Data-Driven Autonomous Manipulation

by

Peter Pastor

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements of the Degree
DOCTOR OF PHILOSOPHY
(Computer Science)

Committee:

Prof. Stefan Schaal (Chair)	Computer Science
Prof. Gaurav S. Sukhatme	Computer Science
Prof. Nicolas Schweighofer	Biokinesiology and Physical Therapy

Date of submission: Jan 31, 2014
Date of degree conferral: May 16, 2014

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Acknowledgements

The work presented in this thesis would not have been possible without the help and support of my advisor, committee members, colleagues, friends, and family, each of whom contributed in their own ways and at different stages in my academic journey. I therefore would like to take this opportunity to express my gratitude.

First and most of all, I want to thank my advisor **Stefan Schaal** for being nothing but the best advisor I could have hoped for. In the past six years, there was not a single moment in which I regretted my decision to join his research group to pursue my PhD studies. I am very grateful about the working environment that he was able to provide; I had the opportunity to work with amazingly driven people on high-profile projects such as LittleDog and ARM-S. Most of all, I very much enjoyed the scientific freedom of being able to pursue research on the topic of my choice. I am grateful for his guidance and encouragement which always ensured that my work was fruitful. I want to stress that these points raised are clearly not the norm and manifest the extraordinary opportunity that I was provided. I am also grateful for the valuable lessons that I have learned from him over the years outside of the lab, for example on our many (bottom less) coffee runs. Finally, I want to stress that many of the core ideas presented in this work (not so much the details) have been part of his lab's research agenda. Therefore, I decided to write this thesis in the "we" perspective. I am also thankful to my thesis committee member **Gaurav Sukhatme** and **Nicolas Schweighofer** for their feedback and support. I am also grateful to my qualifier committee members **Fei Sha** and **Laurent Itti** for their time and effort.

If getting a PhD is like getting married, then my colleague **Mrinal Kalakrishnan** was undoubtedly my best man. I very much enjoyed working with him for the past six years. It was truly an amazing learning experience. I am thankful for his spot-on answers to all my questions and likewise for asking the right kind of questions to me (also known as "playing the bad cop"). I very much enjoyed the long hours and late nights during the third phase of LittleDog and both phases of the ARM-S project. Similarly, I am thankful to **Ludovic Righetti** for many fruitful discussions and especially for the collaboration that resulted in our IROS 2011 paper. I am grateful for his contributions to the control framework during the first phase of the ARM-S project. Likewise, I am thankful to **Jonathan Binney** and **Jonathan Kelly** for their work on perception and calibration during the first project phase. I am thankful to **Jonas Buchli** for the fun times with

LittleDog, **Evangelos Theodorous** for the lectures in control theory, and **Freek Stulp** for his support and feedback. I am thankful to **Heiko Hoffmann** for the collaboration on the movement primitive formulation. I am thankful to **Jun Morimoto** for hosting my visit to ATR in Japan and I am thankful to **Erhan Oztop**, **Sang-Ho Hyon**, **Jun Nakanishi**, and **Emre Ugur** for having made my stay as fun as it was. I am very thankful to **Sachin Chitta** for being my mentor during my two internships at Willow Garage. I also want to acknowledge my former supervisor and long term friend **Tamim Asfour** for paving the way for me to pursue my PhD at the Computational Learning and Motor Control (CLMC) lab.

Last but not least, I want to thank the rest of the CLMC family - **Franziska Meier**, **Vince Enachescu**, **Bharath Sankaran**, **Nick Rotella**, and **Sean Mason** - for the little assists and for the friendly atmosphere in the lab. Similarly, I am thankful to the people in the Autonomous Motion Department (AMD) in Tübingen, especially to **Manuel Wüthrich** for his work on object tracking, which was crucial to our success in the second phase of the ARM-S project, **Daniel Kappler** for his work on sensor prediction as well as his contributions to the second phase of the ARM-S project, **Alexander Herzog** for his work on grasp planning, and **Jeannette Bohg** for her work on 3D perception. I very much enjoyed the fruitful discussions as well as the fun times at conferences and retreats.

Finally, I am thankful for those people who ensured that I got my mind off robotics problems every now and then. To name a few explicitly, these were **Jnaneshwar Das** aka **JD**, **Tim Doppelgänger**, my burning man camp, and my roomies in Venice - **Don Lopez**, **Manuel Griffin**, **Mark Sparkwave**, and **Kobe (Drew) Bryant**.

Dedication

To my parents,
Adelheid and Miguel,
and to my brother and sister,
Daniel and Heidi.

Table of Contents

List of Tables

List of Figures

Abstract

The problem of an aging society is real and will affect everyone. There will be too few young people that can ensure adequate living conditions for the elderly. Personal robots have the potential to assist in day-to-day tasks whenever there are too few humans to cope with societal needs. However, for personal robots to become useful they need to be able to skillfully manipulate objects in their environment. Unfortunately, the problem of autonomous manipulation is very complex and progress towards creating autonomous behaviors seems to have reached a plateau.

In this thesis, we will present a data-driven approach to movement generation. We argue that movement generation (motor output) and perceptual processing (sensor input) are inseparably intertwined and that the ability to predict sensor information is essential for skillful manipulation. Movement generation without sensor expectations defaults to open-loop execution which is prone to failure in dynamic and unstructured environments. However, predicting sensor information for an increasing number of sensor modalities including force/torque and tactile feedback through physics based modelling is challenging given the variety of objects, the diversity of possible manipulation behaviors, and the uncertainty in the real world. Instead, our approach leverages a key insight: Movement generation can dictate expected sensor feedback. Similar manipulation movements will give rise to sensory events that are similar to previous ones. Thus, stereotypical movements facilitate to associate and accumulate sensor information from past trials and use these sensor experiences to predict sensor feedback in future trials.

We will call such movements augmented with associated sensor information Associative Skill Memories (ASMs). We will present a coherent data-driven framework for manipulation that implements this paradigm. First, we will introduce a modular movement representation suitable to encode movements along with associated sensor experiences. Second, we will show how stereotypical movements can be learned from demonstrations and refined using trial-and-error learning. Third, we will show how ASMs can be used to monitor task progress, to realize contact reactive manipulation, and to purposefully choose subsequent movements. Finally, we will present a method that can learn forward models for these stereotypical movements.

Chapter 1

Introduction

Robots capable of safely operating among humans will become very beneficial for society. When facing the problem of an aging society as it is the case in many countries around the world, especially in Asia, and Japan in particular, such robots could provide much needed assistance (Yamazaki et al., 2012). For example, robots could function as personal assistants for the elderly and take care of many day-to-day tasks. With round-the-clock assistance, such robots could facilitate an independent lifestyle and allow the elderly to remain in their homes, rather than being forced to enter unfamiliar nursing facilities. In fact, a recent study has shown that an elderly person may even prefer assistance from a robot for particular day-to-day tasks over assistance from a human (Mitzner et al., 2011). Besides elderly care, capable and human-centered robots can become beneficial for the society in a wide spectrum of possible scenarios (Schaal, 2007), ranging from playmate robots in child education to personalized rehabilitation trainer for stroke patients. They can become part of our life, taking over functions where there are too few humans to cope with societal needs. Nevertheless, as of today, robots equipped with necessary manipulation capabilities have exclusively been utilized in the automation industry. Up to now, they have typically been used in well-structured environments like assembly lines, where they are programmed by experts to do a variety of repetitive tasks.

1.1 Challenges in Autonomous Robotic Manipulation

The research area of autonomous robotics is still fairly young. Yet, tremendous achievements can be reported in various disciplines: Cars can drive autonomously through urban environments (Urmson et al., 2008; Montemerlo et al., 2008), quadruped robots can walk autonomously over difficult terrain (Wooden et al., 2010) (Kalakrishnan, Buchli, Pastor, Mistry, & Schaal, 2010, 2011), even a car sized robot was landed autonomously on Mars (Bell, 2012). Comparatively, the progress in autonomous grasping and manipulation in uncertain and dynamic environments remains minor. Unfortunately, exactly this ability, to physically interact with objects in an unstructured environment, is an important

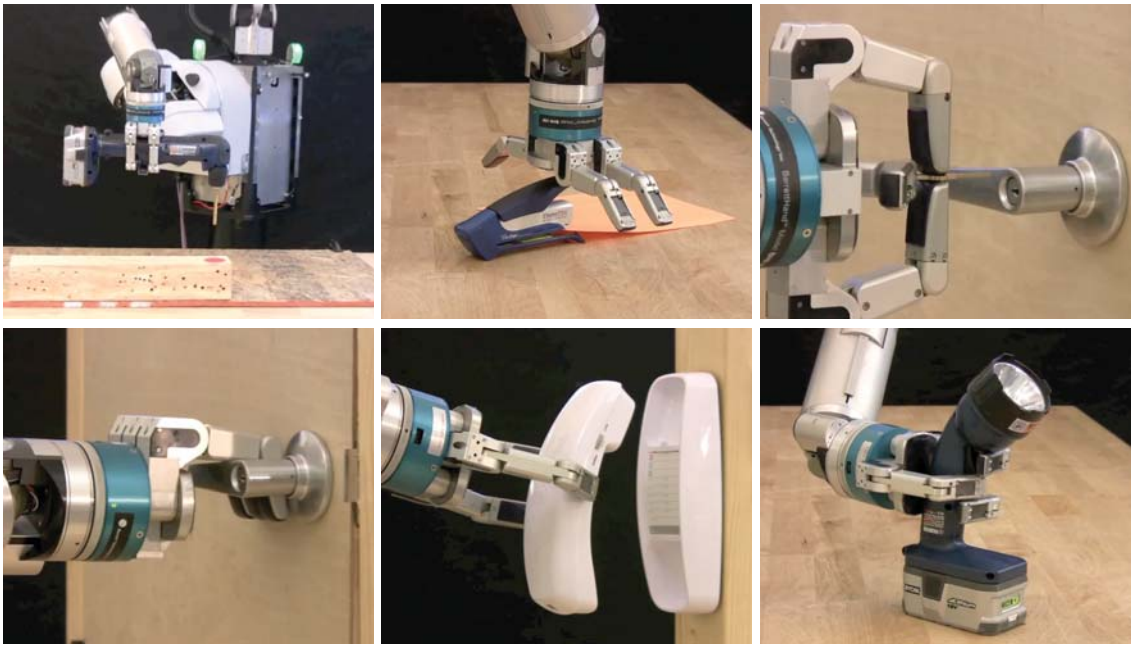


Figure 1.1: The six manipulation tasks considered during the first phase of the ARM-S project: Drilling, stapling, unlocking a door, opening a door, hanging up a phone, and turing on a flashlight. The corresponding video is available at `http://youtu.be/VgKoX3RuvB0`.

prerequisite to enable personal robots. Despite major research efforts, the capabilities of today's robots in terms of (even) simple manipulation behaviors are nowhere near those of humans. This gives rise to the question of why this is the case. How come we managed to successfully send a car sized robot to Mars and have it land autonomously in a single trial, yet we have not managed to build a robot that can reliably perform simple manipulation tasks on a table top? To be fair, that is not quite true. There are a indeed several considerable achievements that show that autonomous interaction with objects and the environment is possible. There are robots that can fetch drinks from a fridge and load a dishwasher (Asfour et al., 2006), can make pancakes (Beetz et al., 2011), and even can do laundry (Maitin-Shepard et al., 2010). Nevertheless, as in many other contributions that show experimental results on a real system, the amount of true autonomy is questionable. According to the Oxford dictionary, the definition of the word autonomy is "freedom from external control or influence; independence". Given this definition, one can develop a scale that quantifies the degree of autonomy of a particular system: A completely autonomous system can take any action at any instant of time *free from external control*, a completely non-autonomous system strictly follows pre-determined actions without making any decision at any time. Systems with a lower degree of autonomy rely on structured environments, typical examples are the previously mentioned factory robots. Open-loop

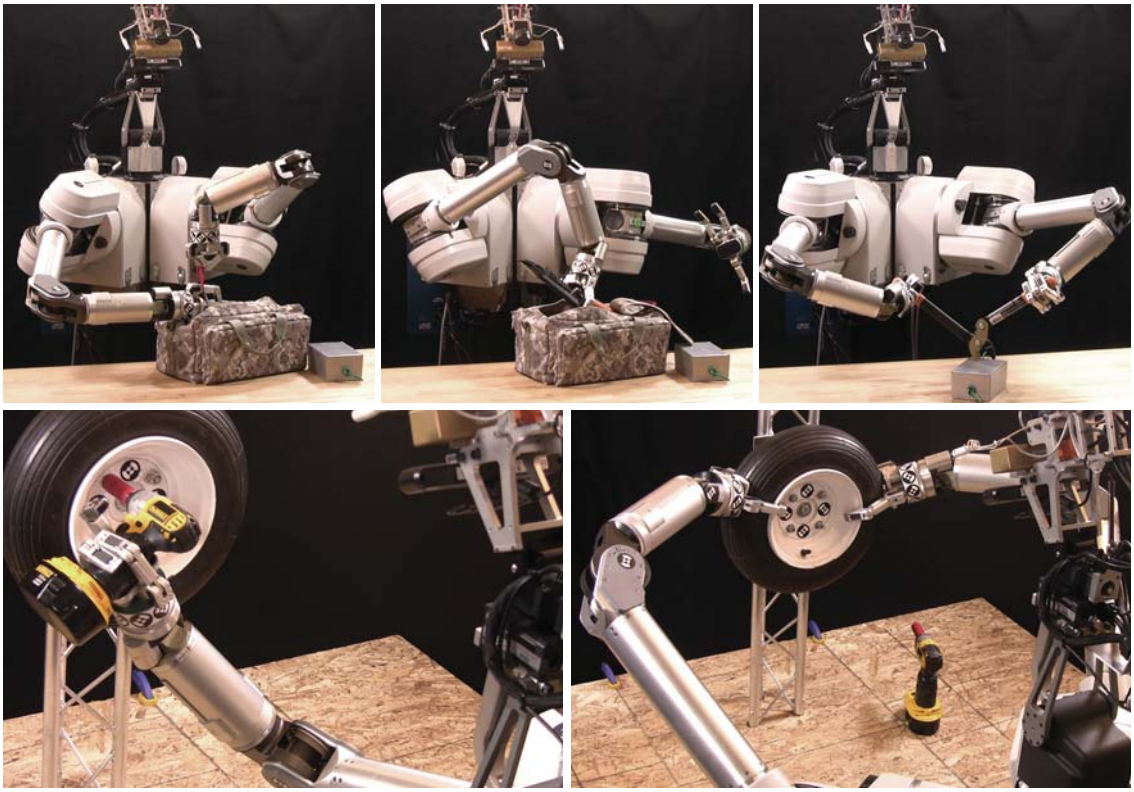


Figure 1.2: The two manipulation scenarios considered during the second phase of the ARM-S project: Opening a bag, retrieving a cutter, and cutting a cable (top) and unscrewing a set of screws and removing a tire (bottom). The corresponding videos are available at `http://youtu.be/TEk4DXuFcyc` and at `http://youtu.be/26mt6j-EMYM`.

execution* is prone to fail in unstructured and uncertain environments simply because the robot cannot react and adapt its behavior. While open-loop task executions may appear very impressive whenever the task succeeds, the necessity of autonomy becomes apparent whenever slight changes in the initial conditions lead to task failures. Thoughtful hardware design can achieve robust performances for particular tasks despite ignoring sensor feedback (Dollar, Jentoft, Gao, & Howe, 2010; Deimel & Brock, 2013), however, these approaches are also very limited in the variety of tasks they can perform.

The research challenge termed Autonomous Robotic Manipulation (ARM) was specifically setup to push these boundaries of autonomy. In the first of two phases, six teams located within the US were given the same experimental platform and were asked to provide software enabling it to accomplish 12 grasping and 6 manipulation tasks (see Fig. 1.1). The emphasis of the project was to evaluate the teams' performances on the same robot at a remote location and without any control over the poses of the relevant objects on the

*For the context of this thesis, open-loop execution refers to ignoring external sensor measurements. For this purpose, joint encoder readings are not considered external sensors rendering standard joint control loops (e.g. PID) not to be considered closed-loop.

table. Thus, teams were forced to develop approaches that are robust to slight mechanical and environmental differences and that can successfully perform these tasks independently of how and where the objects were placed on the table. The first phase of the project yielded considerable results (Hudson et al., 2014; Righetti et al., 2014; Bagnell et al., 2012). Three teams successfully completed at least 64 of the 72 grasping and manipulation tasks. In the second phase of the project, the top three teams from the first phase, were provided with a second arm and the focus shifted towards longer and more complex manipulation sequences. The two considered scenarios were changing a tire and cutting a cable with a pair of pliers hidden inside a bag (see Fig. 1.2). However, the degree of autonomy in the presented solutions was again rather low. After all, the environments were fairly structured, usually a single object on a table. Yet, two interesting approaches have proven to be successful. First, using visual feedback to estimate the location of the hand in camera coordinates, and second, deliberately interacting with the table and the object itself to further minimize uncertainty. Note that both these approaches increase autonomy: the system uses external sensor information and adapts its behavior accordingly. Unfortunately, the developed solutions were often times engineered and tailored to work really well for particular manipulation tasks leveraging some regularities of the environment and/or the robot itself. Although teams seemed to have developed a single

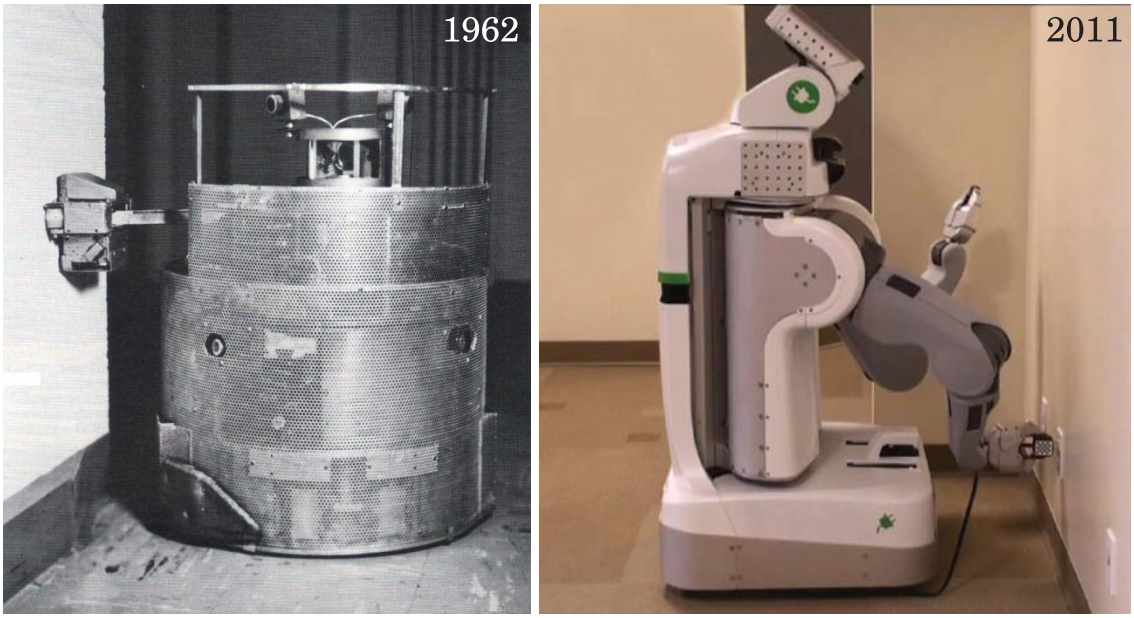


Figure 1.3: Robots plugging itself in: The "Beast" was developed in the Appplied Physics Lab at Johns Hopkins University in 1962 (left). It was controlled by dozens of transistors and able to wander around in white hallways using sonar. It was further able to find black wall outlets using special photocell optics and plug itself in by feel with its special recharging arm. The PR2 robot was developed by Willow Garage (right). In 2011 (almost fifty years after the Beast) it was able to travel 138.9km in 13 days without human intervention navigating in an office environment and recharging itself whenever needed.

behavior that achieved most of the grasping tasks, the manipulation skills usually lacked generality and were developed independently of each other. Such *human-in-the-loop* trial-and-error approaches to find suitable task/control parameters have been common practise already fifty years ago (see for example Fig. 1.3). Therefore, we have all the reasons to ask ourselves whether significant progress has been made in the discipline of autonomous manipulation. Advances in hardware design and computational capacities are without question, however advances towards more autonomy are less obvious.

These observations are not a criticism of the research conducted in the past fifty years, but to illustrate the difficulty of the problem. Autonomous manipulation is "extremely high-dimensional, inherently complex, and riddled with uncertainty" (Brock, 2011). In fact, it might be harder than many researchers (even within the robotics community) have assumed or are willing to accept. The complexity of many tasks is due to the objects to be manipulated. Even when dealing with very simple non-articulated objects, there can be many object properties that need to be considered to successfully perform the task at hand. Many of these object properties (e.g. pose, shape, inertia, friction coefficients) can only be inferred from noisy and incomplete sensor data. Modifying the state (e.g. its pose) of the object can only be achieved through (complex) contact interactions. Thus, autonomous robotic manipulation is inherently interdisciplinary: to enable a robot to perform even simple tasks requires tight integration of many different research fields including computer vision, control theory, machine learning, and algorithms. These findings, and potentially many more (Brock, 2011), may explain the relatively minor progress in the research field of autonomous manipulation. The intention of this work is to identify potential limitations of previous approaches and provide a novel perspective on ways to overcome these barriers.

1.2 Motivation

The aim of this thesis is to develop a data-driven framework for autonomous grasping and manipulation (see Fig. 1.1 and Fig. 1.4).

Figure 1.4: The Barrett WAM arm grasping a ball, a shovel, a pelican case, a rock, and a floodlight.

To motivate the data-driven nature of our approach to autonomous manipulation, we first will illustrate the current paradigm in 1.2.1 and discuss limitations of model-based approaches in 1.2.2. Alongside, we will identify a set of requirements that we think are important to realize autonomous manipulation. In particular, in 1.2.3, we show how to cast robotic manipulation into a memory-based approach to address limitations of model-based approaches. We will stress the necessity of the approach's ability to quickly learn new skills in 1.2.4, and the importance of closing feedback control loops in 1.2.5. Finally, we will motivate the need for a data-driven approach in 1.2.6.

1.2.1 The Current Paradigm

Current state-of-the-art approaches in robotic grasping (Srinivasa et al., 2010; Chitta et al., 2012)* are usually composed from the following four components: (1) visual scene understanding including object recognition and pose estimation, (2) methods to compute a goal grasp configuration, (3) motion planning algorithms to generate a collision free trajectory from the current configuration to this goal configuration, and (4) controllers that track this trajectory (see Fig. 1.5). The advantage of such processing pipelines is that

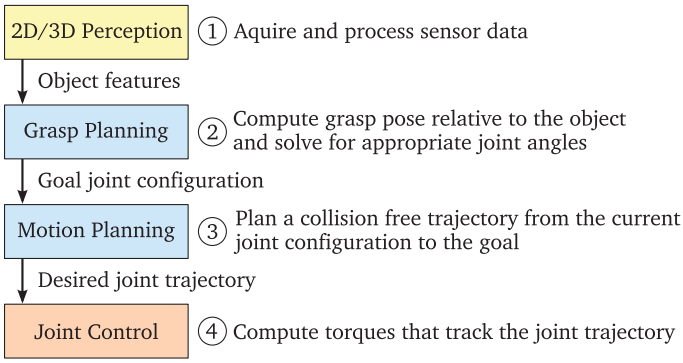


Figure 1.5: Simplified sense-plan-act process diagram used for grasping: (1) Visual sensor information is processed to compute feasible grasp poses. (2) The best grasp hypothesis is chosen and inverse kinematics is used to compute corresponding joint angles. (3) A motion planner is invoked to compute a collision free joint trajectory. (4) The resulting joint trajectory is passed to the joint controller to compute appropriate torques. Note that the diagram shown is a simplified version of Fig. 2 in (Srinivasa et al., 2010), Fig. 2 in (Chitta et al., 2012), and to some extend of Fig. 2 in (Righetti et al., 2014).

they are straight forward to implement and to debug. Well defined interfaces between each software module (visual perception, grasp planning, motion planning, and control) allow

*These approaches have been chosen because they have shown extensive experimental results on integrated systems and constitute a complete approach to autonomous grasping and manipulation. We do not intend to downgrade these achievements. Instead use the obtained results and valuable lessons learned to identify limitations to motivate our design decisions. There are potentially many other approaches that follow a similar concept and could have been listed instead.

to first solve each sub-problem in isolation and then integrate these solutions (execute one after the other) to solve the original problem. However, such *sense-plan-act* processing pipelines have a number of limitations which will be discussed in the following.

The very same software interfaces that allow for modularization also limit the systems flexibility and adaptability. Specifying the inputs and outputs of each of these modules pre-define the interactions between them. The process flow is usually unidirectional and the information bandwidth at these interfaces is rather low (see Fig. 1.5). Nevertheless, surprisingly these interfaces have become accepted and established in some research communities: Research in grasp planning usually revolves around how to compute feasible grasp pose hypothesis* given visual sensor data. Research in motion planning usually evolves around how to compute collision free trajectories from the initial joint configuration to the goal. Both these research communities have shown great progress in addressing exactly these problems, e.g. (Herzog et al., 2012, 2014; Kalakrishnan, Chitta, Theodorou, Pastor, & Schaal, 2011), however, it remains questionable how much these approaches are applicable to unstructured and uncertain environments. For example, objects can be partially occluded (e.g. due to clutter) or even fully occluded (e.g. inside a gym bag) which imposes major challenges for most grasp selection algorithms. Yet, stable grasps might still be achievable through deliberate contact interactions with the object and its environment (Hudson et al., 2012, 2014; Righetti et al., 2014). Similarly, motion planning algorithms usually generate the entire trajectory from the start to the goal using batch processing according to the current collision model of the environment. Thus, such methods rely on static environments, where neither moving obstacles can appear nor the goal of the movements can change online. Most of these kinds of limitations (and potentially many more) come from the fact that each of these four processing steps are usually handled in isolation.

1.2.2 Limitations of Model-based Approaches

Arguably, the most interesting aspect of grasping and manipulation is the moment when the robot actually makes contact with the object and begins manipulating it. Note that this has not been addressed yet in Fig. 1.5 since motion planning is trying to do the exact opposite, which is to avoid contacts with objects. Instead, after reaching the pre-grasp pose (after step 4 in Fig. 1.5), a Cartesian controller is used to move the hand the remaining distance to the object.

To use motion planning algorithms to generate plans that include contacts would require (precise) dynamics models† of the task-relevant objects and their environment. Estimating such parameters is very challenging, especially considering the vast amount of potential

*Grasp pose hypotheses are usually defined as object relative hand positions and orientations along with finger joint configurations.

†The dynamics model might include geometric information, inertia parameters, and friction coefficients.

objects. In fact, the process of estimating the dynamics parameters of the robot's own manipulator might still require inspections from an expert. Nevertheless, even assuming perfect knowledge about the world and assuming that planners can handle the extra degrees of freedom introduced by the objects being manipulated, it has yet to be shown that the generated plans will actually yield good performance on a real system. One of the few approaches that use motion planning to plan contacts with an object for the purpose of grasping or pushing it (Dogar & Srinivasa, 2010), uses very simplified assumptions: the object needs to be on a horizontal plane, it needs to be movable, it needs to be circular, it needs to be approachable from the side, and it should not fall over if pushed from the side. Therefore, it seems that modelling the physical interactions for the purpose of planning is only suitable for very simple manipulation tasks in very structured environments. Nevertheless, whenever contact interactions with the environment are planar/ideal and especially when the dynamics of the task are dominated by the robot itself, model-based approaches have shown good performances (Stilman, 2010; Lengagne, Vaillant, Yoshida, & Kheddar, 2013). However, as contact interactions become more complex, as it is the case in grasping and object manipulation, modeling errors are inevitable and will significantly degrade the performance of the planner (Weisz & Allen, 2012; Balasubramanian, Xu, Brook, Smith, & Matsuoka, 2012). The difficulty of modelling complex contact interactions of the robot's endeffector with objects and its environment is due to the fact that two very similar initial conditions can lead to two very different final states. Thus, the uncertainty about the manipulated object increases exponentially fast and therefore becomes intractable unless smart heuristics/biases (Brock, 2011) have been introduced (Dogar & Srinivasa, 2010).

An interesting alternative to model-based approaches, which require an explicit model of the world, is to follow ideas of morphological computation (Pfeifer & Iida, 2005) and acquire an implicit model of the world through experience. The idea is to let the physics of the real world "compute" the complex contact interactions between the mechanism of the robot and its environment (Eppner & Brock, 2013) and memorize the experienced sensor feedback. Provided similar task executions, these sensor experiences can become a robot-centric implicit model that allow to predict subsequent executions (Pastor, Kalakrishnan, Chitta, Theodorou, & Schaal, 2011; Pastor, Righetti, Kalakrishnan, & Schaal, 2011). We will discuss how to cast manipulation into such a memory-based approach in the following section.

1.2.3 Enabling Memory-based Approaches

An important observation is that for subsequent grasp attempts the processing pipeline (see Fig. 1.5) is simply repeated, i.e. each attempt is treated as a unique problem that is solved on its own. Information from previous trials is often discarded. Although not as obvious as on the factory floor, there is also a high degree of repetition in the real world. For example, day to day manipulation tasks in a kitchen environment show highly repetitive movement behaviors (Tenorth, Bandouch, & Beetz, 2009). Similarly, in

almost all activities of daily living, related tasks are encountered over and over again. We open doors, grasp cups, or pour liquid dozens of times a day, i.e. we "tend to solve similar or even identical instances over and over, so we can keep recycling old solutions with minor modifications" (Horswill, 1993). This observation suggests modularization of complex behaviors into primitives similar to the decomposition of speech and language into phonemes and words. Composing complete behaviors from simpler elements seems to be a promising approach for robotics applications to deal with the complexity in visual perception, action, and cognition (Flash & Hochner, 2005). A robot that is equipped with a rich repertoire (vocabulary) of basic skills alleviates the need to generate solutions every single time a certain task is encountered. Instead, the robot could reuse pre-computed solutions in form of movement primitives.

The idea of using movement primitives for robotics has been inspired by neuroscience, in particular, human motor control. In this community these movement primitives have been termed "motor schemas" (Arbib, 1992), "building blocks of movement" (Ghahramani, 2000), "basic unit of actions" (Buchanan et al., 2003), "action-phase controller" (Johansson & Flanagan, 2009a), and "neural control modules" (Wolpert et al., 2011). While neuroscientists are interested in understanding human motor control, and in doing so are developing models that can closely explain experimental data, the challenges for developing a movement representation suitable for robotics are different. In robotics, the movement representation needs to be able to encode arbitrarily complex movements. Nevertheless, the generated movement needs to be sufficiently smooth to be suitable for execution on a robot. Importantly, to generate complex behaviors through sequencing several movement primitives, the encoding scheme should guarantee smooth transitions (i.e. continuous velocity or acceleration profiles) when switching between subsequent movement primitives. Note, these transitions between individual movement primitives may occur at any moment of time, not just at the end, and may need to happen instantly. Furthermore, the movement representation needs to be spatially and temporally invariant, i.e. encode movements such that they become invariant to changes of the start and goal position as well as its duration. To facilitate reusability of movement primitives, the encoding scheme needs to generalize locally. That is, despite changes in the relative position between start and goal, the representation should be able to generate appropriate movements that initiate from the start and converge at the goal. The representation should further guarantee convergence to the goal even if the goal changes after movement onset. The ability to instantly adapt the generated movement online is crucial especially when dealing with dynamic environments. Finally, and most importantly, the movement representation needs to offer mechanisms to modulate the generated trajectories from external signals, for example to realize (moving) obstacle avoidance. All these features constitute a powerful model to encode movements.

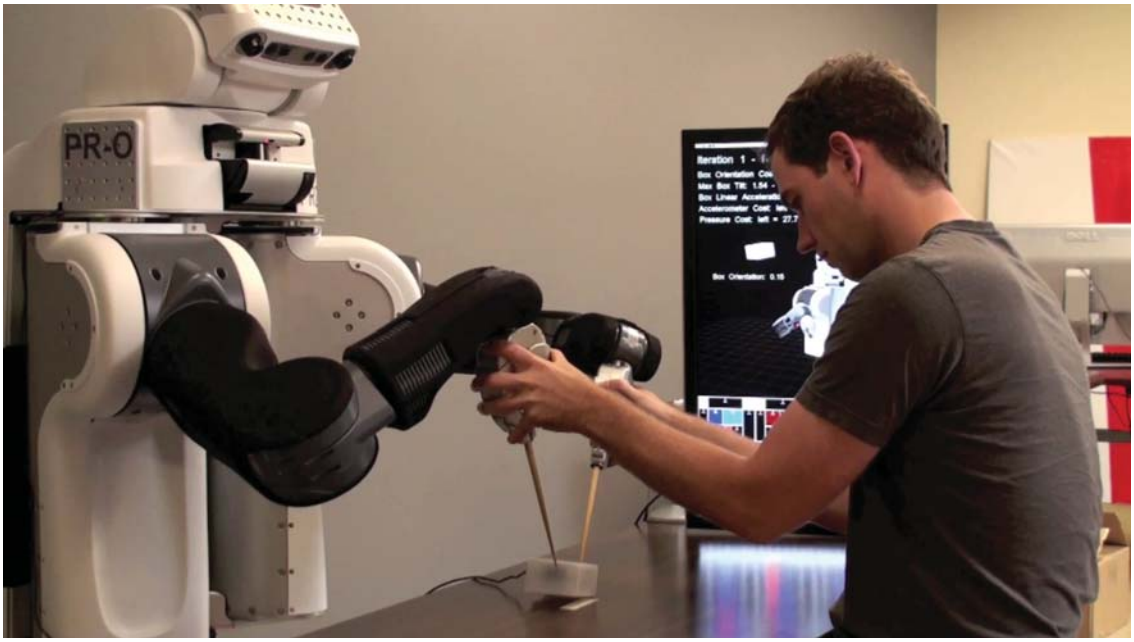


Figure 1.6: Kinesthetic teaching: The human teacher guides the robot to perform a manipulation task, here flipping a box with chop sticks.

1.2.4 Importance of Bootstrapping Manipulation Skills

Another important feature of a movement representation for robotics is that new movement primitives should be learnable from data, i.e. example trajectories. A promising and straight forward approach to obtain these example trajectories is learning from demonstration. A human teacher demonstrates the manipulation behavior once and the robot learns this particular behavior (see Fig. 1.6). Note that the learned manipulation behavior is specific to the task at hand and the objects involved, however, the process of teaching this skill is very general, i.e. independent of the task and the objects involved. This ability to teach a robot a set of skills simply by demonstrating it once provides an intuitive and much needed way of adding prior knowledge to the system. As previously discussed, given the complexity and the number of possible manipulation tasks, it seems to be infeasible to create algorithms that can generate appropriate behaviors from scratch. Instead, the ability to bias the system towards a particular solution, for example using demonstrations, and to use this bias to guide the search becomes crucial (Schaal, 1999).

Complex manipulation skills, like flipping a box with chopsticks (see Fig. 1.6), are difficult to learn from a single example trajectory. Imperfect demonstration, missing information about the applied forces, and/or the stochastic nature of the task itself may result in unsuccessful task executions. Nevertheless, provided a task specific cost function, this initial solution can be used for trial-and-error learning. That is, the robot could explore the space around the initial demonstration, i.e. the imposed bias, while optimizing the reward function to improve task performance.

1.2.5 Importance of Perception-Action Loops

Overall, we have identified the requirements for flexible movement representation that can be initialized from demonstration and improved over time using reinforcement learning. However, using a different encoding scheme and different methods to obtain trajectory data alone does not increase the amount of autonomy of a system. Even if such an approach is capable of learning a rather complex manipulation task, the output is after all an open-loop trajectory which determines the action sequence of the robot for every time step. To account for unstructured and dynamic environments and therefore to achieve true autonomy, it is important to sense and control at the same time. Approaches that follow the sense-plan-act paradigm (see Fig. 1.5) and process each module in sequence and in isolation from the others are limited to only work well for the domain they have been designed for. This insight is not novel, yet the complexity of completely integrated and autonomous manipulation systems have forced researchers to break down the problem into separate entities each of which is executed one at a time. Even the researchers themselves have admitted/reported the limitations of such an approach (Bohren et al., 2011; Srinivasa et al., 2012; Righetti et al., 2014).

Interestingly, closing feedback loops using external sensors has been an integral part in many contributions in the research field of autonomous manipulation ever since its beginning. Visual servoing for example is a prominent approach to close the loop between (visual) perception and action (Hutchinson, Hager, & Corke, 1996; Kragic & Christensen, 2003). To reduce uncertainty in the endeffector pose, the system simultaneously track in camera coordinates a reference point on the robot itself (e.g. the tool center point) and a reference point in the environment (e.g. the object of interest) and generate appropriate control commands to align them visually. An impressive demonstration of the power of closing such perception-action control loops in the context of manipulation has been reported in (Furukawa, Namiki, Senoo, & Ishikawa, 2006). Similarly, approaches have been proposed that incorporate tactile and force feedback to account for positional uncertainty of the object, (Platt, 2007; Hsiao, 2009; Jain, Killpack, Edsinger, & Kemp, 2013). Even auditory feedback has been used successfully to recognize objects (Sinapov et al., 2011). Each of these contributions use external sensor signals (visual, haptic, auditory) to determine subsequent actions *free of external control*, thus increase autonomy. Note that each of these methods close a perception-action loop and therefore require a reference signal for each of the sensor measurements for comparison. These reference sensations are needed to compute the error in camera coordinates in case of visual servoing, to determine whether or not contact has been established in case of contact-reactive grasping, or to correctly classify objects based on sound. Furthermore, these reference signals are spatially and temporally coupled with the particular robot movement and change depending on the interactions between the robot and its environment, e.g. touching an object with different parts of the hand will result in different haptic feedback, similarly, lifting a lighter object

will result in less force feedback as opposed to lifting a heavy one. Computing these reference sensations for a particular manipulation task seems (again) infeasible* especially considering the amount of sensors modern robots are (and will be) equipped with. Therefore, researchers have defaulted to manually specifying points of interest (visual servoing), determining force thresholds (contact-reactive grasping), and labeling object classes (auditory classification). Unfortunately, these parameters are only valid for the task they have been tuned for. Furthermore, for dexterous object manipulation for example it is crucial to obtain force profile predictions (Johansson & Flanagan, 2009a) instead of binary information about whether or not the object is being touched (Hsiao et al., 2010), i.e. the force threshold has been exceeded.

1.2.6 Power of Data-Driven Approaches

Tight coupling between visual, haptic, even auditory perception and action is undoubtedly a promising approach to autonomous manipulation. Especially when dealing with uncertainty, it is crucial to be able to continuously monitor task progress and potentially adapt the systems behavior online to correct for unexpected events. However, without reference signals, i.e. without sensor predictions, it is impossible to detect and account for such events. The interesting information can only be obtained by computing the *difference* between the predicted and measured sensor information. Deviations of the measured sensor information from the predicted sensor information indicate an unusual event.

To generate appropriate sensor predictions for each particular manipulation behavior we can leverage from the proposed concept of using movement primitives. The idea is to have each movement primitive not only encode the desired movement trajectory, but also encode all associated sensory events acquired from previous executions. That is, in sync with the executed movement primitive, the measured sensor information is being recorded and subsequently encoded along with the movement primitive. Thus, sensor information recorded and encoded in previous executions serve as sensor prediction in subsequent executions of the same movement primitive. Whenever a movement primitive is being executed and new sensor information becomes available, the current sensor prediction model can be updated. Furthermore, the statistics over several trials provide valuable insights for each predicted sensor signal at each time instant. Information about the variance at each time step may translate to which amount predicted sensor signals can and should be trusted at that time step. Low variance over several trials suggest high confidence in predicted sensor values.

As previously argued, there is a reasonable amount of repetition in day to day manipulation tasks (Tenorth et al., 2009) and related tasks are encountered over and over again (Horswill, 1993). Movement primitives can naturally leverage such structure. For

*The problem of generating (visual, haptic, auditory) reference sensations (e.g. using physics based simulation) for a given manipulation task is even more complex than planning desired trajectories that involve contacts.

instance, when trying to open a door, instead of treating the task as a completely new instance and solve it from scratch every single time it is encountered (see Fig. 1.5), the system could reuse a movement primitive since similar tasks afford similar movement plans. Following a similar manipulation strategy each time a similar task is encountered is not a limitation of the approach. Instead, the resulting stereotypical behavior bears the true power: Stereotypical movements give rise to stereotypical sensory events. This simple yet profound insight ensures that sensor information experienced from previous task executions will be meaningful in subsequent ones. Sensor predictions (or expectation) are key to autonomous manipulation as they provide the often missing ingredient to close perception-action loops. For example, task progress can be monitored by comparing the expected sensor signal with the currently measured one. Similarly, failure conditions, indicated by significant deviations, can be detected online as they occur. Furthermore, provided appropriate task Jacobians, these deviations can be used to adapt and correct the movement plan online by servoing these sensor predictions. Finally, subsequent movements can be chosen based on these associated sensor information allowing to determine robust sequences of acquired skills online to achieve a complete task.

Interestingly, neuroscientists (Johansson & Flanagan, 2009a) have found that humans follow a fairly similar paradigm. In (Wolpert & Flanagan, 2001), the authors note that (for humans) "the discrepancy between actual and predicted sensory feedback is essential in motor control." Especially in the context of object manipulation (Flanagan et al., 2006) "skilled manipulation relies on predictions". The authors further note that "mismatch between expected sensory signals in tactile afferents and the actual sensory event triggers a learned corrective action pattern that decreases or increases fingertip forces, respectively".

Biological plausibility is popular to motivate robotics research contributions. However, it is not our intention to fall under this category. Instead, we want to emphasize on the fact that the proposed approach to autonomous manipulation is data-driven. Data-driven approaches have gained interest in many communities with increasing computational power as well as increasing data sets (Lohr, 2012). Often times, these data-driven approaches have outperformed model-based approaches. For instance, in linguistics, researchers have struggled for a long time to get decent speech recognition performance. Only after vast amounts of training data became available and machine learning algorithms were developed, the recognition rate skyrocketed. Note that, although the speech recognition performance improved, the linguists arguably did not achieve a better understanding or gained much theoretical insights about the underlying principals of speech recognition. Therefore, combining data-driven approaches and "big data" (Manyika et al., 2011) oftentimes focuses on engineering a workable solution without the claim of having understood how the solution came to be (Boyd & Crawford, 2012). Nevertheless, evaluating the progress in the past fifty years of research in autonomous manipulation, it remains questionable whether significant progress towards *understanding* autonomy will be achieved in the coming fifty years (Brock, 2011). Therefore, it is our belief that the ability of leveraging massive amounts of sensor data is key towards creating truly autonomous manipulation systems. Our approach supports this paradigm in that it can learn new skills

and improve both, task execution as well as task models, through experience as new data becomes available. Unfortunately, as of now, there has not been much experimentation with "big data" in autonomous manipulation. Although there are indeed domains where gathering data is expensive and sufficient data might never become available, e.g. Oceanography (Das et al., 2012), we believe that the domain of autonomous manipulation will not fall under this category. Nevertheless, data sets have been kept rather small and researchers often times focused on developing approaches that can learn from rather few examples. This thesis focuses on developing a data-driven approach to autonomous manipulation that can benefit from more data, it is however not in the scope of this thesis to show experimental evaluations on large data sets.

1.3 Problem Statement

Development of a data-driven approach to autonomous manipulation.

1.4 Thesis Contribution

The contribution of this work is twofold: The development of a coherent framework suitable for autonomous grasping and manipulation tasks as well as its implementation and evaluation on real robotic systems.

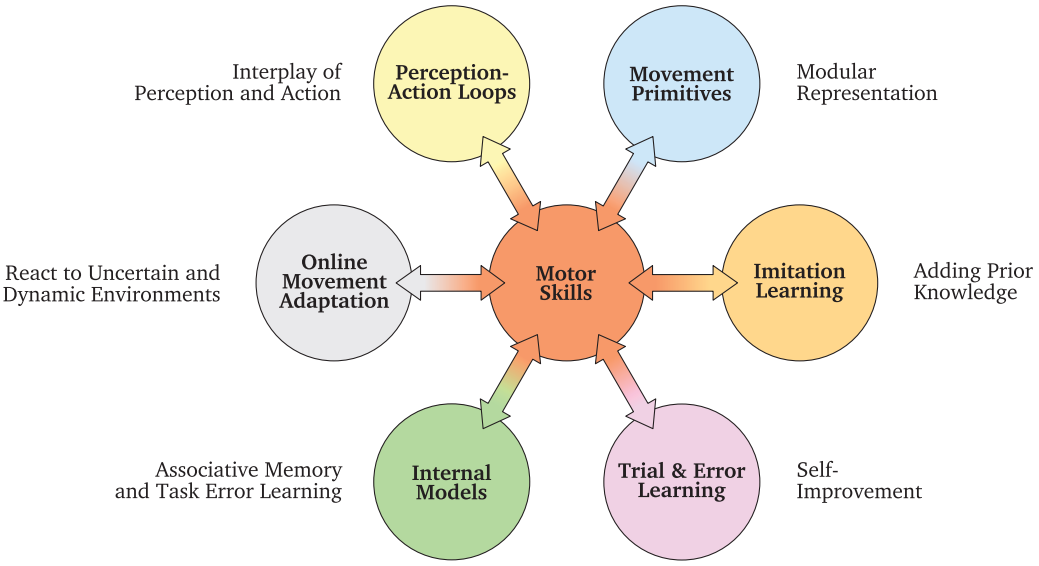


Figure 1.7: Conceptual overview diagram: The six proposed contributions that have been identified and addressed to achieve progress towards autonomous grasping and manipulation.

In particular, we have identified six crucial components that contribute to realize dexterous motor skills. An overview diagram is depicted in Fig. 1.7 and a brief description of each component is provided in the following:

Representing Motor Skills with Attractor Dynamics
A particular realization of movement primitives, called Dynamic Movement Primitives (DMPs), provide a modular and compact movement representation. This thesis will focus on the discrete variant, i.e. movements that encode goal-directed behaviors. The thesis will review important features such as structural stability, spatial and temporal invariance, ability to encode multiple degrees of freedom, synchronization and entraintment phenomena, and online modulation. Alongside, a modified formulation is introduced. This modified formulation changes the rather linear generalization behavior to mimic (local) human generalization capabilities. Note that all experiments have been conducted using this modified formulation. Furthermore, extensions to compose continuous movement sequences are introduced. Finally, experimental results on movement recognition tasks are shown.

Adding Prior Knowledge with Imitation Learning
The thesis presents several experiments on using imitation learning to bootstrap motor skills for object manipulation. In particular, we show results on the Sarcos master-slave system, the Willow Garage PR2 robot, and the ARM-S robot (see platform descriptions in Appendix A). The considered imitation learning setups include master-slave as well as kinesthetic teaching of positional and force skill.

Trial and Error Learning with Reinforcement Learning
Experimental results on using reinforcement learning are presented on two tasks: Learning a strong and accurate pool stroke and flipping a box with chopsticks.

Learning Sensor Predictions and Task Error Models from Experience
This thesis introduces the idea of associative memory for manipulation. Stereotypical movements encoded as DMPs and augmented with experienced sensory events, called Associative Skill Memories (ASMs), are introduced. ASMs facilitate progress monitoring and instant failure detection, allow for very contact reactive behavior, and provide a sensible way of choosing subsequent movements online. Furthermore, in vein with the spirit of stereotypical movements, a non-parametric method to learn task error models for manipulation to improve kinemathic calibration is introduced.

Reactive Control with direct Feedback from the Environment
To account for unexpected contact with the environment, sensor predictions are used inside real-time feedback control loops to online adapt the movement plan. To encode the orientation of the manipulator a quaternion formulation has been introduced that guarantees the generation of unit quaternion necessary for tight integration in feedback control loops.

Closing Perception-Action Loops
Associated sensor information facilitates loop closure. Here, we present results on using Associative Skill Memories (ASMs) to online sequence manipulation movements that capable to adapt for changes in the environment.

1.5 Proposed System Architecture

The following section will provide the historical background on the development of movement primitives for robotics. In Sec. 1.5.2, we present a conceptual overview of how these movement primitives are embedded into our proposed architecture for autonomous manipulation.

1.5.1 Movement Primitives for Robotics

A particular realization of movement primitives, called Dynamic Movement Primitives (DMPs) has originally been introduced by (Ijspeert, Nakanishi, & Schaal, 2003; Schaal, Mohajerian, & Ijspeert, 2007; Ijspeert, Nakanishi, Pastor, Hoffmann, & Schaal, 2013). The approach is motivated from basic ideas of optimal control. Following the classical control literature from around the 1950s and 1960s (Bellman, 1957; Dyer & McReynolds, 1970), the goal of motor control and motor learning can generally be formalized in terms of finding a task-specific control policy:

$$\boldsymbol{u} = \boldsymbol{\pi}(\boldsymbol{x}, t, \alpha) \tag{1.1}$$

that maps the continuous state vector x of a control system and its environment, possibly in a time t dependent way, to a continuous control vector u. The parameter vector α denotes the problem specific adjustable parameters in the policy $\boldsymbol{\pi}$, e.g., the weights in neural network or a generic statistical function approximator. In simple words, all motor commands for all actuators (e.g., muscles or torque motors) at every moment of time depend (potentially) on all sensory and perceptual information available at this moment of time, and possibly even past information. We can think of different motor skills as different control policies $\boldsymbol{\pi}$, such that motor control can be conceived of as a library of such control polices that are used in isolation, but potentially also in sequence and superposition in order to create more complex sensory-motor behaviors. From a computational viewpoint, one can now examine how such control polices can be represented and acquired. Optimal control theory offers one possible approach. Given some cost criterion $r(\boldsymbol{x}, \boldsymbol{u}, t)$ that can evaluate the quality of an action $\boldsymbol{u}$ in a particular state $\boldsymbol{x}$ (in a potentially time t dependent way), dynamic programming (DP), and especially its modern relative, reinforcement learning (RL), provide a well founded set of algorithms of how to compute the policy $\boldsymbol{\pi}$ for complex nonlinear control problems. In essence, both RL and DP derive

an optimal policy by optimizing the accumulated reward (in statistical expectation $E\{\}$) over a (potentially $\tau > 0$ discounted*) long term horizon (Sutton & Barto, 1998):

$$J = E\left\{\int_{t=0}^{T} e^{-t/\tau}\, r(\boldsymbol{x}, t, \alpha)\, dt\right\} \quad . \tag{1.2}$$

Unfortunately, as already noted in Bellmans original work (Bellman, 1957), learning of $\boldsymbol{\pi}$ becomes computationally intractable for even moderately high dimensional state-action spaces, e.g., starting from about 6 to 10 continuous dimensions, as the search space for an optimal policy becomes too large or too nonlinear to explore empirically.

In many theories of biological motor control and most robotics applications, the full complexity of learning a control policy is strongly reduced by assuming prior information about the policy. The most common priors are that the control policy can be reduced to a desired trajectory, $[\boldsymbol{x}_d(t), \dot{\boldsymbol{x}}_d(t)]$. Optimal control or reinforcement learning† approaches for trajectory learning are computationally significantly more tractable (J. Peters & Schaal, 2008) than state-space approaches. For instance, by using a tracking-error driven feedback controller (e.g., proportional-derivative (PD) with positive definite gain matrices K), a (explicitly time dependent) control policy can be written as:

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{\pi}\big(\boldsymbol{x}, \alpha(t), t\big) = \boldsymbol{\pi}\big(\boldsymbol{x}, \big[\boldsymbol{x}_d(t), \dot{\boldsymbol{x}}_d(t)\big], t\big) \\ &= K_x\left(x_d(t) - x\right) + K_{\dot{\boldsymbol{x}}}\left(\dot{\boldsymbol{x}}_d(t) - \dot{\boldsymbol{x}}\right) \quad . \end{aligned} \tag{1.3}$$

For problems in which the desired trajectory is easily generated and in which the environment is static or fully predictable, such a shortcut through the problem of policy generation is highly successful. However, since policies like those in Eq. (1.3) are usually valid only in a local vicinity of the time course of the desired trajectory $\big[\boldsymbol{x}_d(t), \dot{\boldsymbol{x}}_d(t)\big]$, they are not very flexible. For instance, an unforeseen external perturbation, e.g. holding the robot, may introduce a huge error between the desired and actual position of the robot the longer the perturbation lasts. Such a tracking error translates easily into huge motor commands that are potentially dangerous to the environment or may break the robot itself. Thus, when dealing with a dynamically changing environment in which substantial and reactive modifications of control commands are required, one needs to adjust desired trajectories appropriately, or even generate entirely new trajectories by generalizing from previously learned knowledge.

Given that the concept of time-indexed desired trajectories has its problems, both from a computational and a biological plausibility point of view, one might want to look for other ways to generate control policies. From a behavioral point of view, a control policy is

*The discount factor causes rewards far in the future to be weighted down - setting τ to a very large value essentially eliminates discounting.

†As will be discussed later, in contrast to optimal control approaches, reinforcement learning can operate model-free, i.e., without knowledge of the dynamics of a motor system and the equations of the reward function.

supposed to take the motor system from an arbitrary start point to the desired behavior. In most biological studies of arm movements, the desired behavior is simply a goal state for pointing or grasping. But there is also the large class of cyclic movements, like walking, swimming, chewing, etc.. Both behavioral classes can be thought of as attractor dynamics, i.e., either a point attractor as in reaching and pointing, or a limit cycle attractor as in periodic movement. Systems with attractor dynamics have been studied extensively in the nonlinear dynamic systems literature (Guckenheimer & Holmes, 1983; Strogatz, 2008). A dynamic system can generally be written as a differential equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \alpha, t) \tag{1.4}$$

which is almost identical to Eq. (1.1), except that the left-hand-side denotes a change-of-state, not a motor command. Such a kinematic formulation is, however, quite suitable for motor control if we conceive of this dynamic system as a kinematic policy that creates kinematic target values (e.g., positions, velocities, accelerations), which subsequently are converted to motor commands by an appropriate controller (Nakanishi et al., 2008). Planning in kinematic space is often more suitable for motor control because kinematic plans generalize over a large part of the workspace since nonlinearities due to gravity and inertial forces are taken care of by the controller at the motor execution stage. Kinematic plans can theoretically also be cleanly superimposed to form more complex behaviors, which is not possible if policies code motor commands directly. It should be noted, however, that a kinematic representation of movement is not necessarily independent of the dynamic properties of the limb. Proprioceptive feedback can be used online to modify the attractor landscape of the policy in the same way as perceptual information (Ijspeert et al., 2003; Righetti & Ijspeert, 2006; Pastor et al., 2009; Pastor, Righetti, et al., 2011).

Most dynamic systems approaches also emphasize removing the explicit time dependency of π, such that the control policies become "autonomous dynamic systems":

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \alpha) \tag{1.5}$$

Explicit timing is cumbersome; it requires maintaining a clocking signal, e.g., a time counter that increments at very small time steps (as typically done in robotics). Besides that it is disputed whether biological systems have access to such clocks, there is an additional level of complexity needed for aborting, halting, or resetting the clock when unforeseen disturbances happen during movement execution. The power of modeling motor control with autonomous nonlinear dynamic systems is further enhanced, as it is now theoretically rather easy to modulate the control policy by additional, e.g., sensory or perceptual, variables, summarized in the coupling term $\mathbf{C}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \alpha) + \mathbf{C} \tag{1.6}$$

This ability, to modulate movement plans using external and internal sensor signals facilitate reactive behaviors. The term $\mathbf{C}$ can for example represent a potential field and realize obstacle avoidance (Khatib, 1986).

Adopting the framework of dynamics systems theory for policy generation connects to a large body of previous work. For invertebrates and lower vertebrates, research on central pattern generators (Selverston, 1980) has a long tradition of using coupled oscillator theories for modeling. From a behavioral point of view, many publications in the literature deal with coupled oscillator theories to explain perceptionaction coupling and other behavioral phenomena (Kugler & Turvey, 1987; Kelso, 1995). Thus, at the first glance, one might expect a straightforward and experimentally well-established framework to approach control policies as nonlinear dynamic systems. Unfortunately, this is not the case. First, modeling with nonlinear dynamics systems is mathematically quite difficult and requires usually very good intuition and deep knowledge in nonlinear systems theory - optimization of time-index trajectories is often much easier to handle with well established algorithms (Dyer & McReynolds, 1970). Second, with very few exceptions (Bullock et al., 1988; Schöner, 1990), dynamic systems approaches have only focused on periodic behavior, essentially assuming that discrete behavior is just an aborted limit cycle. The goal of this thesis is to demonstrate how a particular dynamic systems approach based on Dynamic Movement Primitives (DMPs) can offer a simple and versatile approach for motor skill generation.

In the following, we will present a conceptual overview of our approach and show how movement primitives are embedded into a larger context.

1.5.2 Conceptual Overview

The core of our architecture, the motion library, comprises the movement primitives, as illustrated in Fig. 1.8. Similar to words and phonemes in speech and language, these movement primitives realize the common computational representation that serves for movement generation as well as for movement recognition. To emphasize on the commonalities between language and motion, in (Del Vecchio, Murray, & Perona, 2003) movement primitives were termed "movemes" as the parallel of speech phonemes for movement. Given a particular language, there are many (but not infinitely many) different words. Similarly, given a particular embodiment and domain, we assume that the amount of (necessary) movements is limited, too*. Complex movements are composed from sequencing several movement primitives just like sentences are constructed from a sequence of words. Finally, sentences are understood by breaking them down into a set of known words. Equally, observed behaviors can be recognized and understood by parsing them into a set of known primitives. The motion library can therefore be understood as the parallel of a vocabulary.

*In fact, our approach aims to identify the minimum set of movements by preferring movements that are being reused more often. These stereotypical movements are being motivated later in this thesis.

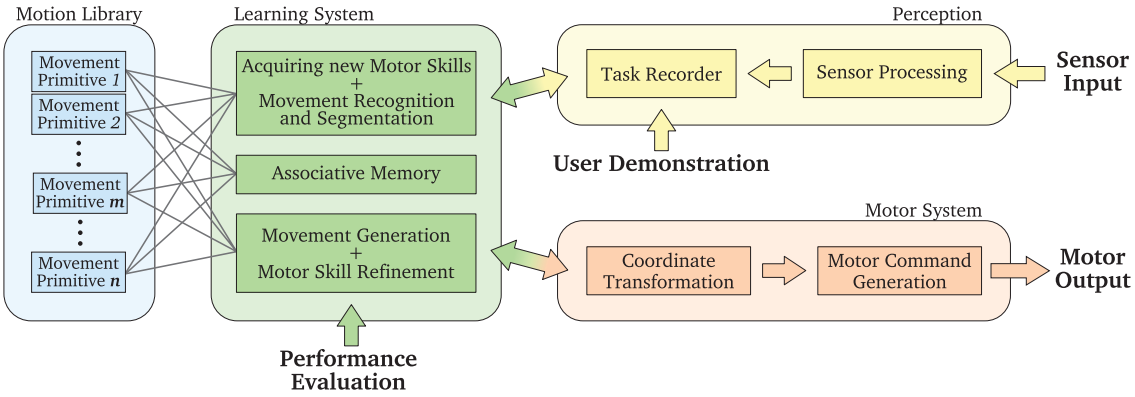


Figure 1.8: The movement primitives in the motion library (blue) realize the common computational representation in which perceptual representations (yellow) serve motor representations (red), motor representations facilitate perception, and learning (green) provides mutual constraints between them. New movement primitives are learned from observation and complex motions are generated by composing several movement primitives. Perceived movements are recognized and segmented based on motor representation. Performance evaluations are used to refine previously learned movement primitives.

Each movement primitive encodes a particular behavior, either rhythmic or discrete, and is specific to only a particular level of representation, for example end-effector or joint space. For movement generation, the movement primitives need to be retrieved from the library, setup according to the task at hand, and passed on to the robot controller. The setup involves specifying the target state and the duration of the movement, whereas the execution stage involves generating the desired trajectories and converting them to motor commands using appropriate controllers. On the other side, the very same representation serves for movement recognition. Similar to speech recognition, movement recognition involves finding the movement primitive within the library that best fits the observed motion data. Visual perception is required to extract the movement of the teacher and classification may be achieved by computing the correlation of the teacher's movement and the system's movement repertoire (Meier, Theodorou, Stulp, & Schaal, 2011; Meier, Theodorou, & Schaal, 2012; Sanmohan, Krüger, Kragic, & Kjellström, 2011). If none of the pre-learned movement primitives is a good fit, a new movement primitive is learned and added to the motion library. Thus, teaching a rich set of motor skills to a robotic system only requires the user to demonstrate movements once and hence does not require an expert with domain knowledge. The teaching process is simplified further, if the robotic system is capable of automatically segmenting observed behavior into a small set of movement primitives, as it alleviates the need of the user to demonstrate each movement separately. Instead, complex movements can be taught in a very natural way, since programming a robotic system is achieved without pressing a single button. Segmentation of complex behavior into a sequence of primitives is (again) based on the system's current movement repertoire, represented by the movement primitives contained in the motion

library. The problem of movement segmentation then becomes finding the set of pre-learned movement primitives that in sequence best describes the observed motion data. The interesting property of such an approach is that the system perceives observed behavior directly through its own motor representation (Meier et al., 2011, 2012; Sanmohan et al., 2011; Niekum, Osentoski, Konidaris, & Barto, 2012). Of course, including spatial information about manipulated objects may also be used to determine discontinuities in the observed behavior. For instance, the change in the relative pose and/or velocity of the teacher's hand and manipulated objects, e.g. when grasping a cup or unscrewing the lid of a bottle, could also be used to identify movement primitives, as in (Niekum et al., 2012). Furthermore, similar to speech recognition, the success rate of recognizing individual movements might mainly be inferred in the context of preceding and succeeding movements, for example, a pouring motion is more likely to occur after a grasping than after a releasing movement. However, to enable reasoning of that sort, additional information in form of semantic need to be assigned to each movement primitive. On the other side, such semantic information attached to each movement primitive can also be used for planning purposes: A pick-and-place operation for example is accomplished by selecting a proper sequence of movement primitives, that is, first a grasping, then a placing, and finally a releasing primitive. Associating sensor information with each movement primitive furthermore allows to learn forward models. Finally, based on such evaluations, reinforcement learning can be employed to further refine each movement primitive until a satisfactory skill level is achieved.

1.6 Thesis Structure

The thesis is meant to be read in sequential order. Nevertheless, each chapter was written to be as self-contained as possible, and references to previous chapters are provided whenever necessary. The remainder of this document is structured as follows: An overview over related work will be provided in Chapter 2. The proposed movement representation is introduced in Chapter 3. Experimental evaluation on learning manipulation skills are presented in Chapter 4. The concept of Associative Skill Memories is introduced in Chapter 5. An extension to learning forward models for manipulation is presented in Chapter 6. Finally, the thesis is concluded in Chapter 7.

Chapter 2

Related Work

This chapter will contrast the proposed approach with previous approaches in the domain of autonomous manipulation. The focus is on related work that have shown experimental results on real robotic systems. Nevertheless, we will also reference previous work that show promising results in simulation studies. In particular, we will review movement representations in Sec. 2.1 and approaches to trajectory modulation in Sec. 2.2. In Sec. 2.3, we will review approaches to imitation learning and in Sec. 2.4, we will present related work in trial-and-error learning. Approaches that allow for task outcome prediction are presented in Sec.2.5. Sensor-based approaches to grasping are presented in Sec. 2.6. Finally, related work on forward models is presented in Sec. 2.7.

2.1 Movement Primitives

Ever since experiments on human motor control* revealed statistical regularities and invariances across subjects (Flash & Hogan, 1985), the quest for a modular movement representation began. Among the first to model human point-to-point reaching movements with a point attractor system were (Bullock et al., 1988). A large body of work on using dynamical systems to model movement phenomena, emphasizing dynamic phenomena like phase transitions, was suggested by (Schöner & Kelso, 1988; Schöner, 1990). Only much later, the robotics community adopted similar models that have been developed to explain movement patterns in biological systems to generate desired trajectories for robotic systems. Several approaches have been suggested ever since, see (Ajallooeian et al., 2010) for an overview. In this section, we will review research on movement representations using dynamical systems.

*The experiments included unconstrained point-to-point arm movement in a horizontal plane. The results showed that human reaching movements are "approximately straight with bell-shaped tangential velocity profiles and that curved motions (through an intermediate point or around an obstacle) have portions of low curvature joined by portions of high curvature" (Flash & Hogan, 1985).

Our work on learnable dynamical systems, namely Dynamic Movement Primitives (DMPs), originated from the desire to model elementary motor behaviors, called motor primitives, in humans and robots as attractor systems, an approach that has a long tradition in neuroscientific modeling (Kelso, 1995; Ijspeert, 2008). Initial work addressed imitation learning for robotics (Ijspeert, Nakanishi, & Schaal, 2002; Ijspeert et al., 2003). Subsequent investigation examined the approach for biped locomotion (Nakanishi et al., 2004), adaptive frequency modulation (Pongas, Billard, & Schaal, 2005; Buchli, Righetti, & Ijspeert, 2006; Righetti & Ijspeert, 2006; Gams, Ijspeert, Schaal, & Lenarčič, 2009), combined discrete and rhythmic behavior generation (Degallier, Righetti, Gay, & Ijspeert, 2011), models for biological movement generation (Schaal et al., 2007; Hoffmann, Pastor, Park, & Schaal, 2009), generating libraries of motor skills (Pastor, Hoffmann, & Schaal, 2008; Pastor et al., 2009; Ude, Gams, Asfour, & Morimoto, 2010). Various other projects in robotics created work that resembles our basic approach - the generation of kinematic movement primitives as attractor systems using basis function approaches to model the nonlinear dynamics (e.g., (Billard & Matarić, 2001; Schaal, Ijspeert, & Billard, 2003; Stulp, Oztop, Pastor, Beetz, & Schaal, 2009; Kulvicius, Ning, Tamosiunaite, & Wörgötter, 2011).

An interesting variant was presented by (Calinon, Guenter, & Billard, 2007; Calinon & Billard, 2008; Hersch, Guenter, Calinon, & Billard, 2008) and later by (Khansari-Zadeh & Billard, 2010; Gribovskaya, Khansari-Zadeh, & Billard, 2011; Khansari-Zadeh, 2012). In this work, the authors use Gaussian Mixture Models (GMM) to directly learn nonlinear attractor landscapes for movement primitives from demonstrated trajectories as a state-space approach; they avoid any explicit or implicit timing system like our canonical system, and they do not include stabilizing dynamics as accomplished with the spring-damper system in our transformation system equations. As an advantage, they obtain a movement primitive representation that depends on only observable states, which can be considered a direct policy learning approach, as discussed in (Schaal, 1999). Such a representation can learn much more complex attractor landscapes, where the attractor landscape can strongly vary throughout the state-space. As a disadvantage, the authors have to spend significant numerical effort on a nonconvex optimization problem to ensure stability of their movement primitives, they have to address learning of mixture models in potentially rather high-dimensional and ill-conditioned spaces where the number of mixture components may be hard to determine, and they require many more data throughout the state-space to represent the attractor dynamics. At this point, movement recognition and periodic motion are not addressed, and it is not entirely predictable how their movement primitives generalize in areas of the state-space that have few or no data. In essence, this approach differs in a similar way from ours as state-space-based optimal control and reinforcement learning differs from trajectory-based optimization approaches (J. Peters & Schaal, 2008). State-space approaches have a more powerful representation but quickly degrade in high-dimensional settings. Trajectory-based approaches work well in very high dimensions and generalize well throughout these spaces, but they have less representational power. It is really up to a particular application which properties are beneficial.

2.2 Coupling Phenomena

Using non-linear coupled dynamical systems to model movement has attracted many researchers due to their abilities to compactly encode complex coordinated patterns without the need to explicitly plan or supervise the details of such pattern formation. Among the first approaches that exploit another key feature of dynamical systems, namely coupling phenomenas, was presented in (Khatib, 1986). Repulsive and attracting potential fields have been used to modulate trajectories to account for obstacles while reaching a target state.

Dynamic Movement Primitives (DMPs) offer trajectory modulation mechanisms that can affect both, temporal as well as spatial evolution. In (Ijspeert et al., 2002), the authors propose a coupling term that modulates the temporal evolution. Similarly, in (Pongas et al., 2005; Gams et al., 2009), approaches have been presented that synchronize a learned rhythmic motion with an external signal. Modulation of the spatial evolution has been exploited in (Park, Hoffmann, Pastor, & Schaal, 2008) to realize obstacle avoidance for reaching movements using a velocity-dependent potential field. In (Hoffmann et al., 2009), the coupling term was adapted such that convergence to the goal is guaranteed despite arbitrarily many (point) obstacles. In this work, the coupling term has been adopted from empirical observations on human obstacle avoidance while walking (Fajen, Warren, Temizer, & Kaelbling, 2003; Fajen & Warren, 2003). A similar adaptation term has been used in (Giese, Mukovskiy, Park, Omlor, & Slotine, 2009) to simulate human walking behavior. In (Gams et al., 2009), the authors presented an approach that adapts rhythmic motions necessary for wiping the surface of a table using force feedback. In (Pastor, Righetti, et al., 2011), a generalized feedback term has been presented that adapts discrete movements. In (Gams et al., 2013), the authors proposed to use iterative learning control to adapt the gains of the feedback term. An online adaptation mechanism to realize obstacle avoidance using a state-space representation of a dynamical system has been suggested in (Khansari-Zadeh & Billard, 2012).

2.3 Imitation Learning

Imitation is a common mechanism for transferring knowledge from a skilled agent (the *teacher*) to an unskilled agent (the *student*) using direct demonstration rather than manipulating symbols. Thus, the ability to imitate has a drastically lower cost of programming than one which requires the domain knowledge of an expert. There are various forms of imitation ranging from imitation of low-level features, namely imitation of trajectories, to imitation of higher level features, such as imitation of complete tasks and behaviors (Billard & Siegwart, 2004). The simplest form of imitation, imitation of trajectories, requires the student to gather information about the task in form of perceptual inputs ($\boldsymbol{x}$) and action outputs ($\boldsymbol{u}$) and learns the control policy ($\pi : \boldsymbol{x} \rightarrow \boldsymbol{u}$) directly. These approaches learn *how* a particular task is accomplished. Thus, learned behaviors

will always be similar to the demonstrated behavior. Approaches in higher level of imitation however reason about the intention of the demonstrated task and therefore infer *what* is being accomplished. Thus, these approaches try to infer the objective function that defines the task (Abbeel, Coates, & Ng, 2010; Kalakrishnan, Pastor, Righetti, & Schaal, 2013; Kalakrishnan, 2014), a problem known as Inverse Reinforcement Learning (IRL). In these approaches, the resulting behavior is generated indirectly by optimizing the objective function and can therefore be very different from the demonstrated behavior. Interestingly, neural processing of vision in the human brain is assumed to follow a similar distinction. The two-streams hypothesis (Goodale & Milner, 1992) suggests that the dorsal stream carries information about the *how* while the ventral stream is involved when trying to understand *what* is being accomplished. Our approach belongs to the category of direct policy learning methods and we will therefore present related work accordingly. Nevertheless, our approach also uses (manually specified) objective functions to improve over an initial policy.

One of the challenges that need to be mastered in direct policy learning is the correspondence problem, i.e. how to overcome the potentially different embodiments between the teacher and the student. A common approach to is to represent trajectories in endeffector space (Calinon et al., 2007; Calinon & Billard, 2008; Pastor et al., 2009). Another popular approach to circumvent the correspondence problem all along is kinesthetic teaching, which requires the teacher to manually guide the student through physical contact (Ijspeert et al., 2002; Hersch et al., 2008). This approach facilitates to encode movements in joint space (Kober, Mohler, & Peters, 2008). An interesting approach has been presented in (Hersch & Billard, 2008) where a combination of end-effector and joint space trajectories has been encoded. DMPs have been used to encode joint trajectories (Ijspeert et al., 2002; Kober et al., 2008), endeffector trajectories (Pastor et al., 2009; Gams, Do, Ude, Asfour, & Dillmann, 2010), and task space trajectories including desired nullspace postures (Pastor, Kalakrishnan, et al., 2011; Pastor, Righetti, et al., 2011; Pastor et al., 2012).

Especially in the context of manipulation, choosing an appropriate coordinate frame to encode movements is crucial (Gienger, Goerick, & Koerner, 2010). End-effector representations have been favored since the allow for more intuitive generalization (Calinon et al., 2007). To facilitate exploration without violating constraints special coordinate frames need to be chosen (Pastor, Kalakrishnan, et al., 2011). An interesting avenue of research is to infer such coordinate transformation automatically from demonstration (Niekum et al., 2012; Niekum, Chitta, Marthi, Osentoski, & Barto, 2013). In (Mühlig, Gienger, & Steil, 2012), the authors propose to choose among a pool of pre-determined transformations. In (Howard, Klanke, Gienger, Goerick, & Vijayakumar, 2009), a method is presented that discovers constraints in manipulation tasks from multiple demonstrations. Recent approaches (Schmidts, Lee, & Peer, 2011; Kormushev, Calinon, & Caldwell, 2011; Pastor et al., 2012) have proposed to also encode demonstrated force information using specialized haptic interfaces.

Approaches to movement sequencing have been presented in (Pastor et al., 2009; Nemec, Tamosiunaite, Wörgötter, & Ude, 2009; Nemec & Ude, 2011; Kulvicius et al., 2011; Shukla & Billard, 2012; Niekum et al., 2012).

2.4 Reinforcement Learning

Reinforcement Learning (RL) approaches can be divided into two classes, trajectory-based RL and state-space RL. As explored in (J. Peters & Schaal, 2008) and discussed in Sec. 1.5.1, trajectory-based RL offers a set of algorithms that have successfuly been applied to high dimensional movement systems. However, in contrast to state-space RL (Sutton & Barto, 1998), trajectory-based RL comes at the cost that only locally optimal solutions can be found. Nevertheless, state-space RL has only been successfully applied to rather low dimensional and discrete problem domains, usually 2D grid worlds. Therefore, in this thesis, we will focus on trajectory-based RL approaches.

Previous work on trajectory-based RL focused on gradient-based RL methods, e.g., policy gradient methods (J. Peters & Schaal, 2008). While successful in various example applications, policy gradients are notoriously hard to tune due to a variety of open meta parameters in the learning procedure, e.g., the gradient descent learning rate. Recently, novel probabilistic reinforcement learning algorithms were suggested (Theodorou, Buchli, & Schaal, 2010; Kober & Peters, 2011). These algorithms employ parameterized policies that guarantee attractor properties towards the goal. The DMP equations belong to this class of dynamic system equations. Their attractor landscapes can be adapted by only changing a few parameters (Kober & Peters, 2009; Pastor, Kalakrishnan, et al., 2011; Stulp, Theodorou, Buchli, & Schaal, 2011). DMPs have been successfully applied in many real world RL experiments including biped locomotion (Nakanishi et al., 2004), ball-in-cup manipulation (Kober & Peters, 2008), table tennis, dart throwing (Kober, Oztop, & Peters, 2010), pan cake flipping (Kormushev, Calinon, & Caldwell, 2010), and billiard (Pastor, Kalakrishnan, et al., 2011). The initial set of policy parameters for the motor primitives were obtained from human demonstration in all these cases. In particular the method of Path Integral RL (Theodorou, 2011) has demonstrated exceptional promise for learning motor skill with DMPs. Recent approaches have focused on not using trial-and-error learning to find good position trajectories but rather find suitable force trajectories (Kalakrishnan, Righetti, Pastor, & Schaal, 2011, 2012). Finally, similar approaches have also been used to learn optimal gain schedule (Buchli, Stulp, Theodorou, & Schaal, 2011).

2.5 Task Outcome Prediction

Failure detection methods from sensor data has mostly been explored in industrial settings such as milling operations (Cho, Asfour, Onar, & Kaundinya, 2005), machine vibration analysis (Althoefer, Lara, Zweiri, & Seneviratne, 2008), or automated assembly (Rodriguez, Bourne, Mason, Rossano, & Wang, 2010). Recently a series of approaches have been proposed that select a few task relevant sensor signals and use supervised learning methods to learn a probabilistic model of success for grasping (Rodriguez, Mason, Srinivasa, Bernstein, & Zirbel, 2011; Paolini, Rodriguez, Srinivasa, & Mason, 2013). In (Kormushev, Ugurlu, Colasanto, Tsagarakis, & Caldwell, 2012) a method was proposed that enabled a bipedal robot to detect unexpected perturbation while walking. Peters et al. (R. Peters, Bodenheimer, & Jenkins, 2006) proposed a method to discriminate successful from unsuccessful trials from sensor recordings of a teleoperated robot. Jain et al. (Jain & Kemp, 2013) have presented an approach to task outcome prediction for a door opening task. Their approach is similar to ours in that it is object-centric, data-driven, and task-specific. Interestingly, the authors developed an object relative probabilistic model for haptic interactions that can be shared among different robots.

Nevertheless, all previously mentioned approaches either only consider few sensor signals (less then 10) such that is was possible to learn a model (given enough data) that can predict the task outcome for future trials. Other approaches have incorporated domain knowledge to allow for good prediction performance. Our approach (Pastor, Kalakrishnan, et al., 2011) does not try to learn a predictive model that can explain all possible future trials, instead, the aim is to use sensor information from previous trials for each stereotypical movement separately and predict the task outcome for those. We would also like to point out the approach by Calinon et. al (Calinon & Billard, 2008), where Gaussian Mixture models were used to form motor primitives, and the statistics of multiple demonstrations during imitation learning was used to find important sensory features in a movement. This approach has similarities to the way we do data mining in sensory trajectories for performance prediction.

2.6 Contact-Reactive Grasping

In recent years, more and more robotic grippers and hands have become available with tactile and force sensing capabilities built-in. Since then, many approaches to grasping that incorporate sensor feedback have been proposed (Natale & Torres-Jara, 2006). Hsiao (Hsiao, 2009) used tactile feedback to localize the position of a known object model for grasping. Platt (Platt Jr, 2006; Platt, 2007) used force feedback to adjust an initial grasp until it is locally stable. Other approaches have been proposed that incorporate sensor feedback to account for positional uncertainty of the object (Ciocarlie et al., 2010; Felip & Morales, 2009; Hsiao et al., 2010). Contact reactive pretouch for grasping using

special sensors has been presented in (Mayton, LeGrand, & Smith, 2010). Adaptive grasping of a multi-fingered hand including tactile sensing has been presented in (Takahashi et al., 2008). Grasping of unmodeled objects using finger torque information has been proposed by (Maldonado, Klank, & Beetz, 2010). These approaches use manually tuned thresholds to detect early collisions in order to trigger a replanning phase for subsequent grasp attempts. Dollar et al. (Dollar et al., 2010) uses a similar pre-scripted reactive behavior in combination with a passively compliant hand. This work shows that not only the grasp range can be increased, but also the forces applied to the object can be reduced.

Alternatively, uncertainty in the object pose can be minimized by first pushing the object on the table (Dogar & Srinivasa, 2010), by controlling the applied forces (Righetti et al., 2014), by adapting the grasping movement online (Pastor, Righetti, et al., 2011), by learning robust grasp configurations from demonstration (Bohg & Kragic, 2010; Goldfeder, 2010; Herzog et al., 2012, 2014), or by optimizing an initial grasp configuration using trial-and-error (Stulp et al., 2011). See (Bohg, Morales, Asfour, & Kragic, 2014) for an overview.

2.7 Learning Forward Models

Accurate modelling of kinematic chains is important for robots to successfully perform grasping and manipulation tasks. Therefore, a large body of research has been conducted to calibrate the kinematic parameters of robotic manipulators, see (Hollerbach, Khalil, & Gautier, 2008) for a comprehensive overview. The goal is to obtain very high precision for primarily industrial robots. In (Bennett, Geiger, & Hollerbach, 1991) and (Zhuang, Wang, & Roth, 1995), the authors present methods to simultaneously calibrate the geometric parameters of the robot as well as the extrinsic and intrinsic parameters of the robot's camera. Joint calibration of multiple sensors is considered in (Le & Ng, 2009). Similarly, work presented in (Pradeep, Konolige, & Berger, 2014) is inspired by the bundle adjustment approach, and generalized to estimate robot system parameters by including measurements from various types of sensors. In (Hubert, Stückler, & Behnke, 2012) the authors present an optimization approach that can make use of prior knowledge of the system. However, most of these methods only consider geometric parameters and therefore cannot account for potential non-linearities in kinematic chains. To achieve high accuracy even in the presence of non-geometric errors, e.g. due to gear transmission, friction, temperature, and compliance, research has been conducted towards also modelling these quantities and also optimizing those parameters, see (Majarena, Santolaria, Samper, & Aguilar, 2010) for an overview. However, accurate modelling of all possible non-geometric effects is tedious and potentially infeasible. In contrast, in (Sturm, Plagemann, & Burgard, 2008), the authors propose to learn the body schema by determining the topology of the system through self-perception.

Although this approach is interesting and ambitious, we argue that discarding the readily available forward model completely may harm accuracy and generalization. Instead we

propose to account for non-geometric effects by learning a non-parametric error model on top of the existing (uncalibrated) forward model. Thus, our approach can be seen as a trade-off between having to manually model non-geometric effects and completely discarding the oftentimes readily available kinematic model. Essentially, using the analytical model achieves generalization across the entire workspace and the (local) error correcting model provides accuracy for the relevant part of the state-space. This paradigm has been exploited for learning the dynamics model in (Nguyen-Tuong & Peters, 2010). In (Aoyagi, Kohama, Nakata, Hayano, & Suzuki, 2010), the authors propose to learn the residual error using a three layered feedforward neural network. However, the experiments conducted consider a rather small workspace with limited non-linear effects achieving an improvement on the order of about 0.1 mm. In (Axelrod & Huang, 2012) the authors present their approach to account for the non-linearities present in their copy of the ARM-S robot. Their approach computes a position offset for the arm using k-nearest neighbors for different parts of the workspace. Orientation corrections are only computed for horizontal and vertical endeffector orientations. Furthermore, their approach cannot account for errors due to cable stretch given that two similar end-effector poses can have two very different pose errors depending on the configuration of the arm, the particular reaching trajectory as well as its execution speed.

Especially in the context of manipulation, one might argue that arm/hand tracking and simultaneous object tracking by means of visual servoing is sufficient to achieve many of the considered tasks, even with uncalibrated cameras (Wang, Jiang, Chen, & Liu, 2012). This holds true, nevertheless, we argue that an accurate initial estimate of the hand pose can only improve performance. Furthermore, precise forward models can be used whenever occlusions are present.

Chapter 3

Dynamic Systems for Movement Generation

Nonlinear dynamic systems are suited to model discrete and rhythmic movements as they exhibit two elementary behaviors which are point attractive and limit cycle. The goal of control in both cases is to attain the particular attractor state. For point attractors, the control policy (CP) is required to reach the goal state with a particular trajectory shape, independent of its initial conditions, as for example in reaching movements from different start positions. For limit cycles, the CP is also required to reach the goal from any start state, however, to encode rhythmic movements the goal is given as the trajectory shape of the limit cycle, as for example in a complex drumming beat hitting multiple drums during one period (Degallier, Santos, Righetti, & Ijspeert, 2006). Although it is possible to construct nonlinear differential equations that could realize both these behaviors in one set of equations (Schöner, 1990), for reasons of robustness, simplicity, functionality, and biological realism, an approach was chosen that separates these two regimes.

This chapter focuses on using point attractive systems to model discrete movements, i.e. movements that have a particular start position, end position, and duration. Work on using dynamic systems, that implement limit cycle attractors as a model for rhythmic movements is presented in (Ijspeert et al., 2002; Nakanishi et al., 2004; Pongas et al., 2005; Righetti & Ijspeert, 2006). In the following section we will outline our model equations for discrete movements, previously introduced in (Ijspeert et al., 2002, 2003), please refer to (Ijspeert et al., 2013) for an overview. Furthermore we will present variations, previously introduced in (Hoffmann et al., 2009; Pastor et al., 2009) and list all the interesting properties.

3.1 Dynamic Movement Primitives: Learnable Nonlinear Point Attractor Systems

Formalizing nonlinear dynamic equations such that they can be flexibly adjusted to represent arbitrarily complex behaviors without the need for manual parameter tuning while

avoiding the danger of instability of the equations is non-trivial. The challenge is due to the parameter sensitivity of these systems, their complex phase transitions in response to subtle parameter changes, and the difficulty to analyze and predict their long-term behavior. Therefore, our approach uses an analytically well understood dynamical system with convenient stability properties and modulates it with nonlinear terms such that it achieves a desired attractor behavior.

A discrete movement is generated by integrating the following set of differential equations*, which model a damped spring and a nonlinear forcing term:

$$\begin{aligned} \tau\dot{v} &= K(g-x) - Dv + (g-x_0)f \\ \tau\dot{x} &= v \ , \end{aligned} \tag{3.1}$$

where $\dot{v}$, v, and x correspond to desired acceleration, velocity and position of the system. The spring acceleration is $K(g-x) - Dv$, where K is the spring constant, and D is the damping term, which is chosen such that the spring system is critically damped (convergence to the goal without oscillations). The nonlinear forcing term is $(g-x_0)f$, where x_0 is the start, g the goal position, and f is a nonlinear function which can be adapted to allow the generation of arbitrarily complex movements. Since the forcing term transforms the simple dynamics of the unforced system into a desired nonlinear behavior, we refer to this first set of equations as *transformation system*.

For appropriate parameter settings and $f = 0$, these equations form a globally stable linear dynamic system with $(v, x) = (0, g)$ as a unique point attractor. Adding a nonlinear function f in Eq. (3.1) that changes the rather trivial exponential convergence of x to allow more complex trajectories on the way to the goal, enters the domain of nonlinear dynamics. The arbitrary complexity of the resulting equations which comes along with such a change has prevented research from employing generic learning in nonlinear dynamic systems so far (Schaal, Peters, Nakanishi, & Ijspeert, 2005). However, this issue is addressed by introducing an additional differential equation referred as *canonical* system

$$\tau\,\dot{s} = -\alpha\,s \tag{3.2}$$

and the nonlinear function

$$f(s) = \frac{\sum_i \psi_i(s)\,\theta_i\,s}{\sum_i \psi_i(s)} \ , \tag{3.3}$$

where

$$\psi_i(s) = \exp\bigl(-h_i(s-c_i)^2\bigr) \ . \tag{3.4}$$

*A different notation is used as in (Ijspeert et al., 2002, 2003) to high-light the spring-like character of these equations.

The canonical system (3.2) is a first order dynamic system, where α is a pre-defined constant and τ is the same temporal scaling factor as in Eq. (3.1). The phase variable s (going from 1 towards zero*) anchors the Gaussian basis functions $\psi_i(s)$ (characterized by a center c_i and bandwidth h_i) and drives the nonlinear function f to ensure two properties: first it prevents the nonlinear function from depending directly on time. Thus, the duration of a movement can be altered simply by changing τ. Second, (assuming boundedness of the adjustable weights θ_i and with the choice of positive constants for K, D, and α) it guarantees global convergence to the unique point attractor g. This can be shown by analyzing separately for the (s) and the (x, v) dynamics. It is clear that the (s) dynamics are globally asymptotically stable. Formally, by following the input-to-output stability analysis in (Khalil, 1996), it can be conservatively concluded that the equilibrium state of the entire system is asymptotically stable. Intuitively, when $t \to \infty$, the transformation system is reduced to $\dot{v} = K(g - x) - Dv$ and $\dot{x} = v$, since $\frac{\sum_i \psi_i \theta_i}{\sum_i \psi_i}$ is bounded (due to the presumption that θ_i are bounded) and $s \to 0$ as $t \to \infty$. Thus, it can be seen that $(v, x) \to (0, g)$ as $t \to \infty$. The reduction of the external force towards the end of the movement transforms the system, in the limit of $s = 0$, into a linear spring-damper with known stability properties. Another path to prove stability for our approach was suggested in (Perk & Slotine, 2006).

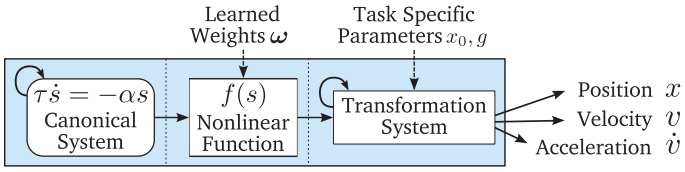


Figure 3.1: Sketch of a one dimensional DMP: The canonical system drives the nonlinear function f which perturbs the transformation system.

Although specific realizations of the canonical system, the nonlinear function approximator, and the transformation system which is driven by the forcing term have been presented above, it is the design principle that is most important. The canonical system is a simple (or well-understood) dynamic system that generates a behavioral phase variable, which is our replacement for explicit timing. Its output drives the nonlinear function approximator to generate a forcing term. Locally weighted regression was chosen to generate the function approximator, as it is very easy to fit to desired behavior, and has variants which automatically determine the necessary number of basis functions, as well as their centers c_i and bandwidths h_i (Schaal & Atkeson, 1998). However, other function approximators could be used, too, e.g., radial basis function networks, mixture models, or Gaussian Process regression, etc. (Bishop, 2006). Finally, the transformation system should be a dynamic system that, when excited with a forcing term, is easily analyzed and

*Actually slightly larger than zero, α is chosen such that s is close to zero (something like $s_{\min} = 0.001$) towards the end of the movement, $\alpha = -\log(s_{\min})/\tau$.

manipulated. For instance, as introduced in (Hoffmann, Pastor, Park, & Schaal, 2009; Pastor, Hoffmann, Asfour, & Schaal, 2009), the transformation system can be replaced by

$$\begin{aligned} \tau \dot{v} &= K(g - x) - D\, v - K(g - x_0)s + K f(s) \\ \tau \dot{x} &= v \ . \end{aligned} \tag{3.5}$$

Note, the phase variable s and the nonlinear function $f(s)$ are computed using the same canonical system (3.2) and function approximator (4.1). The prove for global convergence follows the same line of argumentation as for the previously mentioned transformation system in Eq. (3.5): in the limit $s \to \infty$, both forms are the same. However, these accelerations result from a damped spring that links the current position x of the movement system with a virtual trajectory, i.e. a moving equilibrium point. Thus, the dynamics of the two presented transformation systems are inherently different. Changes in the parameter for the start x_0 and the goal g for example result in different behavior. The pre-multiplication of f with $(g - x_0)$ in Eq. (3.1) scales the movement amplitude linearly, whereas the formulation in Eq. (3.5) is designed such that it compares to human behavior (Hoffmann & Schaal, 2007; Hoffmann, 2011).

In the remainder of this chapter, we will consider both formulations and highlight their differences whenever necessary. For this purpose we will refer to the formulation in Eq. (3.1) as basic transformation system (BTS) and the one in Eq. (3.5) as anthropomorphic transformation system (ATS). Both transformation systems follow the same procedure of learning the equations from observed behavior (described next in Sec. 3.2) and generating movement plans (described in Sec. 3.3). The quantitative difference between these two transformation systems is the way the attractor landscape adapts when changing the start and goal position with respect to each other. We will highlight some differences in Sec. 3.6.

3.2 Learning Equations from Observed Behavior

Learning Dynamic Movement Primitives (DMPs) corresponds to finding the weights θ_i in the nonlinear function f such that it forces the transformation system to follow the observed behavior. Given that f is a normalized basis function representation with linear parametrization, it allows applying a variety of learning algorithms. For instance, if a sample trajectory is given in terms of $y(t), \dot{y}(t), \ddot{y}(t)$ and a duration T, as typical in imitation learning (Schaal, 1999), a supervised learning problem can be formulated. The target for f is obtained from the transformation system by plugging in the arrays y, $\dot{y}$, $\ddot{y}$

(evaluated at each discretization step t) for $x, v, \dot{v}$ and solving for f. Thus, the target for f in case of BTS (3.1) is given by

$$f_{\text{target}} = \frac{-K(g-y) + D\dot{y} + \tau\ddot{y}}{g - x_0} \ , \tag{3.6}$$

and in case of ATS (3.5) by

$$f_{\text{target}}(s) = \frac{\tau\,\ddot{y} + D\,\dot{y}}{K} - (g-y) + (g-x_0)\,s \ . \tag{3.7}$$

The initial position x_0 is given by $y(0)$ and the corresponding goal g is given by $y(T)$. To obtain a matching input for f_{target}, the canonical system needs to be integrated, i.e. $s(t)$ is evaluated. For this purpose, in Eq. (3.2), the initial state of the canonical system is set to $s(0) = 1$ before integration. The time constant τ is chosen such that the DMP with $f = 0$ achieves 95 % convergence at $t = T$. With this procedure, a supervised learning problem is obtained over the time course of the movement to be approximated with training samples (s, f_{target}). The obtained function approximation problem can then be solved efficiently using a locally weighted learning approach, which fits local linear models weighted with Gaussian basis functions $\psi_i(s)$. In standard Locally Weighted Regression (LWR), these basis functions, more precisely their centers c_i and bandwidths h_i, are fixed such that learning the weights θ_i is sufficient to approximate the nonlinear function f. LWR was chosen due to its very fast one-shot learning procedure and the fact that individual kernels learn independently of each other, which will be a key component to achieve a stable parameterization that can be used for movement recognition in the evaluations (see Sec. 3.7). The accuracy of the approximation can be tuned by the number of basis functions, in general, the more basis function, the better the approximation. In the limit, choosing one basis function for every training point, the movement could be reproduced without any error. Thus, setting the number of these learning parameters to a pre-defined quantity, typically 20, restricts the ability of DMPs to learn and accurately reproduce more complex movements.

3.3 Movement Generation

Both transformation systems are at equilibrium when the current position x is equal the goal position g and $\dot{v} = s = 0$. To trigger a movement and obtain a desired plan $(x_{\text{desired}}, v_{\text{desired}}, \dot{v}_{\text{desired}})$ the system needs to be set up such that the condition for the unique equilibrium point $((x, v, s) = (0, g, 0))$ is no longer satisfied. Therefore, the start position is set to the current position $(x_0 \leftarrow x)$, the goal position is set to the desired goal $(g \leftarrow g_{desired} \neq x)$, and the canonical system is reset by setting $s \leftarrow 1$. The desired attractor landscape is obtained through plugging the learned set of weights θ_i, as well as the learned set of basis functions ψ_i (characterized by their centers c_i and bandwidth h_i)

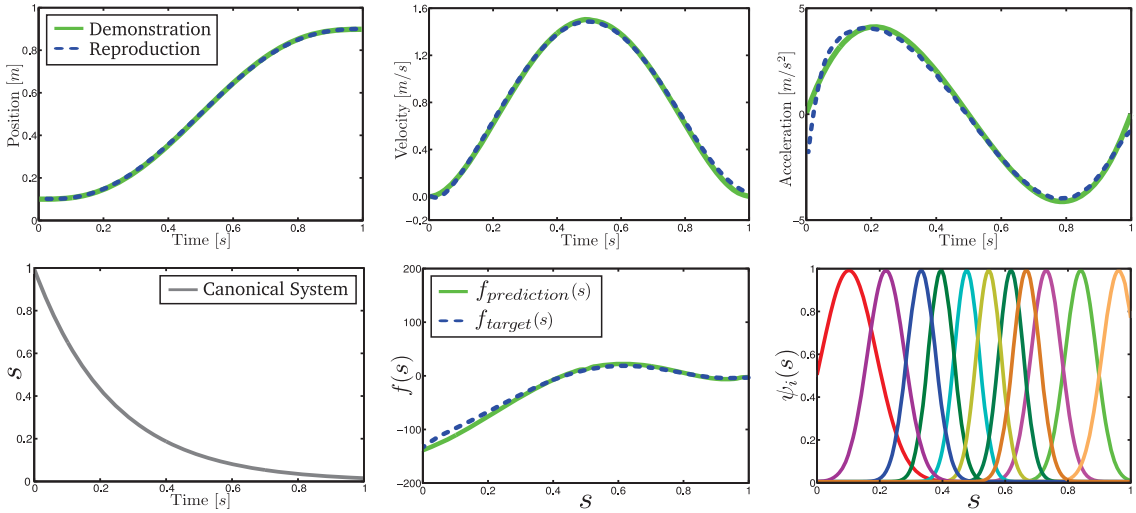


Figure 3.2: Demonstrated minimum jerk movement (top) represented by position $y(t)$, velocity $\dot{y}(t)$, and acceleration profile $\ddot{y}(t)$ (green solid lines) and reproduced movement using a fitted DMP (blue dashed lines). Time evaluation of the corresponding canonical system (bottom left), computed target function f_{target} (green solid lines) vs. learned function f (blue dashed lines) (bottom middle), and 11 basis functions $\psi_i(s)$ (bottom right).

into the nonlinear function f. Finally, the desired movement duration is adjusted using τ. After this setup procedure, the desired movement plan is obtained by integrating the canonical system, i.e. evaluating $s(t)$, computing the nonlinear forcing term $f(s)$ and integrating the transformation system, as sketched in Fig. 3.1. To reproduce an observed trajectory, the start and goal position as well as the movement duration are set to the values obtained from demonstration, see Fig. 3.2.

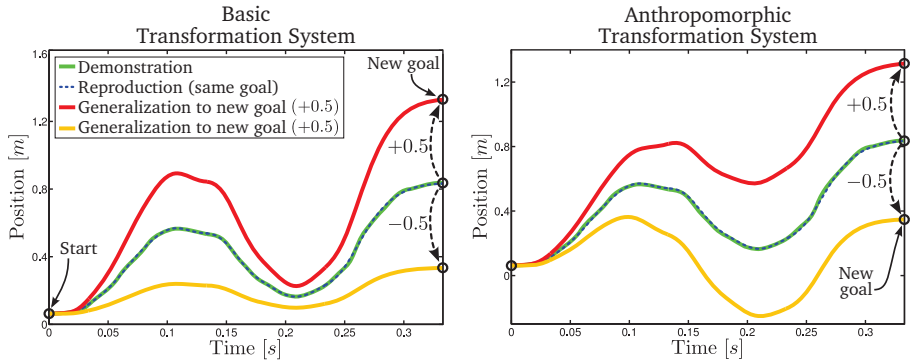


Figure 3.3: Demonstrated movement (green solid line) and reproduced movement using a fitted primitive (blue dashed line). Furthermore two generalizations to new goals obtained by changing the goal parameter to $g_{\text{new}} \leftarrow g + 0.5$ (red solid line) and $g_{\text{new}} \leftarrow g - 0.5$ (orange solid line). The behaviors of the two different transformation systems are compared: The basic transformation system Eq. (3.1) scales the movement amplitude linearly (left). The anthropomorphic transformation system Eq. (3.5) gradually adapts to the movement (right).

The explicit appearance of the goal parameter in both transformation systems allows for movement adaptation to new goals by simply adapting g prior to integrating the dynamic system. As mentioned before, the trajectories obtained by changing the goal variable for the two transformation systems differ, as shown in Fig. 3.3: The basic transformation system (left) generates trajectories that scale linearly with the change of the goal. The anthropomorphic transformation system (right) generates trajectories that adapt to the new goal slowly.

3.4 Special Properties of Dynamic Systems

The presented formulation of DMPs has the following favorable characteristics:

Structural Stability The fundamental property of both transformation systems is that they are structurally stable, i.e. that the qualitative behavior of the trajectories is unaffected by perturbations. The speed at which perturbations are rejected is determined by the spring parameter K and D, which are usually chosen such that the system is critically damped. For such parametrization, it is easy to proof bounded-input-bounded-output (BIBO) stability (Friedland, 1986) of both systems as the magnitude of the forcing function f is bounded by virtue that all terms of the function, i.e., basis functions $\psi_i(s)$, weights θ_i, and phase variable s in Eq. (4.1) are bounded by design. Furthermore, given that $f(s)$ decays to zero, as $s \to 0$, both formulations form a globally stable linear dynamic system with $(v, x) = (0, g)$ as a unique point attractor.

Spatial and Temporal Invariance Another fundamental property of both DMP formulations is that movements can be encoded such that they become spatially and temporally invariant. Translating the start x_0 and goal g variable to a new start $x_0' \leftarrow x_0 + c$ and new goal $g' \leftarrow g + c$ results in a trajectory $\hat{y}'$, that is translated by the same offset c, i.e. $\hat{y}' = \hat{y} + c$, where $\hat{y}$ is the trajectory obtained by integrating the system using start x_0 and goal g. Scaling only the goal variable results in a different trajectory, however, the topology of the attractor landscape is preserved (see Fig. 3.3). Similarly, changing the parameter τ only scales the duration of the movement. Thus, the amplitude and duration of learned patterns can be independently modified without affecting the qualitative shape of the resulting trajectory $\hat{y}$.

Multiple Degrees-of-Freedom DMPs Both DMP formulations presented above encode one dimensional movement plans and are thus only suitable for single degree-of-freedom (DOF) systems. However, an extension to multiple DOF is rather straightforward. The simplest form employs a separate DMP for every DOF. Alternatively, all DMPs could share the same canonical system, but have separate transformation systems as well as separate forcing terms for each DOF as sketched in Fig. 3.4. In this approach, the canonical system becomes a central clock, providing the temporal coupling between DOFs, while the transformation system achieves the desired attractor dynamics for each

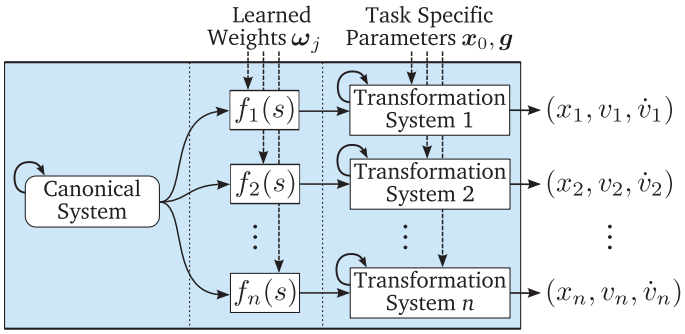


Figure 3.4: Sketch of an n dimensional DMP: The n transformation systems are perturbed by individually learned nonlinear functions f_j, $(j = 1, \ldots, n)$, which are driven and synchronized by a single canonical system.

DOF. Thus, n dimensional DMPs are realized by replacing the scalar variables $x, v, \dot{v}, x_0$, and g with n dimensional vectors $\boldsymbol{x}, \boldsymbol{v}, \dot{\boldsymbol{v}}, \boldsymbol{x}_0$, and $\boldsymbol{g}$, the constants K and D with diagonal matrices $\boldsymbol{K}$ and $\boldsymbol{D}$, where $\boldsymbol{K}_{jj} \leftarrow K$ and $\boldsymbol{D}_{jj} \leftarrow D$, for $j = 1, ..., n$, and f with $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)^\mathsf{T}$.

Synchronization and entrainment phenomena Using dynamic systems for modeling movements facilitates the use of coupling phenomena, i.e. entraining external signals into the trajectory modulation. Coupling terms can affect either the canonical system, the transformation system, or both systems. Introducing a coupling term in the canonical system affects primarily the temporal evolution and allows for instance to synchronize drumming at the beat of music (Pongas et al., 2005). Introducing a coupling term in the transformation system affects primarily the spatial evolution and hence allows to modify the trajectory shape.

Introducing a coupling in both systems can be used to achieve robustness against perturbation. For some tasks for example, it may be desirable to adjust the movement plan such that it accounts for interaction with the environment. Otherwise, large discrepancies between desired position and actual position of the robot will occur, which, as the controller tries to compensate, will cause damage to either the environment or the robot itself. Instead, as introduced in (Ijspeert et al., 2002), the dynamical system allows feeding back an error term between actual $\tilde{x}$ and desired position x into the control policy, such that the time evolution of the policy is smoothly paused during a perturbation, i.e. x is modified to remain close to $\tilde{x}$, and resumes after the perturbation vanishes. Note that other (task-specific) ways to cope with perturbations can be designed. The ability to introduce such coupling terms is one of the most interesting features of using autonomous differential equations for control policies. In Sec. 3.8, we exploit this property to realize online obstacle avoidance, and later in Sec. 5.3, we exploit the same property to realize contact reactive behaviors.

Online Modulation Desired trajectories are generated by integrating differential equations as described in Sec. 3.3. Thus, complete trajectories for a given movement task are not (necessarily) generated beforehand. Instead, desired trajectories are modulated online allowing the previously mentioned coupling terms to adapt trajectories dynamically according to changes in the environment. For example, as shown later in Sec. 3.8, the coupling term that realizes obstacle avoidance adapts the trajectory on the fly without the need for re-planning. Furthermore, trajectories can also be adapted to changes in the task, for example when the goal changes after movement onset. Such behavior can be realized simply by updating the goal parameter g of the dynamical system to the new goal. Different from previously mentioned methods of adapting the shape of the modulated trajectory, changing the goal variable corresponds to changing the attractor dynamics of the system. The re-parameterized system therefore is guaranteed to converge to the new goal. Fig. 3.5 shows a two dimensional DMP generalized to 4 new goal positions before movement onset as well as during the movement.

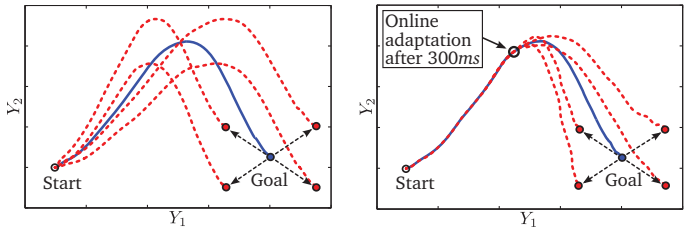


Figure 3.5: Two dimensional DMP (Y_1, Y_2) learned from recorded trajectory (solid blue) with one transformation system for each dimension. The trajectories (red dashed) obtained by changing the goal to 4 new positions before movement onset (left) and during the movement (right). Here, the anthropomorphic transformation system Eq. (3.5) has been used.

Online modulation also allows to adapt the overall movement duration τ in order to speed up or slow down the movement. Importantly, the movement goal g and the movement duration τ can be adapted in every time step. This feature of using dynamic systems for movement generation facilitates the realization of very reactive behaviors, a crucial pre-requisite for many tasks, especially in the context of manipulation.

Composing complex behaviors from multiple DMPs Due to the previously mentioned spatial and temporal invariance of the DMP formulation, a straightforward approach of sequential movement can be realized by setting up each DMP appropriately and executing one after the other. However, since DMPs are required to start and end with zero velocity and acceleration, the resulting movement using this method of sequencing has complete standstills between the individual movements. Although this characteristic might be desirable for some tasks (e.g. a pick and place operation), it may not for others (e.g. between forehand and backhand tennis swings). In this second case, the formulation offers a method to avoid resting in between two DMPs by starting the new DMP before the current one has finished. Choosing appropriate initial conditions for subsequent DMPs allows to join movements such that the overall velocity profile is continuous, as

illustrated in Fig. 3.6. Further extensions that also ensure continuous acceleration profiles are presented in Sec. 3.5.1.

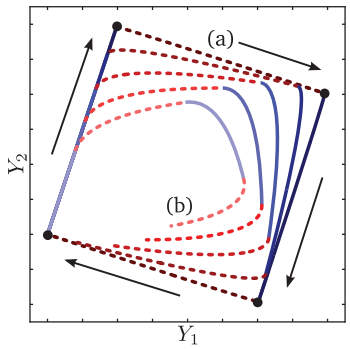


Figure 3.6: Sequencing of 4 straight 2 dimensional minimum-jerk movement primitives. Endpoints are marked by black dots (a). The movements generated by the DMPs are shown in alternating blue solid and red dashed lines to indicate the transition between two successive DMPs. The movement direction is indicated by arrows. The remaining movements (b) result from using different switching times (a lighter color indicates an earlier switching time).

Superposition of DMPs Complex movements are not only achieved through sequencing of primitives but also by their superposition: For instance, a discrete DMP could be employed to shift the setpoint of a rhythmic DMP, thus generating a point-to-point movement with a superimposed periodic pattern. For example, with this strategy it is possible to bounce a ball on a racket by producing an oscillatory up-and-down movement in joint space of the arm, and use the discrete system to make sure the oscillatory movement remains under the ball such that the task can be accomplished (Schaal, Sternad, & Atkeson, 1996). This superposition is obtained by propagating the systems in parallel and summing up the resulting force fields similar to (Giszter, Mussa-Ivaldi, & Bizzi, 1993). Similarly, crawling humanoids can be realized by superimposing rhythmic movements for crawling with discrete modulations for precise hand placements to reach specific marks on the ground(Degallier, Righetti, & Ijspeert, 2007). Other forms of superposition are conceivable, however, evaluation of the most promising strategies remains future work.

3.5 Design Principles and Variations

In the development of our model equations (see Sec. 3.1), we made specific choices for the canonical system, the nonlinear function approximator, and the transformation system. However, it is important to point out that it is the design principle of our approach that is the most important, and not the particular equations that we chose for our realization. As sketched in Fig. 3.1, there are three main ingredients in our approach.

The canonical system is a simple (or well-understood) dynamical system that generates a behavioral phase variable, which is our replacement for explicit timing. For instance,

while we chose a simple first-order linear system as canonical system Eq. (3.2) for discrete movements, it is straightforward tp use a 2nd order system instead (Ijspeert et al., 2003), or even nonlinear equations.

With the input from the canonical system, the nonlinear function approximator generates a forcing term. We used Locally Weighted Regression (LWR) to generate the function approximator Eq. (4.1), as it is very easy to fit to desired behavior, and has variants to automatically determine the correct number of basis functions (Schaal & Atkeson, 1998). However, other function approximators could be used, too, e.g., radial basis function networks, mixture models, or Gaussian Process regression, etc. (Bishop, 2006). Using a function approximator that is linear in the parameters seems to be among the most important criteria for a good choice.

The transformation system should be a dynamical system that, when excited with a forcing term is easily analyzed and manipulated. We used a critically damped linear spring system, but other systems, like higher order or lower order dynamical systems are equally possible. This choice is partially guided by which level of derivatives the output behavior of the entire dynamical system should have. We presented two such realizations of the transformation systems, Eq. (3.1) and Eq. (3.5). For example in (Ijspeert et al., 2013), various modifications to Eq. (3.1) have been proposed. See Sec. 3.6 for a discussion on the choice of transformation system.

Finally, a multitude of coupling terms can be incorporated to realize very reactive closed loop behaviors. In Sec. 3.8 we will present one particular realization that implements obstacle avoidance.

All systems together should fulfill the principle of structural equivalence, should be autonomous, and should have easy analyzable stability properties. Obviously, a large variety of model equations can be generated, tailored for different contexts.

3.5.1 Extensions

When switching the goal $\boldsymbol{g}$ to a new goal $\boldsymbol{g'} \neq \boldsymbol{g}$, the transformation system Eq. (3.1) and Eq. (3.5) generate a discontinuity in the accelerations $\dot{\boldsymbol{v}}$. This can be avoided by filtering the goal change with a simple first order differential equation

$$\tau \dot{\boldsymbol{g}} = \alpha_g \left(\boldsymbol{g}_0 - \boldsymbol{g} \right) \ . \tag{3.8}$$

In this formulation, $\boldsymbol{g}_0$ is the discontinuous goal change, while $\boldsymbol{g}$ is now a continuous variable. The time constant α_g determines the speed of convergence to the new goal $\boldsymbol{g}_0$. This modification does not affect the scaling properties and stability properties of the system, and is easily incorporated in the learning algorithm with LWR.

Movement primitives can be sequenced, as shown in Fig. 3.6, such that the overall velocity profile is continuous. However, unless the movement system comes to rest in between subsequent movements the accelerations will have discontinuities at the transitions. To

ensure continuity, one can either employ a higher order transformation system, similar to (Nemec et al., 2009), or use the following set of filters:

$$\tau \dot{\boldsymbol{g}} = \alpha_g \left(\boldsymbol{g}_0 - \boldsymbol{g} \right) \tag{3.9}$$

$$\tau \dot{\boldsymbol{f}} = \alpha_f \left(\boldsymbol{f}_0(s) - \boldsymbol{f}(s) \right) \tag{3.10}$$

$$\ddot{\tau} = \alpha_\tau \left(\beta_\tau (\tau_0 - \tau) - \dot{\tau} \right) \ . \tag{3.11}$$

The time constants α_g, α_f, α_τ, and β_τ determine the speed of convergence to the new values of $\boldsymbol{g}$, $\boldsymbol{f}(s)$, and τ, respectively. Various modified formulations have been suggested by (Nemec & Ude, 2011; Kulvicius et al., 2011), that also provides continuous accelerations.

3.6 Movement Generalization

One key feature of the presented DMP equations (see Sec. 3.1) is the ability to generalize to new goals. Changing a single control parameter, the goal $\boldsymbol{g}$, adapts the trajectory such that it ends at this new goal. Importantly, the generated movement trajectory is guaranteed to converge at this new goal. However, different realizations of the transformation system, e.g. Eq. (3.1), or Eq. (3.5), generate different trajectories. As noted in (Pastor et al., 2009), pre-multiplying the forcing term $\boldsymbol{f}(s)$ with $(\boldsymbol{g} - \boldsymbol{y}_0)$ has three drawbacks: first, if start point $\boldsymbol{x}_0$ and goal $\boldsymbol{g}$ coincide in at least one dimension, then the forcing term $\boldsymbol{f}(s)$ in Eq. (3.1) cannot drive the system away from its initial state in that dimension; thus, the system will remain at $\boldsymbol{x}_0$. Second, the scaling of $\boldsymbol{f}(s)$ with $(\boldsymbol{g} - \boldsymbol{x}_0)$ is problematic if $(\boldsymbol{g} - \boldsymbol{x}_0)$ is close to zero; here, a small change in $\boldsymbol{g}$ may lead to huge accelerations, which may not be desirable. Third, whenever a movement adapts to a new goal $\boldsymbol{g}_{\text{new}}$ such that $(\boldsymbol{g}_{\text{new}} - \boldsymbol{x}_0)$ changes its sign compared to $(\boldsymbol{g}_{\text{original}} - \boldsymbol{x}_0)$ the resulting generalization is mirrored, see Fig. 3.7 (left). This problem can be circumvented by pre-multiplying

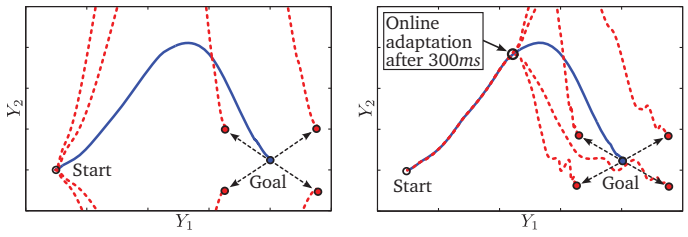


Figure 3.7: Two dimensional DMP (Y_1, Y_2) learned from recorded trajectory (solid blue) with one transformation system for each dimension. The trajectories (red dashed) obtained by changing the goal to 4 new positions before movement onset (left) and during the movement (right). This example illustrates the aforementioned drawbacks of pre-multiplying the forcing term with $(\boldsymbol{g} - \boldsymbol{x}_0) \approx 0$ in Eq. (3.1), here in Y_2; changing the goal either scales the trajectory significantly, or mirrors the trajectory as shown in the left plot.

$\boldsymbol{f}(s)$ with a fixed amplitude, for example the movement amplitude $\mathbf{A} = \max \boldsymbol{x} - \min \boldsymbol{x}$. Similarly, one could adjust the scaling depending on the task.

The anthropomorphic transformation system uses such a fixed scaling. We tested how the adaptation to a changing goal $\boldsymbol{g}$ in Eq. (3.5) compares with human behavior. Therefore we recorded human movements with a stylus on a graphics tablet (Hoffmann & Schaal, 2007). Subjects made curved movements towards a target displayed on a screen. During some of the trials (200 ms after movement onset), the target jumped to a different location, and we observed how the subjects adapted their movement to the new goal. The observed movement adaptation could be explained by changing $\boldsymbol{g}$ in Eq. (3.5) - see Fig. 3.8 and (Hoffmann, 2011). The DMP equations were fitted from the average of the subject's average movement to the original goal.

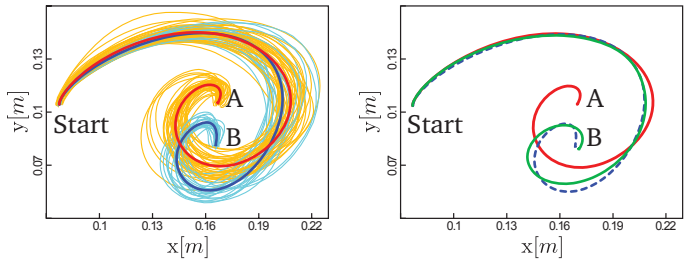


Figure 3.8: Goal adaptation of the anthropomorphic transformation system compared to human behavior. Raw trajectories on a graphics tablet for one subject (orange and cyan) and their means (red and blue) are shown on the left. The red curve is the movement to the original goal (A), and the blue curve is the adaptation to the switching target (B). The adaptation of our dynamical system to the new target (green) is compared with the experimental data (blue dashed) on the right. Plot has been adopted from (Hoffmann & Schaal, 2007).

It is important to note that movement generalization is very task specific. Furthermore, even for particular tasks, e.g. drawing a spiral to a switching targets as shown in Fig. 3.8, there might be a huge variance across subjects, suggesting that there are many *correct* ways of generalizing movements. Our approach is to exploit the formulation's ability to generalize to new targets only in a localized setting.

In conclusion, the invariance properties in our dynamical systems model are mathematically well-founded, but the choice of coordinates for representing a model can make a big difference in how generalization of the dynamics systems appears. From a practical point of view, one should first carefully investigate what properties a model requires in terms of temporal and spatial invariance, and then realize these properties by choosing the most appropriate variant of the dynamical systems model and the most appropriate coordinate system for modeling.

3.7 Movement Recognition

Given the spatial and temporal invariance of our policy representation, trajectories that are topologically similar tend to be fit by similar parameters $\boldsymbol{\theta}_i$. This property holds for both transformation systems and therefore opens the possibility of using our representation for movement recognition (Ijspeert et al., 2002). To illustrate this idea, we carried out a simple task of fitting trajectories performed by a human user when drawing two-dimensional single-stroke patterns. The first six letters of the Graffiti alphabet used in hand-held computers were chosen and presented to the subject, who was asked to re-draw each of them a total of six times (see Fig. 3.9). These characters are drawn in a single stroke, and are fed as a two-dimensional trajectory $(y_1(t), y_2(t))$ to be fitted by a two degree-of-freedom DMP. Similarities between two characters a and b have been measured by computing the correlation $\boldsymbol{\theta}_a{}^{\mathsf{T}}\boldsymbol{\theta}_b \,/\, |\boldsymbol{\theta}_a||\boldsymbol{\theta}_b|$ between their parameter vectors $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$. These vectors are the union, i.e. $\boldsymbol{\theta}_a = [\boldsymbol{\theta}_a^{y_1}, \boldsymbol{\theta}_a^{y_2}]$ and $\boldsymbol{\theta}_b = [\boldsymbol{\theta}_b^{y_1}, \boldsymbol{\theta}_b^{y_2}]$, of the parameter vectors for the $y_1(t)$ and $y_2(t)$ trajectories.

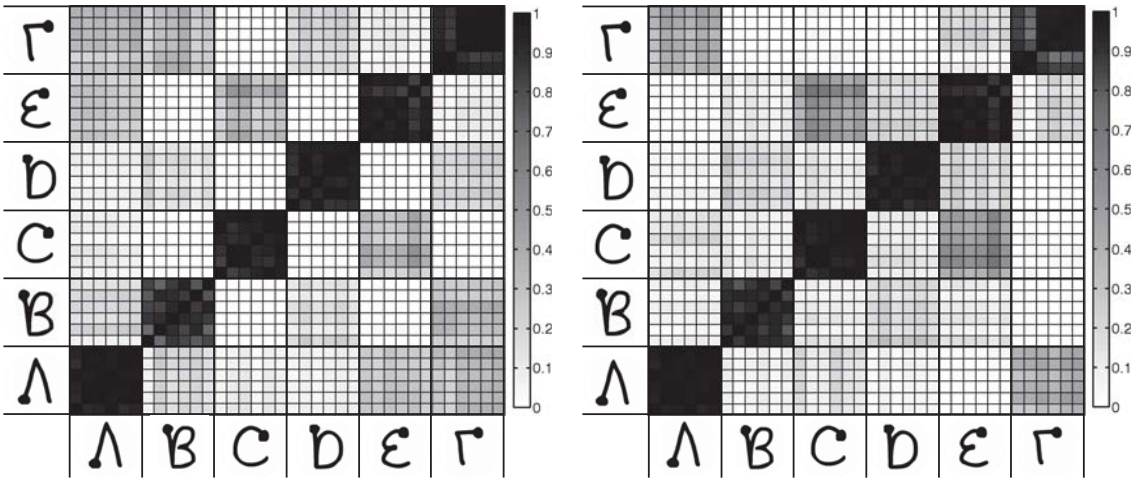


Figure 3.9: Correlation between the parameter vectors of different instantiations of the Graffiti characters (6 instances of each of the first 6 characters of the graffiti alphabet) using the BTS (left) and using the ATS (right). A gray scale value is used with black corresponding to a correlation of 1.0 and white corresponding to a correlation of 0.0 or below. It is important to note that spatio-temporal patterns are recognized, not just spatial patterns. For example, both formulations indicate some correlation between the instantiations of 'A' and 'F', which have different spatial patterns, due to the similarities of their acceleration profiles.

Fig. 3.9 shows the correlations between all 36 instantiations of the characters using both formulations. As illustrated by high correlation values in the diagonal of the correlation matrix, the correlations between instances of the same character tend to be systematically higher than correlations between instances of different characters. These similarities in weight space can therefore serve as basis for recognizing demonstrated movements by fitting them and comparing the fitted parameters $\boldsymbol{\theta}_i$ with those of previously learned

movements in memory. It should be noted that such recognition is recognition of spatio-temporal patterns, not just spatial patterns. For example, the correlation values obtained from computing the similarities between 'A' and 'F', which show different spatial pattern, are positive (see top left and lower right 6x6 square in both graphs in Fig. 3.9) due to the similarities of their acceleration profiles. Hence, the same representation that allows to generate desired trajectories for motor output also serves to recognize observed behaviors. An interesting approach to movement segmentation and recognition using DMPs has been suggested in (Meier et al., 2011, 2012). However, extending this concept to our robotic manipulator remains future work.

3.8 Spatial Coupling for Obstacle Avoidance

Obstacle avoidance is nicely suited to demonstrate the power of coupling terms in our approach (Hoffmann et al., 2009; Pastor et al., 2009). Coupling terms can affect either the transformation system, the canonical system, or both systems. In this section, we address a coupling term in the transformation system only, which will primarily affect the spatial evolution $(\boldsymbol{x}, \boldsymbol{v}, \dot{\boldsymbol{v}})$. Practically, we add a coupling term $\boldsymbol{C}_t$ to the transformation system Eq. (3.5), to become

$$\begin{aligned} \tau \dot{\boldsymbol{v}} &= \boldsymbol{K}(\boldsymbol{g} - \boldsymbol{x}) - \boldsymbol{D}\,\boldsymbol{v} - \boldsymbol{K}(\boldsymbol{g} - \boldsymbol{x}_0)s + \boldsymbol{K}\boldsymbol{f}(s) + \boldsymbol{C}_t \\ \tau \dot{\boldsymbol{x}} &= \boldsymbol{v} \ . \end{aligned} \tag{3.12}$$

A prudent choice of the coupling term is critical and often needs to be specialized for different objectives. The design of coupling terms is a research topic by itself. A typical example from the domain of motor control is obstacle avoidance with the help of potential fields (Khatib, 1986). Traditionally, obstacles are modeled as repelling potential fields which are designed to automatically push a control system to circumnavigate them in an on-line reactive way, instead of pre-planning. Such reactive behavior assumes that obstacles may appear in an unforeseen and sudden way, such that pre-planning is not possible or useful. In a previous study (Park et al., 2008), we explored the negative gradient of a velocity-dependent potential field $\boldsymbol{C}t(\boldsymbol{x}, \boldsymbol{v}) = -\nabla_{\boldsymbol{v}}\Phi(\boldsymbol{x}, \boldsymbol{v})$. However, potential field methods produce a resultant vector that directly controls the movement direction which can cause sudden changes in direction and therefore unsmooth trajectories. Instead, here we present a dynamical model that produces an angular acceleration that controls the steering direction. In (Fajen & Warren, 2003) a model was suggested that empirically models human walking behaviors in the presence of obstacles. Their approach considers the agent's steering angle with respect to the goal and potential obstacles. We adopted the approach in (Fajen & Warren, 2003) and extended it to 3D obstacle avoidance, including moving obstacles. First, we will present our approach for single static obstacle avoidance, then multiple obstacles, and, finally, moving obstacles. We will also proof that

the introduced coupling into the DMP equations does not affect the convergence to the goal g. Real robot experiments are presented in Sec. 4.2.3.

The steering angle φ (see. Fig. 3.10 (right)) is adapted according to

$$\dot{\varphi} = \gamma \, \varphi \exp(-\beta \, |\varphi|) \ , \tag{3.13}$$

where γ and β are positive model parameters (see. Fig. 3.10 (left)).

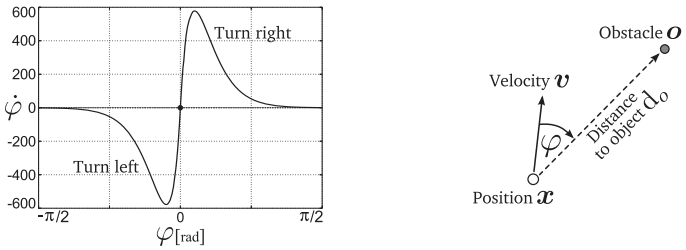


Figure 3.10: Phase plot obtained from Eq. (3.13) illustrating the rate of change in steering angle φ. The unstable fixed point at $\varphi = 0.0$ is theoretically possible, nevertheless real world sensor noise will break the symmetry. The angular turning rate $\dot{\varphi}$ is determined by the angle of attack φ. If the obstacle appears on the right ($\varphi < 0$) the system will turn left and vice versa. The model parameters γ and β determine the amplitude and the bandwidth of the turning rate, here $\gamma = 10000$ and $\beta = 20.0/\pi$.

The coupling term $\boldsymbol{C}_t(\boldsymbol{x}, \boldsymbol{v})$ in Eq. (3.12) is computed from the obtained change of steering angle $\dot{\varphi}$ in Eq. (3.13) according to

$$\boldsymbol{C}_t(\boldsymbol{x}, \boldsymbol{v}) = \mathbf{R} \, \boldsymbol{v} \, \dot{\varphi} \ , \tag{3.14}$$

where $\mathbf{R}$ is a rotational matrix with axis $\boldsymbol{r} = (\boldsymbol{o} - \boldsymbol{x}) \times \boldsymbol{v}$ and angle of rotation of $\pi/2$; vector $\boldsymbol{o}$ is the position of the obstacle. Thus, the vector $\mathbf{R}\,\boldsymbol{v}$ lies in the plane spanned by $(\boldsymbol{o} - \boldsymbol{x})$ and $\boldsymbol{v}$ and is perpendicular to the velocity vector $\boldsymbol{v}$ as shown in Fig. 3.11.

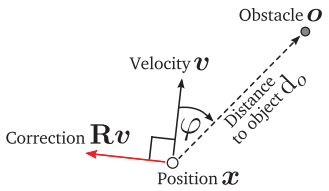


Figure 3.11: The correction vector $\mathbf{R}\,\boldsymbol{v}$ (red) lies in the plane spanned by $(\boldsymbol{o} - \boldsymbol{x})$ and $\boldsymbol{v}$ and is perpendicular to the velocity vector $\boldsymbol{v}$. This vector is scaled by $\dot{\varphi}$ to compute the final object induced change in velocity. Note that the direction of the correction changes by $180\,^o$ if $\dot{\varphi} < 0$.

Thus, the coupling term that induces change in velocity is

$$\boldsymbol{C}_t(\boldsymbol{x}, \boldsymbol{v}) = \gamma \, \mathbf{R} \, \boldsymbol{v} \, \varphi \exp(-\beta \, \varphi) \ , \tag{3.15}$$

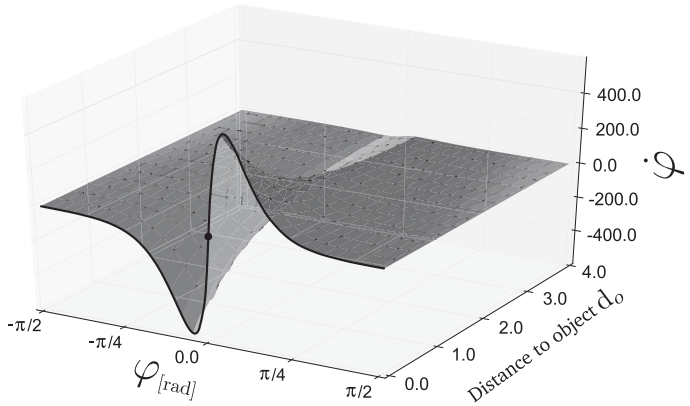


Figure 3.12: Phase plot of $\dot{\varphi} = \gamma\,\varphi \exp(-\beta\,\varphi)\,\exp(d_o)$. The rate of change in steering angle $\dot{\varphi}$ as function of the angle of attack φ and distance d_o to the obstacle ($\gamma = 10000$ and $\beta = 20.0/\pi$). The turning rate increases as the distance to the obstacle decreases.

where $\varphi = \cos^{-1}\big((\boldsymbol{o}-\boldsymbol{x})\cdot\boldsymbol{v}/||\boldsymbol{o}-\boldsymbol{x}||\cdot||\boldsymbol{v}||\big)$; Note, this value is always positive. Thus, the obstacle induced change of velocity is a function of the angle of attack at which the obstacle appears and the current velocity. To decrease the influence of the obstacle with its distance d_o to the current position a penality term can be multiplied with the expression according to

$$\boldsymbol{C}_t(\boldsymbol{x}, \boldsymbol{v}) = \gamma\,\mathbf{R}\,\boldsymbol{v}\,\varphi \exp(-\beta\,\varphi)\,\exp(-d_o)\ , \tag{3.16}$$

where $d_o = \|\boldsymbol{o} - \boldsymbol{x}\|$ is the distance between the obstacle $\boldsymbol{o}$ and the current position $\boldsymbol{x}$, see Fig. 3.10 (right). Fig. 3.12 shows the exponential decay of the influence of the obstacle with distance.

3.8.1 Avoiding Multiple Obstacles

The Eq. (3.16) is extended to consider multiple obstacles by summing over each obstacle's induced change of rate according to

$$\boldsymbol{C}_t(\boldsymbol{x}, \boldsymbol{v}) = \gamma \sum_{i}^{\#\text{obstacles}} \mathbf{R}_i\,\boldsymbol{v}\,\varphi_i \exp(-\beta\,\varphi_i)\,\exp(-d_i)\ , \tag{3.17}$$

where $\mathbf{R}_i$, φ_i, and d_i are the corresponding rotation matrix around $\boldsymbol{r} = (\boldsymbol{o}_i - \boldsymbol{x}) \times \boldsymbol{v}$ and angle of rotation of $\pi/2$, the angle of attack, and the distance to object i. Fig. 3.13 illustrates the obstacle behavior in 2D (Y_1, Y_2). The discrete dynamical system is initialized with a minimum jerk movement (Flash & Hochner, 2005), which is frequently used as an approximate model of smooth human movement. On the way to the goal state, obstacles are positioned randomly that need to be avoided.

The time constants K, D in the DMP equations Eq. (3.12) determine the speed at which the dynamical system returns to the original movement.

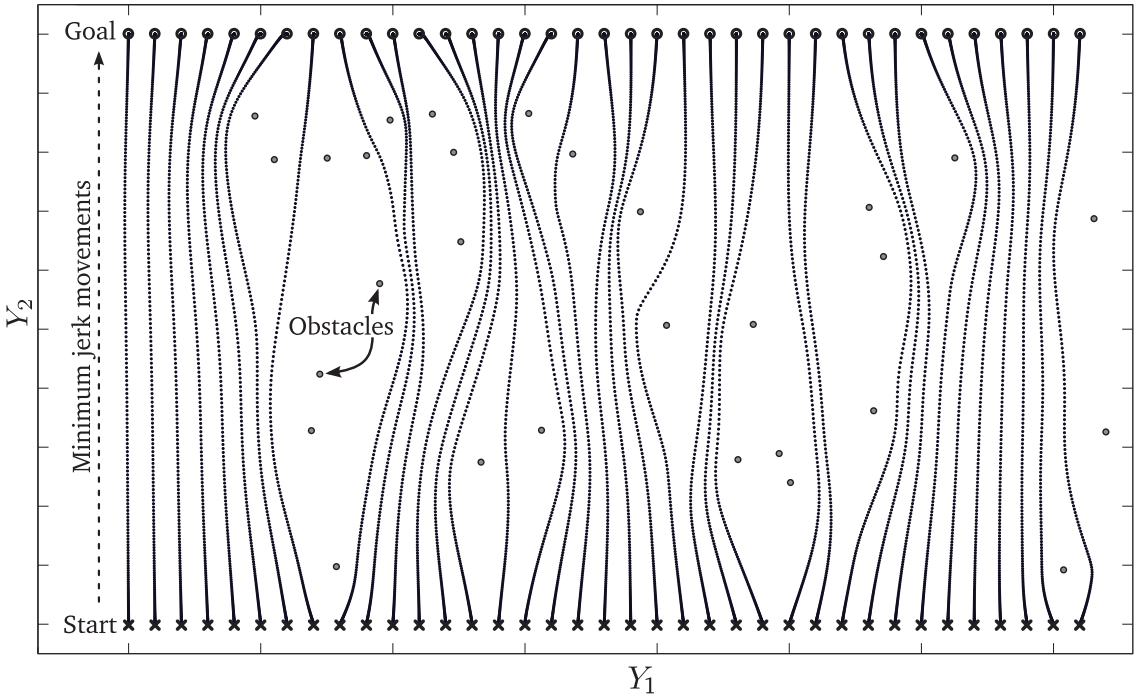


Figure 3.13: Example obstacle avoidance behavior in 2D. The discrete dynamical system (Y_1, Y_2) is initialized with minimum jerk movement and executed from several initial positions (bottom) to several goal positions (top). 30 Obstacles have been placed randomly in between. The obtained behavior illustrates typical avoidance behavior. The same model parameters as in Fig. 3.12 have been used.

3.8.2 Moving Obstacles in 2D

To extend Eq. (3.17) to moving obstacles we need to consider the relative velocity between the position of the movement system $\boldsymbol{x}$ and the obstacle $\boldsymbol{o}$. Therefore, for each obstacle i, we compute φ_i in the reference frame of the obstacle,

$$\varphi_i = \cos^{-1} \frac{(\boldsymbol{o}_i - \boldsymbol{x})^\mathsf{T} (\boldsymbol{v} - \boldsymbol{o}_i)}{|(\boldsymbol{o}_i - \boldsymbol{x})||(\boldsymbol{v} - \boldsymbol{o}_i)|} \ , \tag{3.18}$$

where $\dot{\boldsymbol{o}}_i$ is the velocity of obstacle i. Eq. (3.16) needs to be adapted as well to the relative velocity according to

$$\boldsymbol{C}_t(\boldsymbol{x}, \boldsymbol{v}) = \gamma \sum_{i}^{\#\text{obstacles}} \mathbf{R}_i(\boldsymbol{v} - \dot{\boldsymbol{o}}_i)\varphi_i \exp(-\beta\varphi_i) \ . \tag{3.19}$$

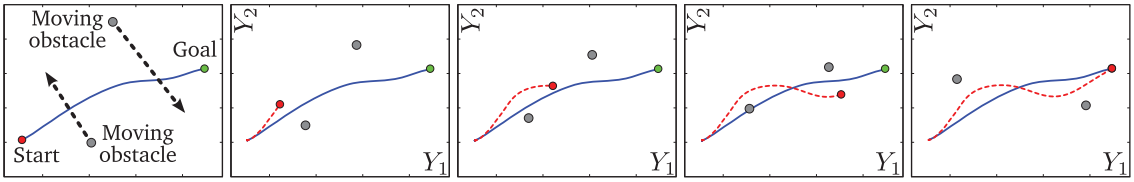


Figure 3.14: Avoiding moving obstacles in 2D space: Two dimensional DMP (Y_1, Y_2) learned from recorded trajectory (solid blue) and perturbed by 2 moving point obstacles (gray) with constant velocities indicated by the arrows in the left most plot.

3.8.3 Proof of Convergence for Static Obstacles

In the following, we show that (3.12) with (3.17) converges to the goal position $\boldsymbol{g}$. First, we demonstrate convergence for one obstacle, and, then, extend to many obstacles. For $t \to \infty$, the terms in (3.12) that depend on s approach 0 exponentially; thus, we just need to study convergence of the reduced equation

$$\dot{\boldsymbol{v}} = \mathbf{K}(\boldsymbol{g} - \boldsymbol{x}) - \mathbf{D}\boldsymbol{v} + \gamma \mathbf{R}\boldsymbol{v}\varphi \exp(-\beta\varphi)\, \exp(-d_o) \ . \tag{3.20}$$

The state $(\boldsymbol{x}, \boldsymbol{v}) = (\boldsymbol{g}, 0)$ is a stationary point in the equation. All other states converge to this point, which we show by constructing a Lyapunov function (Khalil, 2002). As Lyapunov function candidate $V(\boldsymbol{x}, \boldsymbol{v})$, we use the energy of the linear spring system, $\dot{\boldsymbol{v}} = \mathbf{K}(\boldsymbol{g} - \boldsymbol{x}) - \mathbf{D}\boldsymbol{v}$ (here, using unit mass),

$$V(\boldsymbol{x}, \boldsymbol{v}) = \frac{1}{2}(\boldsymbol{g} - \boldsymbol{x})^{\mathsf{T}}\mathbf{K}(\boldsymbol{g} - \boldsymbol{x}) + \frac{1}{2}\boldsymbol{v}^{\mathsf{T}}\boldsymbol{v} \ . \tag{3.21}$$

To prove convergence, we need to show $\dot{V} < 0$ for $\boldsymbol{v} \neq 0$,

$$\begin{aligned} \dot{V} &= \nabla_{\boldsymbol{x}} V^{\mathsf{T}} \dot{\boldsymbol{x}} + \nabla_{\boldsymbol{v}} V^{\mathsf{T}} \dot{\boldsymbol{v}} \\ &= -(\boldsymbol{g} - \boldsymbol{x})^{\mathsf{T}}\mathbf{K}\boldsymbol{v} + \boldsymbol{v}^{\mathsf{T}}\dot{\boldsymbol{v}} \\ &= -\boldsymbol{v}^{\mathsf{T}}\mathbf{K}(\boldsymbol{g} - \boldsymbol{x}) + \boldsymbol{v}^{\mathsf{T}}\mathbf{K}(\boldsymbol{g} - \boldsymbol{x}) - \boldsymbol{v}^{\mathsf{T}}\mathbf{D}\boldsymbol{v} \\ &\quad + \gamma \boldsymbol{v}^{\mathsf{T}}\mathbf{R}\boldsymbol{v}\varphi \exp(-\beta\varphi) \\ &= -\boldsymbol{v}^{\mathsf{T}}\mathbf{D}\boldsymbol{v} < 0 \ . \end{aligned} \tag{3.22}$$

The damping matrix $\mathbf{D}$ is negative definite by choice to guarantee damping. The term $\boldsymbol{v}^{\mathsf{T}}\mathbf{R}\boldsymbol{v}$ is 0 since the matrix $\mathbf{R}$ is a rotation by 90 degrees*. If $\boldsymbol{v} = 0$ and $\boldsymbol{x} \neq \boldsymbol{g}$, then $\dot{V} = 0$; however, $\dot{V}$ changes if $\boldsymbol{x} \neq \boldsymbol{g}$; thus, according to LaSalle's theorem (Khalil, 2002), $\boldsymbol{x}$ converges to $\boldsymbol{g}$. Therefore, we have shown that the introduced coupling term does not affect the convergence guarantees to the goal $\boldsymbol{g}$ for single obstacles.

For multiple obstacles using Eq. (3.17) the same energy term (3.21) is again a Lyapunov function of the system since each term $\boldsymbol{v}^{\mathsf{T}}\mathbf{R}_i\boldsymbol{v}$ vanishes. Thus, global convergence is guaranteed also for many obstacles.

*The inner product of two orthogonal vectors is zero

For moving obstacles the convergence proof does not hold because $\boldsymbol{v}^\mathsf{T}\mathbf{R}_i\dot{\boldsymbol{o}}_i$ does not vanish. However, in most cases, for $t \to \infty$, the obstacles will either move away from the robot ($\boldsymbol{C}_t(\boldsymbol{x}, \boldsymbol{v}) = 0$) or come to a standstill (reduction to static case). Thus, the robot motion will converge again.

Chapter 4

Learning Manipulation Skills

In this chapter, we present several experimental evaluations of applying our approach to learning attractor systems (see Chapter 3) for object manipulation using both, simulation and robotic studies. The evaluations are intended to demonstrate the properties of our methodology, but also the domain-specific choices that need to be made. In particular, we will present the frameworks ability of learning movement primitives from demonstrations to build up a library of movements, generalizing movements to new targets as well as to dynamically changing targets, realizing online obstacle avoidance with the use of coupling terms, and sequencing of several movement primitives to place a cup on a table as well as to grasp a bottle and pour water into it. Furthermore, in Sec. 4.3, we present experimental results on using trial-and-error to improve the initial policy. Note that for all experiments in this thesis, including those presented in this chapter, the anthropomorphic transformation system (Pastor et al., 2009) introduced in the previous chapter, Eq. (3.5), has been used.

4.1 Adding Prior Knowledge with Imitation Learning

Imitation learning is a intuitive way of introducing prior knowledge for robots to learn a particular manipulation task: A human teacher provides a demonstration that achieves a particular task and the robot encodes this demonstration into a control policy (Schaal, 1997). There are several conceivable ways of recording movement trajectories. The arguably most convenient one would involve visual observation and appropriate processing to obtain the relevant task variables, however, this perceptual component is out of the scope of this thesis. Instead, we follow a more direct approach, termed kinesthetic teaching, which involves moving the robot itself. We have used the same principle of teaching manipulation skills on three different robots: The Sarcos master-slave system consisting of a 10 DOF exoskeleton master arm and a 10 DOF slave robot, the Willow Garage PR2 robot, and the ARM-S robot (see Appendix A for detailed descriptions of these systems).

Figure 4.1: Sarcos Master arm used to record human trajectories in end-effector space. Here, the subject demonstrates a grasping, placing, and releasing movement.

The first experimental evaluations have been conducted using the Sarcos master-slave system. To record a set of movements, we used the exoskeleton robot arm shown in Fig. 4.1. To account for the correspondence problem, when teacher and student do not share the same embodiment, we choose to represent movement trajectories in end-effector space. For object manipulation – here, placing and pouring – besides the end-effector position, we also control the orientation of the gripper, the arm posture, and the position of the fingers. The DMP framework allows to combine the end-effector motion with any additional DOF; thus, adding gripper orientation in quaternion notation, desired arm joint configurations (to be optimized in the nullspace of the task) and finger joint configurations is straight-forward (see Fig. 4.2).

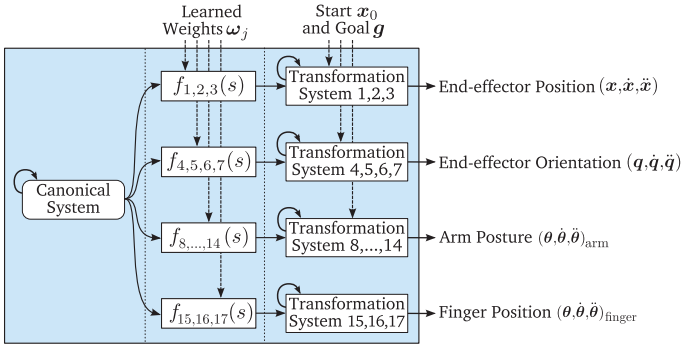


Figure 4.2: Sketch of the 17 dimensional DMP used to generate movement plans for the Sarcos Slave arm.

4.2 Building a Library of Movements

Learning DMPs requires the user to demonstrate characteristic movements. For movement reproduction only a simple high level command - to choose a primitive (or a sequence of them) and set its task specific parameters - is required. Moreover, adaptation to new situations is accomplished by adjusting the start $\boldsymbol{x}_0$, the goal $\boldsymbol{g}$, and the movement duration τ. Thus, such a collection of primitives, the *motion library*, enables a system to generate a wide range of movements. can be employed to facilitate movement recognition

in that observed movements can be compared to the pre-learned ones. If no existing primitive is a good match for the demonstrated behavior, a new one is created (learned) and added to the system's movement repertoire.

4.2.1 Movement Generation

Movement generation involves retrieving an appropriate DMP from the motion library, specifying the task specific parameters, propagating the dynamical system, and converting the obtained movement plan into motor commands. In the presented implementation, shown in Fig. 4.2, the task specific parameters, i.e. the start $\boldsymbol{x}_0$ and the goal $\boldsymbol{g}$, are the end-effector position, end-effector orientation, and the finger joint configuration. The start $\boldsymbol{x}_0$ is set according to the current state of the robot, while the goal $\boldsymbol{g}$ and the duration τ are set according to the constraints of the movement task. For a grasping movement, for example, the goal is set to the position of the object that is to be grasped and the goal configuration of the finger are set depending on the size and shape of that particular object. However, finding an appropriate goal orientation is not straight forward, as the end-effector orientation needs to be adapted to the characteristic approach curve of the movement. Approaching the object from the front results in a different final posture as approaching it from the side. In case of a grasping movement, we developed a method to automatically determine the final orientation by propagating the DMP to generate the Cartesian space trajectory and averaging over the velocity vectors to compute the approach direction at the end of the movement. For other movements, like placing and releasing, we set the end-effector orientation to the orientation recorded from human demonstration. Finally, we use τ to determine the duration of each movement. The resulting desired trajectory is converted into motor commands using task space control.

4.2.2 Task Space Control

To execute DMPs on the robot we used a velocity based inverse kinematics controller (Nakanishi et al., 2007, 2008), which transforms the task space reference velocities $\dot{\boldsymbol{x}}_r$ into the reference joint space velocities $\dot{\boldsymbol{q}}_r$, as sketched in Fig. 4.3.

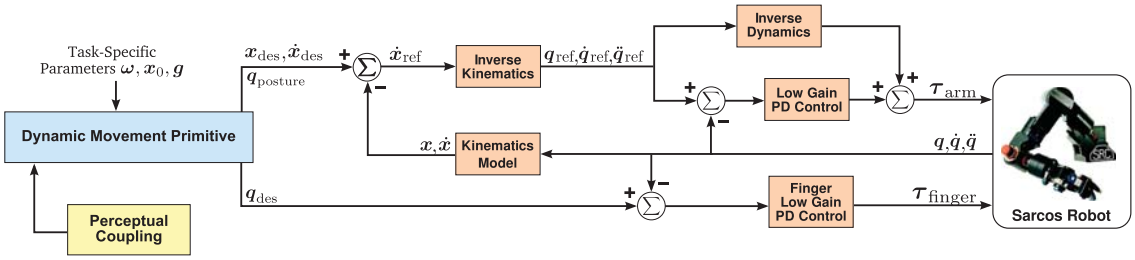


Figure 4.3: DMP control diagram: The desired task space positions and velocities are $\boldsymbol{x}_des$, $\dot{\boldsymbol{x}}_des$, the reference task space velocity commands are $\dot{\boldsymbol{x}}_r$, the reference joint positions, joint velocities, and joint accelerations are $\boldsymbol{q}_r$, $\dot{\boldsymbol{q}}_r$, and $\ddot{\boldsymbol{q}}_r$.

The reference joint position $\boldsymbol{q}_r$ and acceleration $\ddot{\boldsymbol{q}}_r$ are obtained by numerical integration and differentiation of the reference joint velocities $\dot{\boldsymbol{x}}_r$. The desired orientation, given by the DMP as unit quaternions (see Fig. 4.2), is controlled using quaternion feedback as described in (Yuan, 1988; Nakanishi et al., 2008). The reference joint position, velocities and acceleration are transformed into appropriate torque commands $\boldsymbol{u}$ using a feed-forward and a feedback component. The feed-forward component estimates the corresponding nominal torques to compensate for all interactions between the joints, while the feedback component realizes a PD controller. Simulation results for the placing tasks are shown in Fig. 4.4.

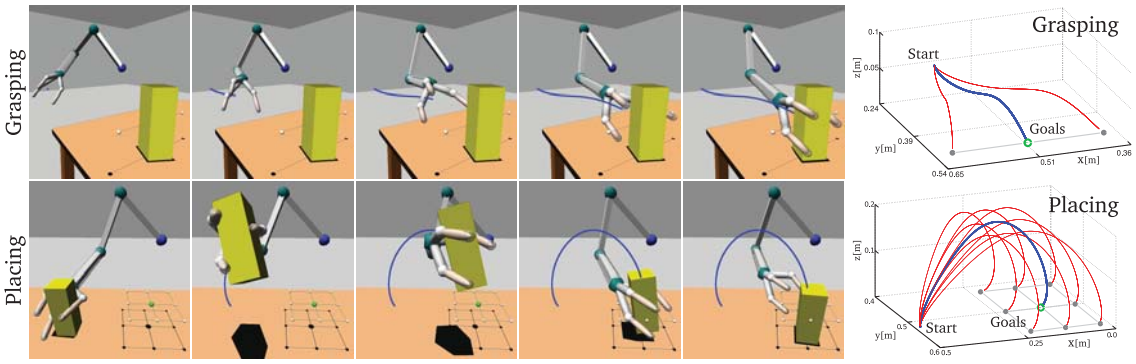


Figure 4.4: Snapshots of the simulated grasping and placing (left) and corresponding 3D plots of obtained movement generalizations (right).

Figure 4.5 shows the Sarcos Slave robot arm grasping the red bottle and pouring water into the green cup. The object positions were obtained using a calibrated color based stereo camera system. It is important to note that the entire motion was generated from sequencing only four DMPs. The goal for the grasping movement was setup to the red bottle and the goal for the pouring was setup with respect to the green cup. The goal for the movement that puts the bottle in an upright position as well as the releasing movement were manually chosen. The location of the cup is changed after each pouring movement to illustrate the abilities of the DMPs to generalize both, in position and orientation. See `http://youtu.be/Ge0GduY1rtE` for a video showing similar experiments performed on the PR2 robot.

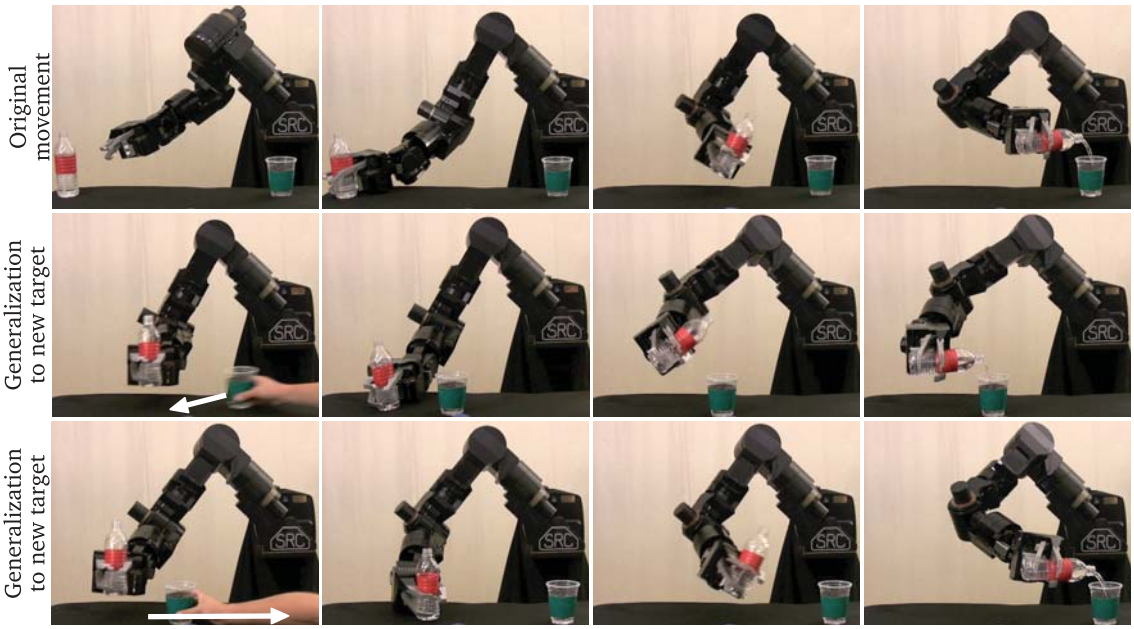


Figure 4.5: Sarcos Slave arm grasping a red bottle and repetitively pouring water into a green cup. While the robot is repositioning the bottle, the cup position is placed on different locations on the table (white arrow). The first row shows the robot reproducing the recorded movement. The second and third row show the generalization abilities of DMPs to new goals. The video of this experiment is available at `http://youtu.be/QIA9nFaU1cc`.

4.2.3 Online Adaptation and Obstacle Avoidance through Perceptual Coupling

As described Sec. 3.4, the DMPs are modulated online in order to take dynamic events from the environment into account, for example, when the task consists of grasping a cup that changes its location while approaching it. Such behavior is simply accomplished by updating the goal variable of the transformation system, which corresponds to changing (moving) the point attractor of the dynamical system. The trajectories obtained through integrating our dynamic system model with changing goal positions smoothly adapt and converge at the final goal. Thus visual-servoing can be realized by updating the goal variable using the perceived position of the object. This ability is demonstrated in the second row of Fig. 4.6: the goal of the placing movement (green coaster) changes after movement onset and the robot arm adapts to the new goal online.

Furthermore, perceptual coupling can also be used to modify the trajectory shape. Here, we exploit the robustness against perturbation of our model for online obstacle avoidance as introduced in Sec. 3.8. In the robot experiment shown in the third row of Fig. 4.6 we used $\gamma = 1000$ and $\beta = 20/\pi$.

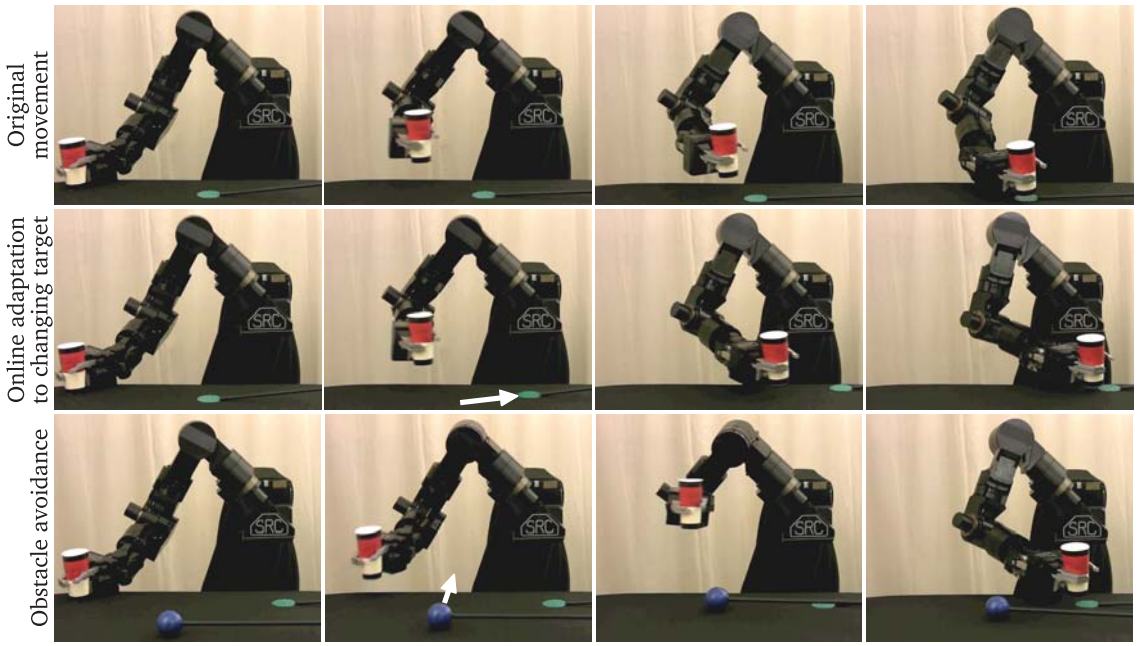


Figure 4.6: Sarcos Slave arm placing red cup on a green coaster. The first row shows the placing movement on a fixed goal. The second row shows the DMPs ability to adapt to changing goals (white arrow) after movement onset. The third row shows the resulting movement as a blue ball-like obstacle interferes with the placing movement. The video of this experiment is available at `http://youtu.be/LuFlWNIcdfM`.

Level of Competence

The presented obstacle avoidance method is a local method and has all the shortcomings of local methods (Khatib, 1986). A feasible path to the goal may not be found due to the local perspective of the task at hand. Feasible paths can only be guaranteed in the presence of convex objects. Non-convex objects can be made convex by considering their convex hull. Such an approach however may render large parts of the free space to be part of the obstacle and are suboptimal.

Further shortcomings are the fact that only avoidance with the robots endeffector (modelled as a point) are considered (Khansari-Zadeh & Billard, 2012; Hoffmann et al., 2009). This approach can be extended to avoid link collisions in the nullspace of the task, similar to a previous study (Park et al., 2008). However, this extension requires redundant manipulators that have a sufficiently large nullspace, which is rarely the case in practise. Finally, additional care needs to be taken to avoid joint limits. Adding repelling force fields at the limit of each joint (Khatib, 1986) may address this issue, but also increase the risk of getting stuck in a local minima. Therefore we want to emphasize that the presented approach is meant to implement very fast and reactive *local* behaviors. To generate a feasible path in more complex environments we initialize the DMP using a global planner, e.g. (Kalakrishnan, Chitta, et al., 2011).

4.3 Learning Motor Skills through Trial-and-Error

Learning motor skills from demonstration is a powerful approach to directly learn control policies as shown in the previous sections. However, depending on the task at hand, these initial policies are often feasible but not very robust, i.e. its success rate on a series of repeated experiments will often be very low. Especially tasks that require contact interactions with objects (see Fig. 4.7) are prone to fail due to the high degree of uncertainty, if the initial policy is reproduced. However, designing specialized controllers or behaviors targeted towards a specific task often times require manual tuning, a painstaking process performed by a human expert.

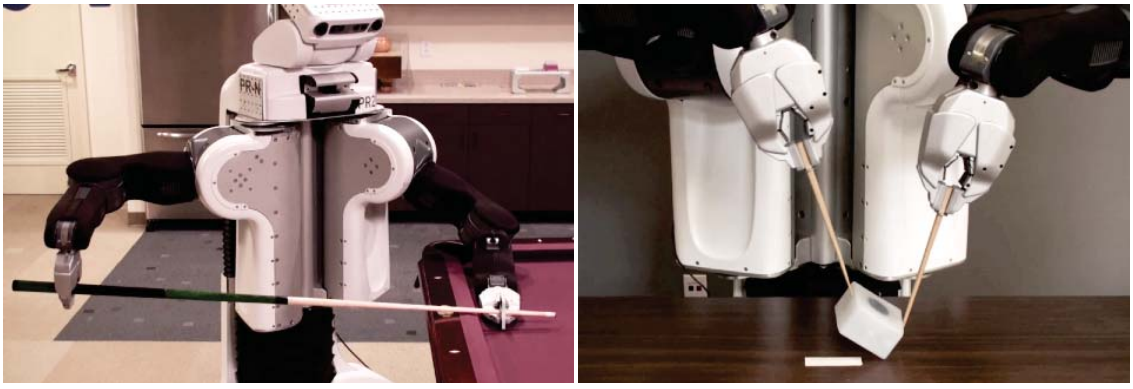


Figure 4.7: The considered manipulation tasks evaluated on the two armed PR2 robot: Learning a strong and accurate pool stroke (left) and gently flipping a box with chopsticks (right).

A promising approach to increase task success without (too much) human intervention is Reinforcement Learning (RL). Provided a task specific cost function, the robot could apply trial-and-error learning, i.e. sample new trajectories, execute these trajectories, evaluate their cost, update the policy accordingly, and repeat until a satisfactory task performance is achieved. Unfortunately, especially in high dimensional spaces such an approach is prone to fail as with a very high probability rather useless trajectories would be generated. Instead, in trajectory-based RL, biasing sampling towards good initial conditions seems to be the key for high dimensional applications (Schaal, 1999). Such an important prior is introduced when using the initial policy learned from demonstration in the previous section. The search for the (locally) optimal solution can focus around this initial policy. In this section, we will present such a trajectory-based approach, that explores a region around an initial policy to obtain an optimal solution, the final policy. Our approach is based on the PI^2 algorithm (Theodorou et al., 2010; Theodorou, 2011), which, due to its minimal number of manual tuning parameters, has been shown to work robustly and efficiently in contrast to previous RL approaches, even in very high dimensional motor systems.

Nevertheless, our approach still requires task-specific input: (1) defining a task-specific cost function and (2) defining a task-specific coordinate transformation. We will expand upon each of these components in this section before illustrating them further with examples in the next section.

4.3.1 Task Space Representation

Manipulation tasks often impose constraints on the degrees of freedom of a robotic system. Most RL approaches to manipulation have used either the joint or end-effector space to represent the task variables. However, task variables can often be chosen appropriately so as to naturally conform to the constraints while *not* eliminating possible solutions. Further, task domain knowledge can also be used to define the most meaningful space for exploration, e.g. the noise level $\Sigma_{\boldsymbol{\epsilon}}$ can be set according to the task.

4.3.2 Cost Function

The specification of a task must include a definition of a cost function to be minimized. Unfortunately, the cost function requires careful tuning. It has to be designed to reward task success appropriately while penalizing task failure. A promising approach to automatically infer these cost functions from demonstrations has been proposed by (Kalakrishnan, 2014), however, the cost functions used in the experiments here are manually specified.

The remainder of this section is structured as follows: First, we will review the PI^2 algorithm in Sec. 4.3.3, second, we will provide and overview of the learning setup in Sec. 4.3.4, finally, we will present experimental results on the PR2 robot in two different manipulation tasks. The first task involves the robot learning a strong and accurate pool stroke. This task was selected to highlight the system's ability to learn a dynamic task. This ability is crucial, since obtaining good demonstrations in dynamic settings is challenging if not impossible. The second task involves the robot to learn how to gently flip a box upright using chopsticks. This task was selected to show that the presented approach can also be used to learn very dexterous manipulation skills.

4.3.3 Policy Improvement using Path Integrals: PI^2

In this section, we will briefly outline the **P**olicy **I**mprovement with **P**ath **I**ntegrals ($\mathbf{PI}^2$) reinforcement learning algorithm and will show how it can be used to optimize DMPs. A more detailed derivation of the PI^2 algorithm as well as comparison with related work in the areas of stochastic optimal control and reinforcement learning can be found in (Theodorou et al., 2010; Theodorou, 2011).

DMPs are parameterized policies that guarantee attractor properties towards the goal. Varying the DMP parameters $\boldsymbol{\theta}$ results in changing the shape of the trajectory while

the initial state x_0 and the goal g remains fixed. An important property of the DMP formalism is that the function approximator is linear in these parameters and therefore can be written as

$$f(s) = \frac{\sum_i \psi_i(s)\,\theta_i\,s}{\sum_i \psi_i(s)} = \frac{\Psi^\mathsf{T}(s)\,s}{\sum_i \psi_i(s)}\,\boldsymbol{\theta} = \boldsymbol{g}^\mathsf{T}(s)\,\boldsymbol{\theta} \ . \tag{4.1}$$

This property has been exploited previously (see Sec. 3.2) to use efficient regression algorithms to learn these shape parameters in a supervised learning setting. Here, this property will be used to realize trial-and-error learning (J. Peters & Schaal, 2008; J. Peters, 2007).

DMPs are part of the general class of stochastic differential equations of the form

$$\dot{\boldsymbol{x}}_t = \boldsymbol{p}(\boldsymbol{x}_t, t) + \boldsymbol{g}^\mathsf{T}(\boldsymbol{x}_t)(\boldsymbol{\theta}_t + \boldsymbol{\epsilon}_t) \ , \tag{4.2}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{1+2n}$ denotes the state of the system at time t, $\boldsymbol{\theta}_t \in \mathbb{R}^m$ is the parameter vector of our DMP function approximator in Eq .(4.1) with m basis functions at time t, $\boldsymbol{p}(\boldsymbol{x}_t, t)$ the passive dynamics, $\boldsymbol{g}(\boldsymbol{x}_t) \in \mathbb{R}^m$ the control transition matrix, and $\boldsymbol{\epsilon}_t \in \mathbb{R}^m$ Gaussian noise with variance $\Sigma_{\boldsymbol{\epsilon}}$. Note, the state of the system $\boldsymbol{x}_t = (s, \boldsymbol{x}, \boldsymbol{v})^\mathsf{T}$, therefore contains the phase variable s from the canonical system Eq. (3.2) and the positions $\boldsymbol{x}$ and velocities $\boldsymbol{v}$ of the transformation system Eq. (3.5). The control parameters $\boldsymbol{\theta}$ partition the DMP into a controlled part $\boldsymbol{x}_t^{(c)} = \boldsymbol{v}$ and an uncontrolled part $\boldsymbol{x}_t^{(u)} = (s\ \boldsymbol{x}^\mathsf{T})^\mathsf{T}$. Thus, $\boldsymbol{g}(\boldsymbol{x}_t)$ corresponds to the control transition matrix and depends on the state, however, it depends only on one of the state variables of the uncontrolled part of the state, that is, s. For fixed length trajectories and without temporal coupling, as considered in the experiments presented in the following, this variable becomes deterministic and is therefore shorten to read $\boldsymbol{g}_t$.

The immediate cost function is defined as

$$r(t) = \phi_t + \frac{1}{2}\,\boldsymbol{\theta}_t^\mathsf{T}\,\mathbf{R}\,\boldsymbol{\theta}_t \ , \tag{4.3}$$

where ϕ_t is an arbitrary state-dependent cost function and $\mathbf{R} \in \mathbb{R}^{m\times m}$ is the positive semi-definite weight matrix of the quadratic control cost*. The total cost of a behavior is defined as

$$R(\tau) = \phi_T + \int_0^T r(t)\,dt \ , \tag{4.4}$$

where ϕ_T is a special terminal cost. The cost $J = \mathbf{E}_\tau\{R(\tau)\}$ to be optimized is the average over all experienced trajectories τ, such that the optimal cost is the value function

*The quadratic control cost matrix $\mathbf{R}$ is a user design parameter and usually chosen to be diagonal and invertible.

$V = \min_{\boldsymbol{\theta}_{0:T}} E\{R(\tau)\}$. The resulting **P**olicy **I**mprovement with **P**ath **I**ntegrals (**PI**^{2}) is summarized in Table 4.1.

Essentially, in step 2.2, the term $P(\boldsymbol{\tau}_{k,t})$ is the discrete probability at time t of each trajectory rollout k that is computed with the help of the cost $S(\boldsymbol{\tau}_{k,t})$ in step 2.1. For every time step of the trajectory, a parameter update $\delta\boldsymbol{\theta}_t$ is computed in step 3.1 based on a probability weighted average cost over the sampled trajectories. The parameter updates at every time step are finally averaged in step 4 by giving every m-th coefficient of the parameter vector an update weight according to the time steps left in the trajectory and the activation of corresponding Gaussian kernel ψ_m. The final parameter update $\boldsymbol{\theta}^{(\text{new})} = \boldsymbol{\theta}^{(\text{old})} + \delta\boldsymbol{\theta}$ takes place in step 5. The entire formulation allows an iterative updating of $\boldsymbol{\theta}$ until the desired performance is achieved corresponding to the trajectory cost R having converged. The parameter λ regulates the sensitivity of the exponentiated cost and can automatically be optimized for every time step t to maximally discriminate between the experienced trajectories.

4.3.4 System Architecture

An overview of the proposed system that implements the PI^2 Reinforcement Learning algorithm is sketched in Fig. 4.8.

The system consists of four components: The specification of the initial policy, which is achieved through kinesthetic teaching; the task specification, i.e. a cost function and task space representation; the reinforcement learning system that optimizes the initial policy; and a sensor based system that collects statistics of successful trials to predict the outcome of future tasks during execution.

The only modules not shared across applications and that need to be engineered for each particular task are the task-specific coordinate transformation (see Sec. 4.3.1) and the cost function that specifies the desired task outcome (see Sec. 4.3.2).

4.3.5 Learning a Pool Stroke

The goal of the pool task is to learn a pool stroke motion that maximizes the resulting speed of the cue ball while precisely hitting a target location on the pool table.

Experimental Setup

The experimental setup consists of the PR2 robot holding the cue stick in its right gripper and the custom-made bridge in its left gripper (see Fig. 4.9). The demonstration for initialization with imitation learning was obtained through kinesthetic demonstration. During the teaching phase, the controllers of the robot's right arm have been switched off, allowing a human demonstrator to easily guide the cue stick to hit the cue ball.

Input:

Initial DMP parameters $\boldsymbol{\theta}$
Basis function matrix $\boldsymbol{g}_t$ from the system dynamics
Immediate cost function ϕ_t and control cost $\mathbf{R}$
Terminal cost term ϕ_T
Variance $\Sigma_{\boldsymbol{\epsilon}}$ of the mean-zero noise $\boldsymbol{\epsilon}$

Offline: Precompute projection matrices $\mathbf{M}_t$

Step 0: For $t = 1..T$, do:

$\mathbf{M}_t \leftarrow \frac{\mathbf{R}^{-1}\boldsymbol{g}_t\boldsymbol{g}_t^\mathsf{T}}{\boldsymbol{g}_t^\mathsf{T}\mathbf{R}^{-1}\boldsymbol{g}_t}$a

---

Online: Repeat until convergence of the trajectory cost R:

Step 1: Create K rollouts $\boldsymbol{\tau}_k, k = 1..K$ of the system each containing T trajectory points from the same start state x_0 using stochastic parameters $\boldsymbol{\theta} + \boldsymbol{\epsilon}_k$

Step 2: For $k = 1..K$, for $t = 1..T$, do:

Step 2.1: Compute the cost-to-go S of trajectory $\boldsymbol{\tau}_k$ at timestep t

$S(\boldsymbol{\tau}_{k,t}) \leftarrow \phi_{T,k} + \sum_{i=t}^{T-1} \phi_{i,k} + \frac{1}{2}\sum_{i=t}^{T-1} (\boldsymbol{\theta} + \mathbf{M}_i\boldsymbol{\epsilon}_k)^\mathsf{T}\mathbf{R}(\boldsymbol{\theta} + \mathbf{M}_i\boldsymbol{\epsilon}_k)$

Step 2.2: Compute the probability P of trajectory $\boldsymbol{\tau}_k$ at timestep t

$P(\boldsymbol{\tau}_{k,t}) \leftarrow \frac{e^{-\lambda S(\boldsymbol{\tau}_{k,t})}}{\sum_{i=1}^{K} e^{-\lambda S(\boldsymbol{\tau}_{i,t})}}$

Step 3: For $t = 1..T-1$, do:

Step 3.1: Compute the parameter update $\delta\boldsymbol{\theta}_t$ at timestep t

$\delta\boldsymbol{\theta}_t \leftarrow \sum_{k=1}^{K} \left[P(\boldsymbol{\tau}_{k,t})\, \mathbf{M}_t\boldsymbol{\epsilon}_k\right]$

Step 4: Compute averaged parameter update $\delta\theta_m = \frac{\sum_{t=0}^{T-1}(T-i)\,\psi_t^{(m)}\delta\theta_t^{(m)}}{\sum_{t=0}^{T-1}(T-i)\psi_t^{(m)}}$

Step 5: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \delta\boldsymbol{\theta}$

Step 6: Create one noiseless rollout to compute the trajectory cost R

$R = \phi_T + \sum_{t=0}^{T-1} r_t$

Table 4.1: Pseudocode of the $\mathbf{PI}^2$ algorithm for a one dimensional DMP policy.

The duration of the demonstration has been set to 1 second. The cue stick, which is held in the right gripper and constraint by the bridge held in the left gripper, closes a kinematic chain and therefore imposes a constrain on the right gripper orientation during movement execution. Encoding the demonstrated movement in joint or end-effector space would violate these task constraints during exploration and create unnecessary internal torques. Instead, by choosing an appropriate task frame, exploration can be restricted to the constraint-satisfying manifold of the particular task. In case of the pool stroke, the appropriate task variables are the translational offset from the right gripper to the bridge, the roll, pitch, and yaw angles of the cue around the bridge, and the redundant degree-of-freedom in the arm, as illustrated in Fig. 4.9 (right). The recorded joint trajectories first have been transformed into the described task space and then encoded in a single DMP with 5 transformation systems as described in Sec. 3.2. To enable autonomous learning, the pool table has been tilted such that the cue ball rolls back after each pool shot and a special apparatus (see Fig. 4.9) has been installed which guides the downwards rolling ball to its original position. Furthermore, a Hokyuo laser scanner has been mounted on the opposing side of the pool table to measure both where and when the cue ball crosses the scan line. A high-bandwidth Bosch DMA150 digital accelerometer embedded in the palm of robot's gripper and sampled at rate of 3 kHz has been utilized to detect the point

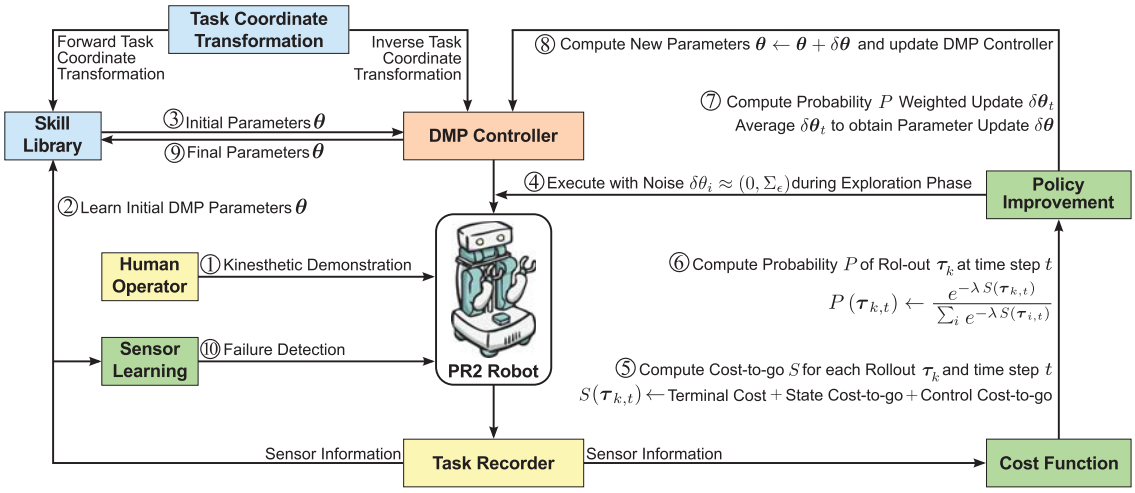


Figure 4.8: Proposed system architecture for learning manipulation tasks: (1) A human demonstrates a particular manipulation skill through kinesthetic teaching. Recorded sensor information is encoded into a DMP (2) and passed to the DMP controller (3). To explore the region around the initial trajectory, noise is added to the DMP parameters $\boldsymbol{\theta}$ (4). During the trials, sensor information is recorded to first compute the cost-to-go (5), second, to compute corresponding probabilities (6), third, to compute the probability weighted parameter update $\delta\boldsymbol{\theta}_t$ (7), and fourth, average these parameter updates to obtain the parameter vector $\boldsymbol{\theta}$ to be used in the next iteration (8). Once satisfactory task performance is achieved the learned DMP parameters are stored in the skill library (9). During subsequent task execution sensor information is collected to facilitate progress monitoring and online failure detection (10).

in time at which the cue stick impacts with the ball. Therefore, the accelerometer signals have been band-pass filtered to contain data between 40-1000 Hz.

The variance of the mean-zero (exploration) noise $\Sigma_{\boldsymbol{\epsilon}}$ for the 5 task space dimensions (see Fig. 4.9) has been set to $\epsilon_{\text{cue tip offset}} = 0.14$, $\epsilon_{\text{cue roll}} = \epsilon_{\text{cue pitch}} = \epsilon_{\text{cue yaw}} = 0.01$, and $\epsilon_{\text{elbow posture}} = 0.1$.

Cost Function

The task objective of the pool stroke is to maximize the cue ball speed, while attempting to have the ball cross the center of the scan line (see Fig. 4.9). Equivalent to maximizing the speed of the cue ball is minimizing the ball travel duration t_{b} which is computed from the difference of the point in time at which the cue stick impacts the ball to the point in time at which the ball crosses the scan line. The target offset error x_{e} is defined to be the offset from the target location. In order to avoid the cue stick possibly slipping out of the bridge during exploration, a high cost has been assigned whenever the backwards displacement of the cue tip $x_{\text{cue tip offset}}$ exceeds a certain threshold. The terminal cost ϕ_T is computed as weighted according to

$$\phi_T = \omega_1 \, t_{\text{b}}^2 + \omega_2 \, x_{\text{e}}^2 + \omega_3 \, f(x_{\text{cue tip offset}}) \ , \tag{4.5}$$

where $f(x_{\text{cue tip offset}})$ is equal to 1 if the backwards displacement boundary has been violated during the rollout, or 0 otherwise. For the learning experiment, the following weights have been used: $\omega_1 = 100$, $\omega_2 = 10000$, and $\omega_3 = 10000$.

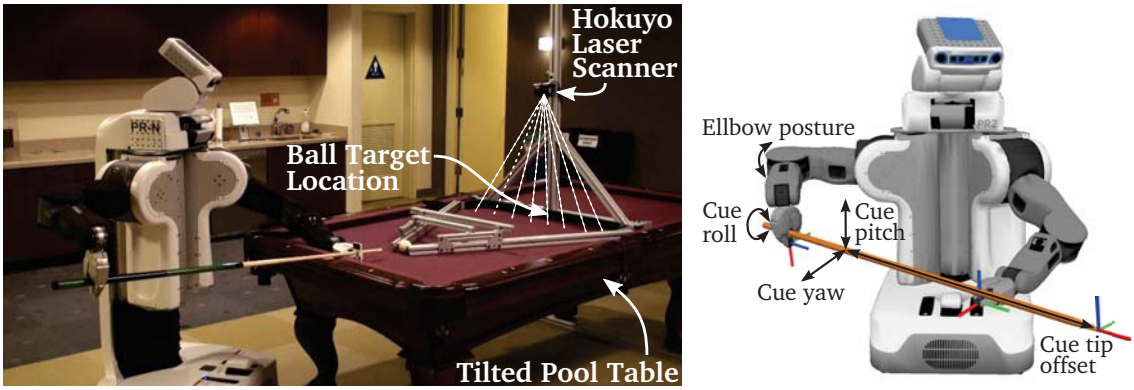


Figure 4.9: The experimental setup (left): The Hokuyo laser scanner is mounted on a stand such that it can measure the offset of the cue ball from the target location. The apparatus on the tilted pool table is used to guide the backwards rolling ball after each pool shot to its original position. Illustration of the pool task frame (right): The stroke motion has been transformed in order to allow constraint-satisfying exploration. In the experiments, the DMP encoded these 5 taskspace dimensions.

Experimental Results

As shown in Fig. 4.10, the initial control policy results in a pool stroke that accelerates the cue ball towards the center of the rail, the ball target location (see Fig. 4.9). However, the reproduction fails to reach the end of the table as the pool stroke was not powerful enough. Note, to show case the system's ability to learn dynamic tasks, the initial policy was taught without first retracting the cue stick in order to build up momentum (see first row in Fig. 4.10).

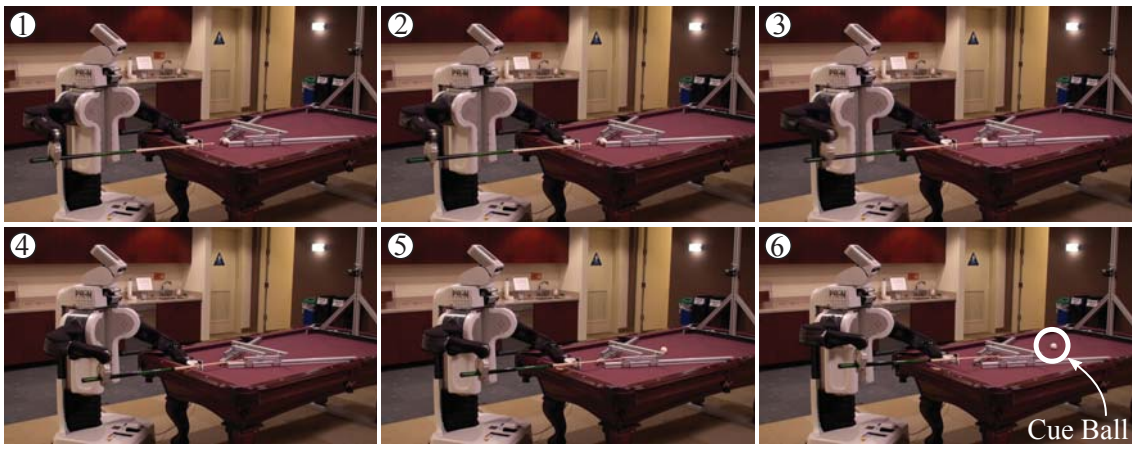


Figure 4.10: The PR2 robot learning a pool stroke: The apparatus on the tilted pool table automatically positions the cue ball after each trial allowing for autonomous learning. The initial policy fails to hit the target on the tilted table (see cue ball position in image 6).

The robot was able to improve the initial policy completely autonomously within approximately 20 minutes, as shown in Fig. 4.11.

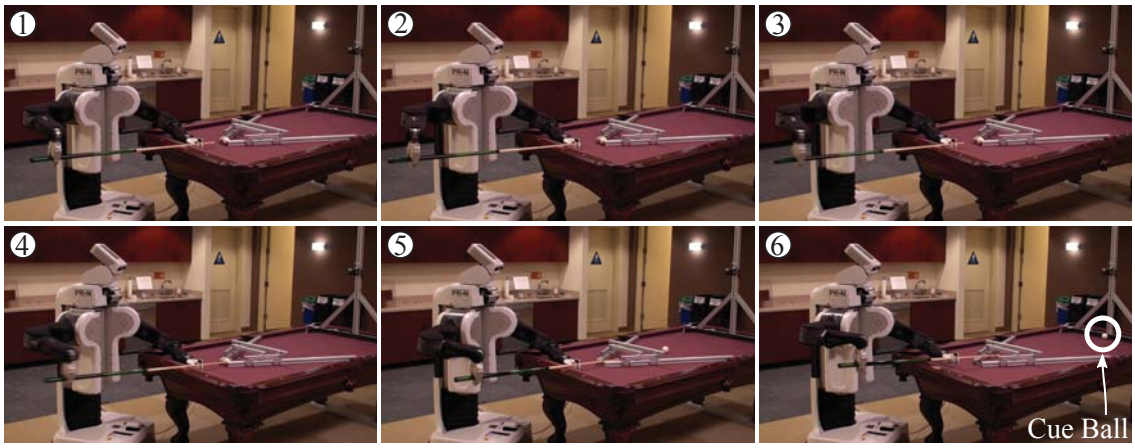


Figure 4.11: After 20 minutes of trial-and-error without human intervention the robot learned to retract the pool cue first allowing for a more powerful shot such that the cue ball hits the target location precisely (see cue ball position in image 6).

The system learned to retract the cue stick in order to gain momentum without loss of precision. Fig. 4.12 confirms this observation since the cost of the noiseless rollouts decrease over time. A video summarizing the pool task experiment is available at `http://youtu.be/8Thdf_7j4dI`.

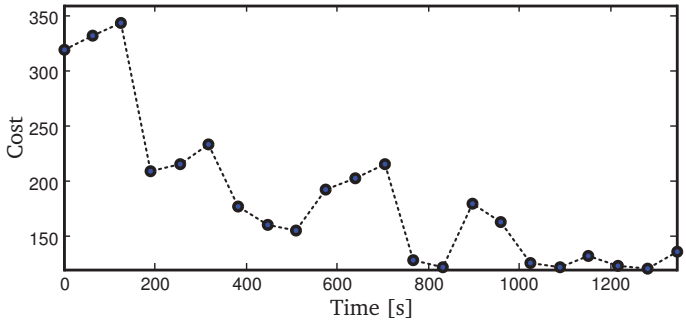


Figure 4.12: The learning curve showing the cost of the 22 noiseless rollouts over the time span of approximately 20 minutes. Note: The robot improved its task execution completely autonomously.

4.3.6 Learning to Flip a Box using Chopsticks

The goal of the second task is to learn the finer manipulation skill of gently flipping a horizontal box upright using a pair of chopsticks (see Fig. 4.7).

Experimental Setup

The experimental setup consists of the PR2 robot positioned in front of a table on which the box is placed, as shown in Fig. 4.13.

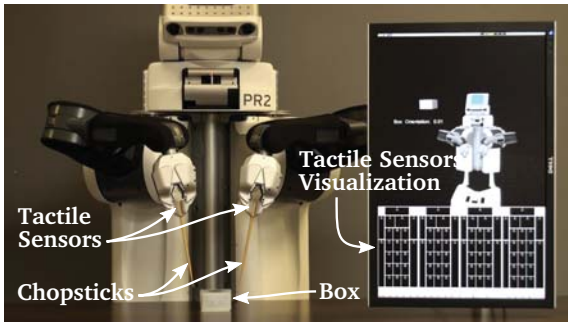


Figure 4.13: The PR2 robot is placed in front of a table holding chopsticks such that they make maximum contact with the tactile sensors on each fingertip (see Fig. 4.14). The manipulation task consists of gently flipping the box upright. Note, the chopsticks were only hold with a force of approximately 10 N.

The two fingertips on each of the parallel jaw grippers are equipped with a pressure sensor array consisting of 22 individual cells covered by a protective layer of silicone

Figure 4.14: Arrangement of the pressure sensors on the finger pads (left) and closeup of the placement of the chopsticks in the gripper (right).

rubber. The arrangements of the individual cells is shown in Fig. 4.14 (left). These capacitive sensors measure the perpendicular compressive force applied in each sensed region and are sampled simultaneously at a rate of 24.4 Hz. In order to overcome the sensor's limitation on detecting shear forces, a washer has been attached to the chopstick as shown in Fig. 4.14 (right). The chopsticks are placed between the two finger pads and the sensor readings are used to servo a force of approximately 10 N. The reasons for not holding the chopsticks with maximum force are two fold: to enable the pressures sensors to sense contact of the chopsticks with the box or table and to impose an interesting aspect on the manipulation task and making it more difficult. In order to measure the box orientation and its linear accelerations during each trial, a MicroStrain Inertia-Link has been attached inside the box. The measurements are obtained at 100 Hz.

Cost Function

The primary task objective is to flip the box upright. Therefore, the terminal cost ϕ_T provides feedback of the maximum roll of the box $\gamma_{\max}$ over all time steps $t = 1, \ldots, T$ as well as its final roll γ_{T} and is computed as a weighted sum according to

$$\phi_T = \omega_1 \, \gamma_{\max}^{10} + \omega_2 \, f(\gamma_{\mathrm{T}}) \ , \tag{4.6}$$

where $f(\gamma_{\mathrm{T}})$ is equal to 1 if $|\gamma_{\mathrm{T}} - \frac{\pi}{2}| < 0.1$, or 0 otherwise. The secondary task objective is to flip the box upright as gently as possible. Thus, the state cost $\phi(t)$ includes the square of the box linear accelerations $a_b(t) = a_{b_x}(t)^2 + a_{b_y}(t)^2 + a_{b_z}(t)^2$, the amount of force sensed on each of the 4 pressure sensors $p(t) = \sum_{i=0}^{88} p_i(t)$, and the sum of the sensed square accelerations in both grippers $a_g(t) = a_{g_x}(t)^2 + a_{g_y}(t)^2 + a_{g_z}(t)^2$. The gripper acceleration signals are band-pass filtered to contain data between 5-1000 Hz so that low frequency signals due to gravity and slow motions are filtered out while impacts of the

chopstick with the table and/or box are still observable. The state cost of a rollout for each time step $\phi(t)$ is also calculated as a weighted sum:

$$\phi(t) = \omega_3 \, a_b + \omega_4 \, p(t) + \omega_5 \, a_g(t) \ . \tag{4.7}$$

The total cost is computed by adding the terminal cost ϕ_T to the last time step of the state cost $\phi(t = T)$. For the learning experiment, the following weights have been used: $\omega_1 = 50.0$, $\omega_2 = 200.0$, $\omega_3 = 0.1$, $\omega_4 = 1.0$, and $\omega_5 = 0.1$. Note that these parameters also account for normalization across different sensor modalities.

As in the previous task, the initial policy is learned from a human demonstrator by directly guiding the robot's gripper. The task variables for this particular task were naturally chosen to be the two tips of the chopsticks. Thus, the joint trajectory, recorded during teaching, is first transformed into the task space and then encoded in a single DMP with 12 transformation systems (position $\mathrm{tcp}_x, \mathrm{tcp}_y, \mathrm{tcp}_z$ as well as orientation $\mathrm{tcp}_{\mathrm{roll}}, \mathrm{tcp}_{\mathrm{pitch}}, \mathrm{tcp}_{\mathrm{yaw}}$ of the tips of each chopstick). The duration of the demonstration was 4 seconds. In the learning phase, the exploration noise $\boldsymbol{\epsilon}$ has been set to $\epsilon_{\mathrm{x}} = 0.01$, $\epsilon_{\mathrm{y}} = 0.04$, $\epsilon_{\mathrm{z}} = 0.06$, $\epsilon_{\mathrm{roll}} = 0.2$, $\epsilon_{\mathrm{pitch}} = 0.5$, and $\epsilon_{\mathrm{yaw}} = 0.2$ for both of the end-effectors task spaces.

To illustrate the difficulty of the task, we conducted an informal user study: 10 subjects have been asked to each perform 20 attempts to flip the box by guiding the robot's gripper as pictured in Fig. 4.15. Although the subjects were using visual and tactile feedback to correct for errors during the execution, the success rate was rather low. Each subject achieved an average of 3 successful trials out of 20 attempts.

Figure 4.15: An informal user study has been conducted: 10 subjects were asked to flip the box by guiding the robot's gripper. Only an average of 3 out of 20 attempts were successful, corresponding to an average success rate of 15 %.

Experimental Results

Although a successful demonstration has been used to initialize the learning experiment, simply replaying the initial policy only succeeded 3 times out of 100 runs. This observation suggests that not all task relevant aspects have been captured from imitation (see Fig. 4.16). However, after 26 iterations (approximately 35 minutes) the robot learned to

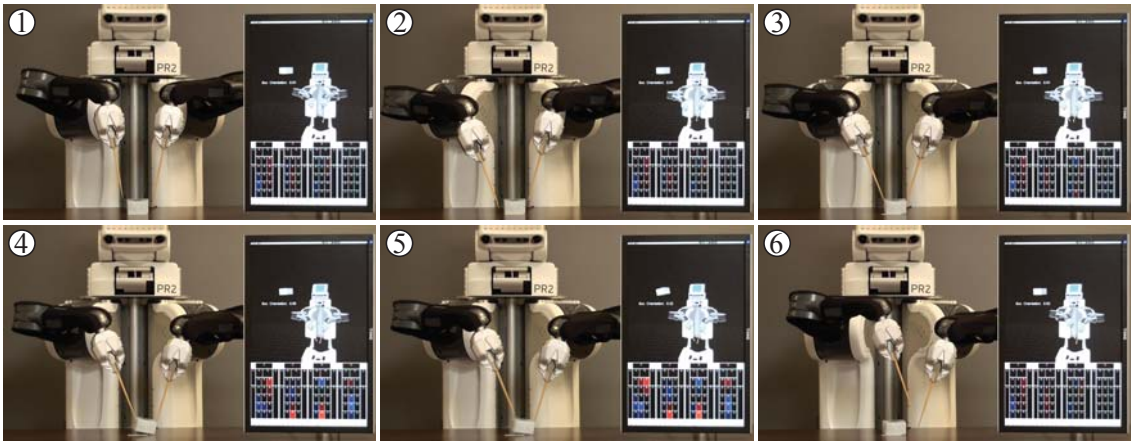


Figure 4.16: The PR2 robot learning a finer manipulation skill of gently flipping a horizontal box upright: The color display on the monitor visualize the amount of sensed pressure of the tactile sensors. The sequence shows that repeating the initial policy fails to flip the box upright (shown in image 6).

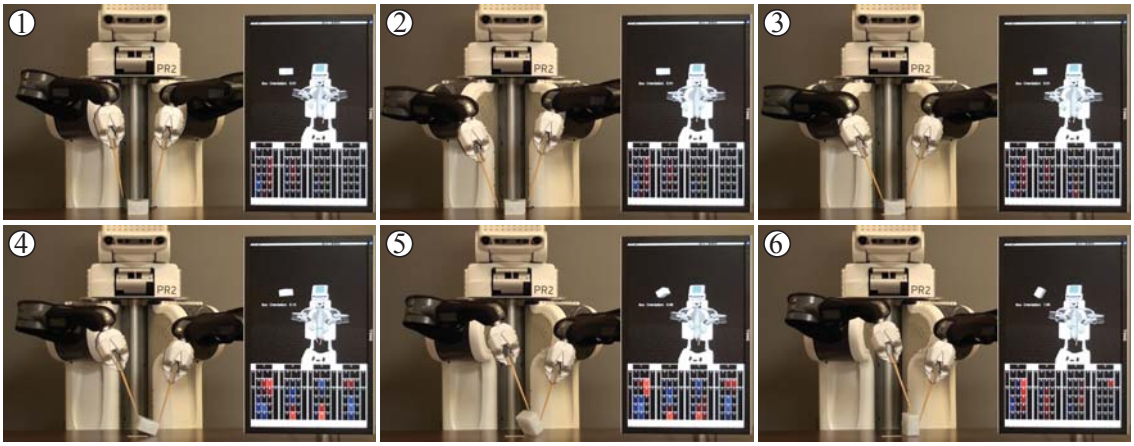


Figure 4.17: The PR2 robot learned to gently flip the box upright with a success rate of 86 % after about 35 minutes. A summarizing video is available at `http://youtu.be/8Thdf_7j4dI`.

achieve the task (see Fig. 4.18). For performance evaluation the learned policy has been executed (without the addition of noise) a total of 200 trials out of which 172 succeeded. This is observation is confirmed when evaluating the noiseless rollouts. Fig. 4.18 confirms this observation since the cost decrease over time. Nevertheless, due to the nature of the task, the learned behavior would (in 14 % of the times) fail to flip the box.

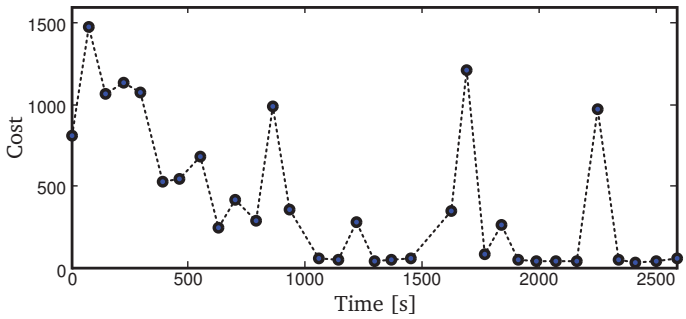


Figure 4.18: Learning curve of the box flipping task showing the cost of 32 noiseless rollouts over the time span of approximately 41 minutes. The cost is minimized over time, however, occasional failures remain due to the stochasticity of the task.

4.4 Conclusion

This chapter demonstrated the feasibility of our approach to learn manipulation skills. Imitation learning provided an intuitive and fast way of directly learning control policies to build up an increasing set of movement primitives to be stored in the motion library. The design of our movement representation allowed for (local) generalization capabilities as well as reactive behaviors through perceptual coupling. To allow learning of dynamic tasks and to account for stochasticity in manipulation tasks that include contact with the environment, we presented a the Reinforcement Learning algorithm PI^2. Our approach was able to significantly improve the task performance over an initial demonstrated policy. We would like to highlight the ease of Reinforcement Learning in the given setup with the PI^2 algorithm, as the update equations have no danger of numerical instabilities as neither matrix inversions nor gradient learning rates are required.

However, the stochastic nature of manipulation tasks will always give raise to failed execution as it was the case in the experiment involving the robot flipping the box with chopsticks. Therefore, in the next chapter, we will investigate methods that can detect such failure condition the instant they occur.

Chapter 5

Associative Skill Memories

The previous chapter presented experimental evaluations on learning manipulation skills in various settings. Imitation learning was used to bias the system towards good solutions and trial-and-error learning was used to improve the initial policy until satisfactory task performance was achieved. However, especially in uncertain and stochastic environments, learned manipulation skills may still fail to achieve the task from time to time. For example, this was the case in the experiment involving flipping the box upright using chopsticks (see Sec. 4.3.6). Even though our approach improved the success rate from 3 % (initial policy) to 86 % (final policy), it was still failing 14 % of the time. Variations in the task, e.g. slightly different initial box poses, are one potential factor that can explain the slippage of the chopsticks resulting in a failed attempt in some of the trials. Another potential factor are variations in the sensorimotor system of the robot: sensors are noisy and imperfect, so are controllers. In any case, even if better sensors and controllers become available, the stochasticity and uncertainty of contact rich manipulation tasks will remain. Especially when executing open-loop trajectories*, as it was the case in the previous chapter, there is always a chance of failing to achieve the task. Thus, the ability to continuously monitor task progress and detect failure conditions as they occur is crucial. However, generating appropriate reference signals that facilitate progress monitoring is non-trivial.

5.1 Sensor Predictions for Manipulation

Humans seem to learn to predict sensory information for most of the tasks they perform on a daily basis (Johansson & Flanagan, 2009a). For example during walking, humans continuously predict the ground reaction forces that they expect to feel at the soles of their feet, enabling them to instantly detect and react to deviations that might arise due

*In this thesis, using sensor feedback from encoders to close a joint position control loop is still considered as open-loop. Only if external sensor information, e.g. force/torque or visual, is used inside a feedback loop, we will consider it as closed-loop.

to changes in ground terrain (like stepping into a pothole). For robots, such prediction of performance and sensory events is largely missing, or if it exists, it is the result of the design of human experts who may have spend a lot of time* analysing the task at hand to find regularities that can be used in combination with manually tuned thresholds to detect failure conditions (Righetti et al., 2014).

Automatically detecting failure conditions the instant they occur is challenging. At each time step, the system requires baseline sensor values to compare against the currently measured sensor values in order to make a decision whether the task outcome will be positive or negative. Note that these baseline values are required for each instant of time and vary across tasks. Furthermore, multiple sensor modalities may need to be considered to increase robustness as well as to avoid perceptual aliasing†. Finally, large variances in certain sensor readings across different trials of the same task need to be considered. For example, when reaching for an object that has uncertainty in its pose, contact might be established earlier or later than expected. Thus, the system also may need specification of confidence intervals for each sensor that account for such variations.

5.1.1 Movements dictate Sensor Feedback

It is important to note that, especially in manipulation, there is a strong correlation between sensory feedback and executed movements. Particular tasks, e.g. grasping an object, can give rise to a variety of different sensor experiences, e.g. different grasp configurations may result in different haptic feedback. Furthermore, the measured sensor readings may also vary significantly depending on the movement speed. Thus, expected perceptual information‡ is both tightly coupled with the motor skill and potentially time aligned.

This chapter introduces a novel concept of how to think about movement generation (Pastor, Kalakrishnan, Meier, et al., 2013) in the context of robotic manipulation. Our approach tightly couples generated movements with expected sensory feedback. The idea is to augment each movement primitive with all sensory information that has been experienced whenever this particular motor skill has been executed. Associating sensory events time aligned with the corresponding motor skill facilitates sensor prediction: Sensor information accumulated from previous executions serve as sensor predictions in subsequent executions of the same movement primitive. We will call movement primitives augmented with all previously experienced sensory events Associative Skill Memories (ASMs).

*Especially long hours at night and on weekends. Sometimes these human experts would even stay overnight.

†The term perceptual aliasing is used to describe the problem when different situations give rise to the same perceptual input, thus become indistinguishable.

‡In this thesis, the term perceptual information refers to all sensory information, not just visual.

5.1.2 Neuroscientific Inspiration

The inspiration for this approach is motivated by neuroscientific investigations, that demonstrated that the human brain can be thought of as a vast associative memory (Hopfield & Tank, 1986; Mitchell et al., 2008). Neuroscientific studies on human motor behavior revealed that especially in the context of object manipulation humans seem to follow a very similar strategy. The following quotation is taken from the paper "Control strategies in object manipulation tasks" written by Flanagan, Bowman, and Johansson and published in 2006 in the journal "Current Opinion in Neurobiology".

> *We suggest that contact events encoded in multiple sensory modalities represent sensorimotor control points that have three crucial functions[...] First, by comparing actual and predicted sensory events in multiple sensory modalities, the sensorimotor system can simultaneously monitor multiple aspects of task performance and, if prediction errors arise, respond to the pattern of errors observed in different modalities. Second, because contact events give rise to salient sensory signals from multiple modalities that are linked in time and space, they provide an opportunity for sensorimotor integration and intermodal alignment that might facilitate learning and upholding of multimodal sensorimotor correlations that are necessary for prediction of purposeful motor commands. Third, the predicted sensory consequences of contact events can directly furnish initial state information for subsequent phases of the manipulation tasks. This enables smooth phase transitions, in contrast to the stuttering transitions that would occur as a result of neuromuscular delays if peripheral afferent information about subgoal completions always triggered, reactively, the execution of the next phase.*

This neuroscientific evidence may give us important insights on how autonomous robotic manipulation might be achieved, given that, besides certain animals, human beings are the only existing instantiations that are capable of dexterous manipulation.

This chapter will show how ASMs implement each of these "three crucial functions" mentioned in the quotation above. First, in Sec. 5.2, we will show how associated sensor information can be used to continuously "monitor multiple aspects of the task performance" by "comparing actual and sensory events in multiple sensory modalities". Second, in Sec. 5.3, we will further exploit sensor predictions for "sensorimotor integration" inside feedback control loops to generate "purposeful motor commands". Finally, in Sec. 5.4, we will use "predicted sensory consequences of contact events" to select among applicable ASMs for "subsequent phases of the manipulation tasks" and realize "smooth phase transitions".

5.2 Online Task Outcome Prediction

As discussed previously, the nature of stochastic and dynamic environments will always leave room for acquired manipulation skills to fail at achieving the task. The ability to detect at which point in time the task will no longer be executed successfully is important, because it allows for appropriate corrective actions limiting potentially negative consequences. In this section, we will show how ASMs can be used to predict failure conditions online using sensor information from multiple senor modalities (Pastor, Kalakrishnan, et al., 2011). Our emphasize is on using all available sensor information despite the fact that some sensors might be redundant and do not provide any information about the task outcome. Note that, while most state-of-the-art robots employ multiple sensors, they are rarely used together. The aim of this approach is to provide the means to leverage from sensor rich robotic platforms.

We will present results on predicting the task outcome for the experiment that involved the robot flipping a box using chopsticks (see Sec. 4.3.6). Although trial-and-error learning improved the task success rate from initially 3 % to finally 86 %, there was enough stochasticity in the manipulation task causing the chopsticks to slip 14 % of the time (see Fig. 4.16). Therefore, after the manipulation task has reached satisfying performance, we propose to record all available sensor information time aligned with the behavior. Thus,

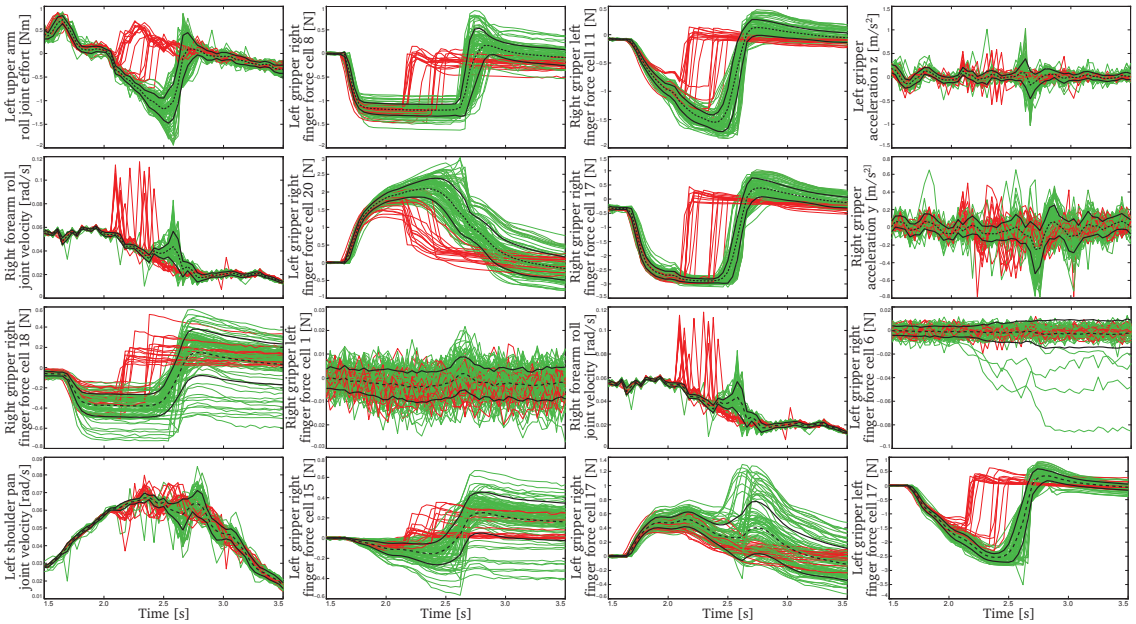


Figure 5.1: A subset of the 132 recorded signals of each of the 100 training trials are shown. This training data set contains 73 successful (green) and 17 unsuccessful (red) trials. The weighted mean (dashed line) and the weighted ± one standard deviation (solid line) are computed from the successful trials. The subset shows examples of signals that can be used to detect failure conditions, however, it also shows examples of redundant signals that may not provide useful information.

our approach leverages from the fact that similar task executions will result in similar sensory events. To demonstrate the feasibility of our approach, we recorded a total of 132 task relevant sensor signals along with each of the 200 noiseless rollouts that have been used to evaluate the task performance (see Sec. 4.3.6). These sensor signals include the joint positions, joint velocities, and joint efforts of both 7 DOFs arms, the linear accelerations of both grippers, and each individual pressure sensor cell of all 4 finger pads (shown in Fig. 4.14). The recorded signals have been time aligned with the executed DMP and were re-sampled such that each signal only contains 100 samples. A subset (16 out of 132) of the recorded sensor data is shown in Fig. 5.1.

The first observation is that there is clearly a strong correlation across trials indicating that similar task executions indeed give rise to similar sensory events. Furthermore, the sensor signals carry enough information (some more than others) that can be used to detect failure conditions online. To demonstrate this ability, we divided the data recorded from 200 trials into a training and test set of equal size. The sensor data shown in Fig. 5.1 served as training data. The idea of our approach is to use successful trials to compute confidence intervals for each time step and for each sensor. These confidence intervals represent the range of sensor values that indicate positive task progression. Conversely,

Input:

The critical value for the cost ϕ_c that determines task success.

Iterative Training:

Step 1: Execute rollout and record sensor information along with the achieved cost ϕ_t.

Step 2: Use ϕ_c to compute whether task was successful, i.e. $\phi_t < \phi_c$.

Step 3: For successful rollouts: compute a weight according to $w = (\phi_c - \phi_t)/(\phi_c)$.

Step 4: Update the weighted mean and weighted standard deviation for each time step over all rollouts for all signals.

Online Testing:

Step 1: During task execution, monitor all sensor signals and compute at each time step for each signal a z-test using the weighted mean and weighted standard deviation from the training phase under the null-hypotheses with confidence c.

Step 2: If the z-test fails, then the perceived sensor value is with confidence c not correlated with the statistics computed from successful trials. If this occurs for n consecutive time steps in more than m signals, a failure condition is predicted.

Table 5.1: Online detection of failure conditions from sensor information.

deviations of measured sensor signals from these predicted ones indicate failure conditions. A summary of our algorithm is summarized in Table 5.1.

5.2.1 Experimental Results

In the training phase, the 73 successful trials contained in the training data set have been used to compute the weighted mean and weighted ± one standard deviation for each time step, see Fig. 5.1. In the testing phase, 100 test trials (containing 89 successful and 11 unsuccessful trials) have been presented to the proposed method in a sequential manner. Even though the testing phase has been carried out offline, the data has been input into the system in the same temporal order as perceived by the robot during execution. The method sketched in Table 5.1, setup with the parameters $c = 10e - 9$, $n = 3$, and $m = 5$, managed to detect all 11 failures in the test set without triggering any false positives. It is important to note, that the system was able to detect these 11 failure conditions the moment more than 5 signals were outside the confidence interval for more than 3 consecutive time steps. These 11 failure conditions have been detected on average after 2.4 seconds, which roughly corresponds to the instant the chopsticks lost contact with the box causing it to fall back, see Fig. 5.2.

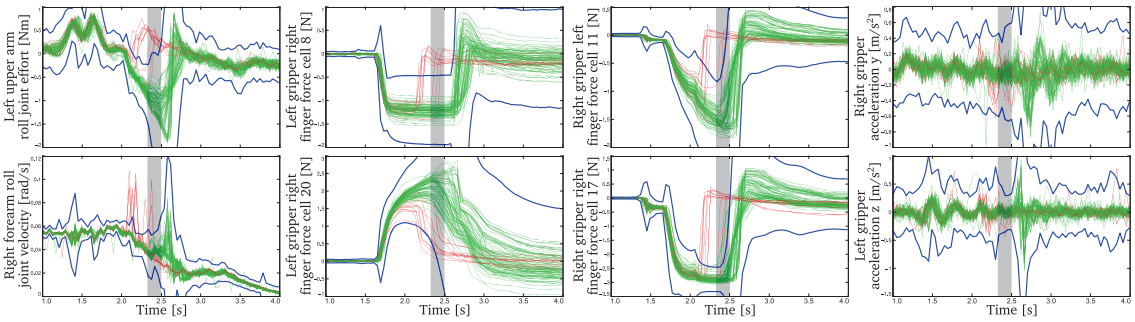


Figure 5.2: The plots show a subset of the training set. The 6 ± weighted standard deviation (blue line) is computed from the training set (shown in Fig. 5.1). The failure condition for all 11 unsuccessful trials (red lines) have been detected correctly after an average of 2.4 seconds (shaded area).

5.2.2 Conclusion

The experimental results show that there is indeed a strong correlation between manipulation behaviors and expected sensory feedback. This insight allowed us to build a successful and rather simple performance predictor from simple sensory statistics. The parameters for n and m have been chosen specifically for the task at hand and will need to be adjusted in future tasks. One possible solution might involve inferring the optimal set of parameters from the training data, but this is not in the scope of this thesis. Instead, we want to

emphasize the overall approach rather then the details of the method. Online task outcome prediction for arbitrary manipulation tasks is an unsolved problem. Our algorithm was designed to showcase the importance of associating sensory information along with movement plans. The experimental results show that such a purely data-driven approach can yield good performance despite a rather simple method. Importantly, the very same predictive model can be used to generate purpuseful motor commands to avoid failure conditions in the first place (see next Section).

5.3 Online Movement Adaptation Based on Previous Sensor Experiences

As previously discussed in the motivation section (see Sec. 1.2) of this thesis, current approaches to grasping (Srinivasa et al., 2010; Chitta et al., 2012) typically involve a four-step processing pipeline (see Fig. 1.5). First, visual perception is used to estimate the object pose, second, a goal grasp pose along with corresponding joint angles is computed, third, a motion planning algorithm is invoked to generate a collision free trajectory from the current joint configuration to this goal configuration, and finally joint controllers are employed to track this trajectory. Usually, this processing pipeline is repeated for each new grasp attempt, i.e. each attempt is treated as a unique problem that is solved on its own. More importantly, information from previous grasp attempts is often discarded. Such approaches underestimate the importance of experienced sensor information from previous trials.

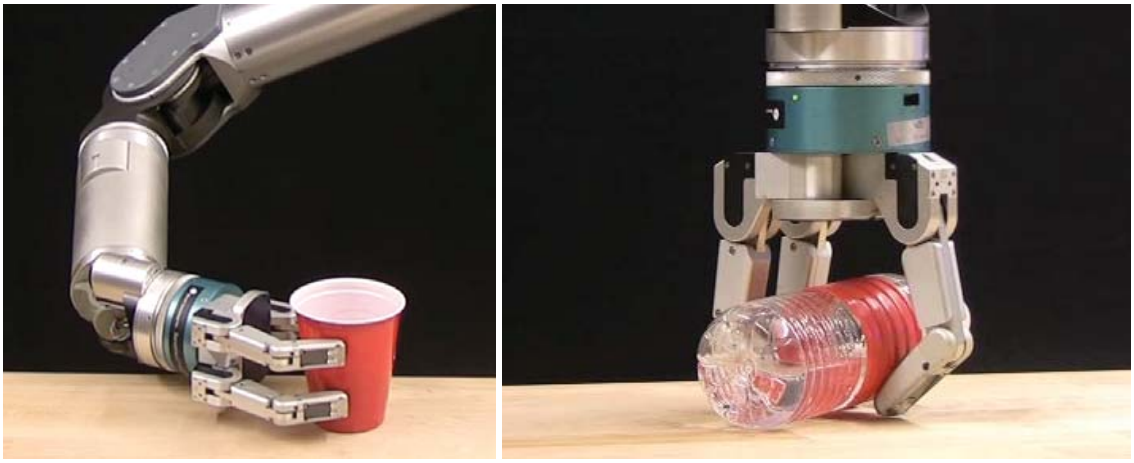


Figure 5.3: The Barrett WAM arm grasping a cup (left) and a bottle (right).

Associating experienced sensor information with corresponding robot action is very important as it allows to generate a predictive model of sensor traces that can be used to improve grasping over time (i.e. new grasp attempts can be evaluated using information from similar past grasp attempts, see Sec. 4.3). It also allows early detection of failure

conditions during task execution (see Sec. 5.2) and to adapt movement plans online to unforeseen perturbations. However, associating sensor information with a set of similar grasp attempts becomes very challenging when employing commonly used stochastic motion planning algorithms, e.g. (Cohen, Chitta, & Likhachev, 2010), in the third step of the grasping pipeline. For two very similar grasp scenarios these algorithms may return two very different approach trajectories resulting in two very different sensor experiences. Thus, transferring knowledge from previous trials may become impossible. Additionally, improving task performance over time and designing online adaptation algorithms become similarly challenging simply because the relation to previously successful trials is missing.

5.3.1 Stereotypical Movements facilitate Memory

In contrast, learning a set of stereotypical movements that are applicable in certain grasping scenarios provides the basis to accumulate sensations of previous trials. The key insight is that stereotypical movements give rise to stereotypical sensory events (see Fig. 5.1). Associating experienced sensor traces with stereotypical movements allows for building up a predictive model that can be used in future trials. Stereotypical movements facilitate the design of a system that can improve over time simply because new trials can be related to previous trials. Sensor traces from successful movements generate an implicit model* of what the robot should *feel* for subsequent similar movements. Furthermore, such a predictive model can be systematically used to realize reactive behaviors that compensate for unexpected events. For example, during grasping, movement plans can be adapted online in case of early collisions with the object. The combination of reactive behaviors with stereotypical movements can create a rich set of possible motions that account for external perturbations and perception uncertainty to generate truly robust behaviors.

In this section, we will encode stereotypical movements into Dynamic Movement Primitives (DMPs), as described in Sec. 3.2. Additionally, we will augment each movement primitive with experienced sensory events from previous executions to form Associative Skill Memories (ASMs). These sensory experiences serve as sensor predictions in subsequent similar executions in sync with the performed motor skill. Thus, in contrast to previous approaches, movement generation and sensor prediction do not follow a sequential order. Generating sensor predictions time aligned with the corresponding motor skill facilitates tight integration in feedback control loops to realize online movement adaptation. Whenever the expected sensor prediction deviates from the measured sensor values, the movement plan can instantly adapt accordingly.

In particular, we present a control framework suitable for grasping, using DMPs with sensory feedback and nonlinear position and force control. First, in Sec. 5.3.2, we extend the DMP formulation for quaternion control, used to generate desired endeffector orientations. Then, in Sec. 5.3.3, we propose a feedback law for DMPs that adapts the

*These sensor traces can be understood as a motor-centric representation of the physical properties of the object in the particular context.

desired trajectories based on the difference between the forces experienced in a previous successful trial and the currently measured forces. In Sec. 5.3.4, we present a low-level position and force control system that seamlessly integrates with DMPs allowing for very reactive and compliant behaviors. In Sec. 5.3.6, we present experimental evaluations on a Barrett WAM (see Fig. 5.3) that highlight the advantages of this control framework for grasping under uncertainty. In our experiments we simulate errors in the vision system 1) and grasp pose estimation 2) by misplacing the object to be grasped while keeping the grasp goal configuration fixed. We show that open loop execution fails to grasp the object in most of the cases, while the proposed approach succeeds even for relatively large perception errors. The experiments successfully show that previous sensor information can be used to create robust grasping behaviors. Our control approach significantly improves the robustness of grasping while creating a very compliant control system.

5.3.2 DMPs for Quaternion Control

In control, the use of unit quaternions to represent rotations in $\mathbb{R}^3$ is very convenient (Yuan, 1988). Unit quaternions allow for non-singular representations of rotations and are easier to compose than for example Euler angles. Moreover, compared to rotation matrices, they are more compact and numerically stable. It is therefore desirable to have DMPs use a quaternion representation for orientations.

However, using 4 separate transformation systems Eq. (3.5) to represent an orientation trajectory generates quaternions that are not of unit length. Such a method is mathematically incorrect and requires an ad-hoc post-normalization step. Furthermore, it is difficult to interpret quaternion derivatives in terms of physical quantities. Indeed, the group of spatial rotation $SO(3)$ is a 3-dimensional manifold and therefore each of its element has a tangent space that is equivalent to $\mathbb{R}^3$. It is therefore unnecessarily complicated to use 4 derivatives to represent a 3-dimensional vector space.

In the following we propose a DMP formulation that guarantees the generation of unit quaternions by the dynamic system. Furthermore, the new integration rule offers a more intuitive representation of the derivatives allowing for tight integration in feedback control loops. The orientation error between two quaternions is defined as

$$\mathbf{e}_o(\{\eta_1, \mathbf{q}_1\}, \{\eta_2, \mathbf{q}_2\}) = \eta_1 \mathbf{q}_2 - \eta_2 \mathbf{q}_1 - \mathbf{q}_1^{\times} \mathbf{q}_2 \quad , \tag{5.1}$$

where $\{\eta, \mathbf{q}\}$ is a unit quaternion with $\eta \in \mathbb{R}$ and $\mathbf{q} \in \mathbb{R}^3$ and $\mathbf{q}^{\times}$ is the skew-symmetric matrix given by

$$\mathbf{q}^{\times} = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \quad . \tag{5.2}$$

Note that the orientation error $\mathbf{e}_o(\{\eta_1, \mathbf{q}_1\}, \{\eta_2, \mathbf{q}_2\}) \in \mathbb{R}^3$ can be interpreted as an error in angular velocities (Yuan, 1988).

Here, we use this error computation to compose a single transformation system which generates unit quaternions $\mathbf{q}$, as well as angular velocities $\boldsymbol{\omega}$ and angular accelerations $\dot{\boldsymbol{\omega}}$. Replacing the error computation in Eq. (3.5) leads to

$$\begin{aligned} \tau\ \dot{\boldsymbol{\omega}} &= -\mathbf{K} \cdot \mathbf{e}_o(\mathbf{g}, \{\eta, \mathbf{q}\}) - \mathbf{D} \cdot \boldsymbol{\omega} + \ \mathbf{K} \cdot \mathbf{e}_o(\mathbf{g}, \{\eta_0, \mathbf{q}_0\})\ s + \mathbf{K} \cdot \mathbf{f}(s) \\ \tau \begin{bmatrix} \dot{\eta} \\ \dot{\mathbf{q}} \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\boldsymbol{\omega}^{\times} \end{bmatrix} \begin{bmatrix} \eta \\ \mathbf{q} \end{bmatrix} \ . \end{aligned} \tag{5.3}$$

Note, the scalars in Eq. (3.5) have been replaced by vectors. See (Yuan, 1988) for more details on quaternion propagation. The proposed quaternion integration rule guarantees the generation of normalized unit quaternions. Furthermore, it ensures that the generated quaternion orientations and corresponding angular velocities and accelerations are differentially compatible. Moreover, this new integration rule facilitates a tight integration of the dynamical system into control loops, as described in the following subsection.

5.3.3 Online Trajectory Generation using Sensory Feedback

The main idea of our approach is to online modify the desired trajectory generated by the DMP using sensor information from previous task executions. Here, the concept of stereo typical movements plays a key role: Similar tasks afford similar movement plans. This simple yet profound insight enables our system to accumulate past sensor experiences in form of sensor traces and use them in future similar scenarios, e.g. for failure detection (Pastor, Kalakrishnan, et al., 2011). These sensor traces compose a growing associative memory relating task execution with sensor experiences. Predictive models of how things should *feel* during task execution are particularly relevant during contact interactions of the robot with its environment. Our approach uses this information to online modify the desired trajectory, i.e. the movement plan, such that the measured sensory experience remains close to the expected one.

Given information of force, torque, or pressure, the corresponding desired generalized forces acting on the task variables can be computed. For example, a force/torque sensor at the wrist provides a direct relation between sensing and endeffector accelerations. The desired acceleration can be modified to compensate for sensing errors.

The general feedback function is given by

$$\boldsymbol{\zeta} = \mathbf{K}_1 \mathbf{J}_{sensor}^T \mathbf{K}_2 (\mathbf{F} - \mathbf{F}_{des}) \ , \tag{5.4}$$

where $\mathbf{J}_{sensor}$ is the Jacobian of the task controlled by the movement primitives with respect to the sensors, $\mathbf{F}$ is the generalized forces read from the sensors and $\mathbf{F}_{des}$ is the corresponding desired forces, that have been acquired in previous task executions. $\mathbf{K}_1$

and $\mathbf{K}_2$ are, potentially time-varying, positive definite gain matrices. The feedback is added to the transformation system in Eq. (3.5) as follows

$$\begin{aligned} \tau \dot{\boldsymbol{v}} &= \mathbf{K}(\boldsymbol{g} - \boldsymbol{x}) - \mathbf{D}\,\boldsymbol{v} - \mathbf{K}(\boldsymbol{g} - \boldsymbol{x}_0)s + \mathbf{K}\,f(s) + \boldsymbol{\zeta} \\ \tau \dot{\boldsymbol{x}} &= \boldsymbol{v} \ . \end{aligned} \tag{5.5}$$

Similarly feedback to the quaternion DMPs is added directly in Eq. (5.3). For example, in the case of using DMPs to control the hand position and orientation and considering a force/torque sensor located at the wrist of a manipulator, the Jacobian of the task becomes $\mathbf{J}_{sensor} = \mathbf{I}_{6\times 6}$ and $\mathbf{F}$ is the 6-dimensional reading from the sensor.

Remark The gain matrix $\mathbf{K}_1$ defines the sensitivity of the task variables with respect to predicted forces. Each dimension (position and orientation) can be weighted differently. For example, we can imagine a task where one direction should adapt very quickly to sensing while the other directions should be more stiff. The gain matrix $\mathbf{K}_2$ defines the relative sensitivity to different sensors, depending for example on their resolution or adequacy to the task. We would imagine that such a gain matrix will be related to the known variance of the desired forces acquired during learning (Pastor, Kalakrishnan, et al., 2011; Kalakrishnan, Righetti, et al., 2011).

5.3.4 Position and Force Control

We separate the low-level controller into two distinct parts: a controller for the endeffector, i.e. the hand, and a controller for the fingers.

Endeffector Position Control

In order to track the desired positions and orientation generated by the DMP we use a velocity-based operational space controller together with an inverse dynamics law and feedback error compensation in joint space. Please refer to (Nakanishi et al., 2008) for more details. The control law is written as

$$\boldsymbol{\tau}_{arm,p} = \mathbf{M}\ddot{\mathbf{q}}_d + \mathbf{h} + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q}) + \mathbf{K}_d(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \tag{5.6}$$

with

$$\dot{\mathbf{q}}_d = \mathbf{J}^{+}\Big(\dot{\mathbf{x}}_d + \mathbf{K}_x(\mathbf{x}_d - \mathbf{x})\Big) + \mathbf{K}_{post.}\big(\mathbf{I} - \mathbf{J}^{+}\mathbf{J}\big)(\mathbf{q}_{post.} - \mathbf{q}), \tag{5.7}$$

where $\mathbf{M}$ is the rigid-body inertia matrix of the arm, $\mathbf{q}$ and $\mathbf{q}_d$ are the vectors of measured and desired joint angles, $\mathbf{x}$ and $\mathbf{x}_d$ are the measured and desired endeffector position and orientation, $\mathbf{h}$ is the vector of Coriolis, centrifugal, and gravitational forces, $\mathbf{K}_p, \mathbf{K}_d, \mathbf{K}_x$, and $\mathbf{K}_{post.}$ are (diagonal) gain matrices and $\boldsymbol{\tau}_{arm,p}$ is the vector of torques. $\mathbf{J}$ is the endeffector Jacobian, $\mathbf{J}^{+}$ denotes the Moore-Penrose generalized inverse and $\mathbf{q}_{post.}$ is the vector of default posture optimized in the nullspace of the endeffector motion. The desired

acceleration $\ddot{\mathbf{q}}_d$ and desired position $\mathbf{q}_d$ are obtained by numerical differentiation and integration of the desired velocity $\dot{\mathbf{q}}_d$.

Remark This controller uses differential inverse kinematics to generate desired joint position, velocities and accelerations (Eq. (5.7)) and a standard inverse dynamics controller with PD error feedback to achieve the desired trajectories (Eq. (5.6)). Inverse dynamics control allows for a significant reduction in the error feedback gains, for compliant control, while ensuring high tracking performances.

For the position control we assume that there are no contacts with the environment, therefore our inverse dynamics control law does not take into account possible external forces. The contacts with the environment are dealt with using the following force controller.

Endeffector Force Control

The tracking of desired contact forces is achieved with a PI controller

$$\boldsymbol{\tau}_{arm,f} = -\mathbf{J}^T \Big(\mathbf{F}_{arm\_des} - \mathbf{F}_{arm} + \mathbf{K}_i \int_{t-\Delta t}^{t} (\mathbf{F}_{arm\_des} - \mathbf{F}_{arm}) \; dt \Big) , \tag{5.8}$$

where $\mathbf{F}_{arm}$ and $\mathbf{F}_{arm\_des}$ are the measured and desired forces at the endeffector, $\mathbf{K}_i$ is a (diagonal) positive definite gain matrix and Δt represents the time-window during which the force error is integrated. Note that we use unitary gains for the proportional case since it exactly eliminates the sensed force and replaces it by the desired one assuming a perfect rigid body system. The integral controller will compensate for steady-state errors during contact.

Combined Position and Force Control

The final command sent to the arm joints is computed according to

$$\boldsymbol{\tau}_{arm} = \boldsymbol{\tau}_{arm,p} + \boldsymbol{\tau}_{arm,f} \quad . \tag{5.9}$$

By choosing an appropriate desired position and force, these controllers act together to achieve the desired task. Our design of the feedback at the DMP level creates force-consistent desired trajectories. The online adaptation of the DMP, thus, allows us to simply use the addition of both position and force controller.

Remark The resulting controller is simple but not as general as an impedance controller would be. However, it has the advantage that we do not need to explicitly design a desired impedance behavior in our control. Indirectly, the choice of (potentially time-varying) gains related to the DMP adaptation and to the force controller will define a

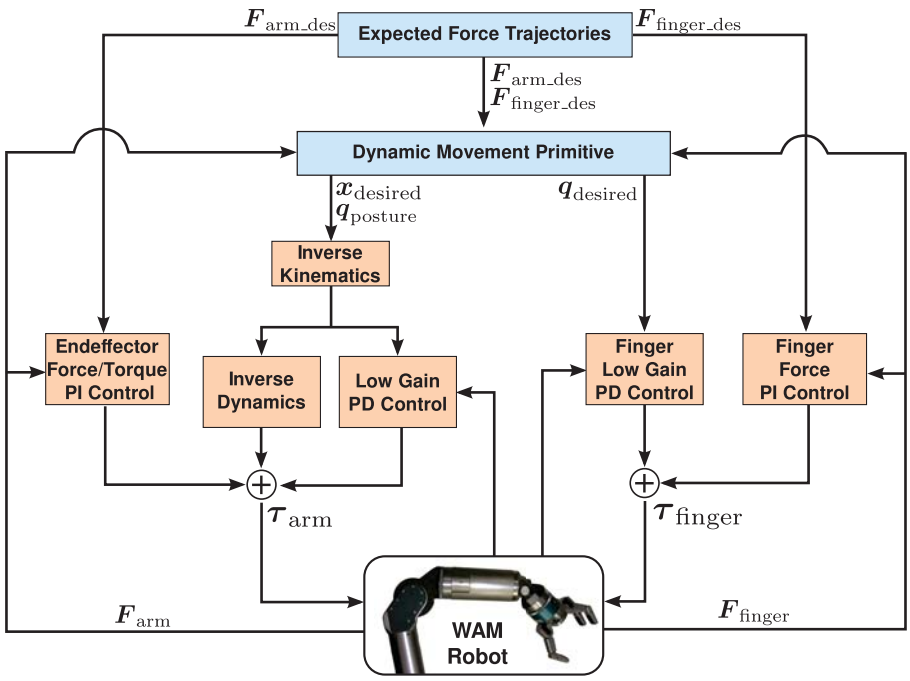


Figure 5.4: Diagram of the proposed control architecture.

desired impedance for the controller. This controller turns out to give us sufficiently good performance in practice. Note that we would not be able to use a hybrid force/position controller since we don't want to decouple the directions of control for the position and the force.

Finger Position and Force Control

The fingers are controlled with a position PD controller and a force PI controller. The force readings are obtained from the strain gage sensors located in the second joint of each finger. The spread between the left and right fingers is only controlled in position. The total torque command for the fingers is computed according to

$$\boldsymbol{\tau}_{finger} = \mathbf{PD}_{position} + \mathbf{PI}_{force} \ . \tag{5.10}$$

Remark This controller renders the fingers compliant even if they are non-backdrivable (which is the case in our experiments). It complements the finger DMP adaptation to ensure a fast response to perturbations.

An overview of the presented control architecture is shown in Fig. 5.4.

5.3.5 Learning from Previous Experience

Our approach aims to generate a rich set of movement primitives that allow online adaptation of the desired behavior based on *external* forces and torques. To acquire new motor skills we propose a two staged approach: First, use imitation learning to bootstrap the learning process and encode all kinematic information. Second, execute the learned behavior in open loop mode, i.e. setting the feedback terms $\zeta = 0$ in Eq. (5.5), and record the experienced *internal* forces and torques. These recorded force and torque trajectories resemble a predictive model of expected sensor readings *without* external perturbations. Thus, the desired kinematic trajectory will only adapt if there is a deviation between the measured and the predicted forces and torques. Furthermore, accurate modeling of the inertia parameters of the endeffector, e.g. hand and fingers, can be omitted since they are taken into consideration during the open loop execution. Our approach therefore works seamlessly when additional weight needs to be compensated, e.g. when lifting a cup.

5.3.6 Experimental Results

In the following, we present a series of grasping experiments that highlight the feasibility of ASMs to account for uncertainty in the object pose.

Experimental Setup

The experiments were conducted using the 7 DOF WAM robot arm and the 4 DOF Barrett Hand BH-280, both built by Barrett Technology, Inc. Additionally, we used the force/torque sensor mounted on the wrist of the arm and the joint-torque sensors embedded in the knuckles of the 3 fingers to close the force control loops (Fig. 5.4). The overall position and force control loop including reading all sensor values and sending desired torques runs at 300 Hz. See Appendix A.3 for more details about the robot.

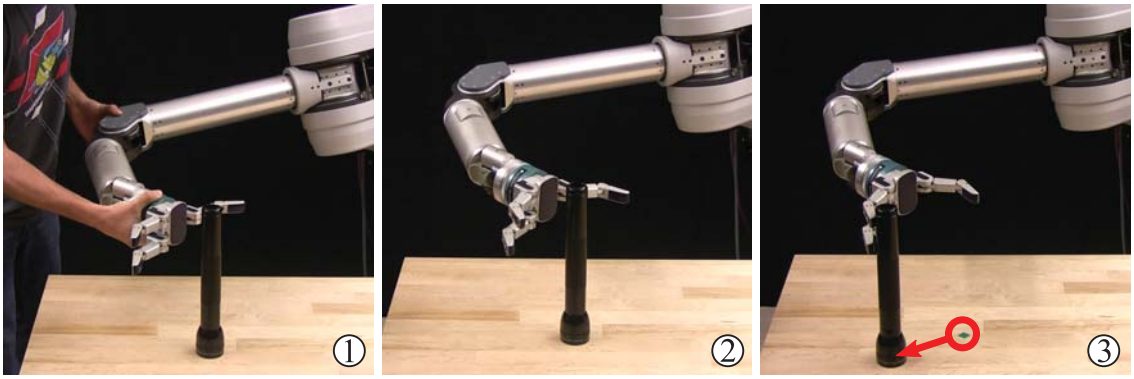


Figure 5.5: Proposed method: (1) Use imitation learning to learn movement plan, (2) execute learned movement and record sensor information, (3) use experienced sensor information to react to unforeseen perturbations, here simulated by varying the flashlight location (red arrow).

In our grasping experiments, we followed the procedure shown in Fig. 5.5. The grasping DMP was learned through kinesthetic teaching, i.e. guiding the gravity compensated WAM robot from an initial position to an appropriate grasp location. From this demonstration, we extracted the endeffector trajectory and learned a DMP in Cartesian space using the new quaternion formulation described in 5.3.2. Additionally, the desired finger joint positions were learned from a minimum jerk trajectory that was generated to ensure synchronous finger closing. Thus, we used kinesthetic teaching to learn a single DMP encoding all kinematics information, i.e. the desired endeffector position and orientation, the desired default posture* and the desired finger trajectories. To obtain a predictive model of the forces, as described in 5.3.5, we executed the learned DMP and recorded the forces and torques sensed at the wrist of the WAM robot arm. These force profiles resemble the *default* sensor experience for this particular grasp (see Fig. 5.4).

We tested the proposed method in three tabletop grasping experiments: Grasping a cup (Fig. 5.3), an upright standing Maglite flashlight (Fig. 5.5) and a lying bottle of water (Fig. 5.3). All objects have been misplaced in order to simulate uncertainty.

Experimental Results

A typical behavior of the proposed approach while grasping a misplaced red cup from the side as depicted in Fig. 5.3 is shown in Fig. 5.7. The results show that the robot significantly adapted its motion (up to 12 cm in position and 20 degrees in orientation) resulting in a successful grasp. The highlights of each grasping experiment are shown in the video available at `http://youtu.be/ZWkF2peI1qw`.

In order to quantify the improvements in grasping under uncertainty more rigorously, we performed a systematic test for grasping a Maglite flashlight. To simulate uncertainty, we placed the Maglite on a grid composed of 4x4 cm squares while keeping the goal grasp configuration fixed. We considered three different control modes: (a) open loop, (b) force control without DMP adaptation, and (c) force control with DMP adaptation of the desired movement plan. A grasp attempt was marked successful if the robot achieved a power grasp. If the flashlight was knocked over, for example due to early contact, the grasp attempted was marked unsuccessful. The result of the systematic test is shown in Fig. 5.6.

As shown in Fig. 5.6, the region of successful grasps is increased if the force control loop is closed. However, only if both, the force control loop as well as the DMP feedback loop are closed the region of successful grasps is increased significantly. Even if the object was misplaced by up to 23 cm, the robot was able to gently give in (without knocking over the flashlight) and successfully grasp the flashlight. Snapshots from this experiment are shown in Fig. 5.8.

*The default posture is used in Eq. (5.7) to resolve the redundancies of the 7 DOF WAM robot.

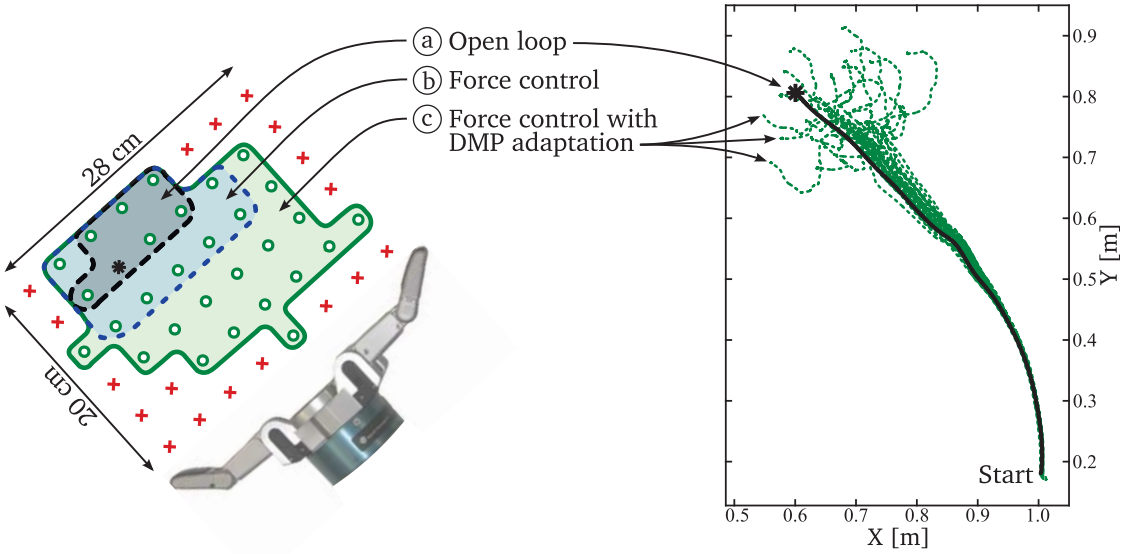


Figure 5.6: Result of each grasping attempt after misplacing the flashlight (left). The grid is 28 cm large and 20 cm wide. There is 4 cm between each adjacent crosses. The region in which open loop (black dashed) execution succeeds is rather small (a). Adding force control (blue dashed) increases the region (b). However, only when the presented DMP adaptation is used, the region of successful grasps increases significantly (c). The thick black line shows the open loop trajectory and green dashed lines show how those trajectories were adapted after touching the object (right).

The same approach was successful in grasping a water bottle lying on the table as shown in Fig. 5.3. This experiment suggest that our approach generalizes to a variety of grasp scenarios requiring important contact interaction and compliant behavior. In this particular experiment, the execution of the learned movement in open loop mode to record the endeffector forces (step 2 in Fig. 5.5) has been done without the table. Indeed, we decided to not execute the movement without the force controllers because of the danger of breaking the fingers of the robot due to missing compensation when touching the table. In contrast, our results with the force controllers and the DMP adaptation show that our approach is safe (see also video-supplement at `http://youtu.be/ZWkF2peI1qw`). Unexpected collisions with the table are compensated in a controlled fashion. This experiment emphasizes the compliant properties of the controller and its ability to generate feasible motion when in contact with the environment. Note, the proposed approach can be extended to additionally improve predicted force trajectories using trial-and-error learning similar to (Kalakrishnan, Righetti, Pastor, & Schaal, 2011).

5.3.7 Conclusion

In this section, we presented a sensory feedback law for DMPs to realize online movement adaptation. Alongside, we derived a mathematically correct DMP formulation for orientation trajectories represented as unit quaternions. We highlighted the importance of

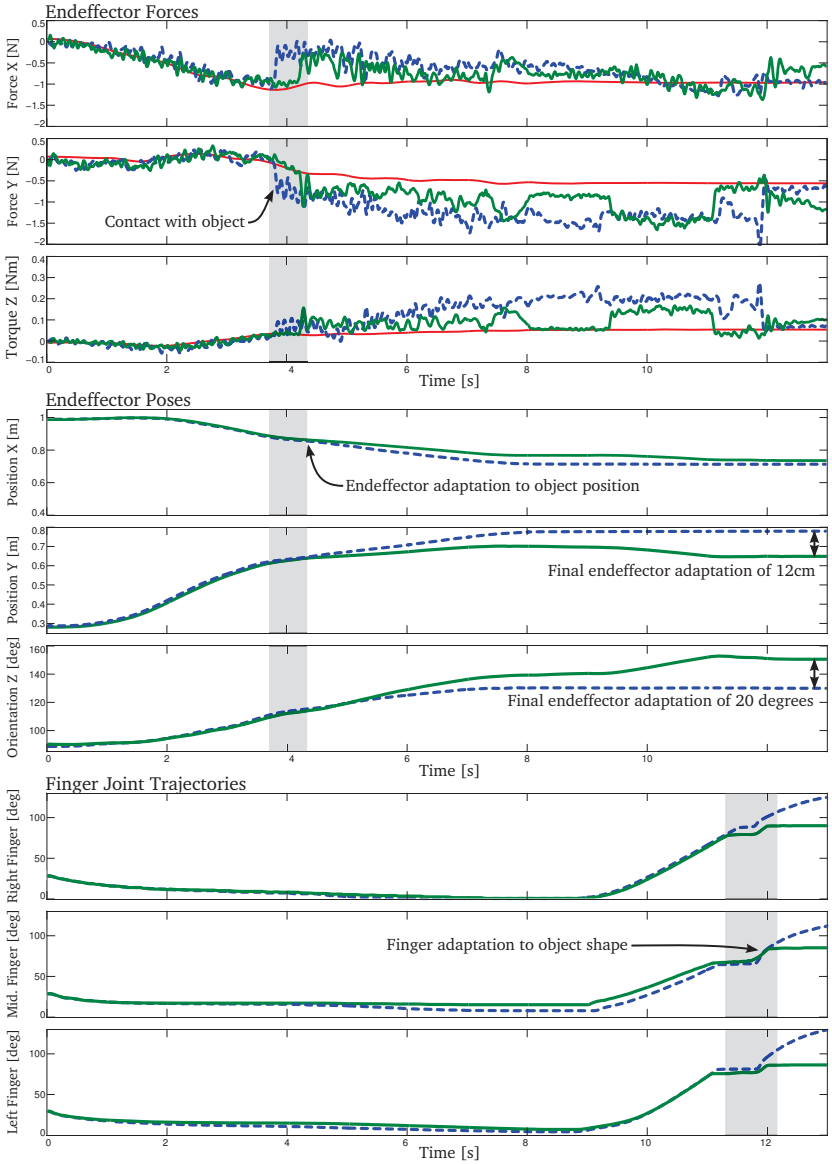


Figure 5.7: Typical behavior obtained from grasping a misplaced cup. The top three plots show the endeffector forces and torque in the horizontal plane, the middle three plots show the endeffector positions and orientation in the same plane, and the bottom three plots show the finger joint trajectories. The green lines were obtained from experiments with DMP adaptation, the blue dashed lines were obtained from open loop execution. The red lines in the top three plots correspond to the expected desired forces recorded from a previous successful grasp. In the beginning of the movement the closed loop endeffector trajectories remain close to the open loop endeffector trajectories. After unexpectedly touching the misplaced cup (gray region around 4 sec) the closed loop controller adapts the endeffector positions and orientation leading to a successful grasp. The finger joint trajectories are adapted to the shape of the cup (gray region around 12 sec).

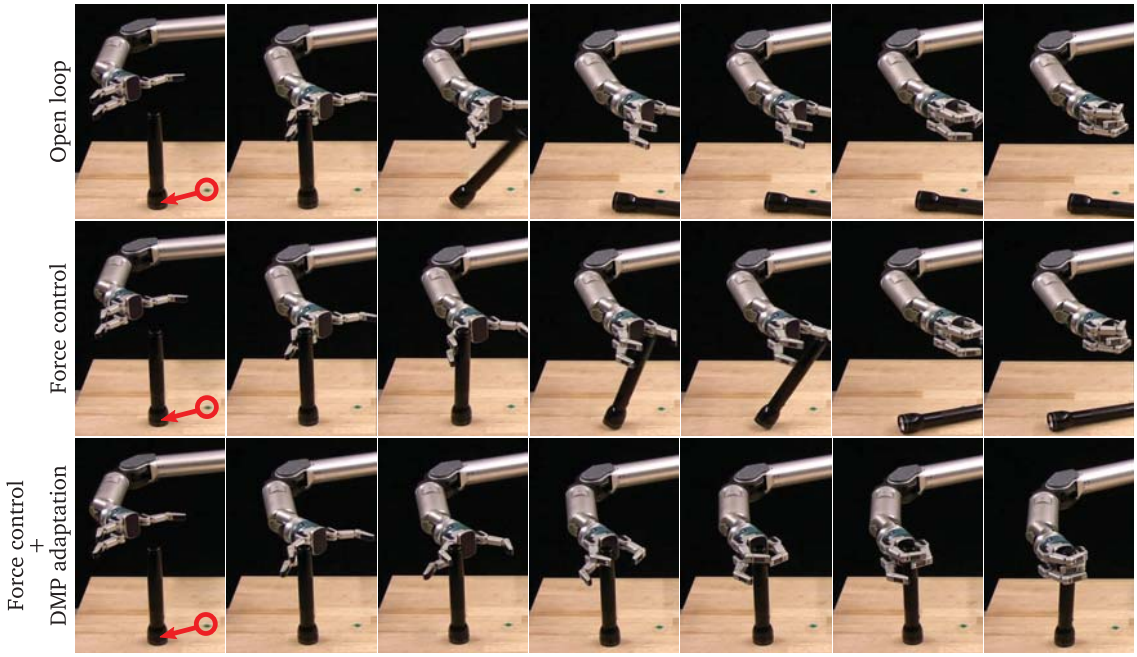


Figure 5.8: Snapshots of the flashlight grasping experiment. Uncertainty is simulated by placing the flashlight off the target (green marker on table) while keeping movement goal fixed. Open loop execution (top row) fails immediately. Including force control to servo the desired force allows for slight adaptation, however, the tracking error overrules the force controller and the grasp fails as well (middle row). Only if the DMP adapts the desired trajectory, the grasp succeeds (bottom row).

stereotypical movements and proposed a general framework that uses sensor information from previous successful trials to systematically compensate for unforeseen perturbations in future trials. The presented approach has been integrated with a low-level position and force control system. The experiments show that our control system is compliant and reactive enough to sense a misplaced upright standing flashlight. Furthermore, our system was able to online adapt its movement plan without knocking over the flashlight. Our approach significantly increased the region of successful grasps. It also allowed to create very compliant behaviors for contact interaction with the environment as we demonstrated in the water bottle grasping that required contact with the table. Nevertheless, the presented adaptation behavior is a local method and therefore has well known limitations. Thus, there are situations in which following the gradient of the Jacobian will not suffice to achieve the manipulation task. In these cases, it is important to realize the necessity to switch to a different manipulation behavior, i.e. movement primitive. The following section will show how associated sensor information can be used to determine such sequences of movement primitives online.

5.4 Online Movement Sequencing using Sensor Predictions

Associating experienced sensor information with corresponding robot action facilitates the generation of a memory-based predictive model for subsequent similar situations. These sensory experiences resemble the expected *nominal* sensation during the movement, including confidence intervals from multiple skill executions. Deviations from the nominal behavior can be detected with statistical hypothesis testing (Pastor, Kalakrishnan, et al., 2011) as described in Sec. 5.2. Furthermore, provided knowledge about appropriate Jacobians associated with the location of the sensors, movement plans can be adapted online to react to unforeseen perturbations (Pastor, Righetti, et al., 2011) as described in Sec. 5.3. In this section, we will present how associated sensor information can be used to determine movement sequences online using an approach inspired from neuroscience.

5.4.1 Neuroscientific Inspiration

Neuroscientific evidence suggests that human hands are equipped with special type of tactile sensors, called fast-adapting type II (FA-II)*, that specifically detect mechanical events, i.e. the making and breaking of contacts of the hand with objects including distant events acting on hand-held objects. Such mechanical events "mark transitions between consecutive action phases" and "represent subgoals of the overall task" (Johansson & Flanagan, 2009a). For example, when lifting an object, contact between the hand and the object marks the end of the reaching phase, while breaking the contact of the object with the support surface marks the end of the load phase and the beginning of the lifting phase. When lifting an unexpectedly light or unexpectedly heavy object, the presence or absence of FA-II sensor predictions (burst responses) at the predicted point elicits a corrective action (Johansson & Flanagan, 2009b). Our framework is developed to mimic this behavior: Unpredicted sensory events result in aborting of the current movement primitive and triggering an appropriate correcting movement primitive. To achieve smooth transitions between subsequent movements, our approach chooses among available candidate actions by comparing the currently measured sensor feedback with previous sensor experiences and selecting the closest match.

An important insight that is being exploited throughout this chapter is that stereotypical movements give rise to stereotypical sensory events (see Fig. 5.1). The concept of stereotypical movements is inspired by the repetitive nature of day to day manipulation movements such as for example reaching for and turning a door knob. Opening a door will - provided stereotypic execution - always result in similar tactile feedback at the finger tips, similar force measurements at the knuckles of the finger, and similar force/torque measurements at the wrist. Similar to the object lifting task from above, discrete haptic

*FA-II is extremely sensitive to mechanical transients and high-frequency vibrations (≈40-400 Hz) propagating through tissues. Furthermore, it is insensitive to static force and respond to distant events acting on hand-held objects, see (Johansson & Flanagan, 2009a) for more information.

events naturally decompose this task into a series of action phases, or movement primitives, i.e. reaching for the door knob, aligning the fingers with it, turning it until its limit, and pulling it to open the door. Each of these intermediate states have distinct sensor signatures which mark task subgoals. Uncertainty in the geometric and physical properties of the door for example can be dealt with by focusing on these sensor signatures and choosing subsequent movements only if task subgoals are reached, i.e. expected sensor feedback has been perceived. Sequencing movements online based on sensory feedback thus creates funnels in the sensor space that guide the movement generation process to reach each task subgoal ensuring overall task success. Ultimately, it is up to the motor skill to ensure that measured sensor values remain within these funnels, i.e. close to the predicted/anticipated sensor values. Such an adaptation mechanism has been proposed in the previous section (see Sec. 5.3.3).

In this section, we propose to use the accumulated sensor information to choose and trigger subsequent movements online, i.e. while the robot is performing the task. Our idea is that a particular movement primitive is only applicable if the associated sensor information corresponds sufficiently with the currently sensed sensor information of the robot. In the door knob task from above, such an approach would, for example, prevent a robot from starting to turn the knob unless the fingers have properly established contact with it (e.g. sensed by finger tip pressure sensors, see Fig. 5.9). If this sensory event is missing, a different movement might be chosen and triggered, such as a re-grasp movement. Thus, this section extends our previous work on stereotypical movements (see previous sections

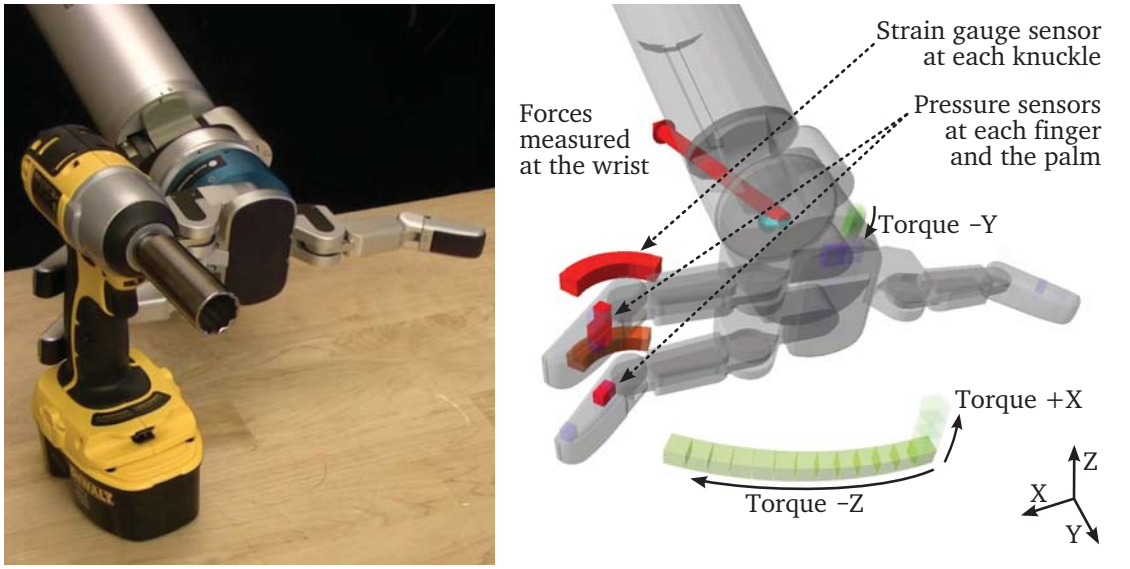


Figure 5.9: Barrett WAM and Barrett hand reaching for a drill (left) and corresponding visualization of measured sensor information (right). Forces and torques at the wrist are visualized with the red arrow (forces) and half circles in front of the hand and around the wrist (torques). Torques in the knuckles of the fingers are visualized with half circles. Pressure values at the palm and each finger tip are visualized with cube markers, red cube markers indicate active cells.

in this chapter) to determining appropriate manipulation sequences *online* to achieve complete manipulation skills. The movement plan is adapted online (Pastor, Righetti, et al., 2011) to servo around these expected sensor values further ensuring that similar task execution indeed result in similar sensory experiences. The combination of reactive behaviors with stereotypical movements can create a rich set of possible motions that account for external perturbations and perception uncertainty to generate truly robust behaviors.

The example task considered is to reach for a drill on a table and turn it on (see Fig. 5.9). Perceptual uncertainties are simulated and the drill position is varied. A series of grasping and re-grasping (correction) movements are being encoded into Dynamic Movement Primitives (DMPs) and provided to the system. Along with the movement plan, each DMP encodes the expected sensor traces as well. The sensor information are recorded from repeatedly executing each DMP under slightly varying conditions (i.e. slightly different object positions). From these multiple trials, mean μ and standard deviation σ are computed and used as the *nominal* behavior that is to be expected if this particular movement primitive is being executed. These accumulated sensor statistics (see Fig. 5.10) that are time-aligned with corresponding stereotypical movements are referred to as *associative memory*. Over time, each stereotypical movement is paired with a growing set of previous sensor experiences.

The remainder of this section is structured as follows: Sec. 5.4.2 will highlight important aspects of encoding ASMs. Sec. 5.4.3 will illustrate how ASMs can be acquired. In Sec. 5.4.4 the overall system is presented. The experimental setup and results are presented in Sec. 5.4.5. Finally, the approach is concluded in Sec. 5.4.6.

5.4.2 Encoding Associative Skill Memories

Associative Skill Memories (ASMs) are stereotypical movements that also maintain a set of associated sensor traces describing the nominal sensor expectations (see Fig. 5.10). The movement plan itself as well as the associated sensor signals are encoded as DMPs*. Thus, ASMs generate desired movement trajectories along and in sync with anticipated sensor feedback. These sensor predictions are used inside feedback control loops (see Fig. 5.11) modulating the desired movement trajectory such that the expected sensor values remain close to the predicted ones, as described in Sec. 5.3.3.

When sequencing movement primitives, it is important to ensure smooth transitions, e.g. continuous velocity or acceleration profiles. Previous experiments, see Sec. 4.2.1, have exploited the DMPs ability to generalize locally (see Sec. 3.6) by setting the initial position and velocity of subsequent DMPs according to the current desired position and velocity. Such initialization facilitates seamless transition from one movement primitive to the next. To also ensure smooth transitions despite additional feedback control loops (see

*As described in the previous section, Sec. 5.3.2, special care is taken to correctly encode quaternion trajectories.

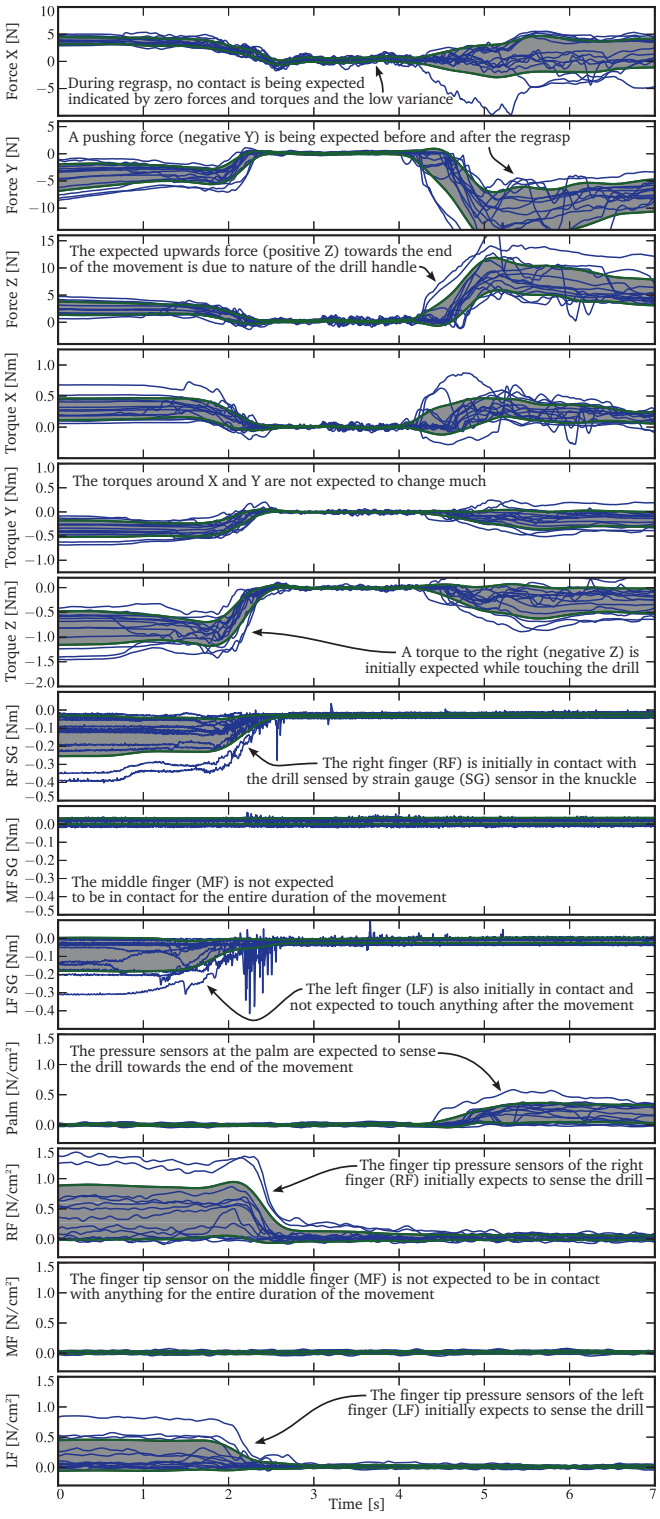


Figure 5.10: Recorded sensor values (blue) after touching the drill with the two fingers (see Fig. 5.9). The executed movement re-aligns the palm of the hand with the drill similar to the correcting movement after step 2 shown in Fig. 5.15. Mean μ and 1-standard deviations σ (green) resemble the nominal sensor experiences for this stereotypical movement. Even though the start and end of the movement as well as the drill position has been changed slightly, the experienced sensor data shows strong correlations across trials. Our approach servos these sensor traces to create even stronger correlations to previous trials (see Sec. 5.3.3) and exploits these to determine manipulation sequences online (see Sec. 5.4.4).

Fig. 5.11), the same DMP property is being exploited (see Sec. 3.4). The initial conditions of those transformation systems that encode sensor predictions are also set according to the currently predicted sensor values. Such initialization ensures that at the point of transition the feedback contribution will remain constant given the difference between measured and predicted sensor values has not changed. Thus, smooth transitions between subsequent ASMs are guaranteed through appropriate initialization of both, movement plan and associated sensor feedback, of the subsequent ASM.

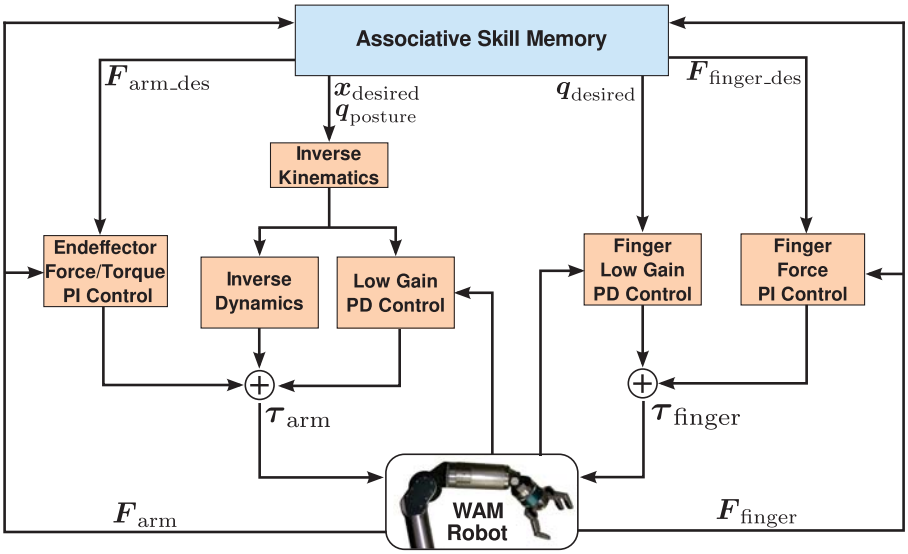


Figure 5.11: Diagram of the proposed control architecture: Desired positions and orientation generated by the ASM are tracked using a velocity-based operational space controller together with an inverse dynamics law and feedback error compensation in joint space, see (Nakanishi et al., 2008) for more details. Tracking of desired arm contact forces is achieved by closing a PI control loop on the force/torque sensor located at the wrist. The fingers are controlled with a position PD controller and a force PI controller using the strain gauge sensors located at the knuckles. The ASM also encodes the expected wrist forces and torques as well as the expected finger torques. These are used as set points of the corresponding force controllers. The ASM feedback adapts the movement trajectories online to remain close to the expected forces and torques.

We use a single ASM to encode the desired position and orientation of the endeffector, the arm posture*, the desired finger joint angles, as well as the expected force and torque trajectories at the wrist and the expected torques at the knuckles of each finger. An overview diagram of our control architecture is shown in Fig. 5.11. The proposed form of trajectory generation allows for very robust movements as it adapts its movement plans online based on expected force trajectories. In the previous section, this has been exploited to successfully grasp a flashlight off the table even though the position of the flashlight has been varied significantly. This section presents an extension to automatically

*The arm posture is encoded to resolve the redundancy in the nullspace of the task similar to the demonstrated movement.

determine the manipulation sequence online based on associative memory. It uses the online movement adaption mechanism (Pastor, Righetti, et al., 2011) to also exploit the fact that servoing around expected forces will ensure that subsequent task executions result in similar sensor experiences. The ability to relate the current sensor experiences to previous executions is key in determining the next movement primitive. Thus, our previous work (Pastor, Righetti, et al., 2011) is a crucial prerequisite. Sequences of movement primitives enable our system to deal with even larger uncertainty, such as changing the object pose mid-way.

5.4.3 Acquiring Manipulation Skills

These manipulation skills are acquired in 3 steps (see Fig. 5.12).

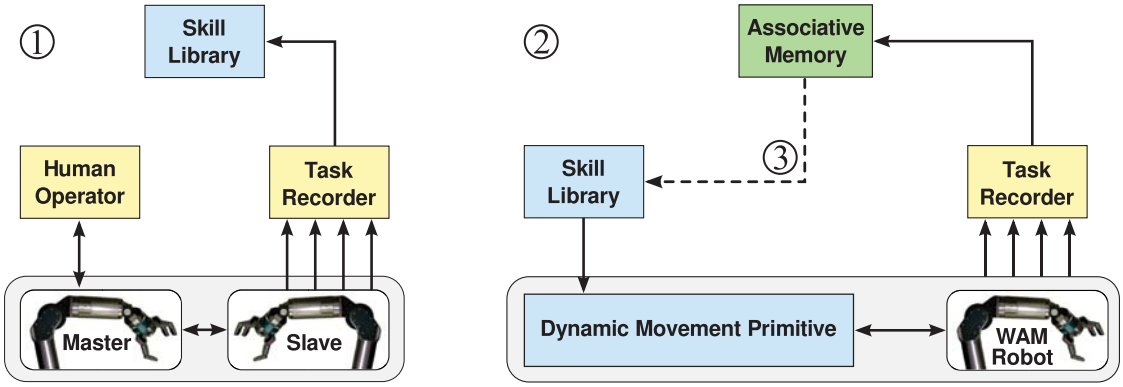


Figure 5.12: Diagram of the skill acquiring procedure: (1) First, a human operator demonstrates a set of movement primitives using the master-slave system. Force feedback enables the operator to also teach meaningful force trajectories. The task recorder time aligns all sensor signals and encodes them into DMPs which are stored in the skill library. (2) Second, these DMPs are retrieved and executed on the robot. The task is varied by changing the object position as well as the start/end of each DMP. Again, all available signals are recorded and stored paired with the DMP that has been executed. The mean and standard deviation are computed which represent the nominal sensor expectations for the associated DMP. (3) Finally, the DMPs are updated with the obtained mean trajectories of the recorded forces and torques.

First, a particular manipulation movement is demonstrated to the robot (e.g. using a master-slave system, see Sec. 5.4.5) while all sensor signals are being recorded. The recorded kinematic variables as well as the measured forces and torques at the wrist and the finger knuckles are encoded into a DMP and stored into the skill library*. Second, the DMP is being retrieved and executed on the robot while the manipulation scenario is slightly varied. This allows to capture task specific variations. The mean and the 1-standard deviation for these sensor traces are computed. These sensor statistics reflect the

*Note, as mentioned above, encoding expected forces and torques into DMPs avoids discontinuous predictions when transitioning between movement primitives by adapting the initial conditions (Pastor et al., 2009).

nominal behavior and are subsequently used to predict each sensor signal at each instant of time for the duration of the movement primitive. Finally, the DMP is updated with the mean of previously recorded forces and torques at the wrist and the finger knuckles.

5.4.4 Sequencing Manipulation Movements based on Associated Sensor Information

After acquiring a set of ASMs our system is able to constantly predict all sensor information and even use it inside the real-time control loop to servo around these predictions. Furthermore, the difference between current measurements and predicted sensor signals allows to constantly confirm successful execution of the task. Failure conditions are immediately detected when the measured signal deviates significantly from the expected behavior (Pastor, Kalakrishnan, et al., 2011). Furthermore, the proposed approach facilitates distinguishing between different kinds of failure conditions, which enables branching into the appropriate recovery action. On the other hand, if the measured signals matches the predictions, the next action in the default sequence is executed. This way of sequencing ASMs ensures successful execution of each movement before proceeding to the next. It subsequently follows that the initial sensor signature for the next ASM matches the currently measured signals (as indicated by the vertical dashed lines in Fig. 5.16). An overview of the proposed approach is shown in Fig. 5.13.

For the presented experiments switching to subsequent ASMs were triggered only after 90% of the movement has been executed. The closest match between currently measured sensor signals and the initial sensor signatures of all ASMs maintained in the library is computed using Nearest Neighbors with squared euclidean distance metric. This closest

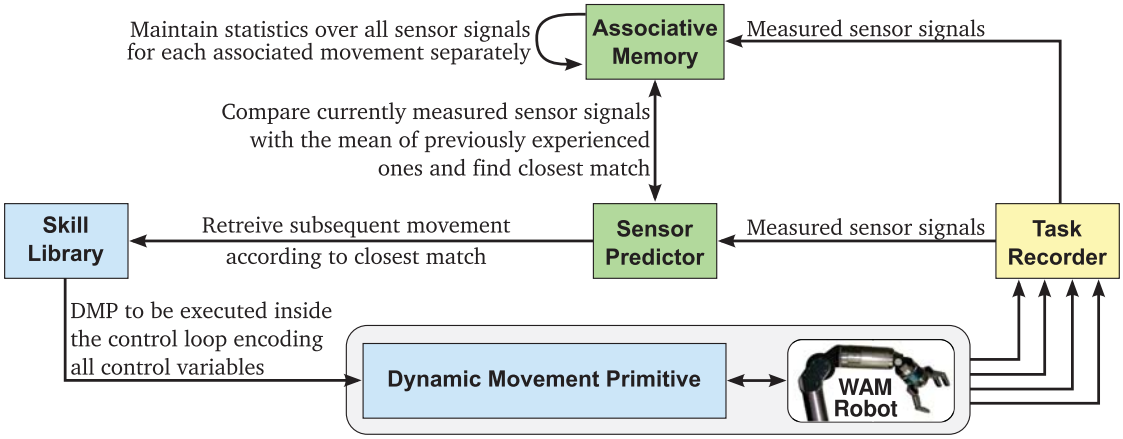


Figure 5.13: Diagram of the proposed approach of sequencing ASMs: The sensor predictor constantly compares the currently measured sensor signals with the mean sensor statistics from previous executions of the same stereotypical movement and find the closest match. Towards the end of each movement primitive, the DMP of the associated best match is retrieved from the skill library and send to the real-time robot controller.

match resembles the most appropriate subsequent movement. After a particular ASM was selected a total of 10 times, the corresponding DMP was sent to the controllers immediately (see Fig. 5.11). Note, the current implementation switches instantly between subsequent DMPs (as described in (Pastor et al., 2009)) which causes the acceleration profiles to be discontinuous. However, transitions that maintain a continuous acceleration profile while switching between DMPs can easily be added, e.g. (Nemec & Ude, 2011). For the presented experiments this did not pose a problem because at the time of switching between movements the robot was moving very slowly.

5.4.5 Experimental Results

The experimental setup consists of two Barrett WAMs equipped with one Barrett BH280 three finger hand each (see Fig. 5.14). The sensor suite of the hands include force/torque and accelerometer sensors at the wrist, pressure sensors on each finger tip and the palm, and strain gauge sensors in the knuckles of each finger (see Fig. 5.3). All the sensors are polled at 300 Hz except the pressure sensors which are polled at 25 Hz. Please refer to Appendix A.3 for more details about the robot.

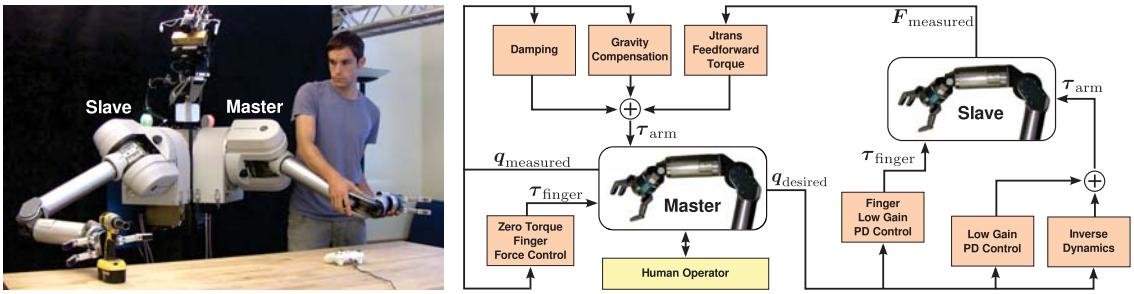


Figure 5.14: Dual arm manipulator used as master-slave system to teach manipulation skills (left) and corresponding control diagram (right). Movements of the master arm are mimicked by the slave arm. Force feedback is provided using the force/torque sensor at the wrist of the slave arm.

To record movements as well as sensor data from the hand we developed a master-slave system in which one arm is used to teleoperate the other. The master-slave system is realized by setting the measured arm and finger joint angles of the master arm to be the desired arm and finger joint angles of the slave arm. The finger joints on the master arm have been made actively compliant by closing a force control loop on the strain gauges. Furthermore, force feedback is provided to the human operator using the force/torque sensor at the wrist of the slave arm (see Fig. 5.14). The master-slave system preserves all the sensor information experienced during the demonstration. Using kinesthetic teaching, i.e. moving one of the robot arms directly, would result in significantly disturbed sensor signals.

The task consisted of turning on a drill standing upright on a table. We taught 5 different reaching movements with the drill placed at 5 different locations to simulate perception

uncertainties. The 5 locations are chosen such that initial contact is made with the palm, with the first link of the middle finger, the fingertip of the middle finger (see image 2 in Fig. 5.15), the first links of the left and right fingers, and the fingertips of the left and right fingers (see Fig. 5.9).

Appropriate re-grasp behaviors were demonstrated for 4 of these object positions. Finally, finger closing and triggering behaviors were demonstrated, resulting in a total of 11 ASMs. Each of these movements has only been demonstrated once using the master-slave system. The demonstrated movement primitives have been chosen to facilitate very robust task execution by appropriately choosing subsequent primitives online (see Sec. 5.4.4). During the ASM acquisition procedure (see Sec. 5.4.3) the start and goal of each DMP was varied by a few centimeters to explore different parts of the workspace. The Jacobian of the task $\mathbf{J}_{sensor}$ in Eq. (5.4) simplified to an identity matrix $\mathbf{I} \in \mathbb{R}^{9\times 9}$ given that the DMP has been encoded in the hand frame and the particular location of the sensors. As shown in Fig. 5.11, the desired position and orientation $\boldsymbol{x}_{\text{desired}}$ has been adapted online based on the difference between the expected forces and torques (red lines in top 6 plots in Fig. 5.16) and those measured at the wrist (corresponding blue lines). Respectively, the desired finger joint trajectories $\boldsymbol{q}_{\text{desired}}$ have been adapted based on the difference between the expected finger torques (red lines in bottom 3 plots in Fig. 5.16) and those measured at the finger knuckles (corresponding blue lines). The recorded sensor traces associated with the "re-grasp right movement" are shown in Fig. 5.10. The proposed method was verified for various initial drill positions. Fig. 5.15 shows one trial in which the drill was misplaced to the left. The corresponding plots for the force/torque sensors at the wrist and the strain gauge sensors at the knuckles are shown in Fig. 5.16, and plots of the pressure sensors are shown in Fig. 5.17. Dashed vertical lines in Fig. 5.16 and Fig. 5.17 indicate transitions between successive ASMs, and the state of the system at these transition points is depicted in Fig. 5.15. At time instant (2), torques at the wrist and middle finger, and pressure at the finger tip indicates that the drill is misaligned. Subsequent ASMs are determined online as described in Sec. 5.4.4, i.e., the past 30 sensor values (corresponding

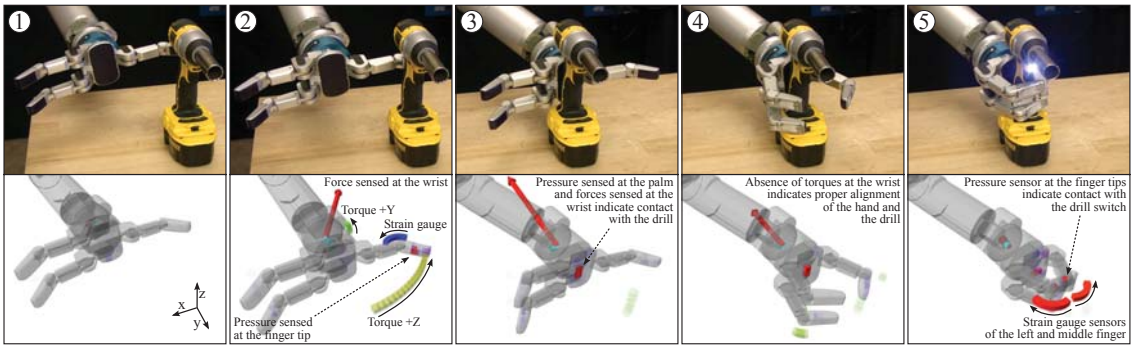


Figure 5.15: Sequenced execution of movement primitives leading to successfully turning on the drill even though the drill was misplaced (top row) and visualized sensor data (bottom row). Fig. 5.16 and Fig. 5.17 show the corresponding plot of expected and measured sensor traces. The video of this experiment is available at `http://youtu.be/lL4-onuLDy0`.

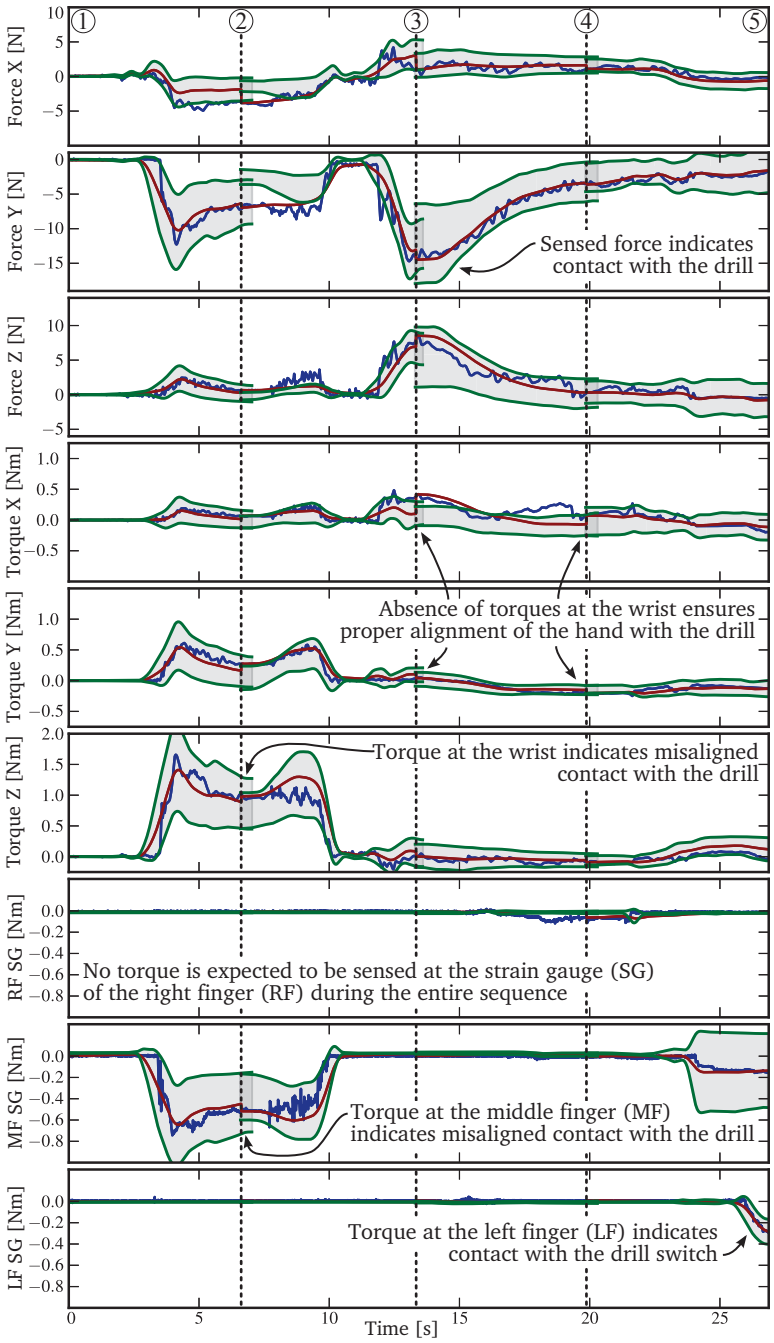


Figure 5.16: The expected sensor traces (red line) used inside the control loop (see Fig. 5.11) are set to the current sensor state (blue line) at the point of switching. The green lines show ± 1-standard deviation. Dashed vertical lines indicate transitions. The plot shows that our method closely predicts all sensor values while performing the task (see Fig. 5.15) and correctly chooses subsequent movement primitives.

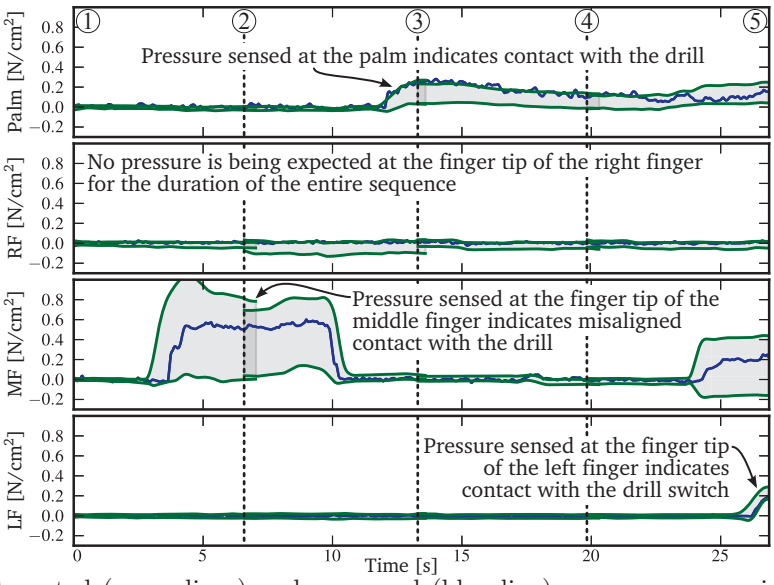


Figure 5.17: Expected (green lines) and measured (blue line) pressure sensor signals while performing the task shown in Fig. 5.15.

to 0.1 seconds) of all available sensors are compared with all initial sensor signatures of all 11 ASMs maintained in the library. The closest match is selected, in this case the "re-grasp left movement", also facilitating continuous (predicted) sensor signals. Execution of this re-grasp behavior aligns the hand with the drill, which is confirmed by the sensed force at the wrist and pressure at the palm, see instant (3). This allows the system to proceed with the finger closing and triggering behaviors, finally bringing the system to the state at time instant (5) where pressure sensed at the left finger tip indicates contact with the drill switch. Determining subsequent ASMs online facilitates our system to cope with significant object pose uncertainty, even if the drill is being moved mid-way as shown in the video available at `http://youtu.be/lL4-onuLDy0`.

The red lines in Fig. 5.16 correspond to the predicted sensor signals that have been used inside the real-time control loop, see Fig. 5.11. As described in Sec. 5.4.3, these sensor signals have been encoded into DMPs. At the point of transition between two subsequent ASMs (dashed lines in Fig. 5.16) the start of the sensor DMP is set to the current state. The obtained online movement adaptations of the desired endeffector and finger joint trajectories drive the current sensor state to remain close to the predicted values. This significantly contributes to generating stereotypical sensor traces even under uncertainty. That is, even if the drill is misplaced causing for example premature contact, the online adaptation of the desired endeffector pose and finger joint trajectories based on force feedback causes the hand and fingers to give in ensuring that subsequent executions result in similar sensor experiences; a key feature that allows our system to successfully retrieve the most appropriate subsequent ASMs. The task could successfully be completed even if the drill was misplaced manually mid-way through the trial, as shown in the

video http://youtu.be/lL4-onuLDy0. We want to stress the generality of the proposed approach. No task specific engineering has been done to achieve these results.

5.4.6 Conclusion

The results highlight a key feature of Associative Skill Memories: The ability to continuously predicting all associated sensor signals. These predictions can be used to determine robust sequences of acquired skills online to achieve a complete task. We have demonstrated the feasibility of this approach on a real robot. Finally, we want to emphasize that our approach presents a general method to close perception-action loops for manipulation tasks.

In the next section we will extend the approach to use a richer set of sensor signals, including visual feedback, as well as additional features and apply it to a bi-manual manipulation task. We will further introduce the concept of a manipulation graph and present extensions to our online sequencing mechanism.

5.5 Incremental Learning of Perception-Action Loops

In this section we will enrich our ASM representation with additional sensors including visual tracking of the hands and object, sound processing, and additional features computed from various sensor information. Furthermore, we will present an extension to our online sequencing method presented in Sec. 5.4.4 to also incorporate failure detection similar to the approach presented in Sec. 5.2. The considered bimanual manipulation tasks involves the ARM-S robot unscrewing the cap of a bottle and screwing it back on as shown in Fig. 5.18.

Our approach aims at making use of all available sensor information even (or especially) if some of these sensor signals are redundant. More sensor feedback enables the robot to acquire a more complete model of the task at hand. Redundant sensor information increases robustness as well as decreases uncertainty. Computing additional features from different sensor signals allow to bias the system to exploit correlations in the data. We will introduce such features in Sec. 5.5.1. However, as the dimensionality of the feature space increases choosing an appropriate distance metric becomes crucial. Therefore, in Sec. 5.5.2 we propose an automatic method to determine the distance using a Naive Bayes approach. In Sec. 5.5.3, we introduce the concept of a manipulation graph to reduce the complexity of the search problem without sacrificing performance. Finally, we show experimental results in a bimanual manipulation task shown in Fig. 5.18.

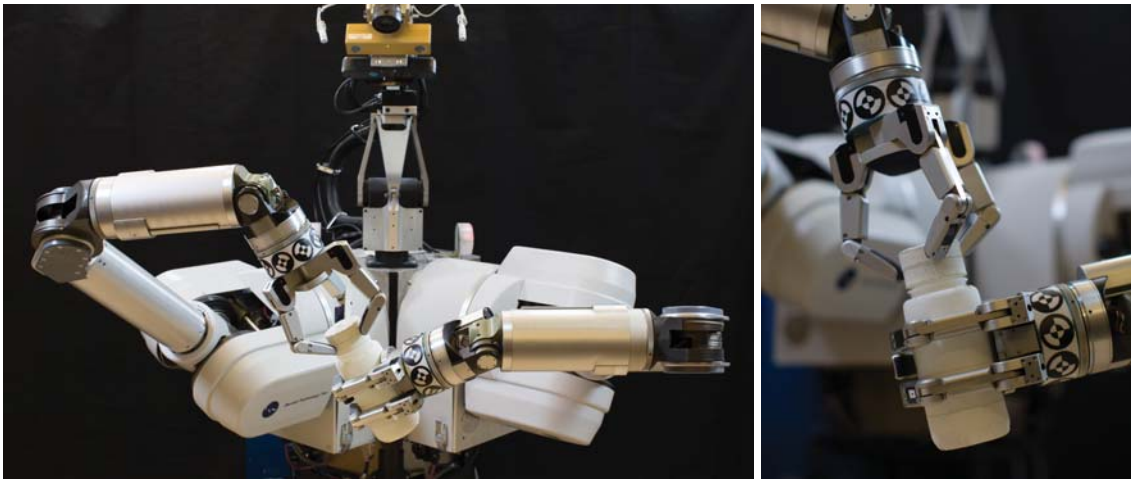


Figure 5.18: The ARM-S robot manipulating a bottle (left) and closeup of the hands (right). The fiducials at the wrist of the robot facilitate tracking using the Bumblebee stereo camera. The Asus Xtion sensor is used to track the bottle. The two white microphones at either side of the camera are used to generate audio features.

5.5.1 Visual Tracking and Feature Generation

We have enriched our ASM representation with visual and auditory (real) sensor information as well as additional (virtual) features. In the following we will describe both additions.

To track the manipulated object we used a real-time object tracking algorithm introduced in (Wüthrich, Pastor, Kalakrishnan, Bohg, & Schaal, 2013). The algorithm matches 3D information from a depth camera against a known object model. An interesting aspect of our approach is that occlusions are explicitly modeled. Thus, despite partial occlusions, which are inevitable during manipulation, our approach can robustly track the object. Finally, our approach allows to incorporate knowledge about the control input for the case when the robot is holding the object, thus reduces the uncertainty of the estimated object pose.

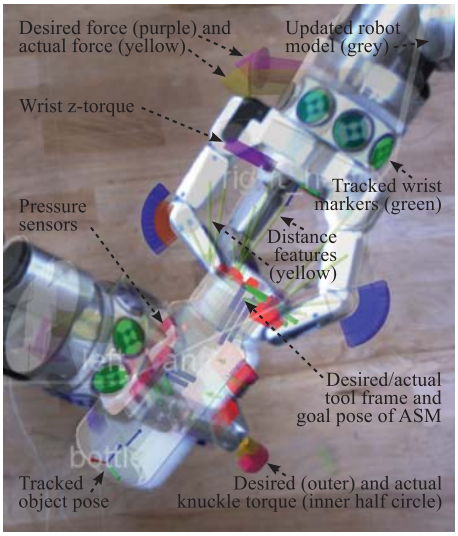
To accurately track the hands during manipulation we use a marker based approach. The markers, or fiducials, are arranged around the wrist and can be seen in Fig. 5.18. The detector* matches an initial pose estimate of the fiducial against a local neighborhood in the camera image to obtain an improved/updated estimate. This process is repeated for each image of each of the stereo camera at the frame rate of the camera, i.e. 15 Hz. The 3D position of each detected fiducial is obtained using triangulation. Provided knowledge about the arrangement of the fiducials we use least-squares to find a good hand pose. Finally, we fuse this information with knowledge about the control input using a Kalman filter. Using the visually tracked hand pose we update our joint angle estimate to match

*The fiducial detector has kindly been provided by Paul Hebert, Jeremy Ma, and Nicolas Hudson from the Jet Propulsion Laboratory, Pasadena.

the observation, see overlaid robot model in Fig. 5.19. Note, the visually tracked hand pose is used as the current estimate of the endeffector, i.e. $\mathbf{x}$ in Eq. 5.7.

Besides visual information, we also enriched our ASM representation with auditory signals. Auditory information is rarely being considered during manipulation, yet we believe it can provide valuable information about the progress of a task. Especially failure conditions are oftentimes accompanied with sound feedback. We have utilized Mel-frequency cepstral coefficients (MFCC) as they are commonly used representation of speech and audio signals (Davis & Mermelstein, 1980). We choose to only use the lower 8 MFCCs (instead of all 12) as they sufficiently covered the frequency spectrum of interest.

Finally, we have enriched our ASM representation with additional (virtual) features that are computed from several (real) sensor signals. These features allow to (manually) uncover and emphasize on task relevant correlations. For example, the proximity of the robot's endeffector to a visually tracked object is strongly correlated with the force/torque measurement at the wrist. Hereafter, were refer to both real sensor information as well as computed features simply as features. The feature count and further description used for the experiments is provided in Tab. 5.19. For more detailed information about the experimental platform please refer to the Appendix A.3.



Feature count and descriptions	
4	Joint angles of the right hand
12	Force/torque measurements at both wrists
3	Torques at the knuckles of the right hand
10	Distance between locations on the visually tracked right hand and the corresponding closest point on the tracked bottle
14	Position and orientation (in quaternion notation) of the right palm and tool frame to the tracked pose of the bottle
12	Relative linear and angular velocities of the right palm and tool frame and the tracked bottle
8	Mel-frequency cepstral coefficients (MFCC) computed using the audio recordings of the two microphones located in the sensor head

Figure 5.19: Image recorded from the left Bumblebee camera of the robot overlaid with sensor and feature information (left) and description and feature count (right).

Note, all features are encoded such that their values do not depend on the absolute pose (in world coordinates) of the hand. Thus, executing a learned manipulation task in different regions of the workspace will result in similarly correlated sensor feedback.

5.5.2 Automatic Relevance Detection

As described in Sec. 5.2, task progress monitoring is an essential requirement for autonomous manipulation systems. Previously, we introduced a statistical method that is able to detect failure conditions indicated by deviations of the measured sensor signals from the expected ones. Similarly, in Sec. 5.4, we introduced a method that uses expected sensor information to branch between behaviors encoded as ASMs depending on the current sensor state. This method was using a Nearest Neighbor approach with squared euclidean distance metric to find the best matching ASM. Thus, the approach requires the sensor signals to be normalized as otherwise sensors with a higher range of values become more important which is undesirable. Determining these normalization values for physical sensor signals, e.g. joint angles or strain gauge readings, might be straightforward, however, deciding on normalization constants for an increasing number of features is non-trivial. Most importantly, these normalization values constitute a global distance metric that penalizes deviations between predicted and measured sensor signals independent of time. However, as the data illustrated in Fig. 5.10 suggests, the importance of sensor information strongly varies with time. Hence, a global distance metric is not suitable for this problem. In this section, we will present an approach that alleviates the need for normalized feature values, instead automatically determines feature scales which capture the relevance of each feature at each time step. The relevance of features can be assessed by examining the variance over all past trials: A small variance in some features indicate important events, a large variance in other features indicate that they are less relevant. To capture this observation, we represent every feature by a one dimensional Gaussian model, which can be fitted in a straightforward manner, using a maximum likelihood estimator for the data mean and standard deviation. This independence assumption facilitates efficient computation of the negative log likelihood of the current state, see Eq. (5.11). Note, our approach neglects the correlation among features as well as the correlations over time, however, our experiments show that this rather simple approximation yields good results. Our approach follows the procedure depicted in Fig. 5.13. However, we propose an improved distance computation that is more suitable for the task at hand. The considered problem is to find the data point $\boldsymbol{d}$ in the active set of ASMs for which the currently measured feature vector $\boldsymbol{x}$, consisting of n features, is most probable. This results in a Nearest Neighbor problem with a different distance function $p(\boldsymbol{x}, \boldsymbol{d})$ for every data point $\boldsymbol{d}$, namely the negative log likelihood of the data

$$\begin{aligned} \arg\min_{\boldsymbol{d}} -\log\, p(\boldsymbol{x}|\boldsymbol{d}) &= \arg\min_{\boldsymbol{d}} -\log \prod_{i=1}^{n} p(\boldsymbol{x}_i|\boldsymbol{d}) \\ &= \arg\min_{\boldsymbol{d}} -\sum_{i=1}^{n} \log\, \mathcal{N}(\boldsymbol{x}_i; \mu_{\boldsymbol{d}_i}, \sigma_{\boldsymbol{d}_i}) \ . \end{aligned} \tag{5.11}$$

Given that we can determine the most likely data point $\boldsymbol{d}$ and thus the time step in a particular ASM, the automatic determination of possible successor ASMs is already

resolved. Most importantly the failure detection problem is reduced to a one dimensional classification problem. The threshold value for the negative log likelihood of the most likely data point at which task failure is more likely than task success can be learned using a maximum-margin classifier. The required negative data samples can be acquired incrementally during the training phase as described in the next section.

5.5.3 Manipulation Graph

As described in Sec. 1.5.2, we think of movement primitives in the context of movement generation to be the analogue to words in language: Complex movements are composed from sequencing several movement primitives similar to how sentences are formulated from sequencing words. Note that not every possible sequence of words results in a correct sentence, only those that obey the rules imposed by a grammar. We think the same rules apply to manipulation, i.e. there is a grammar for manipulation as well. For example, a grasping movement will not succeed before reaching the object with the endeffector, similarly, placing an object requires to be preceded by a lifting movement.

In this section, we will impose such a grammar using a state machine, hereafter referred to as manipulation graph. The nodes of this directed graph are the ASMs, the individual manipulation skills, and the edges between the nodes correspond to valid transitions. The manipulation graph is a directed graph, or finite-state machine, that constraints the possible sequence of manipulation behaviors. The manipulation graph is constructed incrementally. New manipulation skills are learned from demonstration and added to the graph by specifying its predecessors and successors. Arguably, the proposed setup provides an effective method of teaching a manipulation system new behaviors. The incremental nature of the procedure enables the user to teach recovery behaviors as needed, i.e. after the user observed a recoverable failure case. This procedure ensures that (only) failure cases are considered that actually occur on the real system. The manipulation graph that we used in our robot experiments is sketched in Fig. 5.20.

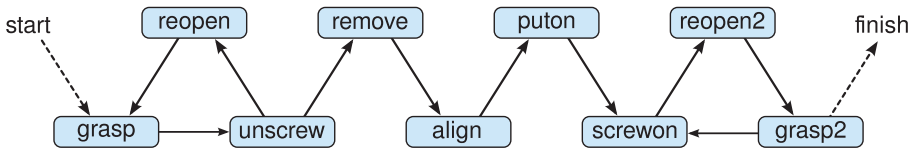


Figure 5.20: Conceptual sketch of the manipulation graph used for unscrewing a cap off a bottle and screwing it back on.

The imposed constraints reduces the search complexity compared to Sec. 5.4.4 as only the valid successor nodes as well as the current execution are used for feature comparison. Furthermore, there are only three different outcomes at any instance in time. First, the current movement is valid and has not converged yet, so we can continue the current execution. Second, a failure case has been predicted for the current movement, meaning the current sensor readings compared to the closest match of the current execution are

classified as a failure case. In this setup, the most likely successor is selected if and only if it is valid. Otherwise the system will stop, because it is in a failure state. Third, the current movement is valid and has converged, thus there is no benefit in performing this manipulation skill any longer. In this case the most likely successor is selected. Similar to the failure case, the most likely successor is only executed if it is valid and thus no failure case has been detected, otherwise the system will halt.

In theory one could always switch to the most likely successor node during any movement. However, if one would assume there exist several recovery behaviors for the same manipulation skill, all of which with the same target configuration, basically a physical exploration of the convergence space to make the higher level manipulation skill more robust. Then it might happen that the two manipulation skills might oscillate, meaning that at consecutive time steps one or the other is always more likely. Since both converge to the same configuration that introduces no theoretical issue, but on a physical system this will introduce problems because the underlying controls have to be adapted. Thus, in our setup we only switch to successor nodes if a failure case has been detected or the manipulation skill converged to the target configuration. Note, our system requires the user to provide this manipulation graph. Automatically inferring the task structure is beyond the scope of this thesis and an research field by itself, see Sec. 5.5.5.

5.5.4 Experimental Results

Our experiments consisted of the ARM-S robot unscrewing a cap off a bottle and putting it back on. The manipulation skills where learned through kinesthetic teaching following the procedure in Sec. 5.3.6. The demonstrated endeffector trajectories have been encoded into ASMs in the frame of the start of the movement. This encoding scheme generates movements that are similar to the demonstration locally. Thus, enforces a strong correlation between subsequent movements relative to the goal. Similarly, the endeffector force/torque trajectories have been encoded in the frame of the goal of the movement. Thus, the force feedback error between the desired $\mathbf{F}_{arm\_des}$ and the measured forces and torques $\mathbf{F}_{arm}$ (see Eq. 5.8) are computed in this task frame and rotated into the base frame to compute the torques. This encoding scheme ensures that the desired forces and torques are independent of the (global) endeffector pose. Instead, the forces and torques are relative to the goal and therefore relative to the object. This goal pose is updated at 30 Hz with respect to the current object pose estimate (see Sec. 5.5.1).

The manipulation graph depicted in Fig. 5.20 has been provided to guide the sequence of the manipulation task. At each time step, all 63 features listed in Tab. 5.19 have been processed and used to evaluate the task progress. Different from our previous experiments described in Sec. 5.4.5, the likelihood of the current state has been assessed at all time steps and compared with all potential successor ASMs. This allowed our system to instantly detect the moment at which the cap was unscrewed allowing it to instantly switch to the appropriate ASM as shown in Fig. 5.21.

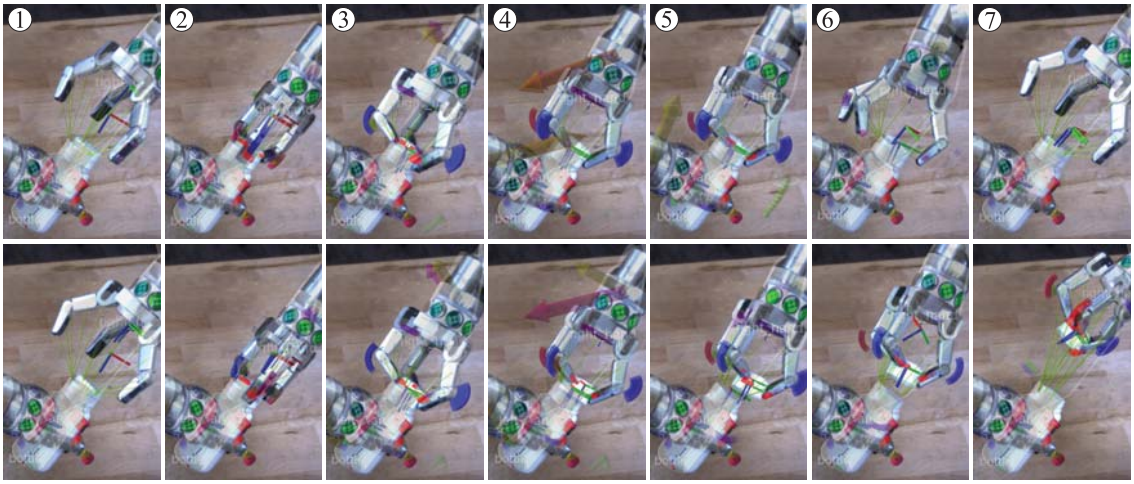


Figure 5.21: Snapshots of the bottle opening task as seen from the left Bumblebee camera with feature information overlaid. Both hands and the bottle are visually tracked and the forces and torques experienced at the wrists are being controlled, so are the finger torques (as sketched in Fig. 5.11). The desired control input has been acquired from past experiences of the same task. The top row shows the robot grasping the cap, unscrewing it, and reopening the hand since the cap did not come off the bottle. The bottom row shows the robot grasping the cap, unscrewing it, and finally removing it. The decision of when or whether at all the unscrew was successful, meaning the cap came off the bottle, was automatically determined by our approach. The difference difference between the predicted sensor information and the currently measured sensor signals indicate this event, see for example the force measurements (yellow arrow) in the 4th row deviate from the predicted forces (purple arrow). The corresponding video is available at `http://youtu.be/QViPYmD3aUQ`.

Finally, to show case the robustness of our approach we introduced perturbations by manually moving the hand that is holding the bottle. Our compliant control including force control and DMP adaptation as well as the visual tracking of both hands and the object ensure that the robot is able to perform the dexterous manipulation task despite significant perturbations, as shown in Fig. 5.22. A video showing the entire manipulation task, including screwing the cap back on is available at `https://vimeo.com/85966383`. This video shows that the computed likelihood of the current state can be used to teach failure conditions. Finally, the video also shows how recovery behaviors can be added to the manipulation graph and automatically triggered. For further details please refer to (Kappler, Pastor, Kalakrishnan, Wüthrich, & Schaal, submitted).

5.5.5 Conclusion

The experimental results show that our approach is suitable to incorporate a significant number of features. We have enriched our sensor repertoire with visual and auditory signals as well as additional hand crafted features. Furthermore, we have presented a simple yet efficient method to automatically determine the relevance of features. Our system

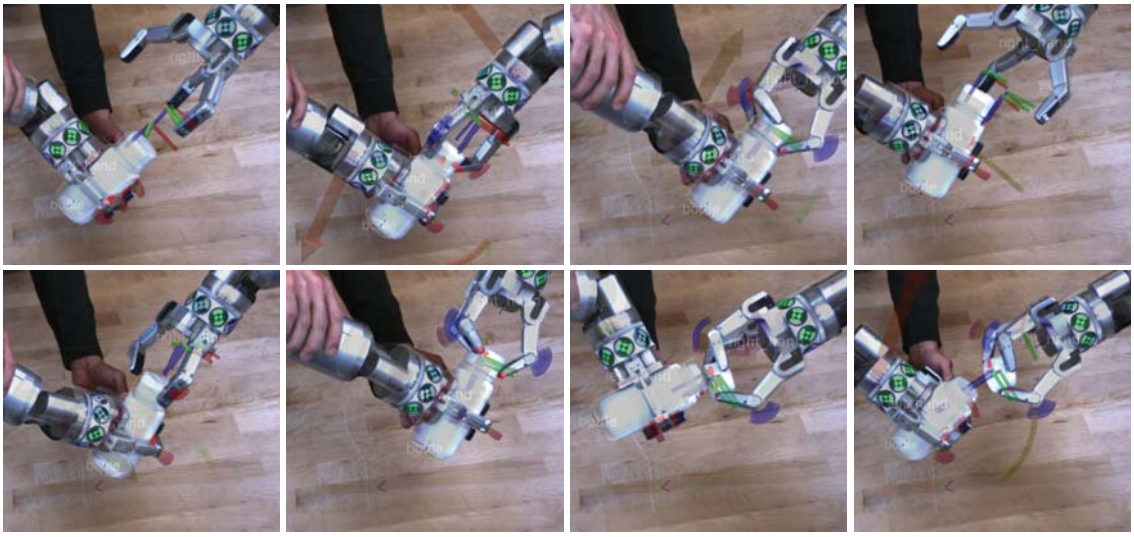


Figure 5.22: Snapshots of the robot performing the bottle opening task despite perturbation introduced by moving the left hand of the robot. The compliance of our control framework including DMP adaptation as well as closed loop visual tracking of the bottle and both hands enable the robot to give in without loosing necessary accuracy to perform the task. The corresponding video is available at `http://youtu.be/IgvKH4v2uvg`.

was able to achieve a very dexterous manipulation task, considered the limitations of the robot. We want to stress that the presented results have been obtained without having to write task specific code. Instead, the manipulation skill were learned from demonstration. The required positional and force reference signals for our feedback controllers have been obtained from a accumulated data of successful task executions. The task specific events, e.g. whether the cap has detached from the bottle while unscrewing, has entirely been learned from data.

Interesting extensions to our approach could involve reexamining the processing of the haptic feedback. We have mostly ignored the high frequency components of haptic signals and therefore missed out on information information for manipulation (Romano, Hsiao, Niemeyer, Chitta, & Kuchenbecker, 2011). An interesting avenue of research is to investigate how to learn task relevant features from observed data. This would not only alleviate the need to hand craft the features, but also likely to improve the quality of these features. Another particularly challenging research question is how to automatically derive manipulation graphs. A potential approach could follow ideas from natural language processing where transition probabilities for words are being inferred by counting large amounts of written text.

Finally, as nicely stated in (Jain & Kemp, 2013), "More Tasks, More Data, More Robots", data-driven approaches thrive with data. It will be interesting to see the performance of such an approach after a serious amount of data has been collected.

Chapter 6

Learning Task Error Models for Manipulation

Precise kinematic forward models are important for robots to successfully perform dexterous grasping and manipulation tasks, especially when visual servoing is rendered infeasible due to occlusions. A lot of research has been conducted to estimate geometric and non-geometric parameters of kinematic chains to minimize reconstruction errors (see Sec. 2 for a brief overview). However, kinematic chains can include non-linearities, e.g. due to cable stretch and motor-side encoders, that result in significantly different errors for different parts of the state-space. Previous work either does not consider such non-linearities or proposes to estimate non-geometric parameters of carefully engineered models that are robot specific. We propose a data-driven approach that learns task error models that account for such unmodeled non-linearities. We argue that in the context of grasping and manipulation, it is sufficient to achieve high accuracy in the task relevant state-space. We identify this relevant state-space using previously executed joint configurations and learn error corrections for those. Therefore, our system is developed to generate subsequent executions that are similar to previous ones, i.e. employs the concept of stereotypical movements.

In this section, we present a method to obtain accurate forward models even in the presence of non-linearities by learning the residual errors using Gaussian Process Regression (GPR). We argue that learning local task error corrections is sufficient if, and only if, the supporting planning and control architecture enforces subsequent executions to remain close to previous executions. It is very important to note, that this is not a limitation of the proposed approach, since the task error model considers the state-space that is relevant for the task. We do not see any advantage of trying to achieve an accurate forward model for the entire state-space. We want to stress that our approach can deal with non-linearities in kinematic chains, for example due to a counter-balancing spring or cable stretch. Calibrating the geometric parameters of the system as proposed in (Bennett et al., 1991) cannot account such state dependent errors. Thus, our approach avoids this tedious calibration procedure altogether and learns (small) model errors together with the non-linear errors.

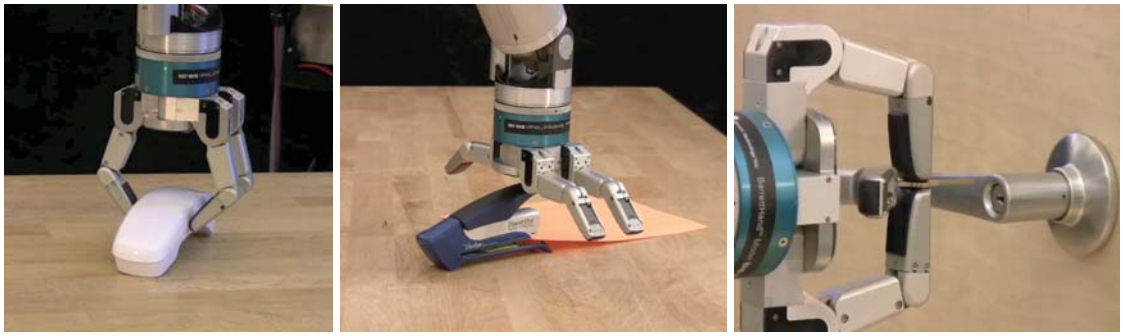


Figure 6.1: A subset of tasks considered during the first phase of the Autonomous Robotic Manipulation (ARM) challenge that required high positional accuracy: Grasping a handset (left), stapling (middle), and unlocking a door with a key (right).

The experiments show that our method successfully captures the non-linearities in the head kinematic chain (due to a counter-balancing spring) and the arm kinematic chains (due to cable stretch) of the considered experimental platform, see Fig. 6.1. We want to note that the presented work (Pastor, Kalakrishnan, Binney, et al., 2013) has been applied in competitive, independent DARPA ARM (Autonomous Robotic Manipulation) testing and provided our system with enough accuracy to successfully perform 67 out of 72 task executions, see (Righetti et al., 2014) for more details on the final results.

The remainder of this section is structured as follows: A problem description is provided in Sec. 6.1; The use of Gaussian Process Regression for model error learning is introduced in Sec. 6.1.1 and its applicability to manipulation is motivated in Sec. 6.1.2; A system overview is provided in Sec. 6.1.3; Experimental results on learning task error models for the head kinematic chain is presented in 6.2 and for the arm kinematic chains in Sec. 6.3; The presented approach is concluded in Sec. 6.4.

6.1 Accounting for Non-linearities in Kinematic Chains

The goal of our approach is to learn error correcting models on top of existing forward models of our experimental platform, the ARM-S robot*, see Fig. 6.1. It is important to note that the considered kinematic chains include non-linearities that cannot be accounted for by aforementioned methods. The head is composed of two pan/tilt units mounted on top of each other. A spring passively counterbalances the head to support the under dimensioned motor of the top pan/tilt unit. This counterbalancing spring creates non-linear effects in the head kinematic chain. The non-linearities in the kinematic chains of the arms are due to cable stretch and the fact that the encoders are on the motor-side, see Fig. 6.2. Motor-side encoders obtain sensor readings directly at the motor as opposed to the joints, as such the cable stretch between the motor and the joints cannot be measured. The resulting error in both head and arm kinematic chains depend on the current joint

*Details about the considered experimental platform are provided in the Appendix A.3

angle configuration of the robot; Explicitly modelling these non-linearities seems difficult and tedious.

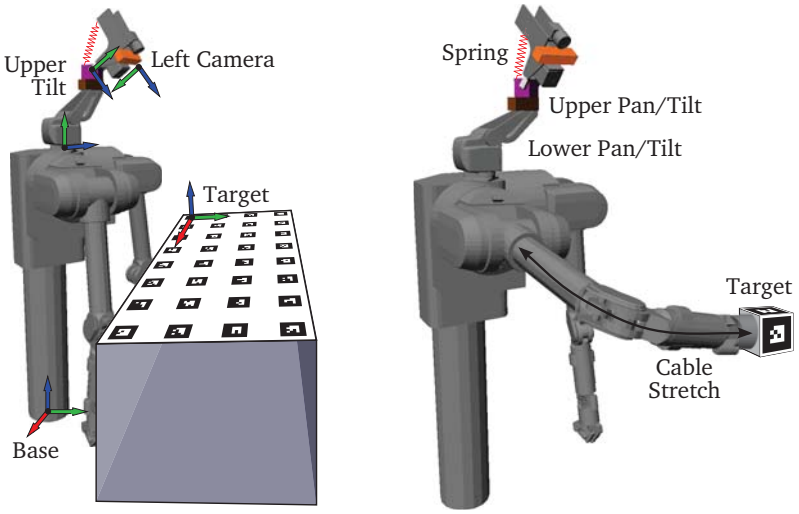


Figure 6.2: Sketch of the two considered experimental setups. The non-linearities in the forward model of the head kinematic chain is introduced by the spring (left). The non-linearities in the forward model of the arm is introduced due to cable stretch and motor-side encoders (right).

The objective of calibrating the head kinematic chain, i.e. the two pan/tilt units, first is to enable our system to accurately merge 3D point clouds of perceived objects from different view angles. The objective of calibrating the arm kinematic chains is to get an accurate object pose estimate in the hand frame necessary for dexterous manipulation. For both these calibration tasks we propose to use the same two staged approach: First, optimize for the best transformation that minimizes the systematic pose error between the forward model and the visually perceived target pose. Second, learn a local non-linear correction term $\Delta\boldsymbol{x}$ that absorbs the remaining task error as a function of the current joint angle configuration $\boldsymbol{\theta}$.

6.1.1 Gaussian Process Regression for Task Error Learning

We use Gaussian Process Regression (Rasmussen & Williams, 2006) (GPR) to learn the remaining task error, i.e. the mapping $\boldsymbol{\theta} \rightarrow \Delta\boldsymbol{x}$. Gaussian Process Regression is particularly suitable because it can accurately fit the data and only very little tuning is required. The considered size of the data set is also well within the computational complexity limits of the standard GPR model. However, more advanced methods that use GPR are readily available for larger data sets (Nguyen-Tuong, Seeger, & Peters, 2008).

The full 6 D pose correction $\Delta\boldsymbol{x}$ is learned using six separate GPR models, one for each task variable, i.e. translation in x, y, z and corresponding rotations $\phi_{\text{roll}}, \phi_{\text{pitch}}, \phi_{\text{yaw}}$. The input to each of these models are the relevant d joint angles $\boldsymbol{\theta}_i \in \mathbb{R}^d$. Let y be the scalar

output variable that represents the correction for one of the six task variables in $\Delta \boldsymbol{x}$. Given a training set of n pairs $\{\boldsymbol{\theta}_i, y_i\}_{i=1}^n$, the goal is to learn a function $f(\boldsymbol{\theta}^\star) = y^\star + \epsilon$ that predicts the correction term $y^\star$ for new joint configurations $\boldsymbol{\theta}^\star$, where ϵ is assumed to be additive noise with zero mean and noise variance of σ^2_{noise}. The observed targets $\boldsymbol{y}$ can be modeled by the Gaussian distribution $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\boldsymbol{\theta}, \boldsymbol{\theta}) + \sigma^2_{\text{noise}}\mathbf{I})$, where $\boldsymbol{\theta}$ is the set containing all joint angles $\boldsymbol{\theta}_i$, and $\mathbf{K}(\boldsymbol{\theta}, \boldsymbol{\theta})$ denotes the $n \times n$ matrix of the covariances evaluated for all pairs of training points $\boldsymbol{\theta}_i$ with all other training points $\boldsymbol{\theta}_j$ for $i, j = 1..n$. A standard choice for the covariance function is the squared exponential covariance function $k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \sigma^2_{\text{signal}} \exp(-\frac{1}{2}(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j)^\mathsf{T}\mathbf{W}(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j))$, where σ^2_{signal} denotes the signal variance and $\mathbf{W}$ the width of the Gaussian kernel. Predictions $f(\boldsymbol{\theta}^\star)$ are obtained by conditioning the joint distribution of the observed targets y_i and the test target $y^\star$ on the observed inputs $\boldsymbol{\theta}_i$, the observed targets y_i, and the query input $\boldsymbol{\theta}^\star$, yielding

$$f(\boldsymbol{\theta}^\star) = k(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) \left(\mathbf{K}(\boldsymbol{\theta}, \boldsymbol{\theta}) + \sigma^2_{\text{noise}}\mathbf{I}\right)^{-1} \mathbf{y} \ . \tag{6.1}$$

Note that predicting $f(\boldsymbol{\theta}^\star)$ can be computed within real-time control loops given that $(\mathbf{K}(\boldsymbol{\theta}, \boldsymbol{\theta}) + \sigma^2_{\text{noise}}\mathbf{I})^{-1}$ can be computed beforehand. Evaluating the kernel function at the query point $\boldsymbol{\theta}^\star$ and all other training points $\boldsymbol{\theta}$ results in negligible overhead. The hyperparameters σ^2_{signal}, σ^2_{noise}, and $\mathbf{W}$ of this Gaussian process remain the only open parameters which can be automatically estimated by maximizing the log marginal likelihood using standard optimization methods. Please see (Rasmussen & Williams, 2006) for more information on Gaussian Progress Regression.

6.1.2 Task Error Learning for Manipulation

Our approach is tailored towards achieving accuracy in the region of the state space that matters. We argue that this is not a limitation of our approach because it perfectly fits our philosophy; We seek to employ stereotypical movements (Pastor et al., 2012) that create stereotypical sensory events which are essential to detect failures (Pastor, Kalakrishnan, et al., 2011) or allow for reactive behaviors (Pastor, Righetti, et al., 2011). To determine a set of *good* joint angle configurations that resemble the workspace of interest, we early on were logging all endeffector poses that required accurate kinematic calibration, e.g. pregrasp poses, whenever the robot was performing a grasping or manipulation task*, see Fig. 6.1. Over the course of about 6 months we collected a total of 1786 such endeffector poses. These collected endeffector poses were converted into corresponding arm postures using an optimization based inverse kinematics (IK) method (Kalakrishnan et al., 2013). This method optimized over the nullspace postures and, in some cases, over a range of possible endeffector poses to find optimal joint angle configurations. Logging the endeffector poses as opposed to joint angles directly allowed us to reuse all stored IK requests, only requiring to recompute the joint angle configurations whenever we decided to adapt the

*A compilation of the considered grasping and manipulation tasks can be seen at `http://youtu.be/VgKoX3RuvB0`.

cost function. Thus, the optimization based IK method tries to enforce that subsequent task executions will result in similar arm postures. Similarly, we employ an optimization based motion planning algorithm (Kalakrishnan, Chitta, et al., 2011)(Kalakrishnan et al., 2013) to favor trajectories that are similar in shape and speed*. To obtain a realistic training set we used k-means clustering algorithm in joint space, where k was set to 150. Our results in Sec. 6.3 show that the selected set of joint angle configurations were suitable to cover the region of interest.

6.1.3 System Overview

In the following sections, we will describe how the presented approach is used to account for non-linearities in the head kinematic chain in Sec. 6.2 and in the arm kinematic chains in Sec. 6.3 of the ARM-S robot. For both experiments we follow the same 3 staged approach. First, we generate a set of joint configurations that are relevant for the considered tasks and record these joint angles as well as the perceived 6 D marker poses. Second, we use this data to optimize for a fixed transform that minimizes the pose errors between the forward kinematics model and the perceived 6 D marker poses to account for systematic errors. Finally, we learn the remaining pose errors using GPR as described in 6.1.1. Next, we will describe the stereo marker pose estimation method that has been used for both experiments.

6.1.4 Stereo Marker Pose Estimation

The task error models are both learned with respect to the stereo camera, a Point Grey Bumblebee2 with 1024x768 pixel resolution. Prior to all experiments we calibrated for the intrinsic parameters. The target for both of the calibration procedures consists of target markers that have been designed to be easily detected by the ARToolKit†. The ARToolKit uses images from a single camera and infers the depth of using the a-priori known size of the detected markers. To increase the accuracy, especially in depth, we applied the ARToolKit to detect the corners of each marker in both images (obtained from the left and the right camera) and used the stereo calibration of the camera to compute the 3 D corner position from the two 2 D image coordinates. Finally, we average the 3 D corner positions to obtain the center position of the marker. The orientation of that marker is computed by averaging the two obtained marker orientations. The full 6 D

*Internal testing using a VICON motion capturing system confirmed that the Barrett arm has a high repeatability, i.e. reaching for a particular joint configuration over and over again results in a small endeffector pose variance. However, varying the approach trajectory as well as the velocity increases the endeffector pose variance slightly. Nevertheless, most of the influence of the cable stretch still depends on the final arm posture.

†ARToolKit (http://www.hitl.washington.edu/artoolkit) was originally developed by Dr. Hirokazu Kato, and its ongoing development is being supported by the Human Interface Technology Laboratory (HIT Lab) at the University of Washington, HIT Lab NZ at the University of Canterbury, New Zealand, and ARToolworks, Inc, Seattle.

Figure 6.3: Example neck joint angle configuration used during training (left) and corresponding image of the left camera with detected fiducials overlaid (right).

marker pose is in the frame of the left camera. In addition to the pose of each marker m_i, the ARToolKit also provides information about which marker i has been detected.

The target for the head calibration was composed of 4 rows with 8 distinct markers each arranged in a plane at a known distance, see Fig. 6.2 (left) and Fig. 6.3. This target has been designed to cover the area of interest. The target for the hand-eye calibration procedure was composed of 4 distinct markers arranged as a cube, see Fig. 6.2 (right) and Fig. 6.5. These markers have been presented to the ARToolKit in a single object such that we could use the internal optimization routines and obtain a single transform. The final transform is obtained by averaging the transforms obtained from the left and right camera image.

6.2 Experiment I : Accounting for Non-linearities in the Head Kinematic Chain

To cover the area of interest we selected 60 neck joint angle configurations for the two pan/tilt units, see Fig. 6.3. The lower pan/tilt unit allows to obtain (slightly) different views of a point of interest and can be used to avoid occlusions where possible. The selected joint angles consisted of the lower pan unit to be at $-90\,^\circ$, $-45\,^\circ$, $0\,^\circ$, $45\,^\circ$, and $90\,^\circ$ and the lower tilt to be straight and all the way forward. For each of these $5 \times 2 = 10$ configurations the upper pan/tilt joint angles were set such that the cameras point at 6 particular locations that cover the table. The actual experiment involved the robot moving to each of these 60 configurations and storing the 4 neck joint angles together with the visually perceived marker poses (using the method described in 6.1.4) and their IDs as shown in Fig. 6.3. The IDs were used to recover the exact marker pose in the target frame shown in Fig. 6.2.

6.2.1 Optimizing for Fixed Transformations to Account for Systematic Errors

First, we optimized for a fixed transformation from the upper tilt (UT) link to the left camera (LC) frame $\mathbf{H}_{\mathrm{UT}}^{\mathrm{LC}}$ to account for systematic errors. The transform from the base (B) frame to the upper tilt (UT) link is $\mathbf{H}_{\mathrm{B}}^{\mathrm{UT}}$ and assumed to be correct. The 3D positions of the markers in the target (T) frame are denoted by $\mathbf{p}_{\mathrm{T}}$ and known since we designed the target. The 3D positions of the markers in the left camera frame are denoted by $\mathbf{p}_{\mathrm{LC}}$ and obtained using the method described in 6.1.4. The 3D coordinates of the markers in base frame are denoted by $\mathbf{p}_{\mathrm{B}}$ and computed either by $\mathbf{H}_{\mathrm{T}}^{\mathrm{B}}\mathbf{p}_{\mathrm{T}}$ or by $\mathbf{H}_{\mathrm{UT}}^{\mathrm{B}}\mathbf{H}_{\mathrm{LC}}^{\mathrm{UT}}\mathbf{p}_{\mathrm{LC}}$. The former requires knowledge about $\mathbf{H}_{\mathrm{T}}^{\mathrm{B}}$ the latter requires knowledge about $\mathbf{H}_{\mathrm{LC}}^{\mathrm{UT}}$, see Fig. 6.2. Assuming a perfect model, i.e. correct transform from the base to the upper tilt link, and perfect perception, i.e. accurate sensing of the 3D marker positions in the left camera frame, then $\mathbf{p}_{\mathrm{B}} = \mathbf{H}_{\mathrm{UT}}^{\mathrm{B}}\mathbf{H}_{\mathrm{LC}}^{\mathrm{UT}}\mathbf{p}_{\mathrm{LC}} = \mathbf{H}_{\mathrm{T}}^{\mathrm{B}}\mathbf{p}_{\mathrm{T}}$ for all detected markers. Thus, an error can be computed by $\sum |\mathbf{H}_{\mathrm{UT}}^{\mathrm{B}}\mathbf{H}_{\mathrm{LC}}^{\mathrm{UT}}\mathbf{p}_{\mathrm{LC}} - \mathbf{H}_{\mathrm{T}}^{\mathrm{B}}\mathbf{p}_{\mathrm{T}}|$. Our approach iteratively solves for these two transformations by minimizing this pose error keeping one fixed and solving for the other using a non-linear optimizer. Finally, this optimized transform $\mathbf{H}_{\mathrm{LC}}^{\mathrm{UT}}$ can be used to compute the remaining pose error for each of the 60 training configurations. This remaining error is mostly due to configuration dependent non-linearities resulting from the spring counter-balancing the top pan/tilt unit.

6.2.2 Learning a Model to Account for Remaining Error

To learn the remaining pose error we use Gaussian Process Regression (GPR) as described in 6.1.1. We learn 6 GP models to predict the 6 correction terms, i.e. translation in x, y, z and corresponding rotations $\phi_{\mathrm{roll}}, \phi_{\mathrm{pitch}}, \phi_{\mathrm{yaw}}$. The input to each of the 6 GP models are all 4 joint angles of both pan/tilt units. Thus, the GP model provides us with a joint configuration depended 6 DOF pose correction which we apply to previously estimated fixed pose $\mathbf{H}_{\mathrm{LC}}^{\mathrm{UT}}$ to account for the remaining unmodeled non-linearities. The use of a zero mean function for the GP ensures that the predicting correction term will be zero for regions of the state-space that are far away from all training configurations. Thus, in the worst case the performance of our system will not degrade more than not using the GP correction, i.e. equivalent to using the forward model alone.

To evaluate the performance of our system we generated 60 random test configurations. These test joint angles are computed using inverse kinematics by pointing the cameras at uniformly sampled locations on the table. The joint angle configurations for the lower pan/tilt unit are restricted to be at one out of the 10 configurations used during training. Thus, we only consider test cases which we determined to be actually relevant for our grasping and manipulation tasks. Furthermore, we noticed that most of the non-linearities are due to the counterbalancing spring which mostly influences the upper pan/tilt, see Fig. 6.2. The primary objective when accounting for non-linearities in the head kinematic chain is the ability to perceive a particular object from many different view angles and

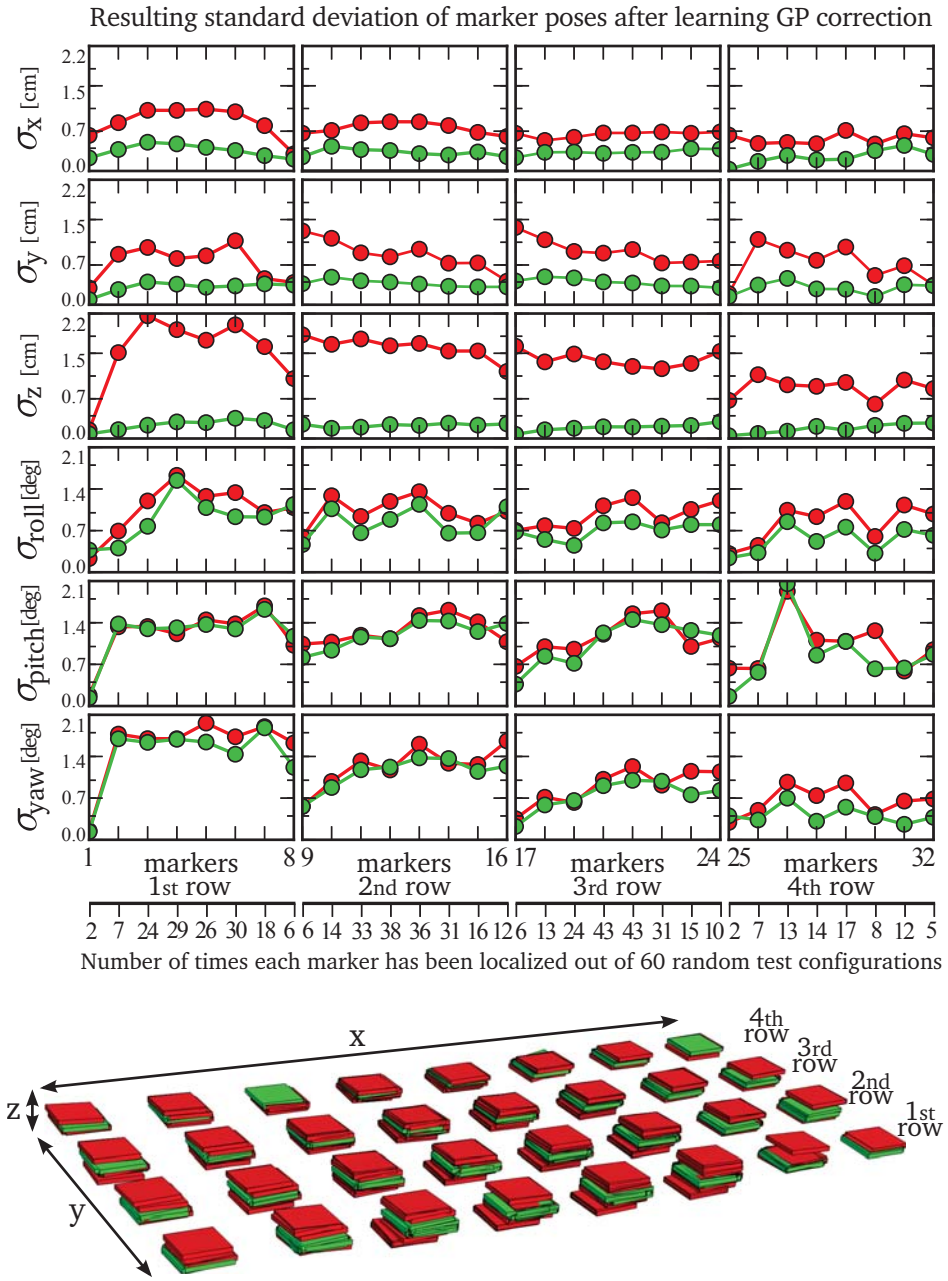


Figure 6.4: Resulting standard deviation of each visually detected marker pose obtained from 60 randomly generated test configurations (top) using the forward model only (red) and including GP correction (green). The marker detection count is provided below. Some markers have been detected fewer times because of non-uniform lighting conditions and simply because the markers at the borders of the target where less often in the field of view of both cameras. A 3D visualization of all detected markers superimposed obtained from 60 different joint angle configurations (bottom). The results show that the GP model significantly reduced the pose variance, especially in position.

obtain a small (ideally zero) object pose variation. Thus, the experiment consisted of accumulating all detected marker poses while moving through the 60 test configurations and computing the 1-standard deviation $\boldsymbol{\sigma}$ in position and orientation for each of the 32 markers. The obtained results, see Fig. 6.4, show that the GP correction successfully accounted for unmodeled non-linearities especially for position. The reduction in variance for the marker orientation is small due to the fact that small errors in the forward model will not cause huge errors in the perceived marker orientation. However, an error of 2° degrees can result in a positioning error of more than 4 cm. The statistics of the obtained results are shown in Tab. 6.1.

1-std-dev σ	x[cm]	y[cm]	z[cm]	roll[deg]	pitch[deg]	yaw[deg]
FK	0.79	0.89	1.53	1.12	1.31	1.3
FK + GP	0.35	0.37	0.24	0.87	1.25	1.17

Table 6.1: Summarized standard deviation over a total of 591 obtained marker poses from Fig. 6.4 for both cases, using forward kinematics only (FK) and using forward kinematics and GP correction (FK + GP). The results show the non-linear nature of the head kinematic chain (especially in z direction) and that the GP correction significantly lowers the error.

6.3 Experiment II : Accounting for Non-linearities in the Kinematic Chains of the Arms

Determining a set of arm postures that cover the relevant state-space for the considered redundant manipulators is non-trivial. Therefore, we propose to store previously executed pre-grasp and pre-manipulation poses while performing the task of interest and use a subset of these configurations to learn a task error model. As described in 6.1.2, we collected a total of 1786 endeffector poses over the course of about 6 months. We used k-means clustering in joint space, i.e. after applying the optimization based inverse kinematics method (Kalakrishnan et al., 2013), with $k = 150$ to obtain a representative set of task relevant arm postures. The 1786 recorded endeffector poses of the right arm have been mirrored and reused for the left arm*. The trajectories for these 150 arm postures for both arms have been computed using the optimization based motion planning algorithm (Kalakrishnan, Chitta, et al., 2011). To not alter the effect of cable stretch by replacing the Barrett hand with the marker cube (see Fig. 6.5) we constructed the cube to weigh exactly the same amount as the Barrett hand†. Due to varying lighting

*We received the left arm only after reaching the second phase of the DARPA ARM-S project, see Fig. 6.1 (left). A video of the one armed version involved during the first phase can be seen at `http://youtu.be/VgKoX3RuvB0`.

†We decided to build a target cube as opposed to directly attaching markers to the Barrett hand primarily for convenience and to reduce the risk of misplaced markers at the remote test site during the ARM-S project.

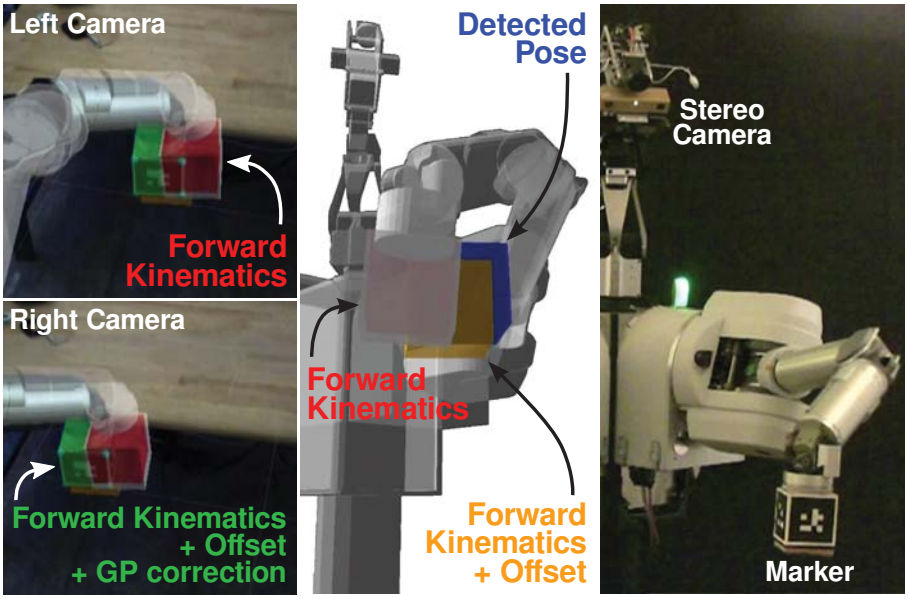


Figure 6.5: Screenshot of an example test configuration of the left arm that shows the significant pose error in the forward model (red). The offset (orange) corrects for the systematic error, however, only with the GP correction (green) the forward model matches the visually detected marker pose (blue) as shown in the camera images on the left.

conditions, our marker pose estimation method described in 6.1.4 did not detect all 150 arm postures. We also did not take special care of filtering out arm postures that violate the visibility constraint and as such, only about 120 data points were collected.

6.3.1 Optimizing for a Fixed Transformation to Account for Systematic Errors

To account for systematic errors in the mounting of the two Barrett arms with respect to the base frame, see Fig. 6.2, we optimized a full 6 D transform. We used the same non-linear optimization technique to minimize the pose error obtained from the difference of the forward model and the visually perceived marker poses during training, similar to 6.2.1. The obtained transform successfully accounted for the systematic error, as shown in Tab. 6.2. The remaining non-linear error is computed using this fixed offset and stored together with the corresponding 7 joint angles of each arm respectively.

6.3.2 Learning a Model to Account for Remaining Error

To learn the remaining pose error we again employed 6 separate GP models and trained them using the corresponding 7 joint angles. The output of each of these 6 GP models are the corrections that absorb the unmodeled non-linearities. We evaluated the system by randomly choosing 150 joint configurations from the pool of 1789 previously executed

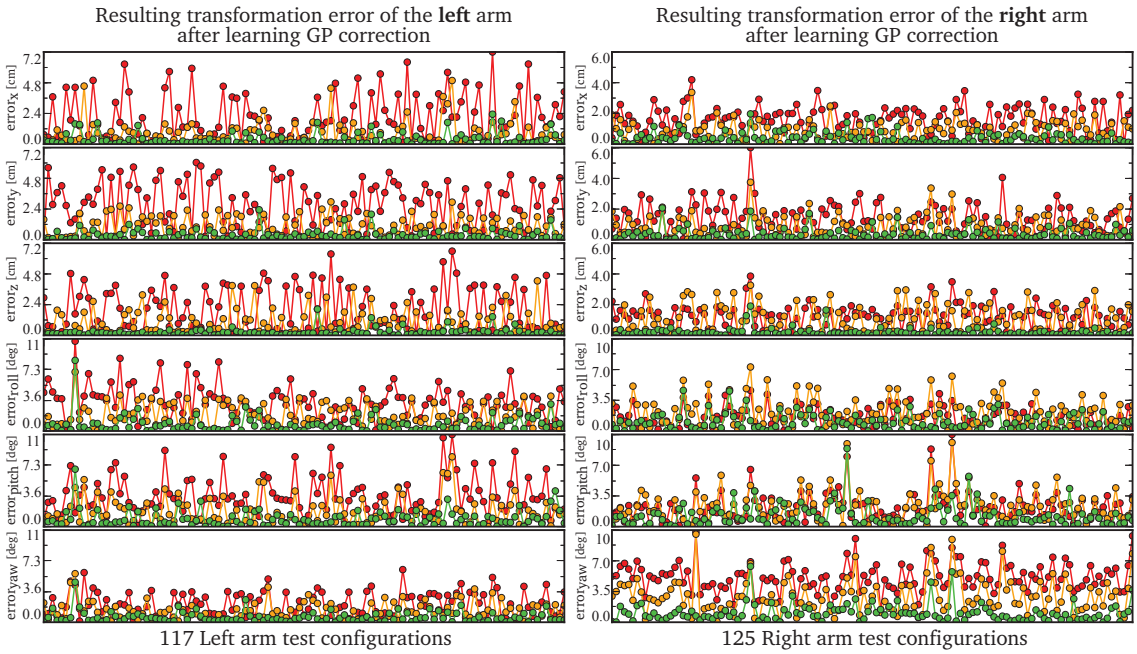


Figure 6.6: Linear transformation errors in x (+right), y (+forward), z (+up) and angular transformation error around the x, y, z axis (roll, pitch, yaw) for randomly chosen test configurations for both arms. The plotted pose error is computed between the visually detected marker pose and the pose obtained using the forward model (red), forward model + calibrated offset (orange), and forward kinematics + calibrated offset + GP correction (green). See Fig. 6.5 for an example configuration. The results are summarized in Tab. 6.2.

and therefore task relevant arm postures. An example arm posture used during testing can be seen in Fig. 6.5. The obtained result for the entire experiment for both arms can be seen in Fig. 6.6. The accumulated statistics are shown in Tab. 6.2. The results show that the GP error correction significantly reduced the mean pose error as well as its variance. The obtained accuracy enabled the robot to accomplish precise grasping and manipulation tasks, see `http://youtu.be/VgKoX3RuvB0`.

The source code used for the experiments can be downloaded from our github repositories at `https://github.com/usc-clmc/usc-arm-calibration` and `https://github.com/usc-clmc/usc-clmc-ros-pkg`. A high resolution (HD) version of the video showing the two presented experiments is available at `http://youtu.be/QKKViNGhWWQ`.

6.4 Conclusion

We have proposed a kinematics calibration procedure that uses GP correction to account for non-linearities in kinematic chains. Our procedure does not need accurate kinematic calibration which cannot account for non-linearities in the first place. We want to stress

Error	x [cm]	y [cm]	z [cm]	roll [deg]	pitch [deg]	yaw [deg]
Right Arm:						
FK	1.6±0.8	1.4±1.0	1.2±0.8	1.5±1.1	2.1±1.7	5.1±1.7
FK+O	0.9±0.6	0.9±0.7	1.1±0.9	2.1±1.6	2.3±1.7	2.9±2.0
FK+O+GP	0.4±0.4	0.4±0.4	0.2±0.3	1.0±0.9	1.2±1.3	1.0±1.0
Left Arm:						
FK	2.1±1.8	2.7±1.7	2.0±1.7	3.3±2.0	3.6±2.3	2.1±1.5
FK+O	0.8±0.9	1.0±0.8	1.0±1.0	1.9±1.3	1.8±1.6	1.4±1.1
FK+O+GP	0.4±0.5	0.4±0.4	0.2±0.3	0.8±1.0	0.9±1.0	0.6±0.6

Table 6.2: Mean and 1-standard deviation of the results in Fig. 6.6. The pose errors are computed from the visually detected marker pose and the pose computed using the forward kinematics (FK), using forward kinematics including offset (FK+O), and forward kinematics including offset and GP correction (FK+O+GP). The results show the non-linear nature of the kinematic chain of both arms and that the GP correction significantly lowers the remaining pose error.

the importance of stereotypical behaviors within our framework to ensure that subsequent executions remain close to previous experiences. Nevertheless, we want to emphasize that the presented data-driven approach can easily be used to learn new tasks and new regions of the state space. The feasibility of the presented approach has been validated in independent DARPA ARM-S testing. Both procedures presented in this section have significantly contributed to the success in the final test. Future work will include considering the computed covariance of obtained predictions as part of the cost function for finding good inverse kinematics configurations as well as motion plans, similar to (Kalakrishnan et al., 2013). We are also working towards using the obtained improved forward model to better initialize an arm/hand tracking algorithm to implement visual servoing.

Chapter 7

Conclusion and Discussion

This thesis presented a data-driven approach to autonomous robotic manipulation. To justify our design criteria, in Chapter 1, we first discussed the challenges of autonomous manipulation and identified a set of requirements that we think are crucial towards addressing these challenges. In particular, we emphasized two crucial prerequisite:

1. **Adding prior knowledge**: The sheer complexity of many manipulation tasks will never allow for brute force methods to generate useful solutions in general. Therefore, we emphasized the approach's ability to easily adopt smart heuristics/biases and use those to guide the search.

2. **Predicting sensor feedback**: The uncertainty of the real world significantly limits the range of tasks that can be accomplished using open-loop executions and usually requires specialized hardware. To achieve robust performances despite noise in the sensory-motor system of the robot as well as in the environment itself, we stressed the importance of closing feedback control loops. The challenge is to provide these feedback controllers with appropriate reference signals, i.e. to predict the sensor feedback for the manipulation task at hand.

We then analyzed the current paradigm, a wide-spread recurring design pattern that has been realized on many robotic platforms, and identified the limits of current model-based approaches. In particular, we argued that model-based approaches cannot fulfill the two previously mentioned requirements: (1) The computational complexity of the planning problem only becomes tractable if smart heuristics/biases have been specified. These task-specific constraints require expert domain knowledge for even very simple interactions (e.g. pushing an object on a flat surface). It is therefore unclear how such a system could acquire new skills autonomously. (2) The necessary accuracy of the world model and the necessary tracking performance during contact interactions with the world that are required to correctly predict the future seems infeasible given the uncertainty of the real world. Thus, it is unclear how model-based approaches will be able to provide feedback

controllers with appropriate reference signals to facilitate loop closure. In contrast, we provided insights in how to cast robotic manipulation into a memory-based approach and argued how such a data-driven method can fulfill the two previously mentioned requirements. Finally, we stated the thesis problem, listed the thesis contributions, provided a conceptual overview of the approach, and, in Chapter 2, referenced related work. In Chapter 3, we presented a modular movement representation that can encode arbitrary movements, can be learned from observed data, can adapt goal positions and movement durations, can be modulated online to realize reactive behaviors, and can be sequenced to achieve complete tasks. In Chapter 4, we conducted several experiments to show that the presented movement representation can quickly and easily be used to bootstrap new manipulation skills using imitation and reinforcement learning - the first of the identified prerequisite of the systems ability to add prior knowledge. In Chapter 5, we motivated the advantages of stereotypical movements and how they facilitate memory that in turn can be used to predict sensor information to be used in feedback control loops - the second prerequisite. In particular, we introduced the concept of Associative Skill Memories to facilitate task outcome prediction, feedback control, as well as online movement sequencing and presented several experimental evaluations. Furthermore, in Chapter 6, we extended our framework to learn task error models.

In conclusion, the work presented in this thesis is the result of carefully analyzing the challenges in autonomous robotic manipulation and alongside identifying regularities which model-based approaches have not been able to exploit. Subsequently, we developed a data-driven approach that was entirely focusing on overcoming these limitations. Finally, we made specific choices to realize and evaluate our approach in various manipulation tasks and on several robotic platforms. Nevertheless, we do not claim to have fully understood the problem of autonomous robotic manipulation, neither to have truly solved the challenges that we have identified. Instead, our aim was to uncover previously ignored structure in manipulation tasks and to present a data-driven approach that can exploit this structure. As such, we hope that this work has provided enough insights and real robot evaluations to open up fruitful research avenues.

7.1 Limitations and Extensions

The power of the presented data-driven approach is that it will thrive with more data: New manipulation skills can be learned from observed data and refined as more data become available. Furthermore, learned manipulation skills can be enriched with experienced sensor data to facilitate prediction of future task executions. Finally, task error models can be learned from data to improve positioning accuracy. However, on the flip side, the limitation of the presented data-driven approach is also its dependency on data being available. For example, performing a completely new task requires a user demonstration in order to acquire the necessary data. Similarly, improving sensor prediction accuracy requires a few executions to acquire necessary statistics. Finally, minimizing

task space positioning errors is only possible if our system has been provided with appropriate data. We do want to stress that our movement representation has been designed to facilitate generalization. However, the obtained generalization may or may not be desirable if the task variables, i.e. start and goal of the movement, have been altered by a large margin. Thus, our data-driven approach is inherently local and as such can only interpolate between already observed data. In contrast, the power of model-based approaches is that they are able to extrapolate, i.e. to generate plans for previously unexplored regions of the workspace. These plans are usually obtained through an optimization process that minimizes a task specific cost function which in turn requires an internal model of the world to be evaluated. As long as the internal model accurately represents the world, the obtained plan will yield good performance on the real robot, e.g. as it is the case in motion planning (Kalakrishnan, Chitta, et al., 2011). The downside, however, is that the quality of the solution degenerates with inaccuracies in the model. Unfortunately, as discussed in Sec. 1.2.2, modelling errors are inevitable especially when dealing with complex contact interactions as it is the case in manipulation.

In summary, our data-driven approach is suitable to predict sensor feedback even for complex manipulation tasks, however it can only do so for already explored regions of the task. In contrast, model-based approaches are able to extrapolate and generate useful solutions across the entire workspace, however, it can only do so if its model of the task is accurate. Thus, a promising extension to our data-driven approach is to combine it with model-based approaches such that they benefit from each other. The extrapolation capabilities of model-based approaches can be used to gather new data outside already explored regions of the manipulation task. This newly acquired data, which is arguably the best approximation to the real world that the system can obtain, can in turn be used to learn and update a (local) model. How to abstract the implicit representation of the task into a state-space model, however, remains an open research question. An interesting first step towards addressing this question has been proposed in (Jain & Kemp, 2013) using simple object-centric state-space models.

Finally, in the process of realizing our approach we made specific choices and assumptions that always leave room for improvement. Please refer to the conclusion sections in previous chapters to read about more specific potential future avenues of research.

Appendix A

The Experimental Platforms

A.1 Sarcos master-slave system

Figure A.1: The Sarcos master arm operated by a human subject (left) and the Sarcos slave arm reproducing the same movement (right). Photo by Luke Fisher Photography.

The experimental platform consists of two Sarcos* robots: a seven DOF exoskeleton robot arm with a three DOF hand controller used to record human movements and a seven DOF

*Sarcos Research Corporation (SRC), 360 Wakara Way, Salt Lake City, Utah 84108, USA and Center for Engineering Design (CED), University of Utah, Salt Lake City, Utah 84104, USA

dexterous robot arm with a three DOF end effector used to perform learned movements (see Fig. A.2).

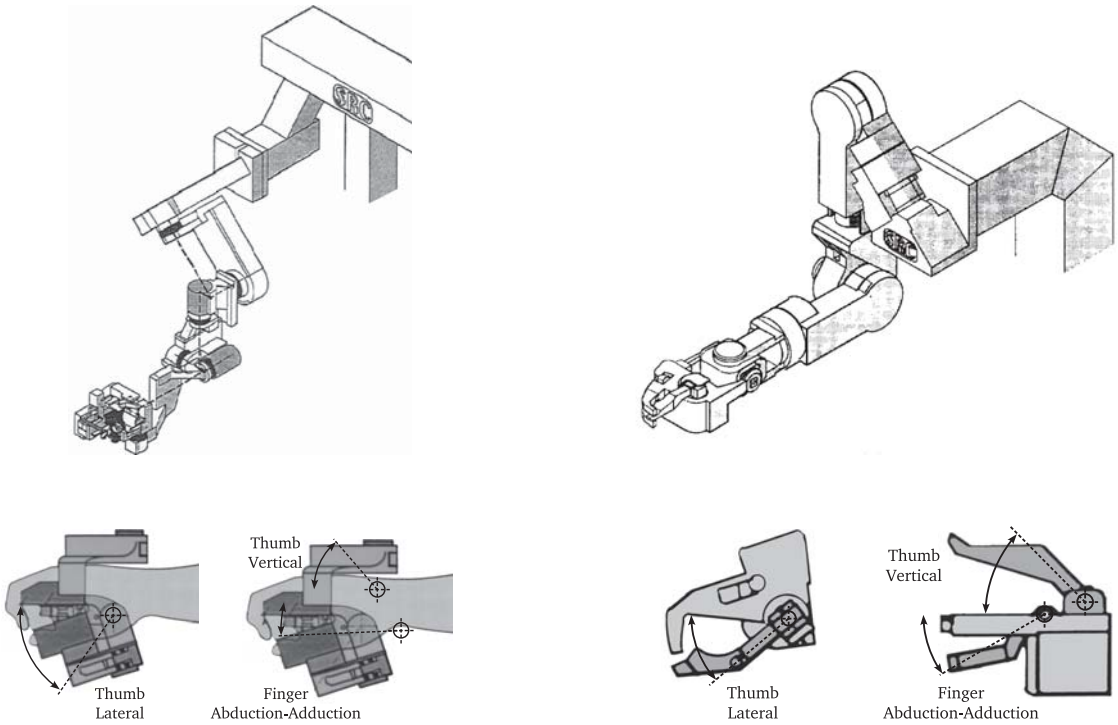


Figure A.2: Sketch of the Sarcos robots involved in the imitation learning setting. Left, a seven DOF exoskeleton robot arm with a three DOF hand controller used to record human trajectories. Right, a seven DOF anthropomorphic robot arm with a three DOF end effector used to perform learned movements.

The control architecture of both robots consists of independent PD servo controllers at each joint (implemented on individual Sarcos Advanced Joint Controller analog circuit boards), with additional feed-forward torque commands computed on a centralized controller running on 2 Motorola PPC 603 parallel processors with the commercial real-time operating system vxWorks (Windriver Systems). Potentiometers and load cells at each joint are sampled at 960 Hz to provide positional and torque feedback, respectively. The centralized controller updates the feed-forward commands and PD set points at 480 Hz.

Human trajectories are recorded using the Sarcos exoskeleton robot arm. Its anthropomorphic design mimics the major seven DOF of the human arm*, such that any joint movement of the user's arm will be approximately reflected by a similar joint movement of the exoskeleton. The robot arm's most proximal joint is mounted to a fixed platform,

*The major joints of the human arm are labeled as follows: shoulder-flexion-extension (SFE), shoulder-abduction-adduction (SAA), humeral rotation (HR), elbow-flexion-extension (EFE), wrist-supination-pronation (WSP), wrist-flexion-extension (WFE), and wrist-abduction-adduction (WAA).

and the user wields the device by holding on to a handle at the most distal joint and by a strapping of the forearm to the equivalent link of the robot just before the elbow. To not (unintentionally) burden the user by the exoskeleton during movement demonstrations, the control scheme compensates for exoskeleton's gravity, centrifugal, Coriolis, and inertia forces as proposed in (Mistry, Mohajerian, & Schaal, 2005).

Besides the seven DOF in the arm, both robots offer additional DOF at their end effectors. In particular, two DOF at the thumb (thumb-lateral (TL) and thumb-vertical (TV)) and one DOF at the lower of both fingers (finger-abduction-adduction (FAA)). Thus, the Sarcos master arm enables an user to record arm movements as well as two dimensional movements of his thumb and one dimensional movements of his middle-finger. During movement execution, the joint positions are sampled at the rate of the centralized controller, i.e. 480 Hz.

The hydraulic activated Sarcos slave arm is a human-scaled, 7 DOF robot arm, with three intersecting axes of rotation at each of the shoulder and the wrist, and one axis at the elbow joint, and a 3 DOF gripper. It is kinematically equivalent to the Sarcos master arm, thus close to a human being. Its anthropomorphic configuration makes its use totally intuitive for the operator and ensures that its "reach space" is not very different from its workspace.

A.2 Willow Garage PR2 robot

The PR2 is a mobile manipulation platform built by Willow Garage*. The PR2 software system is written entirely in Robot Operating System (ROS). As such, all PR2 capabilities are available via ROS interfaces (see `www.ros.org` for more information).

Figure A.3: The Willow Garage PR2 robot. Photo by Luke Fisher Photography.

The first four degrees of freedom in the arm (shoulder pan, shoulder lift, upper-arm roll, and elbow flex) are all driven with Maxon RE40 motors that have single-stage planetary gear-heads which output to a toothed pulley that runs on a belt. The position is measured via an optical encoder on the back of the motor, so the backlash of the gear-head and the stretch of the belt both contribute to the error in the sensed position. All of these drivetrains have fairly high efficiency, which means they can be back-driven. This is made possible by the low force requirements on those joints due to the arms counterbalance. The counterbalance uses a system of springs and linkages to compensate for the effect

*Willow Garage, Inc., 68 Willow Road, Menlo Park, CA 94025, USA

of gravity on the upper arm and forearm links. The compensation should be effective throughout the range of motion of the first four links of the arm.

The forearm roll joint is driven by a motor in the upper-arm with a planetary gear-head and external spur gear. Position is also measured via an optical encoder on the back of the motor. The wrist flex and roll use a differential drive between two motors. If the motors are turned in the same direction the wrist rolls, if they are turned opposite the wrist flexes. The differential bevel gears are attached after a standard encoder, motor, and gear-head assembly.

The gripper consists of two fingers with two joints each. Within each finger, a four-bar linkage constrains the two joints to move together so that the fingertips remain parallel. The two fingers are constrained together with a sector gear that meshes between the first phalanges of the two fingers. The motion of the fingers is driven by a motor which drives a lead-screw in the palm of the hand. This lead-screw is embedded in the four-bar linkage so that the rotation of the motor has a nonlinear relationship to the movement of the joints. The gripper can be back-driven when significant force is applied.

Each motor/encoder on the PR2 has its dedicated Motor Controller Board (MCB). The MCB detects and counts transitions in the encoder signal, measures the motor current, and commands the voltage going to the motor. Each MCB runs a PI-control loop to control the motor current to a desired value, by commanding the motor voltage. This control loop is closed at 100 kHz on an FPGA on the MCB. A shared EtherCAT realtime Ethernet link with the first computer allows all MCBs in the PR2 to communicate with the main computers with deterministic timing.

The gripper of the PR2 is equipped with a Bosch BMA150 digital triaxal accelerometer. The measurement range (2 g, 4 g, or 8 g) and bandwidth (25 Hz-1500 Hz) of the accelerometer can be selected in software

The RoboTouch tactile sensing pads are made by Pressure Profile Systems, each includes 22 tactile sensing elements: fifteen in a 5x3 array on the front surface, two on top, two on each side, and one in back, near the top. Each tactile element has a pressure range of 0-30 psi (0-205 kPa) and sensitivity of 0.1 psi (0.7 kPa). The sensors connect to the robot via an SPI ribbon cable and two screws, and have a maximum scan rate of 35 Hz. The tactile sensors are highly fragile when not protected by the rubber covering, and so care should be taken not to damage the rubber covering.

A.3 The ARM-S robot

The ARM-S robot consists of two fully equipped Barrett* Whole Arm Manipulators (WAMs) and a sensor head, (see Fig. A.4).

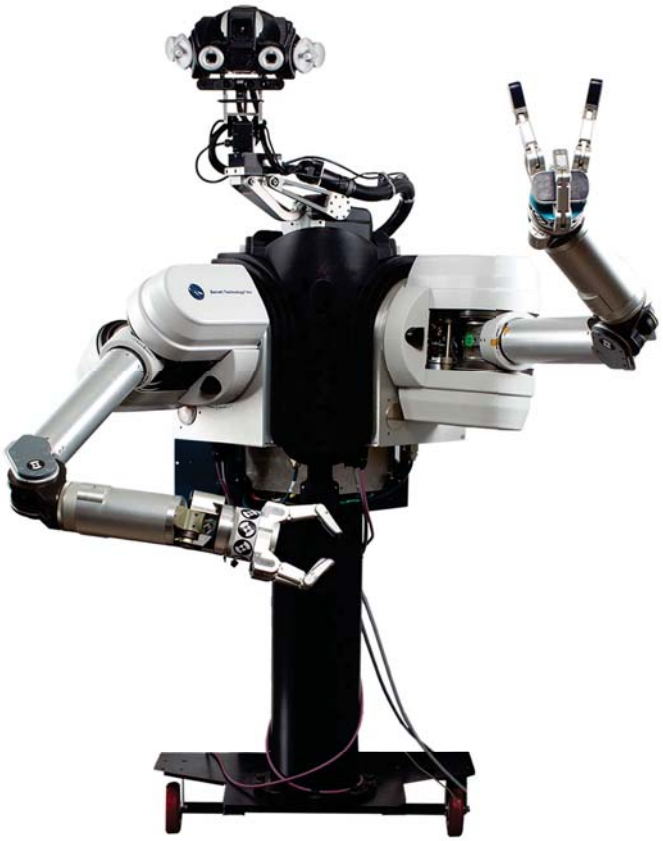


Figure A.4: The two armed ARM-S robot. Photo by Luke Fisher Photography.

Each of the 7-DOF manipulator include a Barrett 6 axis Force/Torque sensor attached at its wrist and a 3-finger Barrett Hand (BH8-280). Each finger has one actuated DOF that moves a finger with 2 articulations. The spread between the left and right finger is actuated and can move these two fingers 180 degrees around the palm. These fingers move in a symmetric way. Each finger knuckle contains a strain gage that measures the applied torque. Both the WAM and the hand can be torque controlled. Nonetheless, there is no direct torque sensing on the 7 joints of the WAM robot and therefore it is not possible to implement a low-level torque feedback controller. We therefore assume that torque is proportional to the motor current. Another issue is that the WAM has position encoders mounted on the motors but no encoders on the joints. Therefore, due to variable stretch

*Barrett Technology, 625 Mount Auburn Street, Cambridge, MA 02138-4555, USA

of the cables depending on the robot pose and its previous motions it is not possible to get an accurate position of the joints for fine manipulation. Internal testing using a VICON motion capture system confirmed a difference of up to 4 cm between the real position of the end-effector and the position computed using joint sensing and forward kinematics.

The neck is composed of two pan/tilt units from DirectedPerception mounted on top of each other and has therefore 4 DOFs. The pan-tilt units are actuated with stepper motors. A spring has been added to the pan/tilt units subsequently since the upper tilt motor was under-dimensioned. The visual sensors on the head are an Allied Vision Technologies Prosilica GC2450C for high resolution images, a Point Grey Bumblebee2 for stereo vision and a Asus Xtion for depth sensing. More details about the ARM-S robot can be obtained at `www.thearmrobot.com`.

References

Abbeel, P., Coates, A., & Ng, A. Y. (2010). Autonomous Helicopter Aerobatics through Apprenticeship Learning. *The International Journal of Robotics Research*, *29*(13), 1608–1639.

Ajallooeian, M., Gay, I. A. J., S., Khansari-Zadeh, S. M., Kim, S., Billard, A., Rückert, E., et al. (2010). *Comparative evaluation of approaches in T.4.1-4.3 and working definition of adaptive module.* (Deliverable D4.1 Available from `http://www.amarsi-project.eu/system/files/Deliverable_4_1_0.pdf` [accessed Jan. 5, 2014])

Althoefer, K., Lara, B., Zweiri, Y. H., & Seneviratne, L. D. (2008). Automated dailure classification for assembly with self-tapping threaded fastenings using artificial neural networks. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *222*(6), 1081–1095.

Aoyagi, S., Kohama, A., Nakata, Y., Hayano, Y., & Suzuki, M. (2010). Improvement of Robot Accuracy by Calibrating Kinematic Model Using a Laser Tracking System - Compensation of Non-Geometric Errors Using Neural Networks and Selection of Optimal Measuring Points Using Genetic Algorithm-. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5660–5665).

Arbib, M. A. (1992). Schema Theory. *Encyclopedia of Artificial Intelligence*, *2*, 1427–1443.

Asfour, T., Regenstein, K., Azad, P., Schroder, J., Bierbaum, A., Vahrenkamp, N., et al. (2006). Armar-iii: An integrated humanoid platform for sensory-motor control. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 169–175).

Axelrod, B., & Huang, W. H. (2012). Improving hand-eye calibration for robotic grasping and manipulation. In *Ieee international conference on technologies for practical robot applications* (pp. 121–126).

Bagnell, J., Cavalcanti, F., Cui, L., Galluzzo, T., Hebert, M., Kazemi, M., et al. (2012). An Integrated System for Autonomous Robotics Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2955–2962).

Balasubramanian, R., Xu, L., Brook, P., Smith, J., & Matsuoka, Y. (2012). Physical Human Interactive Guidance: Identifying Grasping Principles From Human-Planned Grasps. *IEEE Transactions on Robotics*, *28*(4), 899–910.

Beetz, M., Klank, U., Kresse, I., Maldonado, A., Mösenlechner, M., Pangercic, D., et al. (2011). Robotic Roommates Making Pancakes. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 529–536).

Bell, J. (2012). Mars Exploration: Roving the Red Planet. *Nature*, *490*(7418), 34–35.

Bellman, R. (1957). *Dynamic Programming* (1st ed.). Princeton, NJ, USA: Princeton University Press.

Bennett, D., Geiger, D., & Hollerbach, J. (1991). Autonomous Robot Calibration for Hand-Eye Coordination. *The International Journal of Robotics Research*, *10*(5), 550–559.

Billard, A., & Matarić, M. J. (2001). Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, *37*(2), 145–160.

Billard, A., & Siegwart, R. (2004). Special Issue on Robot Learning from Demonstration. *Robotics and Autonomous Systems*, *47*, 65–67.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York: Springer.

Bohg, J., & Kragic, D. (2010). Learning Grasping Points with Shape Context. *Robotics and Autonomous Systems*, *58*(4), 362–377.

Bohg, J., Morales, A., Asfour, T., & Kragic, D. (2014). Data-Driven Grasp Synthesis - A Survey. *IEEE Transactions on Robotics*.

Bohren, J., Rusu, R., Jones, E., Marder-Eppstein, E., Pantofaru, C., Wise, M., et al. (2011). Towards Autonomous Robotic Butlers: Lessons Learned with the PR2. In *IEEE International Conference on Robotics and Automation* (pp. 5568–5575).

Boyd, D., & Crawford, K. (2012, June). Critical Questions for Big Data. *Information, Communication & Society*, *15*(5), 662–679.

Brock, O. (2011, December). Berlin Summit on Robotics. In *Conference Report* (pp. 1–10).

Buchanan, J., Park, J., Ryu, Y., & Shea, C. (2003). Discrete and cyclical units of action in a mixed target pair aiming task. *Experimental Brain Research*, *150*(4), 473–489.

Buchli, J., Righetti, L., & Ijspeert, A. J. (2006). Engineering entrainment and adaptation in limit cycle systems: From biological inspiration to applications in robotics. *Biological Cybernetics*, *95*(6), 645–664.

Buchli, J., Stulp, F., Theodorou, E., & Schaal, S. (2011). Learning Variable Impedance Control. *The International Journal of Robotics Research*, *30*(7), 820–833.

Bullock, D., Grossberg, S., et al. (1988). Neural Dynamics of Planned Arm Movements: Emergent Invariants and Speed-Accuracy Properties During Trajectory Formation. *Psychological Review*, *95*(1), 49–90.

Calinon, S., & Billard, A. (2008). A Probabilistic Programming by Demonstration Framework Handling Skill Constraints in Joint Space and Task Space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 367–372).

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

Chitta, S., Jones, E., Ciocarlie, M., & Hsiao, K. (2012). Perception, Planning, and Execution for Mobile Manipulation in Unstructured Environments. *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, *19*(2), 58–71.

Cho, S., Asfour, S., Onar, A., & Kaundinya, N. (2005). Tool reakage Detection Using Support Vector Machine Learning in a Milling Process. *International Journal of Machine Tools and Manufacture*, *45*(3), 241–249.

Ciocarlie, M., Hsiao, K., Jones, E. G., Chitta, S., Rusu, R. B., & Sucan, I. A. (2010). Towards Reliable Grasping and Manipulation in Household Environments. In *International Symposium on Experimental Robotics* (pp. 1–12).

Cohen, B. J., Chitta, S., & Likhachev, M. (2010). Search-based Planning for Manipulation with Motion Primitives. In *IEEE International Conference on Robotics and Automation* (pp. 2902–2908).

Das, J., Py, F., Maughan, T., OReilly, T., Messié, M., Ryan, J., et al. (2012). Coordinated Sampling of Dynamic Oceanographic Features with Underwater Vehicles and Drifters. *The International Journal of Robotics Research*, *31*(5), 626–646.

Davis, S., & Mermelstein, P. (1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *28*(4), 357–366.

Degallier, S., Righetti, L., Gay, S., & Ijspeert, A. J. (2011). Toward simple control for complex, autonomous robotic applications: combining discrete and rhythmic motor primitives. *Autonomous Robots*, *31*(2-3), 155–181.

Degallier, S., Righetti, L., & Ijspeert, A. J. (2007). Hand placement during quadruped locomotion in a humanoid robot: A dynamical system approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2047–2052).

Degallier, S., Santos, C. P., Righetti, L., & Ijspeert, A. J. (2006). Movement generation using dynamical systems: a humanoid robot performing a drumming task. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 512–517).

Deimel, R., & Brock, O. (2013). A Compliant Hand Based on a Novel Pneumatic Actuator. In *IEEE International Conference on Robotics and Automation* (pp. 2047–2053).

Del Vecchio, D., Murray, R., & Perona, P. (2003). Decomposition of Human Motion into Dynamics-based Primitives with Application to Drawing Tasks. *Automatica*, *39*(12), 2085–2098.

Dogar, M., & Srinivasa, S. (2010). Push-Grasping with Dexterous Hands: Mechanics and a Method. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2123–2130).

Dollar, A. M., Jentoft, L. P., Gao, J. H., & Howe, R. D. (2010). Contact sensing and grasping performance of compliant hands. *Autonomous Robots*, *28*, 65–75.

Dyer, P., & McReynolds, S. (1970). *The Computation and Theory of Optimal Control*. Acad. Press.

Eppner, C., & Brock, O. (2013). Grasping Unknown Objects by Exploiting Shape Adaptability and Environmental Constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (p. 1-7).

Fajen, B. R., & Warren, W. H. (2003). Behavioral Dynamics of Steering, Obstacle Avoidance, and Route Selection. *Journal of Experimental Psychology: Human Perception*

and Performance, *29*(2), 343–362.

Fajen, B. R., Warren, W. H., Temizer, S., & Kaelbling, L. P. (2003). A Dynamical Model of Visually-Guided Steering, Obstacle Avoidance, and Route Selection. *International Journal of Computer Vision*, *54*, 13–34.

Felip, J., & Morales, A. (2009). Robust sensor-based grasp primitive for a three-finger robot hand. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1811–1816).

Flanagan, J. R., Bowman, M. C., & Johansson, R. S. (2006). Control strategies in object manipulation tasks. *Current Opinion in Neurobiology*, *16*(6), 650–659.

Flash, T., & Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, *15*(6), 660–666.

Flash, T., & Hogan, N. (1985). The coordination of arm movements: An experimentally confirmed mathematical model. *The Journal of Neuroscience*, *5*(7), 1688–1703.

Friedland, B. (1986). *Control system design: An introduction to state-space methods*. New York: McGraw-Hill.

Furukawa, N., Namiki, A., Senoo, T., & Ishikawa, M. (2006). Dynamic Regrasping Using a High-speed Multifingered Hand and a High-speed Vision System. In *IEEE International Conference on Robotics and Automation* (pp. 181–187).

Gams, A., Do, M., Ude, A., Asfour, T., & Dillmann, R. (2010). On-line periodic movement and force-profile learning for adaptation to new surface. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 560–565).

Gams, A., Ijspeert, A. J., Schaal, S., & Lenarčič, J. (2009). On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots*, *27*(1), 3–23.

Gams, A., Nemec, B., Zlajpah, L., Wachter, M., Ijspeert, A. J., Asfour, T., et al. (2013). Modulation of Motor Primitives Using Force Feedback: Interaction with the Environment and Bimanual Tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5629–5635).

Ghahramani, Z. (2000). Computational Neuroscience: Building blocks of movement. *Nature*, *407*(6805), 682-683.

Gienger, M., Goerick, C., & Koerner, E. (2010). Movement control in biologically plausible frames of reference. In *International symposium on robotics* (pp. 1–7).

Giese, M. A., Mukovskiy, A., Park, A.-N., Omlor, L., & Slotine, J.-J. E. (2009). Real-Time Synthesis of Body Movements Based on Learned Primitives. In *Statistical and Geometrical Approaches to Visual Motion Analysis* (Vol. 5604, pp. 107–127). Springer Berlin Heidelberg.

Giszter, S. F., Mussa-Ivaldi, F. A., & Bizzi, E. (1993). Convergent force fields organized in the frog's spinal cord. *Journal of Neuroscience*, *13*(2), 467–491.

Goldfeder, C. (2010). *Data-Driven Grasping*. PhD Thesis, Columbia University.

Goodale, M. A., & Milner, A. D. (1992). Separate Visual Pathways for Perception and Action. *Trends in Neurosciences*, *15*(1), 20–25.

Gribovskaya, E., Khansari-Zadeh, S. M., & Billard, A. (2011). Learning Non-linear Multivariate Dynamics of Motion in Robotic Manipulators. *The International Journal of Robotics Research*, *30*(1), 80–117.

Guckenheimer, J., & Holmes, P. (1983). *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields* (Vol. 42). Springer.

Hersch, M., & Billard, A. (2008). Reaching with multi-referential dynamical systems. *Autonomous Robots*, *25*(1-2), 71–83.

Hersch, M., Guenter, F., Calinon, S., & Billard, A. (2008). Dynamical System Modulation for Robot Learning via Kinesthetic Demonstrations. *IEEE Transactions on Robotics*, *24*(6), 1463–1467.

Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Asfour, T., & Schaal, S. (2012). Template-based Learning of Grasp Selection. In *IEEE International Conference on Robotics and Automation* (pp. 2379–2384).

Herzog, A., Pastor, P., Kalakrishnan, M., Righetti, L., Bohg, J., & Schaal, S. (2014). Learning of Grasp Selection based on Shape-Templates. *Autonomous Robots Journal Special Issue : Autonomous Grasping and Manipulation*, *36*(1-2), 51–65.

Hoffmann, H. (2011). Target switching in curved human arm movements is predicted by changing a single control parameter. *Experimental Brain Research*, *208*, 73–87.

Hoffmann, H., Pastor, P., Park, D.-H., & Schaal, S. (2009). Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. In *IEEE International Conference on Robotics and Automation* (pp. 2587–2592).

Hoffmann, H., & Schaal, S. (2007). Human movement generation based on convergent flow fields: a computational model and a behavioral experiment. In *Advances in Computational Motor Control VI, Symposium at the Society for Neuroscience Meeting.*

Hollerbach, J., Khalil, W., & Gautier, M. (2008). Model Identification. In B. Siciliano & O. Khatib (Eds.), *Springer Handbook of Robotics* (pp. 321–344). Springer.

Hopfield, J. J., & Tank, D. W. (1986). Computing with neural circuits: A model. *Science*, *233*(4764), 625–633.

Horswill, I. D. (1993). *Specialization of Perceptual Processes.* PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Howard, M., Klanke, S., Gienger, M., Goerick, C., & Vijayakumar, S. (2009). A novel method for learning policies from variable constraint data. *Autonomous Robots*, *27*, 105–121.

Hsiao, K. (2009). *Relatively Robust Grasping.* PhD Thesis, Massachusetts Institute of Technology.

Hsiao, K., Chitta, S., Ciocarlie, M., & Jones, E. G. (2010). Contact-Reactive Grasping of Objects with Partial Shape Information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1228–1235).

Hubert, U., Stückler, J., & Behnke, S. (2012). Bayesian Calibration of the Hand-Eye Kinematics of an Anthropomorphic Robot. In *IEEE-RAS International Conference*

on Humanoid Robots.

Hudson, N., Howard, T., Ma, J., Jain, A., Bajracharya, M., Myint, S., et al. (2012). End-to-End Dexterous Manipulation with Deliberate Interactive Estimation. In *IEEE International Conference on Robotics and Automation* (pp. 2371–2378).

Hudson, N., Ma, J., Hebert, P., Jain, A., Bajracharya, M., Allen, T., et al. (2014). Model-based autonomous system for performing dexterous, human-level manipulation tasks. *Autonomous Robots Journal Special Issue : Autonomous Grasping and Manipulation*, *36*(1-2), 31–49.

Hutchinson, S., Hager, G., & Corke, P. (1996). A Tutorial on Visual Servo Control. *IEEE Transactions on Robotics and Automation*, *12*(5), 651–670.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, *21*(4), 642–653.

Ijspeert, A. J., Nakanishi, J., Pastor, P., Hoffmann, H., & Schaal, S. (2013). Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*, *25*(2), 328–373.

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2002). Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots. In *IEEE International Conference on Robotics and Automation* (pp. 1398–1403).

Ijspeert, A. J., Nakanishi, J., & Schaal, S. (2003). Learning Attractor Landscapes for Learning Motor Primitives. In *Advances in Neural Information Processing Systems* (pp. 1547–1554).

Jain, A., & Kemp, C. C. (2013). Improving robot manipulation with data-driven object-centric models of everyday forces. *Autonomous Robots*, *35*(2-3), 143–159.

Jain, A., Killpack, M. D., Edsinger, A., & Kemp, C. C. (2013). Manipulation in Clutter with Whole-Arm Tactile Sensing. *The International Journal of Robotics Research*, *32*(4), 458–482.

Johansson, R. S., & Flanagan, J. R. (2009a). Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*(5), 345-359.

Johansson, R. S., & Flanagan, J. R. (2009b). Sensorimotor Control of Manipulation. *Encyclopedia of Neuroscience*, 593–604.

Kalakrishnan, M. (2014). *Learning Objective Functions for Autonomous Movement Generation*. PhD Thesis, University of Southern California.

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., & Schaal, S. (2010). Fast, Robust Quadruped Locomotion over Challenging Terrain. In *IEEE International Conference on Robotics and Automation* (pp. 2665–2670).

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., & Schaal, S. (2011). Learning, Planning, and Control for Quadruped Locomotion over Challenging Terrain. *The International Journal of Robotics Research*, *30*(2), 236–258.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic Trajectory Optimization for Motion Planning. In *IEEE International Conference on Robotics and Automation* (pp. 4569–4574).

Kalakrishnan, M., Pastor, P., Righetti, L., & Schaal, S. (2013). Learning Objective Functions for Manipulation. In *IEEE International Conference on Robotics and Automation* (pp. 1331–1336).

Kalakrishnan, M., Righetti, L., Pastor, P., & Schaal, S. (2011). Learning Force Control Policies for Compliant Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4639–4644).

Kalakrishnan, M., Righetti, L., Pastor, P., & Schaal, S. (2012). Learning Force Control Policies for Compliant Robotic Manipulation. In *International Conference on Machine Learning.*

Kappler, D., Pastor, P., Kalakrishnan, M., Wüthrich, M., & Schaal, S. (submitted). Data-Driven Autonomous Manipulation using Sensor Predictions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Kelso, J. A. (1995). *Dynamic Patterns: The Self Organization of Brain and Behaviour.* Mit Press.

Khalil, H. K. (1996). Adaptive output feedback control of nonlinear systems representedby input-output models. *IEEE Transactions on Automatic Control*, *41*(2), 177–188.

Khalil, H. K. (2002). *Nonlinear Systems.* Prentice Hall.

Khansari-Zadeh, S. M. (2012). *A Dynamical System-based Approach to Modeling Stable Robot Control Policies via Imitation Learning.* PhD Thesis, École Polytechnique Fédérale de Lausanne.

Khansari-Zadeh, S. M., & Billard, A. (2010). Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2676–2683).

Khansari-Zadeh, S. M., & Billard, A. (2012). A Dynamical System Approach to Realtime Obstacle Avoidance. *Autonomous Robots*, *32*(4), 433–454.

Khatib, O. (1986). Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *The International Journal of Robotics Research*, *5*(1), 90–98.

Kober, J., Mohler, B., & Peters, J. (2008). Learning Perceptual Coupling for Motor Primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 834–839).

Kober, J., Oztop, E., & Peters, J. (2010). Reinforcement Learning to adjust Robot Movements to New Situations. In *Robotics: Science and Systems (R:SS)* (pp. 2650–2655).

Kober, J., & Peters, J. (2008). Policy Search for Motor Primitives in Robotics. In *Advances in Neural Information Processing Systems* (pp. 297–304).

Kober, J., & Peters, J. (2009). Learning Motor Primitives for Robotics. In *IEEE International Conference on Robotics and Automation* (pp. 2112–2118).

Kober, J., & Peters, J. (2011). Policy Search for Motor Primitives in Robotics. *Machine Learning*, *84*(1-2), 171–203.

Kormushev, P., Calinon, S., & Caldwell, D. G. (2010). Robot Motor Skill Coordination with EM-based Reinforcement Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3232–3237).

Kormushev, P., Calinon, S., & Caldwell, D. G. (2011). Imitation Learning of Positional and Force Skills Demonstrated via Kinesthetic Teaching and Haptic Input. *Advanced Robotics*, *25*(5), 581–603.

Kormushev, P., Ugurlu, B., Colasanto, L., Tsagarakis, N., & Caldwell, D. (2012). The Anatomy of a Fall: Automated Real-time Analysis of Raw Force Sensor Data from Bipedal Walking Robots and Humans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3706–3713).

Kragic, D., & Christensen, H. (2003). Robust visual servoing. *The International Journal of Robotics Research*, *22*(10-11), 923–939.

Kugler, P. N., & Turvey, M. T. (1987). *Information, Natural Law and the Self-Assembly of Rhythmic Movement*. Erlbaum.

Kulvicius, T., Ning, K., Tamosiunaite, M., & Wörgötter, F. (2011). Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, *28*, 145–157.

Le, Q. V., & Ng, A. Y. (2009). Joint calibration of multiple sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3651–3658).

Lengagne, S., Vaillant, J., Yoshida, E., & Kheddar, A. (2013). Generation of Whole-body Optimal Dynamic Multi-contact Motions. *The International Journal of Robotics Research*, *32*(9-10), 1104–1119.

Lohr, S. (2012). The Age of Big Data. *New York Times*, *11*.

Maitin-Shepard, J., Cusumano-Towner, M., Lei, L., & Abbeel, P. (2010). Cloth Grasp Point Detection based on Multiple-View Geometric Cues with Application to Robotic Towel Folding. In *IEEE International Conference on Robotics and Automation* (pp. 2308–2315).

Majarena, A. C., Santolaria, J., Samper, D., & Aguilar, J. J. (2010). An Overview of Kinematic and Calibration Models Using Internal/External Sensors or Constraints to Improve the Behavior of Spatial Parallel Mechanisms. *Sensors*, *10*(11), 10256–10297.

Maldonado, A., Klank, U., & Beetz, M. (2010). Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2586–2591).

Manyika, J., Chui, M., Bughin, J., Brown, B., Dobbs, R., Roxburgh, C., et al. (2011, May). *Big Data: The next frontier for innovation, competition, and productivity.*

Mayton, B., LeGrand, L., & Smith, J. (2010). An Electric Field Pretouch System for Grasping and Co-Manipulation. In *IEEE International Conference on Robotics and Automation* (pp. 831–838).

Meier, F., Theodorou, E., & Schaal, S. (2012). Movement Segmentation and Recognition for Imitation Learning. *Journal of Machine Learning Research - Proceedings Track*, 761–769.

Meier, F., Theodorou, E., Stulp, F., & Schaal, S. (2011). Movement Segmentation using a Primitive Library. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3407–3412).

Mistry, M., Mohajerian, P., & Schaal, S. (2005). Arm movement experiments with joint space force fields using an exoskeleton robot. In *International conference on rehabilitation robotics (icorr)* (pp. 408–413).

Mitchell, T. M., Shinkareva, S. V., Carlson, A., Chang, K.-M., Malave, V. L., Mason, R. A., et al. (2008). Predicting Human Brain Activity Associated with the Meanings of Nouns. *Science*, *320*(5880), 1191–1195.

Mitzner, T. L., Smarr, C.-A., Beer, J. M., Chen, T. L., Springman, J. M., Prakash, A., et al. (2011). *Older Adults' Acceptance of Assistive Robots for the Home* (Tech. Rep. No. HFA-TR-1105). Georgia Institute of Technology. Human Factors and Aging Laboratory.

Montemerlo, M., et al. (2008). Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, *25*(9), 569–597.

Mühlig, M., Gienger, M., & Steil, J. J. (2012). Interactive imitation learning of object movement skills. *Autonomous Robots*, *32*(2), 97–114.

Nakanishi, J., Cory, R., Mistry, M., Peters, J., & Schaal, S. (2008). Operational Space Control: A Theoretical and Emprical Comparison. *The International Journal of Robotics Research*, *27*(6), 737–757.

Nakanishi, J., Mistry, M., Peters, J., & Schaal, S. (2007). Towards Compliant Humanoids - An Experimental Assessment of Suitable Task Space Position/Orientation Controllers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2520–2527).

Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., & Kawato, M. (2004). Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, *47*(2-3), 79–91.

Natale, L., & Torres-Jara, E. (2006). A sensitive approach to grasping. In *International conference on epigenetic robotics* (pp. 87–94).

Nemec, B., Tamosiunaite, M., Wörgötter, F., & Ude, A. (2009). Task adaptation through exploration and action sequencing. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 610–616).

Nemec, B., & Ude, A. (2011, October). Action Sequencing using Dynamic Movement Primitives. *Robotica*, 1–10.

Nguyen-Tuong, D., & Peters, J. (2010). Using Model Knowledge for Learning Inverse Dynamics. In *IEEE International Conference on Robotics and Automation* (pp. 2677–2682).

Nguyen-Tuong, D., Seeger, S., & Peters, J. (2008). Local Gaussian Process Regression for Real Time Online Model Learning and Control. In *Advances in Neural Information Processing Systems* (pp. 1193–1200).

Niekum, S., Chitta, S., Marthi, B., Osentoski, S., & Barto, A. G. (2013). Incremental Semantically Grounded Learning from Demonstration. In *Robotics: Science and Systems (R:SS).*

Niekum, S., Osentoski, S., Konidaris, G., & Barto, A. (2012). Learning and Generalization of Complex Tasks from Unstructured Demonstrations. In *IEEE/RSJ International*

Conference on Intelligent Robots and Systems (pp. 5239–5246).

Paolini, R., Rodriguez, A., Srinivasa, S., & Mason, M. T. (2013). A Data-Driven Statistical Framework for Post-Grasp Manipulation. In *Experimental Robotics* (Vol. 88, pp. 417–431). Springer.

Park, D., Hoffmann, H., Pastor, P., & Schaal, S. (2008). Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 91–98).

Pastor, P., Hoffmann, H., Asfour, T., & Schaal, S. (2009). Learning and Generalization of Motor Skills by Learning from Demonstration. In *IEEE International Conference on Robotics and Automation* (pp. 763–768).

Pastor, P., Hoffmann, H., & Schaal, S. (2008). Movement generation by learning from demonstration and generalizing to new targets. In *International Symposium on Adaptive Motion of Animals and Machines.*

Pastor, P., Kalakrishnan, M., Binney, J., Kelly, J., Righetti, L., Sukhatme, G., et al. (2013). Learning Task Error Models for Manipulation. In *IEEE International Conference on Robotics and Automation* (pp. 2612–2618).

Pastor, P., Kalakrishnan, M., Chitta, S., Theodorou, E., & Schaal, S. (2011). Skill Learning and Task Outcome Prediction for Manipulation. In *IEEE International Conference on Robotics and Automation* (pp. 3828–3834).

Pastor, P., Kalakrishnan, M., Meier, F., Stulp, F., Buchli, J., Theodorou, E., et al. (2013). From Dynamic Movement Primitives to Associative Skill Memories. *Robotics and Autonomous Systems*, *61*(4), 351–361.

Pastor, P., Kalakrishnan, M., Righetti, L., & Schaal, S. (2012). Towards Associative Skill Memories. In *IEEE-RAS International Conference on Humanoid Robots* (p. 309-315).

Pastor, P., Righetti, L., Kalakrishnan, M., & Schaal, S. (2011). Online movement adaptation based on previous sensor experiences. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 365–371).

Perk, B. E., & Slotine, J.-J. E. (2006). Motion Primitives for Robotic Flight Control. *CoRR*.

Peters, J. (2007). *Machine learning of motor skills for robotics*. PhD Thesis, University of Southern California.

Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Network Society*, *21*(4), 682–97.

Peters, R., Bodenheimer, R., & Jenkins, O. (2006). Sensory-Motor Manifold Structure Induced by Task Outcome: Experiments with Robonaut. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 484–489).

Pfeifer, R., & Iida, F. (2005). Morphological computation: Connecting body, brain and environment. *Japanese Scientific Monthly*, *58*(2), 48–54.

Platt, R. (2007). Learning Grasp Strategies Composed of Contact Relative Motions. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 49–56).

Platt Jr, R. J. (2006). *Learning and Generalizing Control-Based Grasping and Manipulation Skills*. PhD Thesis, University of Massachusetts - Amherst.

Pongas, D., Billard, A., & Schaal, S. (2005). Rapid Synchronization and Accurate Phase-locking of Rhythmic Motor Primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2911–2916).

Pradeep, V., Konolige, K., & Berger, E. (2014). Calibrating a multi-arm multi-sensor robot: A Bundle Adjustment Approach. In *Experimental Robotics* (Vol. 79, pp. 211–225). Springer Berlin Heidelberg.

Rasmussen, C. E., & Williams, C. (2006). *Gaussian Processes for Machine Learning*. MIT Press. Available from `http://www.gaussianprocess.org/gpml/`

Righetti, L., & Ijspeert, A. J. (2006). Design methodologies for central pattern generators: an application to crawling humanoids. In *Robotics: Science and Systems (R:SS)* (pp. 191–198).

Righetti, L., Kalakrishnan, M., Pastor, P., Binney, J., Kelly, J., Voorhies, R. C., et al. (2014). An Autonomous Manipulation System based on Force Control and Optimization. *Autonomous Robots Journal Special Issue : Autonomous Grasping and Manipulation*, *36*(1-2), 11–30.

Rodriguez, A., Bourne, D., Mason, M. T., Rossano, G. F., & Wang, J. (2010). Failure Detection in Assembly: Force Signature Analysis. In *IEEE Conference on Automation Science and Engineering* (pp. 210–215).

Rodriguez, A., Mason, M. T., Srinivasa, S., Bernstein, M., & Zirbel, A. (2011). Abort and Retry in Grasping. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1804–1810).

Romano, J. M., Hsiao, K., Niemeyer, G., Chitta, S., & Kuchenbecker, K. J. (2011). Human-Inspired Robotic Grasp Control with Tactile Sensing. *IEEE Transactions on Robotics*, *27*(6), 1067–1079.

Sanmohan, Krüger, V., Kragic, D., & Kjellström, H. (2011). Primitive-Based Action Representation and Recognition. *Advanced Robotics*, *25*(6-7), 871–891.

Schaal, S. (1997). Learning from demonstration. In *Advances in Neural Information Processing Systems* (pp. 1040–1046). MIT Press.

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

Schaal, S. (2007). The New Robotics - towards human-centered machines. In *HFSP Journal Frontiers of Interdisciplinary Research in the Life Sciences* (pp. 115–126).

Schaal, S., & Atkeson, C. G. (1998). Constructive Incremental Learning from Only Local Information. *Neural Computation*, *10*(8), 2047–2084.

Schaal, S., Ijspeert, A. J., & Billard, A. (2003). Computational Approaches to Motor Learning by Imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *358*(1431), 537–547.

Schaal, S., Mohajerian, P., & Ijspeert, A. (2007). Dynamics Systems vs. Optimal Control – A Unifying View. In T. D. Paul Cisek & J. F. Kalaska (Eds.), *Computational Neuroscience: Theoretical Insights into Brain Function* (Vol. 165, pp. 425–445).

Elsevier.

Schaal, S., Peters, J., Nakanishi, J., & Ijspeert, A. (2005). Learning Movement Primitives. In *Robotics Research* (Vol. 15, pp. 561–572). Springer Berlin Heidelberg.

Schaal, S., Sternad, D., & Atkeson, C. G. (1996). One-handed Juggling: A Dynamical Approach to a Rhythmic Movement Task. In *Journal of Motor Behavior* (pp. 165–183).

Schmidts, A. M., Lee, D., & Peer, A. (2011). Imitation learning of human grasping skills from motion and force data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1002–1007).

Schöner, G. (1990). A Dynamic Theory of Coordination of Discrete Movement. *Biological Cybernetics*, *63*(4), 257–270.

Schöner, G., & Kelso, J. A. (1988). Dynamic Pattern Generation in Behavioral and Neural Systems. *Science*, *239*(4847), 1513–1520.

Selverston, A. I. (1980). Are central pattern generators understandable? *Behavioral and Brain Sciences*, *3*, 535–571.

Shukla, A., & Billard, A. (2012). Augmented-SVM: Automatic space partitioning for combining multiple non-linear dynamics. In *Advances in Neural Information Processing Systems* (pp. 1025–1033).

Sinapov, J., Bergquist, T., Schenck, C., Ohiri, U., Griffith, S., & Stoytchev, A. (2011). Interactive Object Recognition using Proprioceptive and Auditory Feedback. *The International Journal of Robotics Research*, *30*(10), 1250–1262.

Srinivasa, S., Berenson, D., Cakmak, M., Collet Romea, A., Dogar, M., Dragan, A., et al. (2012). HERB 2.0: Lessons Learned from Developing a Mobile Manipulator for the Home. *Proceedings of the IEEE*, *100*(8), 1–19.

Srinivasa, S., Ferguson, D., Helfrich, C., Berenson, D., Collet Romea, A., Diankov, R., et al. (2010). HERB: A Home Exploring Robotic Butler. *Autonomous Robots*, *28*(1), 5–20.

Stilman, M. (2010). Autonomous Manipulation of Movable Obstacles. In *Motion planning for humanoid robots* (pp. 205–250). Springer London.

Strogatz, S. (2008). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Perseus Books Group.

Stulp, F., Oztop, E., Pastor, P., Beetz, M., & Schaal, S. (2009). Compact Models of Motor Primitive Variations for Predictible Reaching and Obstacle Avoidance. In *IEEE-RAS International Conference on Humanoid Robots* (pp. 589–595).

Stulp, F., Theodorou, E., Buchli, J., & Schaal, S. (2011). Learning to Grasp under Uncertainty. In *IEEE International Conference on Robotics and Automation* (pp. 5703–5708).

Sturm, J., Plagemann, C., & Burgard, W. (2008). Unsupervised Body Scheme Learning through Self-Perception. In *IEEE International Conference on Robotics and Automation* (pp. 3328–3333).

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press.

Takahashi, T., Tsuboi, T., Kishida, T., Kawanami, Y., Shimizu, S., Iribe, M., et al. (2008). Adaptive grasping by multi fingered hand with tactile sensor based on robust force and position control. In *IEEE International Conference on Robotics and Automation* (pp. 264–271).

Tenorth, M., Bandouch, J., & Beetz, M. (2009). The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In *IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV.*

Theodorou, E. (2011). *Iterative Path Integral Stochastic Optimal Control: Theory and Applications to Motor Control*. PhD Thesis, University of Southern California.

Theodorou, E., Buchli, J., & Schaal, S. (2010). A Generalized Path Integral Approach to Reinforcement Learning. *Journal of Machine Learning Research*, *11*, 3137–3181.

Ude, A., Gams, A., Asfour, T., & Morimoto, J. (2010). Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives. *IEEE Transactions on Robotics*, *26*(5), 800–815.

Urmson, C., et al. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, *25*(8), 425–466.

Wang, H., Jiang, M., Chen, W., & Liu, Y. (2012). Visual servoing of robots with uncalibrated robot and camera parameters. *Mechatronics*, *22*(6), 661–668.

Weisz, J., & Allen, P. (2012). Pose Error Robust Grasping from Contact Wrench Space Metrics. In *IEEE International Conference on Robotics and Automation* (pp. 557–562).

Wolpert, D. M., Diedrichsen, J., & Flanagan, J. R. (2011). Principles of sensorimotor learning. *Nature Reviews Neuroscience*, *12*, 739–751.

Wolpert, D. M., & Flanagan, J. R. (2001). Motor prediction. *Current Biology*, *11*(18), 729–732.

Wooden, D., Malchano, M., Blankespoor, K., Howardy, A., Rizzi, A. A., & Raibert, M. (2010). Autonomous Navigation for BigDog. In *IEEE International Conference on Robotics and Automation* (pp. 4736–4741).

Wüthrich, M., Pastor, P., Kalakrishnan, M., Bohg, J., & Schaal, S. (2013). Probabilistic Object Tracking using a Depth Camera. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*

Yamazaki, K., Ueda, R., Nozawa, S., Kojima, M., Okada, K., Matsumoto, K., et al. (2012). Home-Assistant Robot for an Aging Society. *Proceedings of the IEEE*, 2429–2441.

Yuan, J. (1988). Closed-loop manipulator control using quaternion feedback. *IEEE Journal of Robotics and Automation*, *4*(4), 434–440.

Zhuang, H., Wang, K., & Roth, Z. S. (1995, October). Simultaneous Calibration of a Robot and a Hand-Mounted Camera. *IEEE Transactions on Robotics and Automation*, *11*(5), 649–660.