.1	General setup for learning a motor task using the Mixture of Motor Primitives (MoMP). A supervisory level creates the augmented state \tilde{s} containing the relevant information of the task based on the state of the system. MoMP selects and generalizes among the movement templates in a library according to the augmented state \tilde{s} which is provided by the supervisory level. As a result we obtain a new motor policy that can be executed. A teacher provides learning signals to the supervisory level as well as the movement generation level such that the system is able to adapt both the generation of the task relevant information and the generation of the movement.	33
3.2	The learning process in the presented MoMP framework. Given a set of demonstrations recorded by kinesthetic teach-in, the system generates the movement library, initializes the gating network as well as the estimation of the augmented states used in the MoMP. The resulting initial system can be used to perform a given motor task. During the execution, the gating network, the augmented states estimation and the primitives are further improved using reinforcement learning.	34
3.3	An illustration of the mixture of motor primitive framework. The gating network weights the single movement templates stored in a movement library based on an augmented state. The weighted sum of these primitives defines the new motor policy which produces the joint positions, velocities and accelerations for one degree of freedom. The resulting movement is then executed using a control law for execution which generates the required motor torques.	35
3.4	Changing the goal position and velocity is essential for adapting the demonstration to new situations. This figure illustrates how the different versions of the dynamical system based movement primitives are modulated by changing the goal position and velocity of the movement as they frequently occur in striking sports. The demonstration of a striking movement was obtained by kinesthetic teach-in in table tennis. After learning, all movement primitive formulations presented were able to reproduce the demonstration to a certain extend. However, the three formulations are differently robust against changes in the desired goal position and velocity. We changed the position by 0.15 m and the velocity by 0.4 m/s. The original formulation of Ijspeert is not able to reach the desired velocity as the system ends with zero velocity. The formulation of Kober et al. [2010] is able to adapt to a new final velocity. However, the accuracy of the adapted movement does not suffice for practical problems. The reformulation presented here reduces this inaccuracy drastically and stays closer to the desired movement shape.	38
3.5	A key problem of the previous movement primitive formulation is the highly uneven distributed accelerations with a peak at the beginning of the movement. Such jumps can affect the position and velocity drastically as shown in this figure. They may result in the attempt to generate infeasible trajectories for real robots. Kober et al. reduced this effect by gradually activating the attractor dynamics. However, if the initial movement amplitude is close to zero in a demonstration jumps will occur when the goal position is changed. By introducing a new scaling term for f , we can avoid jumps at the beginning of the movement and reduce overshooting in the position and velocity profile. The demonstration was obtained by kinesthetic teach-in of a striking movement. Both position and velocity were then changed to a new goal. Note, that the acceleration of the modified hitting primitives is so much smaller that it appears to be zero when compared to Kober's and Ijspeert's original versions.	39
3.6	Sequence of a hitting motion in table tennis demonstrated by a human teacher and reproduced with a Barrett WAM arm with seven DoF. From the left to the right the single pictures represent the system at the end of the awaiting, preparing, hitting and follow through stage respectively.	43

3.7	(a) Velocity error at the movement goal of the single movement primitives and the MoMP. The execution error of each movement primitive in the library was determined. The first bar of each DoF shows the mean error of the individual movement primitives and the corresponding standard deviations. The second bar of each DoF shows the mean error of the motor policies generated by the MoMP. (b) Improvement of the MoMP system with a growing number of movement primitives in the library.	44
3.8	The locations of ball-racket impacts during demonstration and evaluation on the real robot. (a) The ball-racket impacts during the evaluation with the fixed ball launcher. (b) The ball-racket impacts of the evaluation of the oscillating ball launcher.	45
3.9	Distribution on the hitting manifold (i.e., the part of the state-space that defines the ball-racket contacts) of the most dominant movement primitives before (a) and after (b) training. Each colored region corresponds to one movement primitive in the movement library. Each point corresponds to the point of impact during evaluation. The usage of the movement primitives changed after training the system with a ball launcher. While two movement primitives were relocated, three were avoided and replaced by two better suited primitives. Please note that we have displayed only a small excerpt of the served forehand area here in order to visualize the re-organization of a few primitives.	46
4.1	Considered scenario: A table tennis player (agent) plays a game of table tennis. At time point t , he has to decide how to return the approaching ball to the opponents court such that the chance of winning the point will increase. Returning the ball to a specific goal on the opponent's court (with a specific orientation and velocity) corresponds to an action \mathbf{a}_t executed by the agent. The player chooses this action based on his current state \mathbf{s}_t (Figure a). Due to this action, the system will transfer to the state \mathbf{s}_{t+1} defining a new situation for the player (Figure b).	50
4.2	Figure(a): The state of the system is defined by the relative position of the agent (d_{sx} , d_{sy}) and the the relative position (d_{ox} , d_{oy}) and velocity (\mathbf{v}_o) of the opponent towards the table, as well as the the position (d_{bx} , d_{by}) and velocity (\mathbf{v}_b) of the ball when bouncing on the table. Figure(b): In order to compute the table preferences on the opponent's court the table was divided into nine cells. Each cell was assigned a center (red points) \mathbf{c}_i	57
4.3	The bouncing angles α_y and α_z in the xy- and xz-surface define the orientation of the ball. While α_z corresponds to the horizontal bouncing angle, α_y corresponds to the direction of the ball and thereby defines if the ball is played cross to the left, cross to the right or straight.	58
4.4	Experimental setup. A naive player (right side) plays against a skilled opponent (left side). The upper body of both players, as well as the ball are tracked by a motion capture system.	59
4.5	Resulting parameter values for the individual features. Figure a shows the resulting reward function of the table preferences for Algorithm 7 (MM). Figure b shows the weights of all other features for Algorithm 7 (MM) and Algorithm 8 (RE), respectively. Figure c shows the differences of the average reward of the expert and the naive player for each feature separately using the reward function of the max-margin algorithm (green) and the relative entropy algorithm (yellow). Figure d shows the differences of the average rewards for the most important features at different time steps before the terminal state (win or loss) for the reward function yield with the max-margin algorithm.	63
4.6	Histogram of the average reward differences between the expert and sub-optimal players for each player and each feature individually. The reward function was received by the MMS algorithm with a horizon of three.	64

4.7	Possible strategy that distinguished the expert player that won the game, from the non-expert players that lost the game against the opponent. If the expert had the chance, he would play the ball very cross to the backhand area (Figure a). As a result the opponent was forced to move more into the left corner. The expert could then play the ball to the forehand area in order to increase the distance between the ball and the opponent (Figure b).	65
5.1	Diagrams of the employed second-order-system for the hitting DMPs in Laplace space. .	86

List of Algorithms

1	Table Tennis Algorithm	23
2	Generalizing Movements	36
3	Mixture of Motor Primitives (MoMP)	40
4	Imitation Learning of one DMP for MoMP	41
5	General IRL Algorithm	53
6	Maximum Margin for Game Values	54
7	Maximum Margin of States	55
8	Relative Entropy IRL Algorithm	56



List of Tables

2.1	This table shows a few robot table tennis systems as examples. Note that most systems include linear axes to achieve the necessary speed or have an additional stick mounted between racket and the robot's palm.	11
2.2	Root Square Mean Error in centimeters of the deviation of the applied dynamics model of the ball and the vision information of the ball 200 ms after the bounce. The velocity of the model was set to the estimate of the EKF before bouncing. The y direction corresponds to the side direction of the table, x direction corresponds to the long side of the table and z to the height (see Figure 2.3).	27
3.1	Variance of the velocity of the ball before impact in the different experiments of the evaluation. The maximal velocity v_{\max} of the ball before impact is defined by $v_{\max} = \sqrt{ \dot{x} ^2 + \dot{y} ^2 + \dot{z} ^2}$. All values are given in meters per second [m/s].	47
3.2	Analysis of the accuracy of the modified hitting primitives after both imitation and reinforcement learning. The right part of the table shows the changes in the desired final position and velocity compared to the values in the demonstration. On the left side of the table the corresponding mean errors (averaged over all DoFs) in the desired position and velocity are displayed at time T_f . The simulation analysis consisted of several configurations in which the motor policies yield through demonstrations were perturbed. For each configuration, either the final position, final velocity or both were modified using uniform value shifts between [-2,2] and [-1,1]. Please note that these position shifts exceed those which were observed on the real system.	47
4.1	Summary of the results of the evaluations for the different methods. The differences in the average rewards with respect to the expert, define the differences between the reward of the expert and the spared test subject of the non-expert data set. The feature of winning and loosing the game were not included. MMG corresponds to the model-free maximum-margin of game values, MMS corresponds to the model-free maximum margin of states values with an horizon of three and RE corresponds to the relative entropy method (see Section 4.2.2).	60
4.2	Summary of the results for the different horizons with the MMS algorithm. The differences in the average reward with respect to the expert trained with the different horizons H. The differences in the average reward directly before the terminal, define the differences of the reward of the expert and the spared test subject for the state before the terminal or the average reward of the two states before the terminal for the horizons 2 and 3 respectively.	62



Curriculum Vitæ

Education

- 12.2009 - now **Ph.D. Student**, supervised by Prof. Dr. Jan Peters
Max-Planck-Institute for Intelligent Systems and Technische Universität
Darmstadt
Topic: Modelling and Learning of Complex Motor Skills: Case studies
with Robot Table Tennis
- 10.2003 - 11.2009 **Diplom-Informatiker/Bioinformatiker**
Eberhard Karls Universität Tübingen, grade: sehr gut
- 06.2003 **Abitur**, Neues Friedlaender Gymnasium, grade: 1.2

Research and Teaching Experience

- 08.2007 - 11.2009 Undergraduate Research Assistant
Max-Planck-Institute for biological Cybernetics
- 10.2007 - 07.2008 Departmental Student Advisor, Informatik/Bioinformatik
Eberhard Karls Universität Tübingen
- 05.2007 - 08.2007 Lecture Assistant
Undergraduate course: Mathematik II fuer (Bio-)Informatiker
Eberhard Karls Universität Tübingen
- 10.2006 - 02.2007 Lecture Assistant
Undergraduate course: Mathematik I fuer (Bio-)Informatiker
Eberhard Karls Universität Tübingen
- 10.2005 - 02.2006 Lecture Assistant
Undergraduate course: Informatik 1
Eberhard Karls Universität Tübingen

Committee Work

- 11.2003 – 11.2009 Member of the Computer Science Student Senate (Fachschaft)
- 10.2007 – 09.2008 Student representative of the Examination Committee (Prüfungsausschuss), Fakultät für Informations- und Kognitionswissenschaften, Eberhard Karls Universität Tübingen
- 10.2004 – 09.2007 Student representative of the Curriculum Committee (Studienkommission), Fakultät für Informations- und Kognitionswissenschaften, Eberhard Karls Universität Tübingen
- 10.2005 – 09.2006 Student representative at Faculty Meetings (Fakultätsrat), Fakultät für Informations- und Kognitionswissenschaften, Eberhard Karls Universität Tübingen