Learning Sequential Skills for Robot Manipulation Tasks

Lernen von sequentiellen Fähigkeiten für Roboter-Manipulationsaufgaben
Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von M.Sc. Simon Manschitz aus Erbach (Odenwald)
Tag der Einreichung: 18.10.2017, Tag der Prüfung: 19.12.2017
Erscheinungsjahr: 2018
Darmstadt — D 17

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Aude Billard
3. Gutachten: Dr.-Ing. Jens Kober, Assistant Professor

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Intelligent Autonomous Systems

Learning Sequential Skills for Robot Manipulation Tasks
Lernen von sequentiellen Fähigkeiten für Roboter-Manipulationsaufgaben

Genehmigte Dissertation von M.Sc. Simon Manschitz aus Erbach (Odenwald)

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Aude Billard
3. Gutachten: Dr.-Ing. Jens Kober, Assistant Professor

Tag der Einreichung: 18.10.2017
Tag der Prüfung: 19.12.2017

Darmstadt — D 17

Erklärung zur Dissertation

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

Darmstadt, den 18. Oktober 2017

(Simon Manschitz)

Abstract

Most people's imagination about robots has been shaped by Hollywood movies or novels, resulting in the dream of having robots as assistants or household helpers in our homes. However, there is still a large gap between this dream and the actual capabilities of robots. One underlying reason is that every home is unique and largely unstructured, making it impossible to pre-program a robot for all the challenges it might face in such an environment. For instance, floor plans and furniture differ from home to home. Humans and pets walk around, potentially getting in the robot's way and making the environment non-static. Hence, a pre-programmed robot deployed in such an environment will undoubtedly face problems that it cannot solve with its existing knowledge. In order to cope with this issue, researchers started to equip robots with learning capabilities. Ideally, such capabilities allow a robot to adapt skills to new or changing situations or even to learn completely new tasks. Also humans learn new skills over time and are able to adapt them if needed. Therefore, such learning capabilities seem natural to us. If we are not able to master a specific task, we usually would ask another person to demonstrate it or to give instructions on how to perform it. In robotics research, the field of "Learning from Demonstration" tries to mimic this behavior by learning new skills from demonstrations of a task. By applying machine learning techniques, the data perceived from a single or multiple demonstrations are exploited to learn a mapping from perception to the action of a robot.

In this thesis, we concentrate on important Learning from Demonstration aspects that have not gotten so much attention in the research community so far. In particular, we focus on learning methods for robot manipulation tasks. These tasks have two important characteristics. First, they can be naturally decomposed into a set of subtasks and, therefore, can be mastered by performing the individual subtasks in the correct sequential order. Second, they involve physical contact between the robot and objects in its environment. One aim of this thesis is developing methods which allow for learning skills for robot manipulation tasks that generalize well to unknown situations. For instance, a learned skill should also be applicable if positions and orientations of objects differ from those seen in a demonstration.

In the first part of the thesis, we focus on the "sequential" aspect of manipulation tasks. Many approaches assume that subtasks are executed in a purely sequential manner or that the human always demonstrates the same sequence of subtasks. We propose an approach that does not have this assumption. Based on the demonstrations, a graph is generated which connects the subtasks with each other. Each subtask is associated with a movement primitive, a basic elementary movement necessary to perform the subtask. Depending on the environmental conditions, different sequences of movement primitives are executed, allowing the robot to perform tasks which for instance require an arbitrary number of repetitions (e.g., unscrewing a light bulb).

As we concentrate on the sequential aspects of a task in the first part of the thesis, we assume the demonstrations are labeled with the correct movement primitives over time. Additionally, the movement primitives are predefined. In the second part of the thesis, these two assumptions are relaxed. We first present an approach which decomposes the demonstrations into a set of meaningful movement primitives by inferring the underlying sequential structure of the task. The decomposition is based on a probability distribution we call Directional Normal Distribution. By utilizing the distribution, our method infers if a movement should be performed relative to an object in the scene and if a force should be applied in certain directions or not. Forces are especially important when interacting with the environment, for example if the robot has to manipulate objects. By defining movements relative to objects in the scene, the robot is likely to generalize better to new situations, for instance if the object positions differ from the demonstrations. Our task-decomposition method allows for inferring the most likely movement primitives over time and replaces the process of manually labeling the demonstrations. By combining the method with the sequencing concept presented in the first part of the thesis, complex skills can be learned from scratch without further human supervision. Such a learning scheme is an

essential requirement for domestic robots, as not every human teacher might be able or willing to do the tedious labeling of the data.

In both the decomposition and the sequencing part of the thesis, we assume that the teacher performs point-to-point movements and stops between two successive movements. While these assumptions lead to an approach which can learn skills for fairly complex tasks, it also restricts the class of tasks for which the approach can be used. In the third part of the thesis, we therefore introduce the Mixture of Attractors movement primitive representation. Here, a movement is modulated by continuously changing the activations of a set of simple attractors over time. We present a learning algorithm for the representation which learns both the attractors and their activations. An important property of the representation is that the attractors can be defined in different coordinate frames. The continuous activations and the attractors defined in different coordinate frames allow the system to learn movements of arbitrary shape and to generalize them to different object positions. In addition, the transitions between successive movements are smooth. This property reflects an important behavior of humans who often tend to co-articulate between successive movements. In contrast to many existing approaches, movements are learned by solving a convex optimization problem that does not rely on a good initial estimate of parameters.

In summary, the contribution of this thesis to the state-of-the-art in Learning from Demonstration is two-fold. The first contribution is a framework which is able to learn sequential skills for robot manipulation tasks from a few demonstrations. In contrast to other approaches, our method incorporates object-relative movements and force information directly into the skill learning framework. The second contribution is the Mixture of Attractors movement primitive representation. The representation supports co-articulated movements represented in different coordinate frames and outperforms existing movement primitive representations in terms of accuracy and generalization capabilities. Both contributions are evaluated on a wide range of tasks in simulation and on a real single arm robot with seven degrees of freedom. Altogether, this thesis aims at bringing us closer to the dream of having autonomous robots in our homes.

Zusammenfassung

Die menschliche Vorstellung von Robotern wurde überwiegend von Hollywood-Filmen oder Büchern geprägt. Daraus entstand der Wunsch, Roboter als Assistenten oder Haushaltshelfer in unseren Wohnungen einzusetzen. Auch nach Jahren der Forschung besteht jedoch weiterhin eine Diskrepanz zwischen diesem Wunsch und den tatsächlichen Fähigkeiten von Robotern. Ein Grund für diese Diskrepanz ist, dass Wohnungen einzigartig und überwiegend unstrukturiert sind. So unterscheiden sich beispielsweise Grundrisse und Raumaufteilungen von Haus zu Haus. Des Weiteren bewegen sich Menschen und Haustiere frei innerhalb der Wohnung und erzeugen durch ihr Verhalten eine komplexe, dynamische Umgebung. Deshalb ist es quasi unmöglich Roboter so zu programmieren, dass sie allen Anforderungen in der realen Welt gewachsen sind. Es ist sehr wahrscheinlich, dass ein Roboter mit fest programmiertem Verhalten irgendwann auf ein Problem stoßen wird, welches er mit seinem vorhandenen Wissen nicht lösen kann. Aus diesem Grund versuchen Wissenschaftler seit geraumer Zeit, Roboter mit Lernfähigkeiten auszustatten. Im Idealfall ermöglichen solche Fähigkeiten das Anpassen eines Verhaltens an sich ändernde Anforderungen oder sogar das Erlernen von komplett neuen Aufgaben. Menschen besitzen ebenfalls ausgeprägte Lernfähigkeiten, weshalb es uns sehr natürlich erscheint, diese auch auf Roboter zu übertragen. Wenn ein Mensch eine ihm unbekannte Aufgabe nicht lösen kann, so fragt er oftmals eine andere, erfahrenere Person, ob sie ihm die Lösung zeigen kann. In der Robotikforschung beschäftigt sich das Feld „Learning from Demonstration“ damit, wie ein Roboter anhand von Demonstrationen eine Aufgabe erlernen kann.

In dieser Arbeit konzentrieren wir uns auf wichtige „Learning from Demonstration“-Aspekte, die bislang nicht so sehr im Fokus der Forschung standen. Insbesondere konzentrieren wir uns auf Lernmethoden für sequentielle Roboter-Manipulationsaufgaben. Solche Aufgaben haben zwei Charakteristika, die in unserem Kontext relevant sind. Zunächst lassen sie sich üblicherweise in Teilaufgaben zerlegen, die in der richtigen Reihenfolge abgearbeitet werden müssen, um die Gesamtaufgabe erfolgreich zu erledigen. Des Weiteren erfordern Manipulationsaufgaben eine direkte Interaktion des Roboters mit den Objekten in seiner Umgebung. Ein Ziel der Arbeit ist es Methoden zu entwickeln, die es erlauben, generalisierbare Fähigkeiten für Roboter-Manipulationsaufgaben zu lernen, die beispielsweise auch anwendbar sind, wenn Position und/oder Orientierung von Objekten von den Demonstrationen abweichen.

Im ersten Teil der Arbeit konzentrieren wir uns auf den „sequentiellen“ Charakter vieler Aufgaben. Eine Grundannahme vieler Ansätze in diesem Bereich ist, dass Teilaufgaben immer in der gleichen Reihenfolge demonstriert werden, auch wenn diese für das Durchführen der Aufgabe nicht entscheidend ist. Unser Ansatz geht nicht von dieser Annahme aus. Anhand der Demonstrationen wird eine Graph-Repräsentation der Aufgabe erzeugt, die die Teilaufgaben miteinander in Verbindung bringt. Jede Teilaufgabe wird durch en Bewegungsprimitiv repräsentiert, eine simple, elementare Bewegung, von der wir annehmen, dass das Ausführen der Bewegung die Teilaufgabe löst. In Abhängigkeit des aktuellen Zustandes des Roboters und dessen Umgebung werden unterschiedliche Sequenzen von Bewegungsprimitiven ausgeführt, sodass Aufgaben gelöst werden können, die beispielsweise eine beliebige Anzahl von Wiederholungen benötigen, wie das Herausschrauben einer Glühbirne.

Da wir uns im ersten Teil der Arbeit auf das Sequenzieren von Bewegungen konzentrieren, nehmen wir an, dass die Demonstrationen durch den Benutzer vor dem Lernprozess vorverarbeitet werden müssen. So muss der Benutzer beispielsweise die Demonstrationen in logische Segmente unterteilen und jedem Segment ein dazugehöriges Bewegungsprimitiv zuweisen. Außerdem gehen wir davon aus, dass die einzelnen Bewegungsprimitive unserem System bereits bekannt sind. Im zweiten Teil der Arbeit präsentieren wir einen Ansatz, der diese Vorverarbeitungsschritte ersetzt. Unsere Methode erlaubt das automatische Extrahieren von einzelnen Bewegungsprimitiven aus den Demonstrationen und kann jedem Teil einer Demonstration das wahrscheinlichste Bewegungsprimitiv zuweisen. Basierend auf einer von uns „Directional Normal Distribution“ genannten Wahrscheinlichkeitsverteilung kann für jedes Be-

wegungsprimitiv entschieden werden, ob die Bewegung relativ zu einem Objekt ausgeführt werden soll und ob der Roboter eine Kraft aufbringen muss. Durch das Erlernen von objekt-relativen Bewegungen können die erlernten Fähigkeiten auf beliebige Positionen und Orientierungen von Objekten angewendet werden, auch wenn diese sich von den Demonstrationen unterschieden. Durch die Kombination dieser Methode mit dem Sequenzierungskonzept aus dem ersten Teil der Arbeit können komplexe Fähigkeiten anhand von Demonstrationen erlernt werden, ohne dass der Benutzer in den Lernprozess eingreifen muss.

In den ersten beiden Teilen der Arbeit nehmen wir an, dass eine Demonstration einer Aufgabe aus einer Sequenz von Punkt-zu-Punkt Bewegungen besteht und das zwischen zwei demonstrierten Bewegungen eine kurze Pause gemacht werden muss. Auch wenn mit den vorgestellten Methoden Fähigkeiten für komplexe Aufgaben erlernt werden können, so schränken diese Annahmen doch die Menge von Problemen ein, auf denen sie angewendet werden können. Im dritten Teil der Arbeit präsentieren wir daher eine neue Bewegungsprimitivbeschreibung, die wir „Mixture of Attractors“ nennen. Hierbei wird eine Bewegung erzeugt, indem kontinuierlich die Aktivierungen von mehreren simplen Attraktoren gemischt werden. Wir präsentieren einen Lernalgorithmus für die Primitivbeschreibung, die sowohl die Position der Attraktoren als auch deren Aktivierungen anhand von Demonstrationen lernt. Eine wichtige Eigenschaft der Beschreibung ist, dass die Attraktoren in verschiedenen Koordinatensystem definiert werden können. Durch das kontinuierliche Verändern der Aktivierungen solcher Attraktoren können komplexe Bewegungen in Relation zu Objekten in der Umgebung gelernt werden. Des Weiteren wird automatisch auch ein fließender Übergang zwischen zwei aufeinanderfolgenden Bewegungen gelernt. Solch ein fließender Übergang wird Koartikulation genannt und kann auch beim Menschen beobachtet werden. Im Gegensatz zu anderen Methoden ist der Lernvorgang als konvexes Optimierungsproblem formuliert, weshalb die Qualität der Lösung nicht von einem Schätzwert der Parameter abhängt.

Zusammenfassend sind die Hauptbeiträge dieser Arbeit zum Stand der Forschung im Gebiet „Learning from Demonstration“ folgende. Der erste Beitrag ist ein Framework welches es ermöglicht, komplexe sequentielle Fähigkeiten für Manipulationsaufgaben anhand weniger Demonstrationen zu erlernen. Im Gegensatz zu vielen bereits existierenden Methoden konzentrieren wir uns dabei auf die Unterscheidung von Positions- und Kraftkontrolle sowie die Entscheidung ob eine Bewegung relativ zu einem Objekt ausgeführt werden soll. Der zweite Beitrag ist die „Mixture of Attractors“ Bewegungsprimitivbeschreibung, die das Erlernen von komplexen, koartikulierten, objekt-relativen Bewegungen ermöglicht. Die in der Arbeit präsentierten Methoden werden anhand von Simulationsergebnissen und realen Roboterexperimenten mit einem Roboterarm mit sieben Freiheitsgraden evaluiert und validiert. Das Gesamtziel dieser Arbeit ist es, einen Betrag zu leisten, der uns einen Schritt näher an das Ziel bringt, lernfähige und vielseitig einsetzbare Roboter in unseren Alltag zu integrieren.

Acknowledgment

I am very grateful to all the people who contributed in one or the other way to this thesis. First of all, I would like to thank Michael Gienger for his supervision and guidance during my time as a PhD student and the Honda Research Institute for making this thesis possible in the first place. I really liked the working atmosphere and research environment there. Jens Kober supervised me as a postdoctoral researcher at Honda and later stayed in contact with me and provided feedback after leaving the institute for becoming an Assistant Professor in Delft. Thank you very much! Next, I thank Jan Peters for his advice and support throughout the four years I had the pleasure of working with him, as well as Aude Billard for agreeing to be my external committee member and investing her valuable time into reviewing my thesis. Your feedback is greatly appreciated. Similarly, I would like to thank the other members of my thesis committee, Iryna Gurevych, Oskar von Stryk and Kristian Kersting, without whose time investment the defense of the thesis would not have been possible. I would also like to thank my office mates Manuel, Fabio, Andre and Aki for constructive feedback and lots of discussions and suggestions, as well as Martin Heckmann for being my friendly advisor at Honda and providing me with valuable feedback. I further thank all members of the Intelligent Autonomous Systems group. I really liked the time we spent together at conferences and on other occasions. Lastly, I would not have been able to write this thesis without the support of my family, my friends and my beloved wife Maren. Thank you very much!

Contents

List of Symbols

The following tables give an overview of the notation and list most of the variables used throughout the thesis. Symbols that only pertain to a specific section are defined where they are used. Due to the vast amount of variables, sometimes a symbol may be overloaded. In that case, the correct meaning should be apparent from the context.

Notation	Description
x	Scalar
$\dot{x}$	Time derivative
$\hat{x}$	Estimate of x
$\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$	Set of elements x_1, x_2, …, x_n
$\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$	Vector of elements x_i
x_i	Element i of vector $\boldsymbol{x}$
$\boldsymbol{x}_{i:j}$	Series of $j - i + 1$ vectors $\boldsymbol{x}_i$ through $\boldsymbol{x}_j$
$\boldsymbol{X} = \begin{pmatrix} X_{1,1} & \ldots & X_{1,M} \\ \vdots & \ddots & \vdots \\ X_{M,1} & \ldots & X_{M,M} \end{pmatrix}$	Matrix $\boldsymbol{X}$ with elements $X_{i,j}$
$\boldsymbol{X}_i$	Column i of $\boldsymbol{X}$
$\boldsymbol{X}_{i:j}$	Series of $j - i + 1$ matrices $\boldsymbol{X}_i$ through $\boldsymbol{X}_j$
$\boldsymbol{X}^{\mathrm{T}}$	Transpose of matrix
$\boldsymbol{X}^{-1}$	Matrix inverse
$\mathbb{R}$	Real numbers
$\lvert \cdot \rvert$	Absolute value, $\ell 1$-norm, Cardinality of a set
$\lVert \cdot \rVert_p$	p-norm

Variable	Description
τ	A trajectory
J	A cost function of any kind
t	Time
$\boldsymbol{R}$	Rotation matrix
$\boldsymbol{T}$	Transition matrix
$\boldsymbol{f}$	Features
$\boldsymbol{g}$	Goal of an attractor
$\boldsymbol{q}$	Joint angles
$\boldsymbol{x}$	Task-space data
$\upsilon, \dot{\boldsymbol{x}}$	Velocity

Acronym	Description
DMP	Dynamic Movement Primitive
GMM	Gaussian Mixture Model
LfD	Learning from Demonstration
MP	Movement Primitive
MoA	Mixture of Attractors
ProMP	Probabilistic Movement Primitive

1 Introduction

According to the World Robotics Report 2016, an estimated number of 31 million robots will be deployed in households worldwide by the year 2019 [International Federation of Robotics, 2016]. Despite an extensive predicted growth of the sales (see Figure 1.1), domestic robots are still restricted to a relatively small number of mass-market products: lawn mowers, floor cleaning and edutainment robots. In fact, 96 percent of the domestic robot sales are accounted by vacuum and floor cleaning robots. These robots are built for a single specific purpose and have limited interaction capabilities. Therefore, it is possible to pre-program most of their behaviors. For more sophisticated tasks, pre-programming is infeasible. The programmer cannot take into account all possible environmental conditions and tasks the robot will face in domestic homes. Loading a dish washer, for instance, requires handling objects of different size, shape or texture. Some objects might be fragile, while others can be handled with less care. In addition every dishwasher is different, even if they are manufactured by the same company. The variety of challenges a robot might face even increases if it is not built for a single purpose.

It is likely that at some point, a robot deployed in the real world will face a task it cannot solve with its existing knowledge. Current robots do not have the ability of learning new tasks autonomously as their behavior is usually fixed when leaving the factory. As such, we are still far away from universal housekeeper robots or robot butlers depicted in movies and novels. In this thesis, we refer to a skill as the ability of performing a task. For equipping robots with the ability of learning skills for previously unknown tasks, two main research directions exist: Reinforcement Learning and Imitation Learning. The field of Reinforcement Learning deals with learning new tasks autonomously via trial and error [Deisenroth et al., 2013, Kober et al., 2013]. A reward function evaluates the success of a trial and the robot's behavior is adapted to maximize the expected reward. In order to not harm the robot and increase learning speed, reinforcement Learning algorithms usually need a good initial behavior on which they can improve upon. Imitation Learning can be a means to find such an initialization [Argall et al., 2009, Chernova and Thomaz, 2014, Hussein et al., 2017]. The idea of Imitation Learning is to imitate the behavior of others. Such a learning scheme appears natural to us, as it is one way humans and many other mammals transfer knowledge from one generation to the next. For instance, Meltzoff and Moore [1977] showed that babies at the age of 12 days are already able to imitate manual gestures. In robotics, Learning from Demonstration emerged as a subfield of Imitation Learning. In Learning from Demonstration, a teacher demonstrates a task to a robot. While Imitation Learning refers to all possible kinds of learning by imitating others, in Learning from Demonstration the teacher always is aware that he or

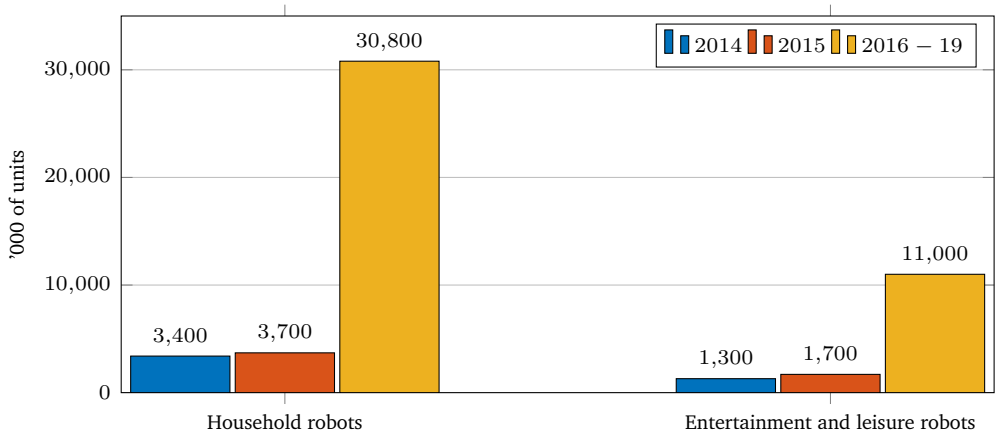


Figure 1.1.: Unit sales of service robots for personal/domestic use according to International Federation of Robotics [2016]. Sales 2014 and 2015, and forecast for 2016-2019.

she is demonstrating a task. Therefore, the teacher might emphasize certain aspects of a task or might perform it at low speed in order to simplify the learning problem. The term Learning from Demonstration is not used consistently throughout the literature. Common synonyms are (Robot) Programming by Demonstration, (Robot) Learning by Demonstration, Guiding and sometimes also Imitation Learning. In this thesis, Learning from Demonstration is the preferred term.

In contrast to Reinforcement Learning methods, Learning from Demonstration allows for rapid learning of a skill. If the essence of a task is captured, only one demonstration of a task can be sufficient for learning. Still, most existing learning algorithms either need many demonstrations for learning or result in skills which fail to generalize to situations different from the demonstrations. This thesis aims at developing methods which allow for learning skills with good generalization capabilities from a few demonstrations. For instance, a learned skill should also be applicable if positions and orientations of objects differ from those seen in a demonstration. We concentrate on skill learning methods for robot manipulation tasks. These tasks have two important characteristics. First, they may involve physical contact between the robot and objects in its environment, a property which is often neglected in learning approaches. Second, they can be naturally decomposed into a sequence of subtasks. The following sections will give an overview of the main contribution of this thesis and describe how our methods address the aforementioned topics.

1.1 Imitation Learning Challenges

Nehaniv and Dautenhahn [2001] summarized the main Imitation Learning challenges in four questions: Whom to imitate? What to imitate? How to imitate? When to imitate? In the following, we will briefly describe these questions and comment on our contribution to answering them. After describing the general challenges in the field of Imitation Learning, we will discuss some important open research problems in Section 1.1.5 which are specifically tackled in this thesis.

1.1.1 Whom to Imitate?

If multiple persons are present in a scene, it is often not clear who the person to imitate is. While answering this question is an interesting research problem, it is out of the scope of this thesis. Throughout the thesis, demonstrations are performed kinesthetically. In a kinesthetic demonstration, a human takes a gravity compensated robot by its arm and guides it through a task, similar to a tennis teacher teaching a task to a student. Such a way of demonstrating a task has two main benefits. First, it allows for recording the robot's joint angles directly. It is therefore easier to find a connection between the perceived state and the desired output of the robot. It also circumvents the correspondence problem [Nehaniv and Dautenhahn, 2002]. Due to the different kinematic structures of humans and robots, learning from non-kinesthetic demonstrations requires finding a relation between the movements of the human body and those of the robot. The relation determines which robot joint angle trajectories will produce movements that are similar to those of the human teacher. Second, if the robot is equipped with an appropriate sensor, kinesthetic demonstrations allow for recording the forces and/or torques directly at the robot. With such data, a decision can be made in which phase of a task the robot has to apply forces or torques. Due to the kinesthetic demonstrations, we do not have to decide whom to imitate. Instead, we concentrate on contributing to the three remaining questions.

1.1.2 What to Imitate?

What to Imitate relates to inferring what the human teacher wants the robot to do. Without knowledge about a task, it is often not obvious which aspects of a demonstration are to be imitated. For a box stacking task, it is important to properly grasp the box. Therefore, the alignment of the fingers and

the position of the teacher's hand are essential for the task, whereas for instance the exact pose of the elbow is not really important. In contrast, the elbow position may be important when playing tennis, as it allows for generating more powerful strokes. In addition, some environmental conditions might be irrelevant for a task, while others might be crucial. For the tennis example, the color of the ball is of much less importance compared to the ball's velocity and spin.

In order to learn a skill that goes beyond pure copying of the teacher's actions, the essential aspects of a task have to be extracted from the demonstrations. Several months old children already understand human intentions [Rao et al., 2007]. Instead of over-imitating human movements (imitating everything), they start to imitate object-related movements. Hence, they learn to generalize a skill to novel object positions or even new objects. In order to equip a robot with similar abilities, one of the core questions to be answered is what are subgoals to be reached in order to master a given task. In this thesis, we tackle this question on different levels of abstraction. On the highest level of abstraction, we try to infer the semantic structure of a task from the kinesthetic demonstrations. Some of the main problems here are: What are the subtasks to be mastered? Do they have to be reached in a certain order or is the ordering arbitrary? On an intermediate level we try to infer the composition of the subtasks. Does a subtask involve an object-related movement? Does the robot have to apply a force or should it be controlled kinematically? On the lowest level, we connect the sensory input of the robot to the corresponding subtasks. When does the robot perform which subtask? What movement is necessary to perform a specific subtask? These two questions are discussed in the following two sections.

1.1.3 How to Imitate?

Knowing the semantic structure of a task (e.g., the individual subtasks and their orderings) is not sufficient for performing the task. Instead, one also has to know *how* to perform the individual subtasks. Therefore, movement representation and movement generation are essential aspects of skill learning. In robotics research, a common assumption is that a subtask can be performed by executing an elementary movement called movement primitive (MP). A MP could for instance correspond to a tennis hitting movement. Based on a small number of parameters (e.g., the current position, velocity and spin of the ball), the MP can generate a trajectory which will allow the robot to hit the ball. Often, MPs are a means for compactly describing a movement without considering the specific embodiment of a robot. Hence, they can be easily be transferred between different robots or end-effectors. Such a description is supported by findings in biology which suggest that the human brain also encodes movements independently of the embodiment [Wing, 2000]. As a result, humans can for instance write the letter s with their hand, but also with a foot in the sand. In robotics, the field of Operational Space Control deals with emulating such a behavior [Khatib, 1987, Nakanishi et al., 2008]. By introducing task-spaces and coordinate frames, movements can be easily transferred between different kinematic structures and can be performed relative to objects in the scene.

In this thesis, we present an algorithm which extracts a set of MPs from the kinesthetic demonstrations of a task in an unsupervised fashion. In order to make the MPs reusable in more situations and to increase the generalization capabilities of the system, they are represented in different task-spaces and coordinate frames. In addition, we introduce a novel MP representation we call "Mixture of Attractors". The evaluation will show that this representation outperforms existing MP representations in terms of accuracy and generalization capabilities.

1.1.4 When to Imitate?

The when to imitate question can be summarized as monitoring and coordinating task execution. It is of high importance when executing a skill on a real robot. Here, the system has to decide when to perform which subtask. It has to monitor if a (sub)task is completed or if an error occurred. Throughout task

execution, it has to make decisions which determine the robot's behavior. The basis for the decision are the sensory input of the robot as well as the decision history. Depending on the environmental situation, the system may change the current behavior, for instance to avoid an obstacle. Most MP representations come with some sort of adaptation capabilities (e.g., a movement can be adapted to a changing goal position). Still, these capabilities are rather limited and usually not sufficient (and not meant) for reacting to a completely new situation. Therefore, a system which monitors task execution can be seen as a higher level system which coordinates the individual MPs. We present an algorithm which continuously coordinates the individual MPs. At every point in time, the system decides which MPs to activate. Based on the sensory input, the system may stop a MP and start to activate a new one.

1.1.5 Open Challenges

While the main Imitation Learning challenges can be summarized with the four aforementioned questions, we would like to point out three important challenges which are specifically tackled in this thesis. The interested reader is also referred to Section 5.2, where we discuss further open research problems which are out of the scope of this thesis.

Learning Sequential Skills

Most real-world tasks can be naturally decomposed into a sequence of subtasks. As they are usually rather complex, dividing them into multiple subtasks can help simplifying the overall learning problem. Finding such a decomposition is not straightforward. We present an algorithm which aims at decomposing a task given a set of unlabeled kinesthetic demonstrations of a task. In contrast to many other approaches, we do not assume that the sequence of subtasks demonstrated by the teacher is the same for all demonstrations. In addition, movements may be stopped prematurely, enabling the system to detect and react to errors.

Learning Force Interaction Skills

One main purpose of a robot is to interact with its environment. Even though this interaction leads inevitably to physical contact, force data is often ignored in Learning from Demonstration approaches. Forces can be useful in two different ways. First, measured forces can be useful for detecting contacts or contact changes. Second, some tasks require the robot to actively apply forces. An example is the task of cleaning a window. For this task, it is beneficial to push against the window while swiping from left to right, as it ensures contact with the window throughout the task. Therefore, it is often not sufficient to learn a kinematic plan of a task. While many approaches neglect the force information, others focus only on forces, for instance by learning desired force profiles. These approaches neglect that for many tasks the physical contact with the environment is restricted to certain phases of the task. For instance when approaching the window in the cleaning task, the robot is not in contact with an object. Hence, for many tasks a decision has to be made when to apply forces. In the remainder of the thesis, we refer to such tasks as force interaction tasks. For these tasks, we present a concept for deciding when to apply forces.

Learning from a few Demonstrations

The practical applicability of many approaches is limited as they require more demonstrations for learning a skill than a typical user is willing to provide. Ideally, only one demonstration should be sufficient for learning. For realistic tasks, it is not sufficient to just replay a demonstrated behavior. Instead, a skill has to capture the essential aspects of a task to ensure the system is able to generalize to novel situations such as varying object positions. Without additional information (e.g., voice commands or task knowledge), it is often not possible to capture these essential aspects from a single demonstration. Therefore, we aim at learning from as few demonstrations as possible. We present two approaches that try to extract as much information as possible from the demonstrations. Our task-decomposition

method extracts a set of point-to-point MPs from the demonstrations and chooses proper task-spaces and coordinate frames for them. Additionally, we present the Mixture of Attractors MP representation which allows for generating movements of arbitrary shape in relation to objects in the environment.

1.2 Main Contributions

In this thesis, we present several methods which contribute to the field of Learning from Demonstration. In this section, we outline the main contributions and comment on if and how they benefit from each other.

1.2.1 Learning to Sequence Movement Primitives with Sequence Graphs

Many manipulation tasks cannot be mastered by simply executing a completely pre-planned movement on a robot. For instance, when unscrewing a light bulb, the unscrewing movement has to be performed until the light bulb is loose. Here, the number of unscrewing repetitions cannot be planned in advance, as it is not clear how firmly the light bulb is screwed in before actually touching it. Therefore, in order to be able to perform such a task, the system has to be able to react dynamically to changes in the environment and adapt its behavior accordingly.

In this thesis, we address this problem by presenting a method which allows for learning to coordinate a set of MPs in order to perform a sequential robot manipulation task. When executing a learned skill on a robot, the system decides at each point in time which of the MPs to activate. This decision is made based on the state of the environment and therefore the system is able to adapt its behavior dynamically. The method is based on a graph representation we call sequence graph. In a sequence graph, each node corresponds to a MP and is associated with a classifier. The task of such a classifier is to decide when to switch between MPs that are connected in the graph. In contrast to many existing approaches, we do not assume that the sequence of demonstrated movements is the same in all demonstrations.

1.2.2 Probabilistic Task-Decomposition based on the Directional Normal Distribution

The sequence graph concept allows for performing a sequential robot manipulation task by coordinating the execution of a sequence of MPs. One assumption of the approach is that the MPs have to be known to the system. In order to bypass the tedious process of defining the MPs by hand, we further present a method that automatically decomposes a task into a set of its generating MPs. Here, we concentrate on tasks that can be represented by a sequence of point-to-point movements. In contrast to state-of-the-art methods, we explicitly incorporate force information and object-relative movements into the decomposition.

The task-decomposition method is mainly based on a novel probability distribution we call Directional Normal Distribution. In the context of the task-decomposition, the distributions allows for deciding if a movement should be performed relative to an object in the scene or if the robot is supposed to apply forces. In this thesis, we show that these decisions greatly improve the generalization capabilities of the learned skill. In general, the Directional Normal Distribution is a probability distribution over a point and its velocity vector. The density function of the distribution is large if the velocity vector of a point is pointing towards the mean of the distribution and small if it is pointing away from the mean. In this thesis, we introduce the distribution and present an Expectation-Maximization algorithm for learning its parameters from data.

In this thesis, we also combine the task-decomposition method with the aforementioned sequence graph concept. While applying the task-decomposition method results in a decomposition of a task into a set of MPs, the sequence graph concept can be utilized for learning to sequence the resulting MPs. By combining both methods, skills for sequential manipulation tasks can be learned from scratch without further human supervision.

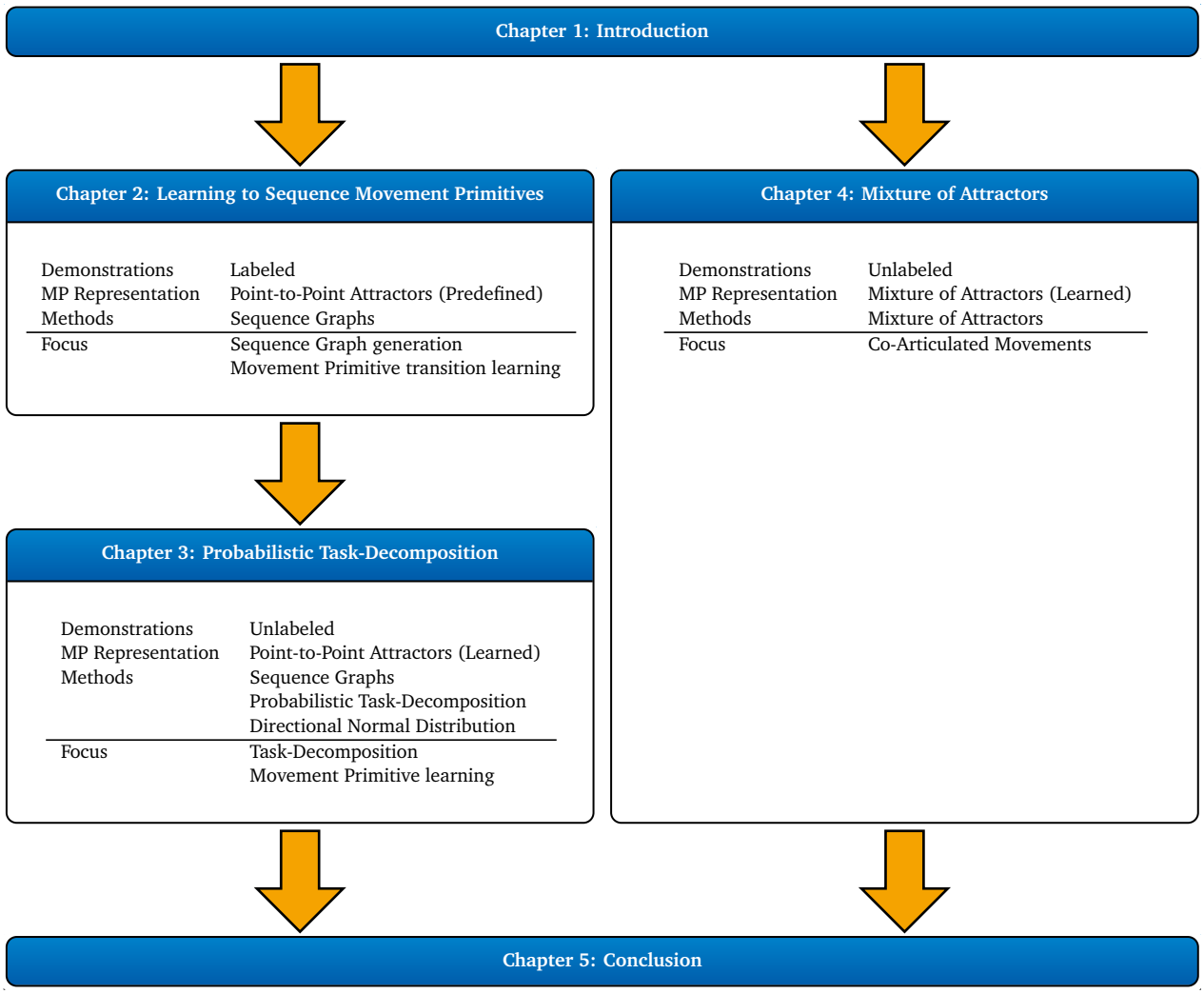


Figure 1.2.: Outline of the thesis. Chapter 4 can be read independently of the previous two chapters, as it concentrates on different skill learning aspects.

1.2.3 Object-Relative Movement Generation with Mixture of Attractors

The third major contribution of this thesis is a novel MP representation we call Mixture of Attractors (MoA). In contrast to the MP representation used within our task-decomposition framework, the movements generated by MoA can be of arbitrary shape. Hence, MoA allows for learning skills for tasks that cannot be represented by a sequence of point-to-point movements. Moreover, it is an important property of the MP representation that the shape of a movement is learned in relation to objects in the scene. Therefore, MoA allows for learning complex skills for tasks that involve interacting with multiple objects. By demonstrating a task multiple times with varying object positions, the system learns automatically which object is important in which phase of the task and transitions smoothly between different task phases.

The basic idea behind MoA is to represent a movement as weighted sum of simple attractors. By continuously changing the weights of the attractors over time, a movement of arbitrary shape can be generated by the system. The attractors can be defined in different coordinate frames, leading to movements which are conditioned on the position and orientation of objects. We show that learning a skill with this framework can be formalized as a convex optimization problem. Due to the convexity, the optimization result does not depend on a good initial estimate for the parameters of the representation. Instead, the importance of a coordinate frame emerges automatically from the optimization process and

can change over time. In addition, the optimization leads to smooth movements and smooth transitions between different object-relative movements. To the best of our knowledge, MoA is the first MP representation which has all of the aforementioned properties.

1.3 Outline

The individual chapters of this thesis can be read largely independently. Still, we recommend to read them in order. Excerpts of the presented research have led to various publications. This section presents the outline of the thesis and clarifies how the individual contributed to the chapters. Figure 1.2 depicts an overview of the thesis.

Chapter 2 introduces sequence graphs. These graphs represent the semantic structure of a task and are utilized to learn sequential skills in a supervised fashion. The method(s) presented in this chapter are based on Manschitz et al. [2014a,b, 2015b].

Chapter 3 introduces our probabilistic task-decomposition method. The method extracts a set of MPs from the demonstrations and finds their most likely composition (e.g., attractor goal and coordinate frame). For sequencing the resulting MPs, the sequence graph concept from the preceding chapter is utilized. Therefore, we recommend to read these two chapters in order. The chapter is based on Manschitz et al. [2016, 2017a,submitted].

Chapter 4 introduces the Mixture of Attractors framework, a novel MP representation. The representation can be used for generating object-relative movements of arbitrary shape. The MP representation relaxes some limitations of the movement primitive representation used in the two preceding chapters, which requires the teacher to demonstrate point-to-point movements. The content of this chapter is based on Manschitz et al. [2017b,accepted] and can be read independently of the two preceding chapters.

Chapter 5 concludes the thesis, gives an overview of open problems and discusses some future directions.

2 Learning to Sequence Movement Primitives

In this chapter, we address the skill learning problem from a higher level perspective. The particular focus is to learn the coordination of a set of MPs, in order to realize complex sequential movement behavior. An illustrative example is the replacement of a light bulb: mastering this task requires performing movements such as reaching towards a lamp, aligning the fingers with the bulb, grasping the bulb or turning it in the thread. A sequential skill coordinates these MPs with a flexible arbitration scheme: It needs to maintain the causal order of the MPs (e.g. reach, pre-shape, grasp), while coordinating the timing of the MP. In case of larger disturbances, the skill may need to adapt the sequential flow to account for the changed situation (e.g. pick up bulb if it drops out of the hand). The chapter is mainly based on the work presented in Manschitz et al. [2015b].

Most previous work on learning sequential skills has focused on tasks that do not require a flexible arbitration scheme. In that case, it is often sufficient to decide when to stop a current MP and start a new one. It is not required to make decisions on *which* MP to start next. We present an approach for learning such sequential robot skills through kinesthetic teaching. From the demonstrations, a graph representation reflecting the potential MP orders is extracted. Finding the transitions between consecutive MPs is treated as multiclass classification problem. We show how the goal parameters of linear attractor MPs can be learned from manually segmented and labeled demonstrations and how the observed MP order can help to improve the movement reproduction. The improvement is achieved by restricting the classification result to the currently activated MP and its possible successors in the graph representation of the sequence. The approach is validated with three experiments using a Barrett WAM robot.

2.1 Introduction

Arguably, one of the key elements for robots to become more autonomous is the ability of adapting skills to new situations. Instead of pre-programming the way a robot handles unknown situations, Learning from Demonstration (LfD) can be utilized for learning how to adapt to them. The idea is to demonstrate the essential aspects of a task by showing a robot some variations of the task. For instance, demonstrating the same task with different object positions allows for inferring which object may be important in which phase of the task. Or, demonstrating a repetitive task multiple times allows for learning when to stop the repetitions. While demonstrating all kinds of variations is infeasible, demonstrating only a few variations can be sufficient for learning skills that can be adapted to a wide range of situations.

Many approaches in the LfD domain focus on learning single movement skills. Hence, they cannot learn skills for repetitive tasks or tasks which require to handle different situations (e.g., error recovery). One way of mastering these tasks is to sequence single movements. In this chapter, we refer to the ability of sequencing movements in order to perform a complex task as a sequential skill. Learning such sequential skills is still an open research topic. These skills are particularly useful in two cases. First, there are tasks which are not representable in a non-sequential way at all. As an example, consider a robot standing in front of a door. Without any additional knowledge, the system does not know whether the robot has to open the door or if the robot just closed it. The reason is that the same state is perceived for both options. This problem is often referred to as perceptual aliasing [Whitehead and Ballard, 1991]. Dissolving perceptual aliasing requires either the previous movement history to be encoded in the perceived state or a policy which activates movements based on the history of movements. Second, even though a task may be representable using a single movement, it may be beneficial to decompose it into smaller (sub-)tasks first. Such a decomposition bounds the complexity of each (sub-)task and the resulting movements are often more intuitive and easier to learn.

In this chapter, we aim at learning sequential skills where the currently activated movement cannot be solely determined from the perceived state, but may also depend on the history of movements. The goal

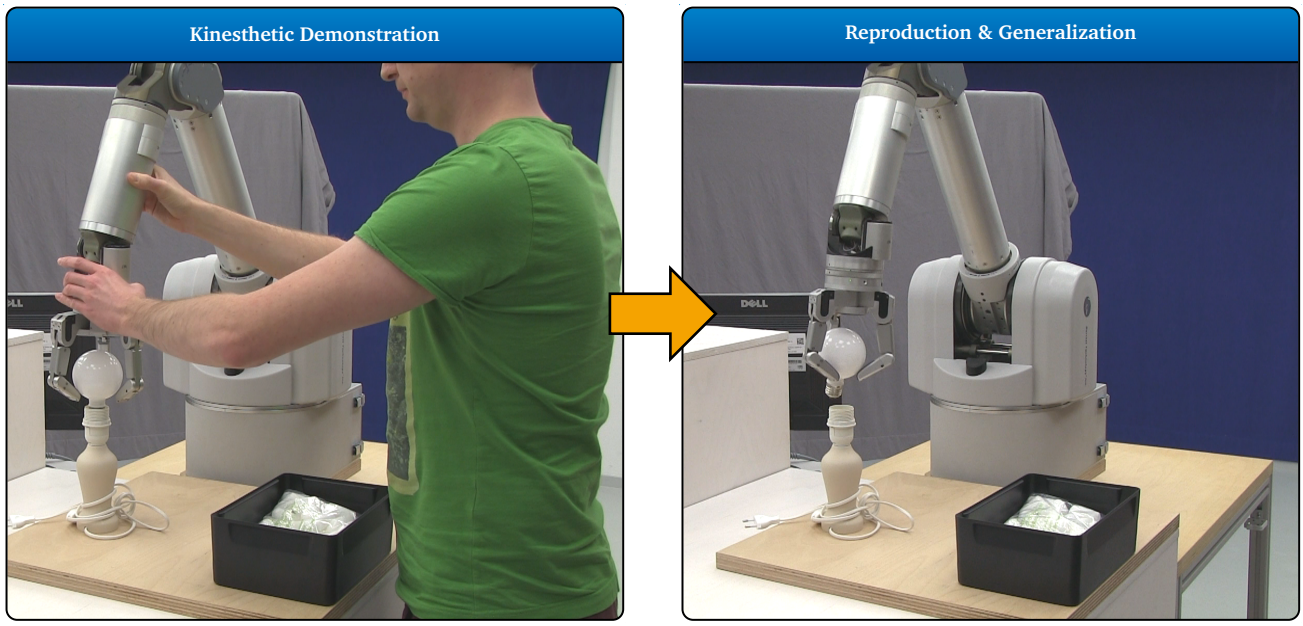


Figure 2.1.: The system is supposed to learn how to unscrew a light bulb from kinesthetic demonstrations. We evaluate our approach on this example using a real seven degrees of freedom (DoF) Barrett WAM robot with a four DoF hand.

is to learn when to activate each movement, based on kinesthetic demonstrations. Kinesthetic teaching is a widely used teaching method in robotics. Here, a teacher guides a robot through movements by physically moving the robot's arm, similar to parents teaching tasks to their children (see Figure 2.1).

2.1.1 Related Work

Various approaches exist which aim at learning skills for sequential tasks. Often, these approaches are fundamentally different in their assumptions about the tasks or demonstrations or aim at different learning aspects. In this section, we review other approaches, classify them into different categories and contrast our approach to this prior work.

Segmenting Demonstrations into Sequences of Movement Primitives

Single elementary movements are often modeled as MPs in literature [Flash and Hochner, 2005]. By chaining or blending a set of MPs, complex tasks can be performed. In order to avoid tedious pre-programming of MPs, it is an interesting research problem to learn MPs from demonstrations of a task. Many approaches in this domain therefore concentrate on the analysis of human demonstrations and aim at finding reoccurring patterns in the time-series data resulting from the demonstrations. For instance, Chiappa and Peters [2010] cluster individual demonstrations using Bayesian mixtures of linear Gaussian state-space models. Their method clusters full demonstrations and hence does not segment demonstrations into smaller building blocks. Meier et al. [2011] assume a MP library is given and use it for movement recognition. Their approach is able to segment a demonstration into its most likely sequence of generating MPs, even if movements are only partially observed. Lioutikov et al. [accepted] additionally learn the MP library from demonstrations. Recently, an extensive survey on segmentation approaches has been published by Lin et al. [2016]. The authors categorize the methods based on their definition of a segment, how the data is collected, what the requirements are, the actual segmentation method and how the method is verified. The survey provides a general overview of segmentation

approaches and is not restricted to LfD and robotics. Among others, also methods from physical rehabilitation [Houmanfar et al., 2016, Karg et al., 2015], activity tracking [Krishnan et al., 2008, Li et al., 2013] or human movement recognition [Ahad et al., 2008, Lara and Labrador, 2013] are discussed.

In general, many segmentation approaches focus on movement analysis and not on movement generation. The approaches try to come up with a set of MPs which may have generated the demonstrations, but do, for instance, not reason about the observed ordering of MPs or the reason for starting a new movement. For these approaches, movement generation often serves as proof of concept for the segmentation. Therefore, the sequence of executed MPs is chosen randomly (e.g., in Kulić et al. [2012]) or is the same as in the demonstrations [Maeda et al., 2014]. The transition behavior between MPs is also either deterministic (e.g., the succeeding movement depends only on the previous movement) or not learned at all [Grollman and Jenkins, 2010].

Learning MPs from demonstrations of a task is an important aspect of skill learning. The main advantage of segmentation methods is that they allow for learning skills in an unsupervised fashion, making the tedious pre-programming of the required movements superfluous. Yet, in order to learn a skill, also the temporal and spatial correlations of the MPs have to be learned. One way of modeling these correlations is a higher-level task representation. Such a representation connects the individual MPs. Hence, it represents potential MP sequences and determines when to perform which of the movements. In this chapter, we exactly aim at extracting such a representation from the demonstrations of a task. As we concentrate on this aspect, we assume the demonstrations are already segmented into a set of MPs. In the succeeding chapter, we will then introduce our own segmentation method. Therefore, segmentation and task-decomposition methods will be discussed in Chapter 3 in more detail.

Sequencing Movement Primitives

While there exists a vast amount of methods for segmenting demonstrations into a set of MPs, only a few approaches aim at sequencing these MPs in order to perform a complex task. Traditionally, methods from this field are inspired by the subsumption architecture [Brooks, 1986]. Here, the behavior of a system is represented by a hierarchy of sub-behaviors. A sequential skill can be composed by a two-level hierarchy, whereby the lower-level MPs are activated by an upper-level sequencing layer. There are various ways of modeling the sequencing layer. Among the options are graph structures Kroemer et al. [2014], Kulić et al. [2008], Finite State Machines (FSMs, [Niekum et al., 2013, Riano and McGinnity, 2012, Sullivan and Luke, 2012]) or Petri nets ([Chang and Kulić, 2013a,b]). Usually, the activation of a new MP is interpreted as discrete event in a continuous system [Chang and Kulić, 2013b, Pavlovic et al., 2000, Peters, 2005]. An alternative view is treating the overall system as continuous entity. For example, Luksch et al. [Luksch et al., 2012] model a sequence with a recurrent neural network. In that architecture, MPs can be concurrently active and inhibit each other. Therefore the sequence is defined implicitly. Although this structure leads to very smooth movements, the model is hard to learn and has to be defined mostly by hand. Levine et al. [2016] present a method which learns a complex skill using a deep neural network. Due to large number of parameters of the network, the method is very data-intensive. As we aim at learning skills from only a handful demonstrations, the method is not applicable in our context.

Pastor et al. [2012] use a nearest neighbor classifier for deciding which MP to activate when the current movement has finished. Kappler et al. [2015] extended this approach by adding an online decision-making process which decides when to stop a MP and which MP to start next. Butterfield et al. [2010] use a hierarchical Dirichlet process hidden Markov model as classification method for determining the next MP based on the sensor information and current MP. Niekum et al. [2013] segment a demonstration with a beta process auto-regressive hidden Markov model in a set of MPs and build a FSM on the sequential level. The transition behavior is learned with k-nearest neighbor classification. The focus of our work lies on incorporating several demonstrations with varying MP sequences into one

model of a task and learning the transition behavior between succeeding MPs. Basis for learning are the manually segmented and labeled sensor data traces from a set of kinesthetic demonstrations.

Learning Skills with Planning Methods

Planning methods are an extensively studied field in robotics. Traditional movement planning methods aim at finding a path which reaches a desired goal under consideration of a given set of constraints [Schwartz and Sharir, 1988] (e.g., avoiding obstacles and joint-limits). Planning is typically done offline using a given model of the environment and the robot. Depending on the problem, finding a globally optimal plan can be very complex, even when no uncertainty is taken into account. Sampling-based methods can be a means for dealing with this complexity [Elbanhawi and Simic, 2014].

In the context of sequential skill learning, planning methods can be utilized for finding a sequence of MPs which is able to perform a task. Kallmann et al. [2004] present such a planning approach. The authors assume that each MP can only be started in a subspace of the robot's configuration space. The execution of a MP then again results in a change of the robot's configuration. The planner generates a tree which connects the MPs (and their start and goal configurations) with each other until a desired goal configuration is reached. Yet, it is assumed that the MPs are given and no uncertainty is considered. An approach that explicitly takes uncertainty into account is that of Konidaris et al. [2015]. The authors present a probabilistic planning method in which the symbols are not discrete but represented using probability distributions. The low-level sensor and actuator space of an agent are described by a fully observable, continuous-state semi-Markov decision process. They evaluate their approach on a $2D$ navigation task and it is not clear how the approach would scale to more complex problems.

A generated plan is usually fixed after the optimization process. For instance, a resulting path may be comprised of a sequence of joint angles which the robot tracks when executing the task. Methods for planning under uncertainty often model the world as Partially Observable Markov Decision Process (POMDP, [Kaelbling et al., 1998]). A POMDP models the system dynamics as Markov Decision Process. The underlying state cannot be directly observed and therefore a probability distribution over the set of possible states is maintained. While being general enough to model a large variety of real-world problems such as moving dirty dishes into a dishwasher [Pajarinen and Kyrki, 2017], the complexity of the models often limits their practical applicability. In addition, solving a POMDP is NP-complete [Littman, 1996] and they usually require a lot of data for learning an optimal policy. As our aim is to learn a skill from only a few demonstrations of a task, we cannot utilize them for our approach.

A different view on task planning is taken by Dantam and Stilman [2013]. The authors adopt methods from the field of formal language theory and present Motion Grammar, a grammar for representing and generating movements. Due to the formal definition as a language, results from language and automata theory are directly applicable. For instance, it is possible to prove completeness and correctness of the language. Yet, it is not clear how the individual tokens, symbols and syntax of the language could be learned from demonstrations of a task.

Learning Task Constraints

Many tasks have sequential constraints on their subtasks. While some subtasks can be performed in an arbitrary order, others are dependent on each other. For instance, before slicing a vegetable, a knife has to be grasped first. Various approaches aim at learning or extracting such constraints from demonstrations of a task. The main difficulty is that it is not straightforward to see if a varying sequential subtask order means the ordering is arbitrary or if, for instance, the difference can be traced back to different environmental conditions.

Pardowitz et al. [2005] extract a Task Precedence Graph (TPG) from demonstrations of a task. A TPG encodes the precedence relations between the subtasks (e.g,. subtask A has to be completed before subtask B may be executed). It can be learned incrementally from labeled demonstrations. Additionally,

the authors provide a metric for measuring the similarity of subtasks. The metric is utilized to evaluate if different subtasks are used across the demonstrations. Two subtasks are considered to be equivalent if the object relations before starting the subtasks and after their completion are similar. The main downside of the approach is that it cannot handle cyclic tasks (e.g., unscrewing). Tenorth and Beetz [2012] present a software framework for extracting and modifying high-level task descriptions. The idea is to construct a task plan by reasoning about object transformations resulting from actions. The authors state that their methods is mainly inspired by two other approaches, namely STRIPS [Fikes and Nilsson, 1971] and Hierarchical Task Networks (HTN, [Erol et al., 1994]). While STRIPS is a plan language for describing the pre- and postconditions of actions, HTN can be utilized for describing the hierarchical composition of the actions. Given an incomplete action description, their method aims at detecting and filling the knowledge gaps of the description.

Hayes and Scassellati [2014] present an approach where an initial graph structure is learned from demonstrations. This graph structure represents potential subtask orders and can subsequently be refined via active learning. Based on a query strategy, the system asks for additional user input where desired. For example, the system may ask the teacher if sugar may be added before adding butter in a cake recipe. The teacher then can respond positively or negatively to such a query, resulting in a change in the graph structure by adding or removing edges between the subtasks. Additionally, a case-study is presented where the authors investigate whether it is preferable to let a single teacher demonstrate the same task multiple times or if it is beneficial to have multiple teachers. Their main finding is that in order to capture more variety of a task, multiple teachers are beneficial.

Task constraints are an intuitive way of describing the relations between the individual subtasks. These relations can often be parsed into a task plan in a standard language such as the Planning Domain Definition Language (PDDL, [McDermott et al., 1998]). As such, these methods are closely related to the approaches presented in the previous task planning section. Still, most approaches in this domain rely on predefined assumptions about the tasks. If these assumptions do not fully apply, they are likely to bias the system towards suboptimal decisions. Task learning is often performed on a high level of abstraction and it is not clear how to connect symbolic task descriptions with the lower level movement generation. The approach presented in this chapter aims at learning the coordination of a set of lower-level MPs based on the perceptual input of the system. Therefore, it can be seen as a means for connecting a high-level task representation with the low-level control signals.

2.1.2 Overview of our Proposed Approach

The aim of our approach is to learn the coordination of a set of MPs, in order to realize complex sequential movement behavior. For mastering a task, our system has to activate a set of MPs in the correct order and has to decide when to stop the current MP and which MP to start next. This decision is made based on a feature state $\boldsymbol{f}$, which is comprised of the state of the robot and the environment. Therefore, learning a skill requires finding a mapping from this feature state to a MP activation vector $\boldsymbol{a}$ which indicates the active MP. As we concentrate on the sequencing aspect in this chapter of the thesis, we assume the demonstrations are already labeled with the active MPs over time. A demonstration of length N is therefore comprised of the triplets $\tau = \{(\boldsymbol{x}_1, \boldsymbol{f}_1, \boldsymbol{a}_1), (\boldsymbol{x}_2, \boldsymbol{f}_2, \boldsymbol{a}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{f}_N, \boldsymbol{a}_N)\} = \{(\boldsymbol{x}_i, \boldsymbol{f}_i, \boldsymbol{a}_i)_{i=1:N}\}$. Here, $\boldsymbol{x}_i$ is the robot state in task-space coordinates, $\boldsymbol{f}_i$ is the feature state and $\boldsymbol{a}_i$ is the MP activation vector. Similar to most other approaches, the transition behavior between MPs is considered to be discrete in this chapter. Therefore, only one MP is active at a time and given K MPs, $\boldsymbol{a}_i$ is a K-dimensional unit vector indicating the active MP.

The feature state $\boldsymbol{f}_i$ can be comprised of arbitrary features such as positions and orientations of objects or the state of the end-effector of the robot. Learning a skill requires finding a mapping from the current feature state to the correct active MP. A straightforward way of applying machine learning methods to this problem would be to train a single classifier h with the labeled demonstrations, resulting in a function $h(\boldsymbol{f}_i) \rightarrow \boldsymbol{a}_{i+1}$. The skill could be subsequently reproduced by choosing the classification result for the

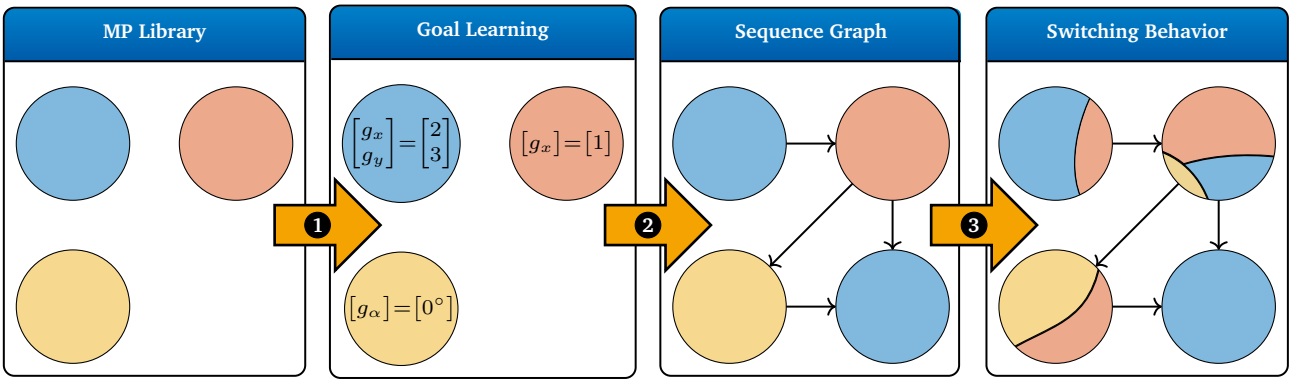


Figure 2.2.: Overview of the learning approach. The approach starts with a predefined movement primitive (MP) library. All parameters of each MP in the library (such as the coordinate frames) are known, but the attractor goals such as a position x/y or an angle α are not. From the kinesthetic demonstrations, first these goals ❶ are learned. Next, a graph representation of the demonstrated sequences ❷ is built. We call the graph representation sequence graph. A sequence graph determines possible successors of a MP during reproduction. Finally, the transition behavior is learned by training one local classifier for each node in the graph ❸.

current feature values as next activated MP for time-step $i+1$. Nevertheless, complex skills involve many different MPs and it is often not possible to find a unique mapping from feature state to MP activations. Due to this perceptual aliasing, the classification may yield unsatisfying results. One remedy for this issue would be to consider also the history of features and MP activations for the prediction. Training an out of the box time-series classifier $h(\boldsymbol{f}_{1:i}, \boldsymbol{a}_{1:i}) \rightarrow \boldsymbol{a}_{i+1}$ such as a Recurrent Neural Network with the additional information is theoretically possible. Yet, these methods are data-intensive and often suffer from over-fitting when being trained on only a few demonstrations. Therefore, we take a different approach. Instead of training a single classifier, we first build a graph structure where a node corresponds to the activation of a single MP. Each node j in the graph is connected to a classifier h_j whose task is to decide when to transition to a succeeding node in the graph, leading to the activation of a different MP. Doing so, the overall classification problem is split into smaller problems which are easier to solve. In addition, the feature set is not global. Instead, features are assigned to the individual MPs. The graph structure allows for training each classifier only on the features assigned to the connected MPs in the graph which further reduces the perceptual aliasing problem, $h_j(\boldsymbol{f}_i^{(j)}) \rightarrow \boldsymbol{a}_{i+1}$.

In this chapter, we assume a predefined set of K MPs is given, whereas the individual MPs are denoted with index k. A MP is a dynamical system (DS) with a linear attractor behavior

$$\ddot{\boldsymbol{x}}^{(k)} = \alpha \left(\beta(\boldsymbol{g}^{(k)} - \boldsymbol{x}^{(k)}) - \dot{\boldsymbol{x}}^{(k)} \right), \tag{2.1}$$

where α and β are controller parameters. Each MP has a goal $\boldsymbol{g}^{(k)}$ in task space coordinates $\boldsymbol{x}^{(k)} \subseteq \boldsymbol{x}$ that should be reached if it is activated. In theory, a goal can be a desired position of a robot body, joint angle, force or a combination thereof and can be defined relative between bodies using coordinate frames. Throughout this chapter, a single MP will always control the position/force and orientation of the robot's end-effector (in world coordinates or relative to an object in the scene) as well as the joint angles of the robot's hand. MPs may be terminated before their goal is reached, for example, if a sensor reading indicates to the system that an obstacle is close to the robot. All parameters of each MP in the library (such as the coordinate frames) are known, but the attractor goals are not. A description of the underlying controller framework is provided in the next section. Please note, however, that the sequencing method (graph structure and classifiers) is kept general and should be applicable to arbitrary MP frameworks and feature sets.

Our proposed approach consists of three stages, as depicted in Figure 2.2. In the first stage, the attractor goals of the individual MPs are learned from the demonstrations (Section 2.2). In the second

stage, a representation of the demonstrated sequences is constructed by connecting the observed MPs in a graph (Section 2.3). Each node in the graph corresponds to a MP and each transition leads to a potentially succeeding MP. In the final stage, the MP transition behavior is learned (Section 2.4). One classifier is linked to each node in the graph. The task of the classifier is to decide when to transition to a new state in the graph during the reproduction of a skill, resulting in an activation of a different MP. An experimental validation of the approach is presented in Section 2.5, followed by a conclusion of the chapter in Section 2.6.

2.1.3 Utilized Controller Framework

For controlling the robot, we use a hybrid position-force controller based on task-level inverse dynamics. The controller as well as the utilized MP representation were initially presented by Luksch et al. [2012] and also used by Kober et al. [2015]. The (desired) joint torque $\boldsymbol{T}$ is given by

$$\begin{aligned}\boldsymbol{T} = \boldsymbol{M}\boldsymbol{J}^{\#}\boldsymbol{S}\left(\boldsymbol{e}_x - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) + \boldsymbol{J}^T(\boldsymbol{I}-\boldsymbol{S})\boldsymbol{e}_f \\ + \boldsymbol{M}\boldsymbol{J}^{\#}(\boldsymbol{I}-\boldsymbol{S})\left(\boldsymbol{e}_d - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}\right) - \boldsymbol{M}\left(\boldsymbol{I}-\boldsymbol{J}^{\#}\boldsymbol{J}\right)\boldsymbol{\xi} + \boldsymbol{g} + \boldsymbol{h}.\end{aligned} \tag{2.2}$$

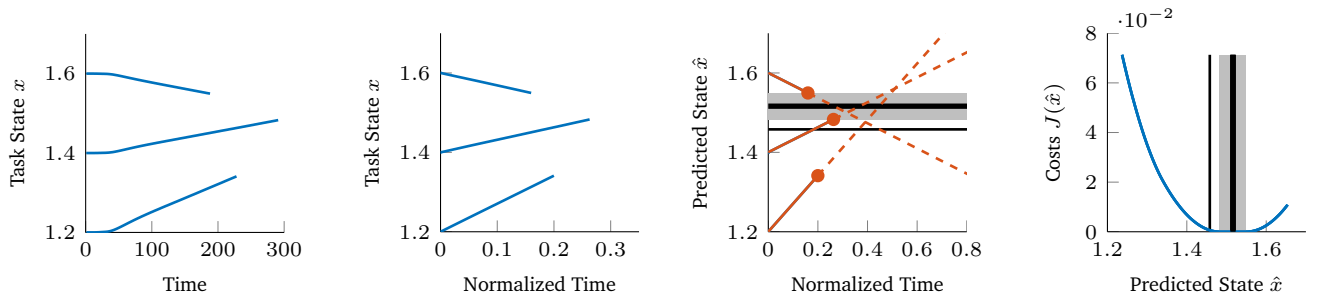
Here, $\boldsymbol{J}$ is the task Jacobian and $\boldsymbol{J}^{\#}$ its pseudo-inverse. Vector $\boldsymbol{\xi}$ accounts for joint speed damping and joint limit avoidance and is projected into the null space of the movement. $\boldsymbol{M}$, $\boldsymbol{h}$ and $\boldsymbol{g}$ denote the mass matrix, Coriolis forces and gravity, respectively. The MPs enter the equation via $\boldsymbol{e}_f$ and $\boldsymbol{e}_x$. Vector $\boldsymbol{e}_f$ contains the concatenated desired forces of the individual MPs. Vector $\boldsymbol{e}_x$ is the output of a PID controller which tracks the desired task-space accelerations of all MPs. If a MP is composed of task-variables controlling force and position of a robot body (e.g., Cartesian x and y position of end-effector and force along z component in a certain coordinate frame), the individual variables are split up and assigned to $\boldsymbol{e}_f$ or $\boldsymbol{e}_x$ according to what they control. $\boldsymbol{S}$ is a diagonal selection matrix which enables selecting either kinematic or force components of a task variable and $\boldsymbol{e}_d$ is a task-level damping term. Based on the activations of the individual MPs, the pseudo-inverse is scaled with a weighting matrix. In theory, the weighting matrix allows for continuously modulating the contributions of the individual MPs, but in this chapters only one MP is active at the same time. As a consequence, only the activated MP influences the robot's movement, while the desired paths of the other MPs are ignored.

In this thesis, all real robot experiments are conducted using a Barrett WAM robot. The robot has seven degrees of freedom (DoF) and is equipped with a three fingers hand as end-effector. The hand has four DoF. In all chapters, we assume a controller such as (2.2) is given and allows for controlling the robot in task-space. In order to transfer the presented approaches also to other robots, one would have to implement such a controller first. Additionally, the robot should allow for kinesthetic teaching. If a task does not require a robot to apply forces, also other teaching methods would be conceivable (e.g., a data glove if the robot has five fingers).

2.2 Learning Movement Primitive Parameters

In this chapter, we assume that most MP parameters are known beforehand. For instance, the number of MPs and their coordinate frames are known, but their attractor goals are not. By learning these goals from the demonstrations, the teacher has some freedom when demonstrating a task. For instance, if the task is to grasp an object, he or she could grasp it from the top or the left. If the attractor goals were predefined, the teacher would have to closely follow an expected behavior, which is difficult as the goals cannot be seen during the teaching process.

As we assume the demonstrations are labeled with the active MP over time, we can split the demonstrations into segments and assign the segments to the individual MPs. Given the segments, each MPs attractor goal can be learned independently. In the following, we will therefore show how the attractor



(a) Trajectory **(b)** Normalization **(c)** Approximation and Prediction **(d)** Goal Learning

Figure 2.3.: Goal learning overview. The trajectories (a) are velocity normalized (b) and approximated by linear functions (solid red lines, c). For each trajectory, the goal is expected to lie on its future path (red dashed lines), which is predicted using the linear functions. The goal (thick black line) is then found by minimizing the cost function (d), which arbitrates between all expected goals. Note that the cost function is zero for the entire gray area. Therefore the center of the interval is chosen as goal of the MP. The thin black line shows the mean of the trajectory end points for a comparison.

goal of one individual MP can be learned. We assume that the MP has been active M times, resulting in M segments

$$\tau_m = \left\{ \left(\boldsymbol{x}_i^{(m)}, \boldsymbol{f}_i^{(m)}, \boldsymbol{a}_i^{(m)} \right)_{i=1:N_m} \right\}, \tag{2.3}$$

with $m = \{1, \ldots, M\}$ and varying length N_m. The attractor goal $\boldsymbol{g}$ of the MP is learned from the task-space data $\boldsymbol{x}_{1:N_m}^{(m)}$ of the segments assigned to it. Note that task state $\boldsymbol{x}$ and feature state $\boldsymbol{f}$ are different. The task state represents the state controlled by the system and is used for learning the MP goals. The feature state instead is decoupled from the controller and can represent anything, such as the state of the environment. It will be used for learning the transition behavior in Section 2.4.

Even when segmenting and labeling demonstrations manually, extracting the goals is often not straightforward. One reason is that the same movements do not always reach the same goal. For example, in a reaching movement that has to be stopped due to a collision with an obstacle, the segment's end point can be far away from the movement's goal. If the attractor goals are not known beforehand, it is often not clear from the demonstrations whether a movement was stopped as the goal was reached or for any other reason. Hence, a simple solution such as taking the mean of all segment end points is often insufficient for learning the goals. In general, there are two possible ways of handling such incomplete movements.

1. Ignore them for goal learning.
2. Predict the goal regardless of completeness.

The first approach requires a method for detecting incomplete movements, e.g., by clustering the end points of the segments and detecting outliers. Clustering usually requires a lot of data. Our goal is to learn from as few demonstrations as possible, as we do not want to overload the demonstrator by requiring tens of demonstrations. Therefore, we take the more sample efficient second approach and predict the goal without detecting and discarding incomplete movements. We argue that even if a movement is stopped prematurely, the movement up to that point still roughly points towards the MP's attractor goal. Our approach utilizes this information as follows. First, all segments are approximated by linear functions of time. For each segment, the goal is expected to lie on its future path, which is predicted using the linear functions. The goal is subsequently found by arbitrating between all expected goals and finding the best compromise between them. In the following sections, we will first show how

a segment is approximated and then how the goal is learned. An overview of the goal learning is shown in Figure 2.3.

2.2.1 Approximating Segments with Linear Functions

Within our system, each dimension of the goal $\boldsymbol{g}$ is learned independently. We have observed that one-dimensional goal learning is reflecting the behavior of a human teacher more naturally. As an example, consider a teacher moving a gravity compensated robot end-effector to a desired 3D-position. The teacher will start by approaching the goal position. Due to an imperfect movement, not all dimensions of the goal position will be reached at the same time. Instead, the teacher may recognize that the desired x-position is already reached, but y- and z-position are not. Therefore, while trying to reach the desired overall position, the end-effector is moved along the yz-plane while the x-position is kept constant. The resulting trajectory will differ from the desired linear attractor behavior of the robot. For a real demonstration, the teacher also has to control the orientation of the end-effector and the hand of the robot. As it is difficult to reach the desired goal state all at once, humans seem to concentrate on a few dimensions first. Therefore, demonstrations usually do not really match the attractor behavior of a real robot movement. To compensate for this mismatch, we learn the goal per dimension. In the following, we therefore use a scalar notation for the task space. We will also drop the m index for indicating the segment, as every segment is approximated in the same way.

We focus on tasks where the velocity of a movement is irrelevant. Therefore, the time does not correspond to the real time axis of the demonstrations, but is computed by normalizing the velocity

$$t_i = t_{i-1} + (x_i - x_{i-1})^2, \quad t_1 = 0, \tag{2.4}$$

as illustrated in the two left plots in Figure 2.3. As model for the approximation of a segment, a simple linear function is used

$$r(t_i) = \alpha t_i + \beta = \hat{x}_i. \tag{2.5}$$

Here, $\hat{x}_i$ is the predicted state of the MP at time t_i and α and β are the parameters of the model. Although a linear function is a simple model, it is notable here that it matches the demonstrations of single attractor movements quite well, as they can be seen as point-to-point movements in task space. We focus on tasks that have such linear characteristics, e.g., pick-and-place tasks. The assumption of a linear model usually does not apply for the transition between two MPs. During demonstration and reproduction of a sequence, a transition can occur with a non-zero velocity. As a consequence, the start of a MP may be influenced by its predecessor. The resulting segment will contain arcs or edges and is an example for a segment that cannot be well represented using a straight line. Nevertheless, note that there is no need to approximate the whole segment well in our approach. Instead, the line only has to pass through the real goal at some future time point. The method has to find the parameters of Equation (2.5) to ensure this property.

Therefore, we chose to use weighted least squares regression (WLSR, Carroll and Ruppert [1988]) for learning the parameters α and β. Compared to least squares regression, WLSR additionally allows for weighting the importance of each data point. By weighting the data points at the end of a segment stronger than at the beginning, it is possible to minimize the influence of the preceding movement, while still matching the overall segment well. The parameters are found by minimizing the weighted sum of squared differences between the sampled states x_i and predicted states $\tilde{x}_i$ from Equation (2.5), resulting in the cost function

$$J(\alpha, \beta) = \frac{1}{2} \sum_{i=1}^{N} w_i \, (x_i - \hat{x}_i)^2 . \tag{2.6}$$

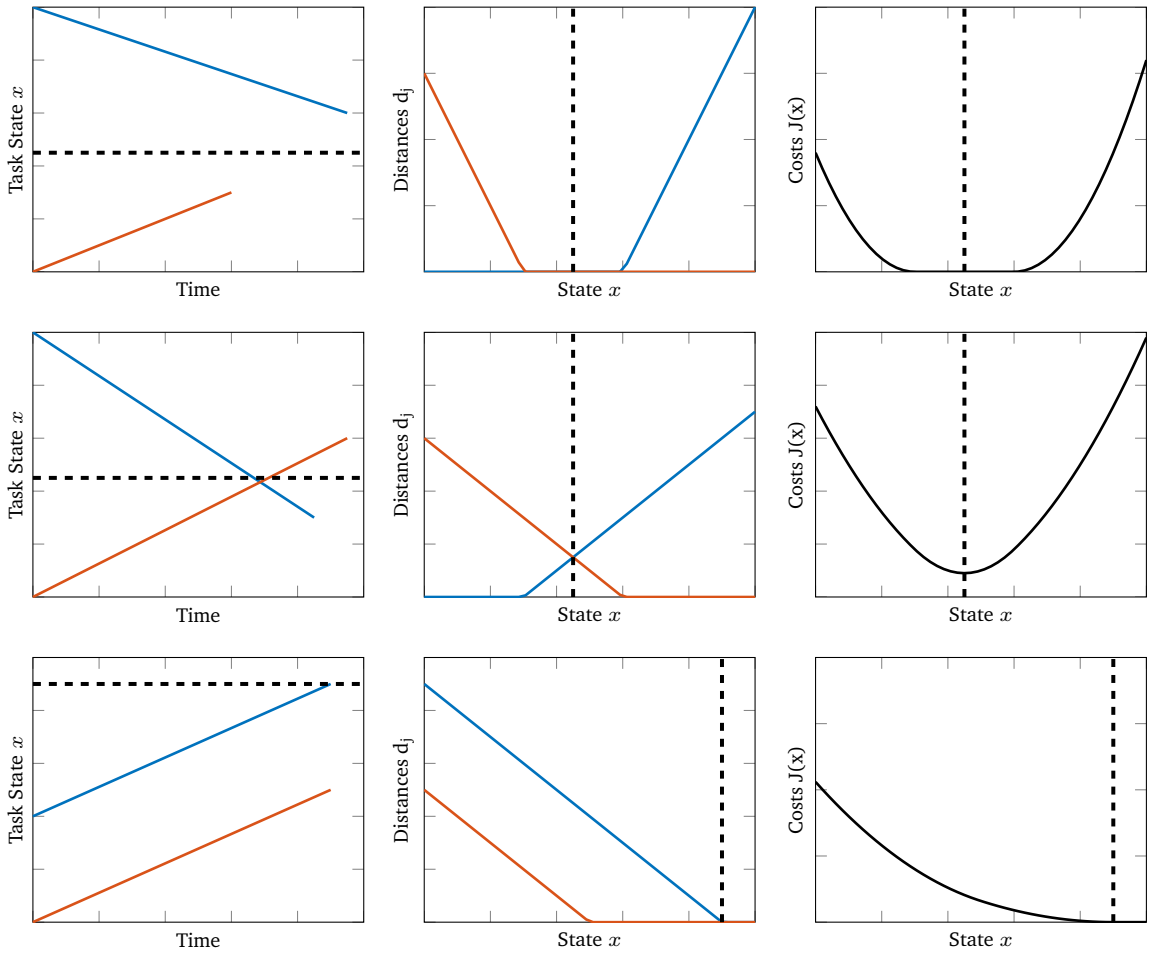


Figure 2.4.: Three goal learning cases and their resulting distance and cost functions. In general, segments may converge (top), diverge (center) or point in the same direction (bottom). The black dashed lines show the goal of the MP.

Here, N is the number of samples and w_i are the weights. We suggest to use $w_i = i^2/(\sum_{i=1}^{N} |w_i|)$, as the quadratic weights minimize the influence of the preceding MP at the start of a segment and focus on the data points closer to the goal. Cubic or even larger weights focus on few data points at the end of a segment and therefore become sensitive to noise. Minimizing the error function (2.6) is straightforward (see Carroll and Ruppert [1988]) and leads to

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \boldsymbol{A}^{\#}\boldsymbol{b}, \quad \boldsymbol{A} = \sum_{i=1}^{N} w_i \begin{bmatrix} t_i^2 & t_i \\ t_i & 1 \end{bmatrix}, \quad \boldsymbol{b} = \sum_{i=1}^{N} w_i x_i \begin{bmatrix} t_i \\ 1 \end{bmatrix}, \tag{2.7}$$

where $\boldsymbol{A}^{\#}$ is the left pseudo-inverse of $\boldsymbol{A}$.

2.2.2 Goal Learning

Approximating each of the M segments results in M linear functions $r_m(t) = \alpha_m t + \beta_m$ with $m = \{1, \ldots, M\}$ and the normalized time steps $t_{1:N_m}^{(m)}$. Figure 2.3(c-d) shows an overview of our goal learning approach. As already mentioned, the basic idea is that for each segment m, the goal is expected to lie on its future path. The future path is predicted using the linear function r_m, which allows us to formulate

the expected goal in terms of the slope parameter α_m and the predicted state u_m at the final time of the trajectory

$$u_m = r_m(t_{N_m}) = \alpha_m t_{N_m} + \beta_m. \tag{2.8}$$

If the slope is positive, the expected goal is equal or greater than u_m. If it is negative, the expected goal is equal or less than u_m. For finding the best compromise between all trajectories, we construct a cost function which penalizes deviations from each expected goal. As a first step, we define the distance between a state x and the expected goal of a segment m as

$$d_m(x) = \begin{cases} 0, & \text{if } a_m > 0 \text{ and } x \geq u_m, \\ 0, & \text{if } a_m < 0 \text{ and } x \leq u_m, \\ |x - u_m|, & \text{otherwise.} \end{cases} \tag{2.9}$$

Then, the attractor goal of one particular dimension of a MP can be defined as the point g where the squared sum of distances becomes minimal

$$J(x) = \sum_{m=1}^{M} d_m^2(x), \tag{2.10}$$

$$g = \min_x J(x). \tag{2.11}$$

Figure 2.4 shows some trajectories and their resulting distance and cost functions as an example. For finding the solution g, we first calculate the derivative of the cost function (2.10). Due to Equation (2.9), the cost function is non-differentiable at each threshold u_m. Therefore, we sort the intervals in ascending or descending order and compute the derivative for each interval $[u_m, u_{m+1}]$ separately. The derivative is given by

$$\frac{d}{dx} J(x) = 2 \sum_{m \in \mathbb{D}} (x - u_m). \tag{2.12}$$

Here, $\mathbb{D}$ is the set of functions for which Equation (2.9) is non-zero. Setting the derivative equal to zero and rearranging for s results in

$$g = \frac{1}{|\mathbb{D}|} \sum_{m \in \mathbb{D}} u_m, \tag{2.13}$$

where $|\mathbb{D}|$ is the number of elements in $\mathbb{D}$. The solution g may lie outside of the interval. In that case, it is clipped to the closest interval border. If there exists an interval for which $\mathbb{D}$ is empty, the error will become zero and hence any value in this interval is a possible solution. In that case, the center of the interval is chosen as goal. The final equation therefore is

$$g = \begin{cases} (u_m + u_{m+1})/2 & \text{if } |\mathbb{D}| = 0, \\ u_m & \text{if } |\mathbb{D}| \neq 0,\ g < u_m, \\ u_{m+1} & \text{if } |\mathbb{D}| \neq 0,\ g > u_{m+1}, \\ g & \text{otherwise.} \end{cases} \tag{2.14}$$

The goal g is computed for every interval $[u_m, u_{m+1}]$ and subsequently inserted into Equation (2.10). The goal resulting in the lowest value of this cost function is then chosen as final goal of the MP.

Due to the quadratic dependency of Equation (2.10) on the expected goals, overshoots may shift the goal away from the desired value. However, our experience is that if a teacher recognizes that the goal was not hit accurately, he/she usually corrects his/her mistake, so that in the end the real goal is approximately reached. If a MP is stopped prematurely, overshoots also do not lead to problems. Additionally, the trajectory approximation with WLSR leads to some robustness against overshoots.

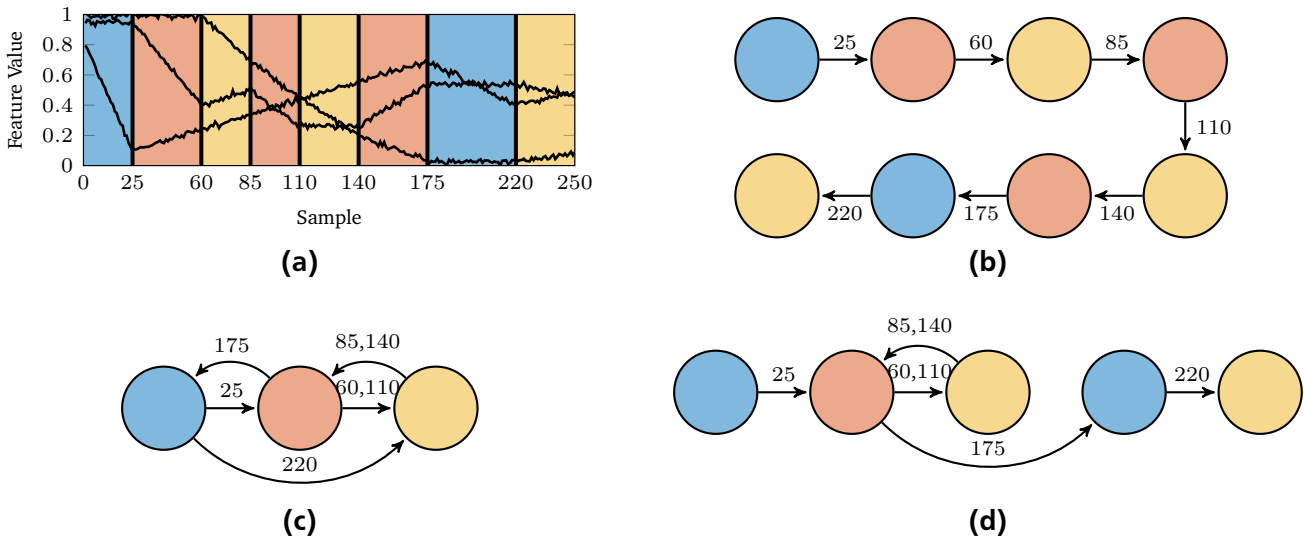


Figure 2.5.: Overview of the graph generation. First, the labeled data from a set of demonstrations (a) is taken to extract the MP sequence (b). This sequence can then be used to generate a sequence graph. We investigate two different sequence graph types. A compact local sequence graph (c) and a more sophisticated global sequence graph (d). The numbers on the transitions correspond to the transitions points (TPs, see upper left figure). A TP is a point in time at which a transition between MPs occurs.

2.3 Sequence Graph Generation

In the previous section, the parameters of the individual MPs have been acquired. In this section, we propose an approach for generating a graph representation of the demonstrated sequences which we call sequence graph. In a sequence graph, every node is linked to a MP. During reproduction, the graph determines which MP may be activated next. A missing transition in the graph might prevent an activation of the correct MP, while too many outgoing transitions might lead to the activation of a wrong MP due to perceptual aliasing. It is therefore crucial to find a good structure for a given set of demonstrations.

Figure 2.5 shows an overview of the graph generation based on a simple toy example with only three different MPs, that will be used throughout this section. The MPs are indicated by different colors. They are chosen arbitrarily and have no further meaning, but show the essential characteristics of our approach. First, we perform at least one kinesthetic demonstration. In general, we assume that M demonstrations have been collected. For each demonstration, we get a labeled (background colors in Figure 2.5a) set of features (black lines). The features are utilized for learning the transition behavior and will be explained in Section 2.4. For generating a sequence graph, only the observed MP sequence is used, as shown in Figure 2.5b. The graph representation will be explained in detail in the following section, where we also present two different types of sequence graphs, both showing different ways of incorporating the sequences into the representation.

A sequence graph is a directed graph $G = (\mathcal{N}, \boldsymbol{T})$ with a set of nodes $\mathcal{N}$ and a transition matrix $\boldsymbol{T}$ indicating the connections between the nodes. Each node N_i is linked to a MP. This mapping is not injective which means a MP can be linked to more than one node. During reproduction, a MP is activated if a linked node is considered active. Transitions in the graph lead to succeeding MPs that can be activated if the current MP has finished. A transition $T_{k,l}$ is connecting the node N_k with N_l. Each transition is linked with the corresponding transition points (TP) at which it was observed during the demonstration (black vertical lines in Figure 2.5a). As the same transition can be observed multiple times, multiple TPs are possible. Having $|\mathcal{N}|$ nodes in a graph, we use a $|\mathcal{N}| \times |\mathcal{N}|$ transition matrix $\boldsymbol{T}$ with elements $T_{k,l}$ to describe one sequence graph. As a MP is usually activated for more than one time step, the transition $T_{k,k}$ exists for all k.

In order to generate a sequence graph, we assume M demonstrations have been performed. For each demonstration m, we start by constructing one directed acyclic graph $G_m = (\mathcal{N}^{(m)}, \boldsymbol{T}^{(m)})$ from the observed MP sequences (see Figure 2.5b). The main step is now to combine multiple of these graphs into one representation of the skill, which can be a difficult problem as the algorithm has to work solely based on the observations. As an example, consider the task of baking a cake. Here, it does not matter if milk or eggs are put in the bowl first. Still, the task may be demonstrated one time with the sequence milk-eggs and one time with the sequence eggs-milk. From an algorithmic point of view it is often not clear if a sequence is arbitrary for a skill or if the differences can be linked to some traceable sensor events. Hence, there are different ways of building the graph structure for a skill. We show two possibilities by investigating two different kinds of sequence graphs. The local graph presumes a sequence to be arbitrary and is not considering it in the representation, while the global graph is trying to construct a more detailed skill description.

2.3.1 Local Sequence Graph

The local sequence graph assigns exactly one node to each activated MP and hence the number of nodes and MPs is equal. The graph is initialized with one node per MP and without transitions. For each observed pair of MPs a transition is added to the graph. As only pairs and no history are considered, it is irrelevant at which point in the sequence a transition occurs. The corresponding graph for the toy example is shown in Figure 2.5c.

The graph contains only three nodes, one for each activated MP. When reproducing the movement, a transition from the ●-MP to the ●-MP one is always possible at this level of the hierarchy and it is up to the classifier to prevent such incorrect transitions. The major drawback of this representation is the strong requirement on the feature set, as it has to be sufficiently meaningful to allow for a correct classification independent of the history of activated MPs.

2.3.2 Global Sequence Graph

The global sequence graph attempts to overcome this issue by constructing a more detailed skill description. One essential characteristic of the global sequence graph is that there is no one to one mapping between MPs and graph states. Instead, a MP can appear multiple times in one representation as depicted in the global sequence graph of the toy example (Figure 2.5d). Here, two nodes are linked to the ●-MP because the sequence was considered to be in two different states when they were activated. The repeated appearance of the ● $\rightarrow$ ● transitions (see Figure 2.5a) is represented by only two nodes as in the local graph. The reason is that consecutive sequences of the same MPs are considered to be a repetition which can be demonstrated and reproduced an arbitrary number of times. Repetitions are also advantageous when describing tasks with repetitive characteristics, such as unscrewing a light bulb. Here, the unscrewing movement has to be repeated several times depending on how firm the bulb is in the holder. As the number of repetitions is not fixed for each single demonstration, the algorithm has to conclude that different numbers of repetitions of the same behavior appeared in the demonstrations and incorporate this information into the final representation of the task.

Note that even if a skill requires a fixed number of repetitions, both presented sequence graphs will contain a cycle in the representation. The system is then only able to reproduce the movement properly if the classifier would find the transition leading out of the cycle after the correct number of repetitions. While an improvement is not possible here for the local graph, a fixed number of repetitions can be modeled with the global graph by skipping the search for cyclic transitions.

Algorithm 1 Graph Folding

Require: Initial transition matrix $\boldsymbol{T}$

$repetition = \text{findRepetition}(\boldsymbol{T})$;
while $repetition.found$ **do**
  $\boldsymbol{T}_f = \emptyset$;
  $repetition.l = repetition.end - repetition.start + 1$;
  for $i = repetition.start$ **to** $repetition.end$ **do**
    $\text{mergeNodes}(\boldsymbol{T}(i + repetition.l), \boldsymbol{T}(i))$;
    $\boldsymbol{T}_f = \boldsymbol{T}_f \cup \boldsymbol{T}(i)$;
  $repetition = \text{findRepetition}(\boldsymbol{T})$;
  if $!repetition.found$ **then**
    $tail = \text{findTail}(\boldsymbol{T}, repetition.end + 1)$;
    if $tail.found$ **then** // Found incomplete cycle
      for $i = tail.start$ **to** $tail.end$ **do**
        $\text{mergeNodes}(\boldsymbol{T}(i + r), \boldsymbol{T}(i))$;
        $\boldsymbol{T}_f = \boldsymbol{T}_f \cup \boldsymbol{T}(i)$;
  $\boldsymbol{T} = \boldsymbol{T} \setminus \boldsymbol{T}_f$; // Remove merged nodes from graph
return Final transition matrix $\boldsymbol{T}_f$

2.3.3 Graph Construction

The local sequence graph is created by adding one node for each observed MP and one transition for every observed MP pair. If a MP pair is observed multiple times, only one transition is added to the graph. For creating a global sequence graph, three steps have to be performed.

1. Create one acyclic graph $G_m = (\mathcal{N}^{(m)}, \boldsymbol{T}^{(m)})$ for each demonstration m.
2. For each graph G_m, replace repetitions of MPs with cyclic transitions (Folding).
3. Combine updated graphs to one global representation $G = (\mathcal{N}, \boldsymbol{T})$ of the skill (Merging).

The first step is straightforward as the acyclic graph represents the MP sequence directly observed in the demonstrations. We call the second point *folding* and its pseudo code is shown in Algorithm 1. The algorithm starts by calling the method findRepetition, which is searching for repetitions of length $l = \lfloor |\mathcal{N}^{(m)}|/2 \rfloor$ in a graph G_m with $|\mathcal{N}^{(m)}|$ nodes. The method starts by comparing the MPs of the nodes $\{N_0^{(m)}, N_1^{(m)}, \dots, N_l^{(m)}\}$ with $\{N_{l+1}^{(m)}, \dots, N_{2l+1}^{(m)}\}$. If both node chains match, the node pairs $\{N_0^{(m)}, N_{l+1}^{(m)}\} \dots \{N_l^{(m)}, N_{2l+1}^{(m)}\}$ are returned. If the chains do not match, the indices are incremented by one and the method starts from the beginning with node $N_1^{(m)}$ as starting point. The shifting process continues until the end of the list is reached. Next, l is decremented by one and all previous steps are repeated. Thus, longer repetitions are preferred over shorter ones. The method terminates if the cycle size is one, which means no more cycles can be found.

If a repetition is found, the corresponding nodes are merged to a single node. When merging two nodes N_A and N_B, the input and output transitions of node N_B become input and output transitions of N_A. If an equal transition already exists for N_A, only the associated TPs are added to the existing transition. Note that a cyclic transition is introduced when merging the nodes $N_0^{(m)}$ and $N_{l+1}^{(m)}$, as the input transition $T_{l,l+1}^{(m)}$ is being rerouted to $T_{l,0}^{(m)}$. After each iteration of the algorithm, the nodes of the latter chain are not connected to the rest of the graph anymore and can be removed from the representation. To allow escaping a cycle not only at the end of a repetition, the algorithm also searches for an incomplete cycle after a found repetition. This tail is considered to be part of the cycle and is also

Algorithm 2 Graph Merging

Require: $G_a = (\mathcal{V}^{(a)}, \boldsymbol{T}^{(a)})$ and $G_b = (\mathcal{V}^{(b)}, \boldsymbol{T}^{(b)})$

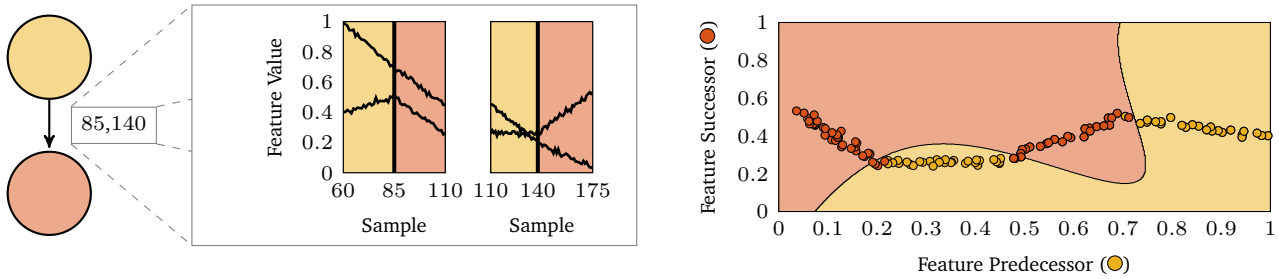
1: $\boldsymbol{\mathcal{U}}^{(a)} = \text{getUniquePaths}(\boldsymbol{T}^{(a)})$; // Set of unique paths
2: $\boldsymbol{\mathcal{U}}^{(b)} = \text{getUniquePaths}(\boldsymbol{T}^{(b)})$;
3: **for all** $\mathcal{U}^{(b)} \in \boldsymbol{\mathcal{U}}^{(b)}$ **do** // Iterate over each unique path
4: $c_{\max} = 0$;
5: **for all** $\mathcal{U}^{(a)} \in \boldsymbol{\mathcal{U}}^{(a)}$ **do**
6: $c = \text{compare}(\mathcal{U}^{(a)}, \mathcal{U}^{(b)})$; // Compute number of matching nodes
7: **if** $c > c_{\max}$ **then**
8: $c_{\max} = c$;
9: $\mathcal{V}^{(b,\max)} = \mathcal{U}^{(b)}_{1:c}$; // First c nodes
10: $\mathcal{V}^{(a,\max)} = \mathcal{U}^{(a)}_{1:c}$;
11: **for all** $V^{(a)} \in \mathcal{V}^{(a)}, V^{(b)} \in \mathcal{V}^{(b)}$ **do** // Iterate over all nodes
12: **if** $V^{(b)} \in \mathcal{V}^{(b,\max)}$ **then** // Nodes match
13: $\text{mergeNodes}(V^{(a)}, V^{(b)})$;
14: **else** // Node $V^{(b)}$ has to be added to G_a
15: $\text{addNode}(G_a, V^{(b)})$;
16: **return** merged graph G_a

merged into the cyclic structure (Algorithm 1, lines 11-15). The toy example also contains an incomplete cycle, as the ● → ● repetitions end incompletely with the ●-MP.

We call the final step of creating a global sequence graph *merging*, as several separate graphs are merged into one representation. Algorithm 2 merges two graphs and thus gets called $M - 1$ times for M demonstrations. The algorithm steps through the graphs simultaneously, starting at the initial nodes, merging equal nodes and introducing branches whenever nodes differ. The algorithm starts by extracting the unique paths from both graphs. A unique path is a path which starts with a node that has no input transition and ends with a node that has either no output transition or only output transitions to nodes that were already visited. The toy example has two unique paths, ● → ● → ● and ● → ● → ● → ● (see Figure 2.5d). Next, the algorithm compares the paths of both graphs with each other from left to right, searching for the longest equal subpath. Two nodes are considered as equal if the columns of the corresponding transition matrices are equal, which means both nodes use the same underlying MP and have the same input transitions. Finally, the nodes of the longest equal subpath are merged, whereas all other nodes of graph $\boldsymbol{T}_{\text{B}}$ are added to $\boldsymbol{T}_{\text{A}}$. By searching for the longest subpath, branches are introduced at the latest possible point in the combined graph. Once branched, both branches are separated and do not get merged together at a later point in the sequence.

2.4 Learning the Transition Behavior

After creating the graph representation, the next step is to train the classifiers – one for each node in the graph. If a node is active during reproduction, its associated classifier decides when to transition to a possible successor node. This multiclass classification problem has the active node and all of its neighbor nodes in the graph as classes. Hence, the classifier corresponding to node N_j can be defined as $h_j(\boldsymbol{f}_t^{(j)}) \rightarrow \boldsymbol{a}_{t+1}$. The feature vector $\boldsymbol{f}_t^{(j)}$ is composed of the features assigned to the MPs of node N_j and its successors in the graph. The activations of all MPs which are not linked to one of these nodes are constrained to be zero. Restricting the number of classes often increases the accuracy of the system as transitions not observed in the training data are prevented. A reduction of the feature vector can be seen as an implicit dimensionality reduction where unimportant features used by uninvolved MPs are no



(a) **(b)**

Figure 2.6.: One classifier is created for each node in the graph. The training is performed by using the data around the transitions between the connected MPs (a). The data is projected into feature space and the classifier learns a separating border between the MPs, as indicated by the background color in (b). As soon as the current feature state crosses such a border during reproduction, a transition in the graph is triggered, leading to the activation of a different MP.

longer considered for the decision. In this chapter, MPs are exclusively activated. Therefore, the MP to be activated at time t is the one with the largest activation $\arg\max_j \boldsymbol{a}_t$.

Figure 2.6a depicts the data used for training a classifier as an example. After the demonstrations, each transition in the acyclic graph is linked to one TP in the sampled data. During the merging and folding process of the global sequence graph or the pair search for the local graph transitions are merged together, resulting in potentially multiple TPs for each transition. For each TP, the data points between the previous and next TP in the overall data are taken from the training and labeled with the MP that was active during that time. As all transitions have the same predecessor for one classifier, the first part of the data will always have the same labels, while the second part may differ depending on the successor node of the transition.

The classifiers learn from the labeled training data how to separate the MPs in feature space (Figure 2.6b). During the reproduction of the MP, the current feature state of the robot is tracked and as soon as it crosses a border between the classes, the corresponding successor will be activated, leading to a transition in the graph and a switch to another classifier. Any classifier is applicable to our method. We evaluated support vector machines (SVMs, [Cortes and Vapnik, 1995]), logistic regression (LR, [Bishop, 2006]), kernel logistic regression (KLR, [Cortes and Vapnik, 1995]), import vector machines (IVMs, a certain type of sparse kernel logistic regression, [Zhu and Hastie, 2001]), and Gaussian mixture models (GMMs, [Bishop, 2006]).

2.5 Evaluations and Experiments

For evaluating our approach, we performed three different experiments. In Section 2.5.1, we evaluate the goal learning on a task where a robot has to move an object in its workspace. In Section 2.5.2, we evaluate the overall performance of the system, including the two sequence graph representations and different classifiers. In the experiment, a robot has to unscrew a light bulb. In Section 2.5.3, the system has to learn to grasp different objects. Additionally, an error recovery strategy for unsuccessful grasps is demonstrated. With the third experiment, we evaluate the system performance on a more complex feature set. All experiments are evaluated using a real seven degrees of freedom (DoF) Barrett WAM robot with an attached four DoF hand. For the first and third experiment, also some simulation results are presented.

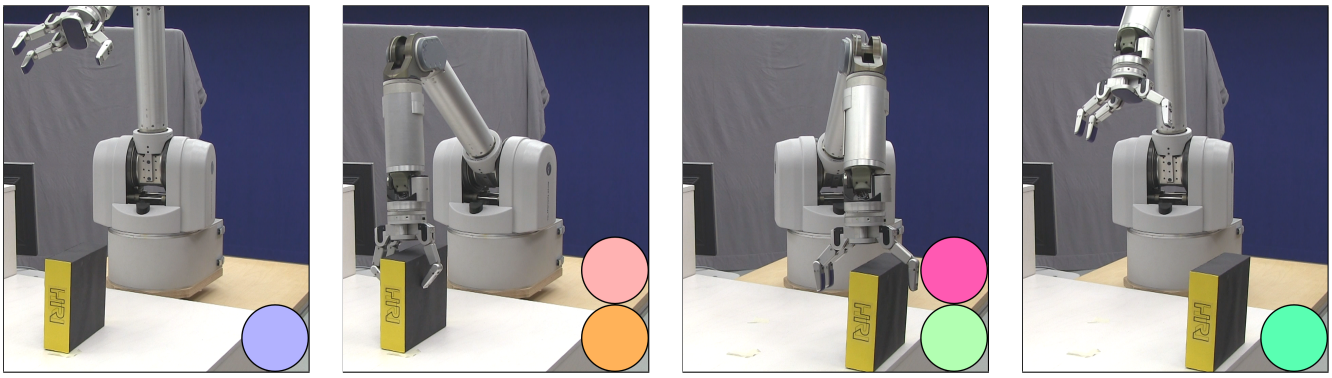


Figure 2.7.: In the first experiment, the robot has to move an object to a certain position. The robot starts in an initial position (●), moves to the object (●) and grasps it (●). Subsequently, it moves the object to another position (●), opens its hand (●) and moves to the final position (●).

2.5.1 Moving an Object

The aim of the first experiment was to evaluate the goal learning algorithm we describe in Section 2.2. Therefore, we chose a rather simple sequence of movements, which does not differ between individual demonstrations. The task is to start in an initial position, move to an object and grasp it. Subsequently, the object has to be moved to another position, it has to be released and finally, it is required to move back to the initial position (see Figure 2.7 for an illustration). As the sequence of movements always is the same and does not include any repetitions or specific patterns of movements, the local and global sequence graph algorithms return the same structure. A MP may control the position and/or orientation of the end-effector as well as the joint angles of the fingers. If an entity is not controlled by a MP, the movement results from the null space criteria (e.g., joint limit avoidance) of the underlying task space controller. Six different MPs have been defined to perform the task, as shown in Table 2.1.

Before the experiments on the real robot are presented, the goal learning is evaluated in simulation first. As kinesthetic teaching is not possible in simulation, we predefine the demonstrated sequence with a state machine and the transition behavior between MPs using thresholds. For realism and variation, Gaussian noise is added to the thresholds. Every time a new MP is activated, new thresholds are computed. The attractor goal of each MP k is set to desired values $\boldsymbol{g}^{(k)}$ and perturbed with additive noise $\mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ for each demonstration. The intention of this experiment is to evaluate the robustness and accuracy of the goal learning by disturbing the transition behavior and MP goals. We perform eight demonstrations with a fixed σ and learn the goals $\hat{\boldsymbol{g}}^{(k)}$ of every MP k after each demonstration with the data of all demonstrations that have been performed up to this point. For each learning instance, an error is computed according to

$$e = \frac{1}{K} \sum_{k=1}^{K} \left\| \hat{\boldsymbol{g}}^{(k)} - \boldsymbol{g}^{(k)} \right\|_1 , \tag{2.15}$$

MP	Position	Orientation	Fingers	Next MP
●	Initial	-	-	●
●	A	Fixed	Open	●
●	Hold	Fixed	Closed	●
●	B	Fixed	Closed	●
●	Hold	Fixed	Open	●
●	Final	-	-	-

Table 2.1.: MPs for the object movement experiment.

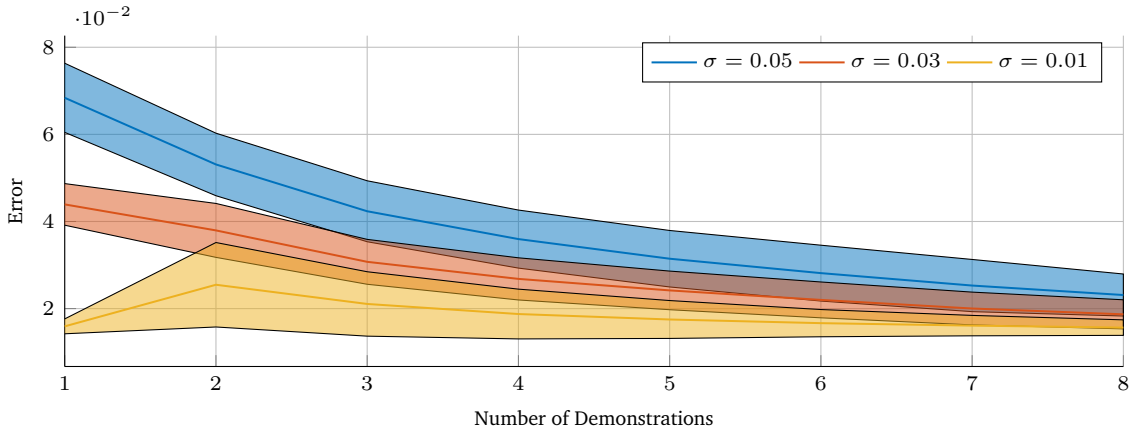


Figure 2.8.: Goal learning error for the simulated toy example. The error degrades when demonstrating a task multiple times. The background color shows the hull of one standard deviation.

which is the mean of the ℓ_1-norm of the difference between predefined and learned attractor goals of all K MPs. The experiment is repeated with three different values for σ 20 times, so that in total 480 demonstrations are performed. We summarize the results for all learning instances according to the number of demonstrations they have been trained with and compute the mean and standard deviations of the errors. The results are plotted in Figure 2.8. In general, the error decreases slightly for more demonstrations, but is always consistent with the amount of noise added to the system.

For the experiments with the real robot, we performed three kinesthetic demonstrations. In order to label the demonstrations, we indicated a movement as complete by pressing a key. Opening and closing of the hand was also initiated by pressing a key. The labeling is performed based on the indicated transitions. Next, the transition behavior was learned with SVMs as classifier and the task was reproduced on the real robot. A one-dimensional feature $f_t^{(k)}$ is assigned to each MP k according to the equation

$$\boldsymbol{\Delta}_t = \boldsymbol{g}^{(k)} - \boldsymbol{x}_t^{(k)}, \tag{2.16}$$

$$f_t^{(k)} = 1 - \exp(-\frac{1}{2}(\boldsymbol{\Delta}_t^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\Delta}_t)). \tag{2.17}$$

Here, $\boldsymbol{x}_t^{(k)} \in \mathbb{R}^{11\times 1}$ is the current state of the robot in the task space coordinates controlled by the MP. $\boldsymbol{\Sigma}_k$ is a 11×11 diagonal matrix with positive parameters. In this chapter, all MPs defined in the experiments control the end-effector position, orientation and the 4 DoFs of the robot's hand. Therefore, the dimension of the task-space is 11 (orientations are represented with quaternions). Equation (2.17) depends on the absolute difference between the state of the robot and the MP's attractor goal. Hence, the feature can be seen as progress indicator and is called goal distance [Luksch et al., 2012]. It is intrinsically in the range $[0, 1]$ and makes further data scaling superfluous. In addition, the variation of the feature around the MP goal can be shaped with the parameters of $\boldsymbol{\Sigma}_k$. To get expressive features, we find the parameters by minimizing the variance of the feature values while constraining the minimum and maximum values to be as close to zero and one as possible.

Figure 2.9 shows the resulting trajectories for some selected MP transitions. The learned goals are consistent with the directions of the movements. If trajectories are constant or diverge slightly, the goal is averaging over the trajectories (Figures 2.9b,c). If all trajectories point in the same direction as it is the case for Figure 2.9a, the goal lies in this direction as well, without conflicting with the trajectories.

The first example also illustrates the difference between the goal state of a MP and the state at which a new MP is activated. A new MP is activated as soon as the classification border is crossed, which is the case when the feature state reaches the value of the first transition. Note that such an early transitioning strategy is only triggered if it was also demonstrated. Therefore, behaviors such as opening or closing a gripper prematurely should not occur unintentionally.

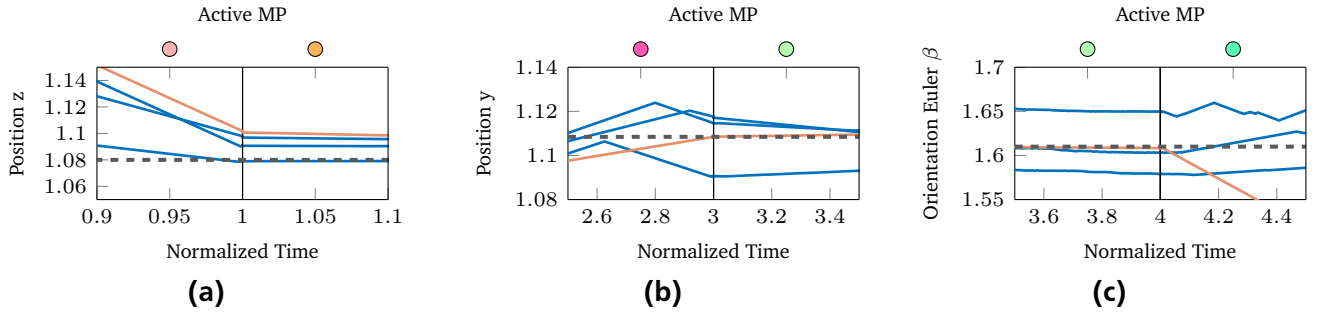


Figure 2.9.: Comparison of trajectories from kinesthetic demonstrations (●) and reproduction (●). The dashed lines show the learned MP attractor goals. All trajectories are aligned in time, so that each MP activation takes the same (normalized) time. The learned goals and the transition behavior are consistent with the demonstrations.

2.5.2 Unscrewing a Light Bulb

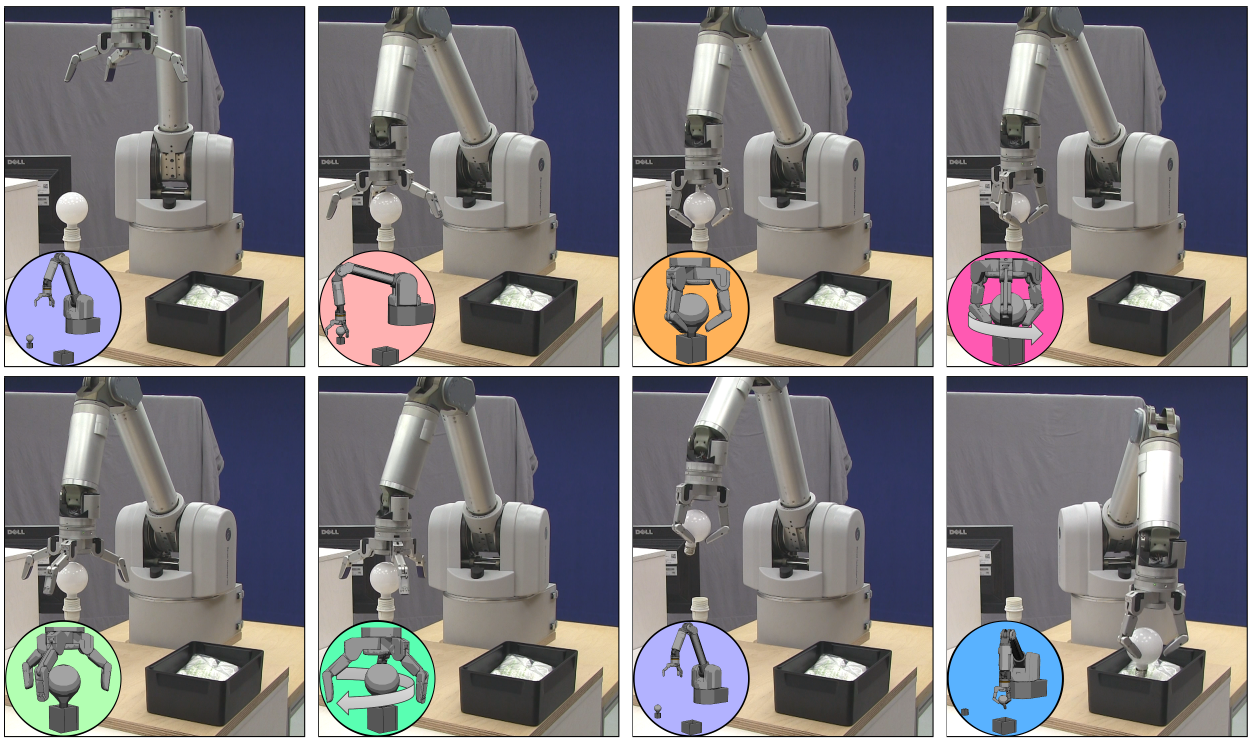
In the second experiment, the system had to learn how to unscrew a light bulb. The focus of this experiment was on evaluating the overall performance of the learning system, including the goal learning, the graph representations and the transition behavior. The task is to approach a light bulb, unscrew it and to put it into a box afterwards. For the representation of the skill, we chose seven different MPs. The detailed task flow and the MPs are illustrated in Figure 2.10.

We choose to unscrew the light bulb by caging it. Here, the robot encloses the bulb with its hand and grasps it below the point with the largest diameter. All MPs control the end-effector's Cartesian position, its orientation as well as the DoFs of the robot's hand. In addition to a global world frame, a coordinate frame is attached to the light bulb holder and the box. These coordinate frames are used for positioning the robot. When opening, closing or rotating the hand, either the three DoFs of the fingers or the angle of the wrist joint are controlled by the MP. The unscrewing MP (rotating the closed hand counterclockwise) additionally applies a force in upward direction to the robot's hand to ensure contact with the bulb. Again, the goal distance feature is assigned to each MP. The goal distance of the ●-MP can be used to detect if the light bulb is still in the holder. As a force is applied in upward direction during unscrewing, this force leads to an acceleration of the robot's arm as soon as the light bulb gets loose. As a consequence, the arm moves away from the MP's goal, resulting in an increasing value of the goal distance.

The system has to learn that an increase of this goal distance should lead to an immediate stopping of the unscrewing MP and a transition to the branch in the graph that puts the light bulb into the bin. As the light bulb is not represented in the feature set and the unscrewing stopping criterion depends implicitly on the height of the end-effector, a slipped light bulb can not be detected. As no slip happened during our experiments, we did not integrate the state of the light bulb into the feature set.

We performed three kinesthetic demonstrations and vary the position of the light bulb holder for each demonstration. For all following experiments, the system is trained separately with the data of each single demonstration, all pairs of demonstrations and all demonstrations. Every time the system is trained, the task is reproduced and the success rate of each MP transition is evaluated. A transition is considered successful, if the system activates the correct successor at the correct state. Incorrect, premature or too late transitions are considered failures. For unsuccessful transitions, we restart the movement, trigger the transition manually and continue with the reproduction from there.

We first evaluated the graph representations by comparing the local and global sequence graph with a baseline graph, which has one node for each MP and is fully connected, as shown in Figure 2.11. Hence, the system is allowed to transition to any MP at every point in time. All graph types are trained with LR as classifiers. The reproduction results are shown in Figure 2.12a. Both presented graph representations

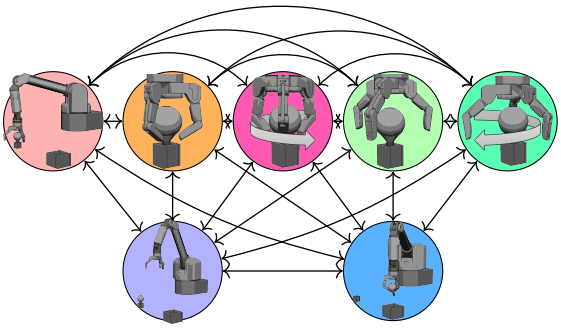


(a) Illustration of a successful task execution.

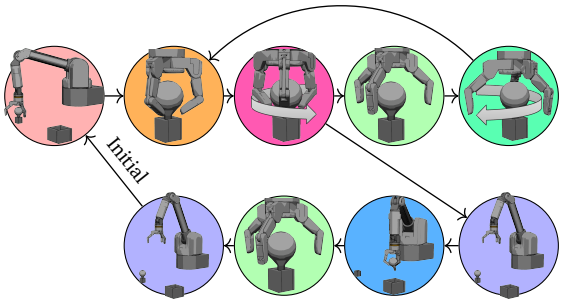
MP	Description	X	Y	Z	Orientation	Fingers	Next MP
●	Approach Initial Position	Initial Position			Initial	Hold	●, ●
●	Approach Light Bulb	Light Bulb			Hold	Spread	●
●	Close Hand (Grasp)	Light Bulb			Hold	Closed	●
●	Unscrew	Bulb	Bulb	Force	Rot. Wrist	Closed	●, ●
●	Open Hand (Release)	Light Bulb			Hold	Spread	●, ●
●	Rotate Wrist Clockwise	Light Bulb			Rot. Back	Spread	●
●	Approach Boxes	Garbage			Hold	Closed	●

(b) Task-space composition of the individual MPs

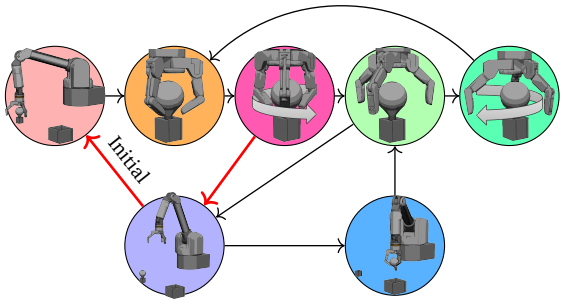
Figure 2.10.: Illustration and description of MPs for the light bulb experiments. The robot starts in an initial position (●) and first moves towards the bulb (●). Then it repeats the unscrewing movement (●, ●, ●, ●) until the bulb loosens (●) and subsequently, the bulb is put into a bin (●) and the robot returns to its initial position (●).



(a) Fully connected Baseline Graph



(b) Global Sequence Graph

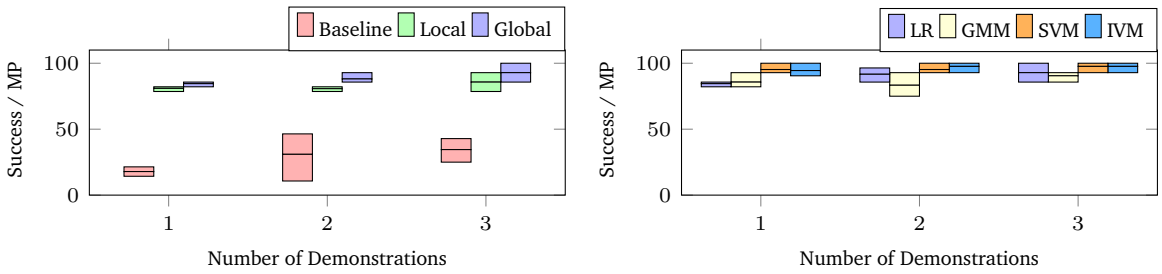


(c) Local Sequence Graph

Figure 2.11.: Graph representations of the light bulb task. We compare the global and local graph with a fully connected baseline graph. Compared to the global graph, the local graph merges several nodes. The merging creates potential paths in the graph which were not demonstrated. An example is the sequence marked as red which leads to a misbehavior of the robot if reproduced.

are clearly better suited than the baseline graph. Due to the reduced number of outgoing transitions for each node, the effect of perceptual aliasing gets reduced, which in turn improves the performance of the classifiers. This effect is also the reason why the global sequence graph slightly outperforms the local sequence graph. The local sequence graph contains paths which have not been demonstrated and lead to misbehavior if reproduced. An example is the red path in the figure. Here, the robot returns to its initial position with the bulb in its hand and immediately goes back to the bulb holder while opening its hand instead of going to the bin.

In a second set of experiments, we evaluated the performance of different classifiers. In addition to LR, we evaluated GMMs, SVMs and IVMs (see Section 2.4). All classifiers were trained using the global sequence graph, as it was the overall winner of the first experiments. The reproduction results for the different classifiers are shown in Figure 2.12b. The results indicate only a slightly better performance of the kernel methods compared to LR and GMMs. When being trained on all demonstrations, the average success rate of SVMs and IVMs is 97.6%, while LR reaches only 92.9%. The main reason for failing is the unscrewing movement, where the system sometimes failed to generalize from the demonstrations. When the light bulb gets loose at the beginning of the unscrewing movement during demonstration, the system is expected to be in a similar state when the light bulb gets loose during reproduction. If both states are different, the system tends to fail triggering the transition to the succeeding MP properly. This effect is reduced if more demonstrations are performed, as different wrist orientations are observed for each transition and therefore this feature becomes irrelevant for the decision. In general, a feature selection method may be helpful for further improving the performance.



(a) Graph Comparison **(b)** Classifier Comparison

Figure 2.12.: Experimental results for the light bulb task. The left plot shows the comparison of the three different graph structures: A fully connected graph used as baseline and the local and global sequence graph. All graphs were trained with Logistic Regression (LR) as classifiers. The right plot shows the evaluation of different classifiers: LR, Gaussian Mixture Models (GMMs), Support Vector Machines (SVMs), and Import Vector Machines (IVMs). Here, the global sequence graph has been used. The bars in both plots indicate the minimum, average and maximum success rate of each MP transition during reproduction of the task.

2.5.3 Grasping Objects with Error Recovery

In a third experiment, the system had to learn to grasp and lift three cylinders with different lengths using different grasps. The intention was to evaluate the approach on a more complex feature set compared to the goal distance features used in the first two experiments. The task was demonstrated as follows (see Figure 2.13). A cylinder was put randomly on a table, with arbitrary position and orientation (e.g., standing or lying). The teacher took the gravity compensated robot by its arm and moved the end-effector to an initial position. Next, it was moved to a pre-grasp position close to the cylinder. Subsequently, the cylinder was grasped and lifted. Depending on the length of the cylinder and its orientation, different grasps and pre-grasps were used for solving the task. If the cylinder was standing, it was grasped from the top. If it was standing upside down, it was grasped at its bottom. We demonstrated the task with three different cylinders of varying lengths, 8, 16 and $24\,\mathrm{cm}$. If the cylinder was lying and had a length of 8 or $24\,\mathrm{cm}$, it was grasped using a power grasp. If the $16\,\mathrm{cm}$ cylinder was lying, it was grasped using a pinch grasp. Pinch and power grasp could be performed with two different wrist angles as shown in Figure 2.14. During the final lifting step, the cylinder could slip. In that case, the lifting was immediately stopped. The end-effector was moved to the initial position and the task was demonstrated from the beginning. Note that even when such an error recovery strategy is demonstrated, the system has no explicit notation of an error when reproducing the task. Instead, a slipped cylinder is supposed to lead to a MP transition which is treated just as any other MP transition. The immediate triggering of an error recovery is also a typical example for a MP that is stopped before reaching its goal state when reproducing the task.

Feature	Dimension
Cylinder Length	1
Cylinder Inclination	1
Cylinder Rotation	1
End-Effector Position	3
Strain Gauges	3
Finger Joint Angles	4

Table 2.2.: Feature set for grasping task.

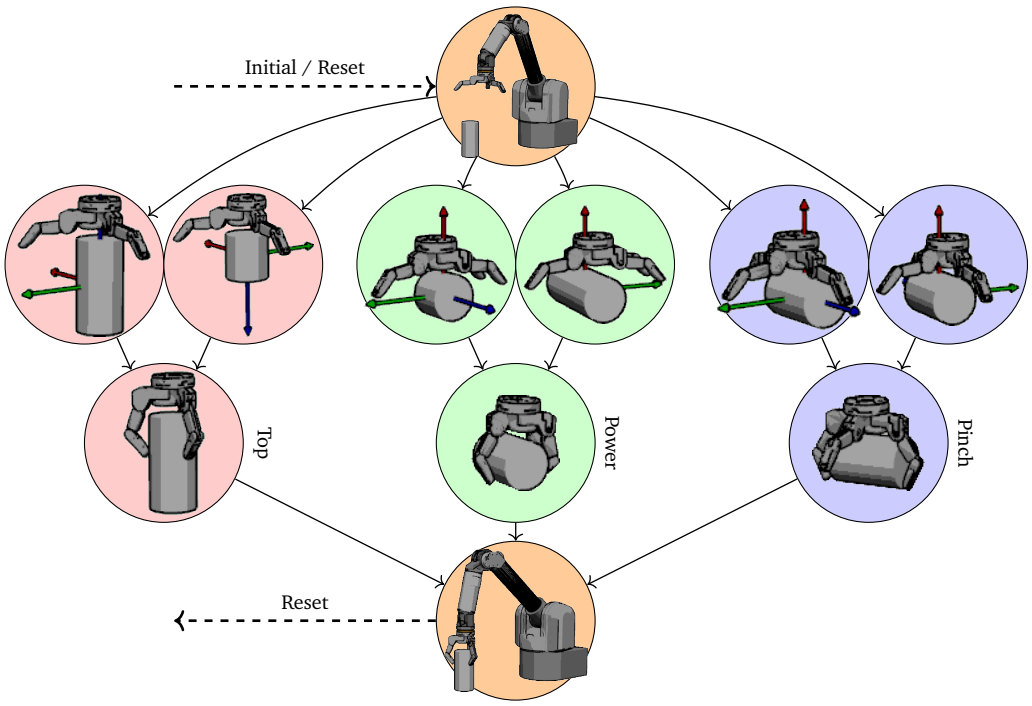
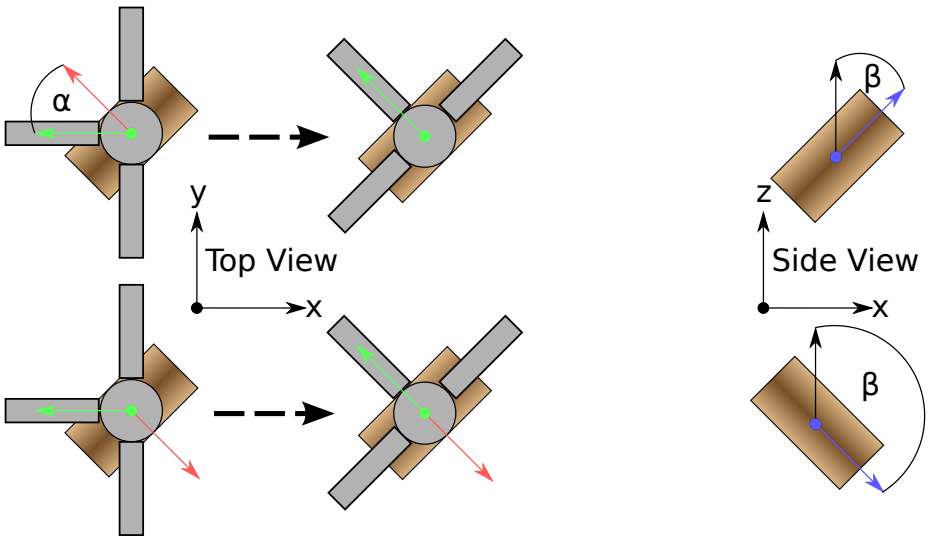


Figure 2.13.: Task flow of the grasping task. The robot starts in an initial position (top). Depending on the size of the cylinder and its orientation, the system approaches the cylinder with different orientations and finger configurations. Subsequently, the cylinder is grasped and lifted. If the contact with the cylinder is lost during lifting, the system has to re-start the task.

Table 2.2 summarizes all features for this task. The position of the end-effector is defined relative to the cylinders. Inclination and rotation are angles that represent the orientation of the cylinders. Both features are illustrated in Figure 2.14. Orientation and position of the cylinder are measured using a six DoF magnetic field tracking sensor system with precisions of approximately $1\,\mathrm{cm}$ and 0.15°. The strain gauges measure the tension in each finger. Together with the finger joint angles, they allow for detecting successful grasps or a slipped cylinder.

Similar to the previous experiments, we trained the system incrementally after each demonstration. Each training was followed by an execution of the learned skill. A successful reproduction of the task requires grasping and lifting the cylinder, as well as successfully detecting a slipped cylinder. Figure 2.15 shows the success rate of the reproduction for all 17 demonstrations. We evaluated the system on the local sequence graph and trained the classifiers with LR in simulation and with SVMs on the real robot. The results indicate that the system was able to learn the task completely, even though more demonstrations had to be performed compared to the light bulb task. The increased number of required demonstrations is not surprising, as the resulting graph has six outgoing transitions for the initial node. The transitions for the lying cylinder depend non-linearly on the cylinder length. The logistic regression models fail to cover this non-linearity. As a result, usually the power grasp is performed when the cylinder is lying. Even though such a strategy might also lead to a successful grasp, it is not the behavior that was demonstrated for the medium-length cylinder and is therefore considered a failure. The experiment shows that the system works well when using positions or joint angles instead of the MP goal distances as features. It also shows that it is easy to integrate arbitrary other features into our framework. The only requirement of our system is that the features are meaningful enough, so that successive MPs can be separated well in feature space.



(a) Rotation **(b)** Inclination

Figure 2.14.: The rotation feature (a) is the angle between the shown axes of the lying cylinder (red) and the hand (green). For avoiding unnecessary rotations of the end-effector, we define two pre-grasp MPs that approach the cylinder with different orientations (top and bottom). Depending on the angle α, the MP that is closer to its target orientation should be activated. The feature is shown for the pinch grasp, but is used for the power grasp in the same manner. The inclination feature (b) is the angle between the negative gravity vector (black) and the shown cylinder axis (blue). It indicates if the cylinder is standing ($\beta = 0$), lying or standing upside down ($\beta = \pi$).

2.6 Conclusion

This section provides a brief summary of our proposed approach and will highlight the main contributions. Finally, an epilogue will cover some suggestions for future work and discusses some open problems.

2.6.1 Summary of this Chapter

In this chapter, we proposed a method for learning sequential skills. In our case, a skill is comprised of a set of MPs and the ability to sequence them in order to perform a complex task. We showed how the

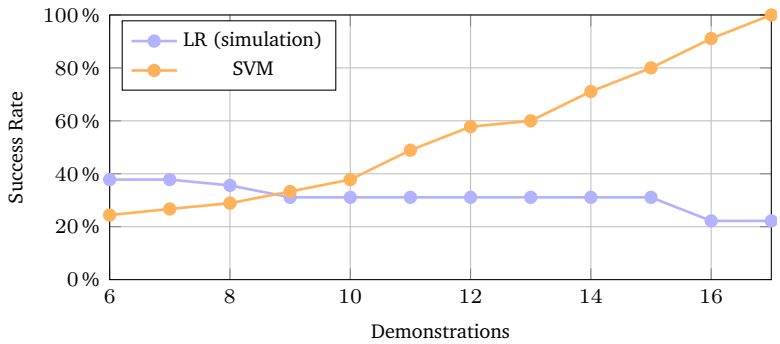


Figure 2.15.: Reproduction results for the grasping task. While an accuracy of 100% can be achieved using SVMs, the logistic regression (LR) models fail to cover the non-linear dependency on the length of the cylinder.

parameters of single linear attractor MPs as well as the transition behavior between them can be learned from labeled kinesthetic demonstrations of a task.

The observed MP sequences are incorporated into a graph representation we call sequence graph. A sequence graph has the individual MPs as nodes. It connects the MPs with each other and uses local classifiers for modeling the transition conditions between them. When executing a learned skill on a robot, the classifiers trigger transitions between the nodes, leading to a change of the activated MP. The major advantage of the graph structure is that it allows for splitting the overall decision of which MP to activate into multiple smaller classification problems.

The approach was validated in three experiments using a real seven DoF Barrett WAM robot with a four DoF hand. Here, we evaluated two different sequence graph structures and different classifiers for their applicability in our skill learning framework. The evaluation shows that our system is able to learn complex skills from a few demonstrations. One important insight from the experiments is that splitting the overall classification problem into many smaller problems helps to reduce the effect of perceptual aliasing.

2.6.2 Epilogue

Even though our system was able to learn skills for fairly complex tasks, some aspects of our proposed approach can be improved in future work. First of all, the approach relies on labeled demonstrations and a set of predefined MPs, which is a strong restriction and limits its practical applicability. In order to alleviate these limitations, we developed an approach which extracts a set of MPs from the demonstrations and labels the demonstrations automatically without further human supervision. This method will be presented in the next chapter.

In Section 2.3.3, we presented two different variants of the sequence graph and algorithms for generating them from labeled demonstrations. Compared to the global sequence graph, the local sequence graph is comprised of fewer nodes. In the experiments, it contained paths in the graph which were not demonstrated and lead to misbehavior. Therefore, the local graph could not be utilized for successful skill learning in all experiments. Even though the global sequence graph performed better for the conducted experiments, its generation still relies on a deterministic algorithm and there may be tasks for which the generated graph is suboptimal. Therefore, an important open problem for future research is finding an optimal sequence graph from a set of demonstrations which does not rely on a deterministic approach. Here, the main difficulty is that often, an optimal sequence graph cannot be derived from the limited amount of data acquired from only a handful demonstrations. In order to resolve ambiguities (e.g., which representation is optimal), either more demonstrations have to be provided or a method for interactive corrections has to be used. A conceivable alternative is to find the optimal sequence graph representation via trial and error, e.g., by expanding the most compact representation until the skill can be successfully executed on the real robot.

3 Probabilistic Decomposition and Skill Learning for Sequential Robot Manipulation Tasks

In the previous chapter, we presented an approach for learning sequential skills from kinesthetic demonstrations of a task. For the approach, our main assumption was that a robot is already equipped with a repertoire of (partially) predefined MPs and we focused on learning to sequence them properly. In order to learn a skill, the demonstrations had to be labeled with the active MPs over time. This labeling process is tedious and time-consuming and may prevent the method from being deployed in real-world application scenarios.

Therefore, in this chapter, we present a method for automatically decomposing a task into a set of MPs sufficient for performing a task. The approach is based on a probability distribution which we call Directional Normal Distribution (DND). The distribution allows to infer the MP's composition (coordinate frames, control variables, attractor goal) from the demonstrations in a principled manner and permits to determine an appropriate number of MPs for a task via model selection. Additionally, the method makes the process of manually labeling the demonstrations superfluous, as it allows for inferring the sequence of most likely MP activations directly from the demonstrations. In contrast to most state-of-the-art approaches, we explicitly deal with force data and do not assume that the sequence of movements performed by the human teacher is the same in all demonstrations.

While the focus of the chapter is the task-decomposition method, we also combine the method with the sequence graph concept presented in the previous chapter in order to learn how to sequence the resulting MPs. Combining both methods allows for learning a skill directly from the demonstrations without further human supervision. We evaluate the approach on three different tasks, unscrewing a light bulb, box stacking and box flipping. All tasks are kinesthetically demonstrated and then reproduced on a Barrett WAM robot. The evaluations will show that our approach allows for learning skills for fairly complex tasks from only a few demonstrations and that the skills can be generalized to situations which were not demonstrated. This chapter is mainly based on the work presented in Manschitz et al. [2017a,submitted].

3.1 Introduction

As robots are increasingly being used for many tasks just a few times instead for the same task a million times, it becomes impossible to pre-program them for all situations they may encounter. Learning methods can reduce the programming effort. By generalizing existing task knowledge to new situations, they even have the potential to allow the realization of tasks with higher complexity than just by programming alone. Therefore, learning manipulation tasks from human demonstrations has many potential application domains ranging from service robotics to industrial applications.

The contribution of this chapter is a concept for improving the human-to-robot skill transfer. Our particular aim is learning skills for tasks that have the following characteristics. First, the task can be represented by a sequence of point-to-point movements. Hence, the particular shape and execution speed of a movement are not relevant, but the target of a movement is. Examples for targets are desired positions or orientations of an end-effector, or a desired grasp. Second, the task involves a direct interaction with objects in the environment. In phases where the robot interacts with an object, it may have to actively apply a force, for instance it may have to push an object. Therefore, movements are not restricted to the kinematic domain. Instead, a target may also be comprised of a desired force or torque. We refer to tasks that have the aforementioned properties as sequential force interaction tasks.

Learning a skill for such tasks requires learning the individual point-to-point movements necessary to perform the task and the composition of the movements (e.g., if a force should be applied or not). In order to be able to perform the task on a real robot, additionally the possible sequential orders of the movements and transition conditions that allow for switching between them have to be learned. Our

particular aim is learning skills that generalize well to situations which were not demonstrated. Hence, the skill should be independent of global positions, such that it can be generalized to object locations which were not demonstrated to the robot. Therefore, we demonstrate a task a couple of times with varying object positions. From these demonstrations, the system learns the individual movements and decides for each movement if it should be performed relative to an object and if a force should be applied or not. After learning, we execute the resulting skill on a real robot with a different setup and, therefore, evaluate the generalization capabilities of the skill.

Throughout this chapter, we will refer to the individual movements as movement primitives (MPs). The focus of the chapter will be on the process of extracting attractor MPs representing the single point-to-point movements from the demonstrations. This process will be referred to as task-decomposition. In order to represent the robot's movements relative to objects in the scene, we use a hybrid position-force task-space controller (see Section 2.1.3). A task-space is composed of a controlled entity (e.g., end-effector), control variables (e.g., position or force) and a coordinate frame which allows for controlling the robot relative to an object. We predefine a set of task-spaces which are sufficient for performing a wide range of tasks. Applying our proposed task-decomposition method to the demonstrations then results is a set of MPs, whereas for each MP an appropriate task-space is selected and the most likely attractor goal is found. Basis of our approach is a novel probability distribution we call Directional Normal Distribution. In the chapter, we present the distribution and an Expectation-Maximization algorithm for inferring its parameters from data. After decomposing a task, we utilize the sequence graph concept presented in the previous chapter for learning to sequence the resulting MPs. By combining the task-decomposition method with the sequence graph concept, the system learns a reactive behavior which allows for performing the task autonomously on setups which were not demonstrated to the robot.

In summary, our system covers the whole process from the acquisition of the demonstrations to the execution of a learned skill on a real robot. The content of this chapter is based on previously published work. Our task-decomposition method was first presented by Manschitz et al. [2016]. The method was later combined with the sequence graph concept by Manschitz et al. [2017a,submitted], where additionally a more in-depth evaluation of the task-decomposition method was provided. Therefore, this chapter is mainly based on this paper.

3.1.1 Related Work

We classify approaches that aim at learning sequential skills into four different categories, depending on their focus:

1. Movement segmentation
2. Coordinate frame selection
3. Choosing control variables

   } Task-Decomposition
4. Sequence or transition learning

Movement segmentation deals with extracting and/or identifying MPs from a set of demonstrations. At the core, methods from this field aim at finding reoccurring patterns in time-series data. Coordinate frame selection methods try to infer if a movement is supposed to be performed relative to an object. This knowledge is important as it allows for generalizing a skill to undemonstrated object positions. Choosing the proper control variables is crucial for many tasks. Manipulation tasks are often comprised of phases where the robot has to apply forces, for instance when pushing a button or turning a valve. In other phases, it may only have to move the end-effector to a current position. In order to learn skills for such tasks, it is crucial to properly switch between these control variables. In this chapter, we focus on finding the decomposition of a task. In our case, such a decomposition is comprised of a set of generating MPs and their potential sequential orders. Sequence or transition learning approaches were

presented in the previous chapter and are therefore not discussed here. The interested reader is referred back to Section 2.1.1. In the following, we review other approaches, classify them into one of the first three fields and contrast them to our approach.

Movement Segmentation

Learning MPs from and identifying them in demonstrations is an important research problem, as it allows for reusing the same movements in multiple, potentially different situations. Among the aforementioned research fields, movement segmentation may be the one which has received the most attention so far. An extensive survey of such segmentation methods is provided by Lin et al. [2016]. The survey reviews various segmentation approaches from different fields (e.g., robotics, rehabilitation, motion analysis). The authors concentrate on detecting segmentation points between successive movements and present different metrics for evaluating the performance of a segmentation algorithm. In this section, we discuss some important segmentation approaches in the domain of robotics.

Time Window Segmentation

Segmenting demonstrations into smaller reoccurring patterns requires the detection of boundaries between successive movements and clustering of the individual segments based on a similarity measure. One way of dealing with the complexity of the overall problem is to first split the demonstrations into segments of fixed length. The resulting segments can subsequently be clustered by for instance interpreting them as single data points (e.g., by concatenating the individual data points of the segments to a single vector). Encapsulating the time-series character of the segments in a single data point allows for modeling the individual segments using non time-series models. Besides the computational advantages, downsides of time window approaches are that they usually require to specify the window size by hand and that the window size is often fixed over the entire demonstration.

Barbič et al. [2004] introduce and compare three different methods for segmenting high-dimensional motion capture data into single movements. The methods make use of Principal Component Analysis (PCA), Probabilistic Principal Component Analysis (PPCA), and Gaussian Mixture Models, respectively. The two underlying ideas of the two PCA methods are that first, during a movement, the individual joint angles are highly correlated with each other. Second, this correlation differs between movements. Hence, by capturing the intrinsic lower-dimensional subspaces of the data, individual movements can be discriminated from each other. Their method computes the PCA for fixed time windows and introduces segmentation points between successive movements when the projection error increases. Taylor et al. [2007] use a Restricted Boltzmann Machine (RBM) for modeling human movements. The authors use a predefined fixed number of past observations as input for the RBM and can learn higher-level movements by stacking several layers of the model. Takano and Nakamura [2006] split demonstrations of human whole-body movements into short sequences of fixed length. Each short sequence is modeled with a HMM. A higher-level HMM is then trained to model the sequences of HMMs and to abstract from the fixed sized data sequences. A similar approach is proposed by Kulić et al. [2008], where a HMM models data partitioned into multiple sliding windows. Their algorithm can be used for online segmentation and the movements can be clustered into different groups representing similar movements. Kulić et al. [2012] extend this approach by incrementally learning the relationships between MPs. These relationships are modeled with a graph representation which can be utilized for the generation of longer behaviors. Pais et al. [2013] split the demonstrations into windows of fixed length. Segmentation points between successive MP are restricted to appear at the border between two segments. A segmentation point is added if either the most likely coordinate frame of the current movement or the variable of interest (e.g., force or position control) changes. No clustering of segments has to be performed as the authors assume the demonstrated sequence of MPs is the same for all demonstrations.

Changepoint Based Segmentation

Some approaches assume initial guesses for potential cuts between the segments. The aim is then to prune out false positives cuts and to find the true positives. Often, such cut candidates are found by analyzing the zero velocity crossings (ZVCs) of the demonstrations. These kind of approaches are supported by Flanagan et al. [2006], who showed the relevance of ZVCs for dexterous manipulation from a biological point of view. The intuition behind ZVCs is that a pause is a natural separation between two movements.

Lioutikov et al. [2015] prune out false positive cuts in a probabilistic manner using an Expectation-Maximization algorithm. A distribution over cut candidates is maintained and treated as latent variable. In the Expectation step of the algorithm, the parameters of the MPs are fixed and the distribution over the cut candidates is updated. In the Maximization step, the distribution is fixed and the MP parameters are updated. An extended version of this approach is presented by Lioutikov et al. [accepted]. Konidaris et al. [2012] present a concept for statistical online changepoint detection based on Hidden Markov Models. Segmenting a single demonstration results in a skill chain, namely a sequence of goal-directed movements. Multiple demonstrations are merged into a skill tree. Their model can also be used in the context of Reinforcement Learning, where a robot actively generates its own demonstrations. Kober et al. [2015] align demonstrations represented in different coordinate frames in time using Dynamic Time Warping (DTW). Subsequently, their approach finds the ZVCs of the demonstrations and extract one MP for each resulting segment. Similar ideas to analyzing the ZVCs are to add segmentation points based on the joint acceleration [Guerra-Filho and Aloimonos, 2007] or angular jerk [Yamamoto et al., 2006]. All of these concepts have in common that pausing a movement results in a logical segmentation point.

Utilizing ZVCs for detecting potential cuts between successive segments is supported by findings from biology and has proven to work well in practice [Calinon and Billard, 2004, Guerra-Filho and Aloimonos, 2007]. Therefore, our approaches also incorporates them into the segmentation process. Compared to many other approaches, our approach allows for learning object-directed movements without requiring that the demonstrated movement sequences are the same for all demonstrations. Therefore, the demonstrations do not have to be aligned in time.

Data-Driven Segmentation

Even though most of the previously mentioned segmentation approaches make use of data-driven machine learning methods, they either assume the demonstrations are pre-segmented or pre-segment the demonstrations using heuristics (e.g., ZVCs). Such heuristics, even though they work well in practice, may not be appropriate for all tasks. Other approaches therefore segment the demonstrations in a purely data-driven way, using as few assumptions as possible about the task or the structure of the demonstrations.

Grollman and Jenkins [2010] partition the demonstrations incrementally using the Chinese Restaurant Process. For each cluster, a Gaussian Process Regressor learns a policy which maps the perceived state to a continuous action of the robot. However, when reproducing a task, the algorithm requires using a hand coded mechanism for selecting the correct cluster at each time step. Butterfield et al. [2010] overcome this issue by also learning the transition behavior between the individual clusters. For this purpose, they use a Hierarchical Dirichlet Process Hidden Markov Model. Alvarez et al. [2011] model single movements with Latent Force Models (LFMs, [Alvarez et al., 2009]). A LFM can be thought of as a set of spring-mass-damper systems being driven by a function sampled from a Gaussian process. A skill is then composed by a set of LFMs and the ability to properly switch between them. The authors present a method for learning a skill from demonstrations which requires to specify the number of LFMs as a hyperparameter. Chiappa et al. [2009] present a Bayesian approach for clustering movements based on Mixtures of Linear Gaussian State-Space Models (LGSSMs). As a Bayesian treatment of LGSSMs is intractable, they introduce a clustering method based on variational Bayes. Their model is generative and thus can also be utilized for generating movements. Yet, no transition conditions between the movements

are learned and therefore, it is only shown that the generated movements for single clusters are similar to those who have been demonstrated. Endres et al. [2011] use Bayesian binning for segmenting human Taekwondo movements into single actions. Actions are modeled as fixed order polynomials. The author's also focus on the segmentation process and not on the actual generation of movements. One downside of their approach is that the individual bins corresponding to the movement are independent of each other and are represented in a singly connected factor graph. Therefore, if the same movement is demonstrated twice in a sequence, it is not considered to be the same movement. Niekum et al. [2012, 2013, 2015b] segment demonstrations using a Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM, [Fox et al., 2009]). After segmenting the demonstrations and assigning the segments to clusters, each cluster is considered a single movement that is subsequently modeled with a DMP. Therefore, the method can also be utilized for reproducing a learned skill on a real robot. Their method does not require specifying the number of MPs beforehand and is able to learn skills from demonstrations with varying movement orders. Medina and Billard [2017] represent single movements with Linear Parameter Varying (LPV) Systems encoded as Gaussian Mixture Models. Movement sequences are encoded with HMMs, where the individual states are represented with LPVs. Each single movement is guaranteed to have a unique global attractor and they also learn the termination conditions of the individual movements.

The segmentation of most of the aforementioned approaches is based on modeling the individual segments. In that case, a segmentation point is implicitly defined as a point were the likelihood of a model corresponding to the current segment decreases and the likelihood of another model increases (hence, the most likely model changes). Another idea is to directly detect the changes between the physical relationships of objects in the environment. Niekum et al. [2015a] do so by using Bayesian online changepoint detection. An example for a physical relationship is one where two objects move together as a rigid body (e.g., two boxes are stacked together and moved from one position to another). Baisero et al. [2015] have similar aims. Their approach is based on Conditional Random Fields [Lafferty et al., 2001], but requires labeled demonstrations.

Recently, also Inverse Reinforcement Learning (IRL) methods have been frequently used in the context of movement segmentation and skill identification. Instead of directly extracting MPs from demonstrations, IRL methods try to infer reward functions corresponding to the goals of the individual sub-tasks. Once found, the reward functions can be utilized for learning a set of MPs which achieve the goals of the sub-tasks. If the reward functions capture the essence of the sub-tasks, the resulting MPs are likely to generalize well to previously unknown situations. Despite these advantages, one downside of IRL methods is that they are data-intensive, making them impractical for our purpose. Ranchod et al. [2015] present an approach for segmenting trajectories using IRL. They adapt the Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM) by replacing the Vector Autoregressive Process as emissions with a Markov Decision Process (MDP). The MDP allows for learning a reward function and the corresponding policy that maximizes the reward for each segment. All model parameters have to be estimated with Monte Carlo methods, which is why the method is computationally very expensive and the evaluations are carried out on simple toy examples. A similar IRL method is proposed by Michini [2013], who use a nonparametric Chinese Restaurant Process for inferring the number of sub-tasks from the demonstrations in a non-parametric way.

In summary, completely data-driven approaches have the ability to learn a skill with very few assumptions about the task or the structure of the demonstrations. However, they are often data-intensive, as they require too many demonstrations for learning a skill. Additionally, the resulting MPs do not necessarily describe meaningful movements in the context of a task (e.g., grasp an object) as they do not incorporate knowledge about robotics or the task itself. As a consequence, the MPs usually explain the demonstrations, but the generalization capabilities of the resulting skill may be limited. In order to learn generalizable skills from a few demonstrations our proposed approach makes use of ZVCs.

Identifying Movement Primitives from a Library

If a system is already equipped with a repertoire of MPs, the segmentation problem reduces to identifying these MPs in the demonstrations. Meier et al. [2011] propose an approach for online movement recognition and segmentation based on a library of predefined MPs. For each new data point, they estimate the open parameters of all MPs in the library and compute the likelihood of the corresponding models. The currently active MP is the one corresponding to the highest likelihood. If all likelihoods fall below a threshold, a new MP is added to the library. This approach is extended by Meier et al. [2012], where segmentation candidates are added by analyzing the velocity profiles of the movements. These candidates allow for segmenting data offline and make the hand-tuning of the threshold parameter superfluous. A similar approach is proposed by Lemme et al. [2014b]. Here, a MP library is built incrementally, given only a single initial MP (corresponding to a simple point to point movement) and a set of kinesthetic demonstrations. The demonstrations are segmented into a sequence of trajectory chunks. Each chunk is assigned to the MP which most likely produced it. The robot then executes the sequence and co-articulates between successive movements. Due to the co-articulation, combining two point to point movements may result in a smooth curved movement. Subsequently, the chunks from the previous demonstration are pairwise combined to form new MPs. The new MPs are added to the library and are again used for segmenting the new trajectory. This process is repeated until the reproduced trajectory matches the kinesthetic demonstrations well. By combining and refining existing MPs, the approach can learn movements of arbitrary shape. The movement identification is based on the geometrical shape of the movements. Therefore, it is not possible to distinguish movements of similar shape but different dynamics. Additionally, it is not clear if the approach can be applied to more than three-dimensional movement spaces. Gräve and Behnke [2012] model individual MPs with HMMs and assume that these MPs have been previously learned. Given a demonstration, their approach tries to find segmentation borders between successive segments and assigns each segment to one of the existing MPs. A segmentation border is defined as the best compromise between the most likely end point of the current segment and the start point of the succeeding segment.

Coordinate Frame Selection

Introducing coordinate frames allows for representing and generating object-directed movements, potentially leading to increased generalization capabilities of a skill. Therefore, various approaches aim at learning in which coordinate frame a MP should be represented. Niekum et al. [2015b] segment demonstrations using an Auto-Regressive Hidden Markov Model. Each state of the model corresponds to a MP. After the segmentation, the end-points of the segments assigned to a MP are clustered in different coordinate frames. The goal of the MP is then defined in the frame corresponding to the largest cluster. In contrast to this approach, our method integrates the coordinate frame selection directly into the segmentation. Another approach that explicitly deals with coordinate frames is that of Rozo et al. [2016]. The authors use a task-parametrized Gaussian mixture model (TP-GMM, [Calinon et al., 2012]) as trajectory representation. A TP-GMM is a hierarchical GMM with two layers. While on the upper layer each mixture corresponds to a MP, the lower layer represents the joint probability of the trajectory (and the velocity) in different coordinate frames. In addition to learning a task-representation in different coordinate frames, they learn to vary the controller stiffness dynamically. As a result, their approach enables a robot to physically interact with a human co-worker. While human-robot collaboration is an interesting research domain, our approach focuses on tasks where the robot is operating autonomously.

Pais et al. [2013], Ureche et al. [2015] extract the coordinate frames and control variables based on the variance of the data. If the variance of a variable is large within a time window for all demonstrations, they consider it to be significant for the task. The reason is that the variable changes its value and does that in a systematic way across all demonstrations. Kober et al. [2015] showed that under certain circumstances this assumption leads to an over-segmentation of the data. Therefore, they suggested to incorporate the convergence behavior of the movement as well. Compared to our method, many

approaches (e.g., Kober et al. [2015], Mühlig et al. [2009], Pastor et al. [2012], Reiner et al. [2014], Ureche et al. [2015]) require demonstrations with identical sequential ordering of the employed MPs. Our approach is instead capable of decomposing a task if the demonstrated MP sequences vary across demonstrations. An example are demonstrations of a light bulb unscrewing task. Here, the number of unscrewing repetitions may vary over demonstrations depending on how firmly the light bulb is screwed in. Furthermore, our approach is able to simultaneously infer all possible MP sequences as well as the composition of the MPs. We will show that decomposing kinesthetic demonstrations of a force interaction task with our approach leads to meaningful and intuitive MPs.

Choosing Control Variables

Only a few approaches incorporate force information into the decomposition of the task. For instance, Abu-Dakka et al. [2015] adapt the dynamic movement primitives (DMPs, Ijspeert et al. [2002]) framework so that the robot follows a desired force profile. Yet, they predefine if an MP is position or force-controlled. Steinmetz et al. [2015] learn a desired position and force profile with a DMP from a single demonstration. They present a control framework that is able to transition between phases of pure impedance control and force control based on the measured external force. The approach is not able to handle multiple demonstrations or sequences of MPs. Rozo et al. [2016] learn to modulate the stiffness factor, which is important for collaborative tasks. Ureche et al. [2015] as well as Kober et al. [2015] explicitly choose between position and force control based on the variance of the data and the convergence properties of the segments, respectively. As these approaches worked well in practice, we adopt some of the ideas of the aforementioned papers and integrate them into our approach.

3.1.2 Learning Sequential Force Interaction Tasks

This section provides an overview of the presented method and the structure of this chapter, Additionally, the notation of this chapter is introduced. In order to learn a skill, two main steps are performed. First, a set of MPs is extracted from the demonstrations. Second, the system learns to sequence the resulting MPs. Our method operates on the data recorded during a set of kinesthetic demonstrations of the task

$$\tau_m = \left\{ \left(\boldsymbol{q}_i^{(m)}, \boldsymbol{ft}_i^{(m)}, \boldsymbol{f}_i^{(m)}, \boldsymbol{o}_i^{(m,1)}, \ldots, \boldsymbol{o}_i^{(m,N_o)} \right)_{i=1:N_m} \right\}. \tag{3.1}$$

Each demonstration m of varying length N_m with a robot that has N_j joints results in a time-series of joint angles $\boldsymbol{q}_{1:N_m}^{(m)} \in \mathbb{R}^{N_j \times 1}$, forces and torques $\boldsymbol{ft}_{1:N_m}^{(m)} \in \mathbb{R}^{6\times 1}$, as well as the 3D positions and 3D orientations of all N_o objects in the scene $\boldsymbol{o}_{1:N_m}^{(m,o)} \in \mathbb{R}^{6\times 1}$. Here, $\boldsymbol{o}_{1:N_m}^{(m,o)}$ contains the positions and orientations of the oth object for demonstration m. Each object and the world are associated with a coordinate frame. The forces and torques are measured at the wrist of the robot. The teacher is required to touch the robot above of the wrist such that he or she does not distort the force measurements. The joint angles include the joints of the robot arm and the joints of the robot's hand.

In a first step, the demonstrations are projected onto a set of predefined task-spaces. For our tasks (as well as many household and/or industrial tasks), it is sufficient to control the position and orientation of the end-effector and the joint angles of the robot's hand. These three entities (position, orientation, fingers) can be controlled largely independently. Therefore, we define the following task-spaces: We use task-spaces representing the position and force of the end-effector in the world frame and relative to every object in the scene. To represent the orientation and torque of the end-effector, also task-spaces are used for the world frame and all other coordinate frames. The fingers of the robot are controlled in joint-space. Therefore, one additional task-space is used to describe the finger joints. The projection results in the task-space data $\boldsymbol{x}_{1:N_m}^{(m,k)}$. Here, the time-series $\boldsymbol{x}_{1:N_m}^{(m,k)} \in \mathbb{R}^{N_k \times 1}$ corresponds to the data of

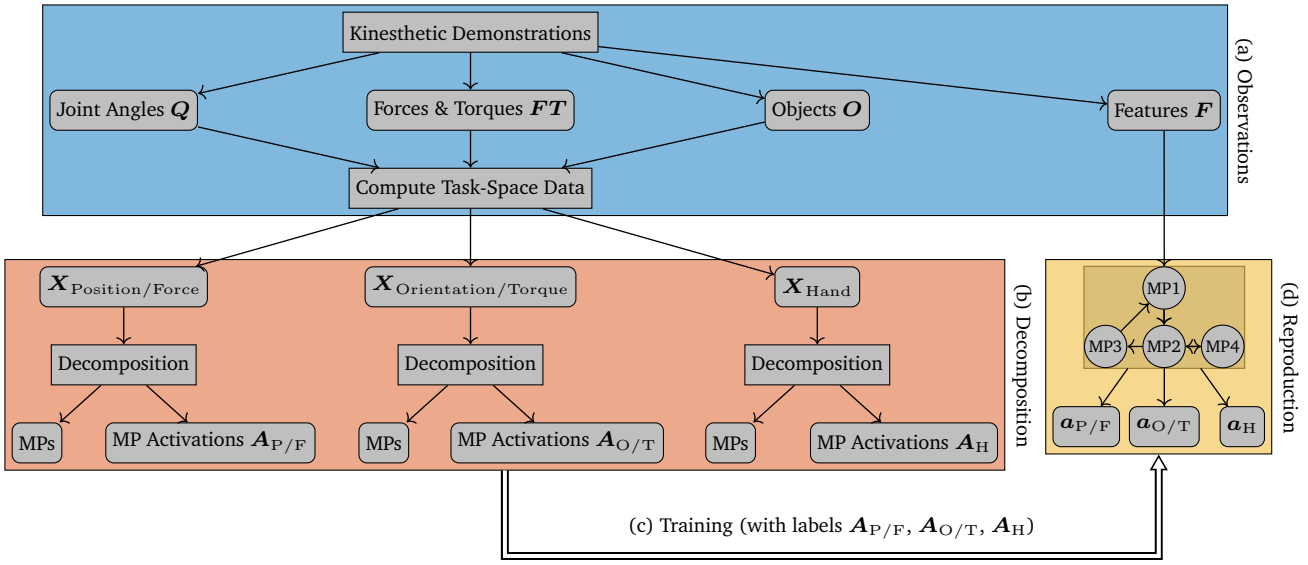


Figure 3.1.: Overview of our approach. From the kinesthetic demonstrations (a), we record the joint angles of the robot, the measurements of the force-torque sensor, the positions and orientations of all objects in the scene, and features over time. The observations are then projected onto a set of predefined task-spaces. The task-spaces are split according to what they control (b). The resulting data is used to decompose the demonstrated task, yielding a set of MPs, and their activation probabilities over time. For reproducing the task on the robot, a graph with local classifiers is responsible for setting the MP activations (d). For training the classifiers (c), the features and MP activation probabilities are used.

the mth demonstration represented in the kth task-space. Note that the number of dimensions N_k can vary for the individual task-spaces.

The next step is to extract a set of MPs from the task-space data. For understanding this step, some insights into our controller and MP framework are beneficial. We use the same hybrid position-force task-space controller for controlling the robot as in the previous chapter (see Section 2.1.3), as well as the same MP representation. Hence, a MP is a dynamical system with a linear attractor behavior

$$\ddot{\boldsymbol{x}}^{(k)} = \alpha \left(\beta(\boldsymbol{g} - \boldsymbol{x}^{(k)}) - \dot{\boldsymbol{x}}^{(k)} \right), \tag{3.2}$$

where $\boldsymbol{g}$ is the attractor goal of the MP and $\boldsymbol{x}^{(k)}$ are the coordinates of task-space k. The controller parameters α and β determine the dynamical behavior of the movement. For each MP, our method has to select an appropriate task-space $k \in \{1, \ldots, K\}$ and has to infer its most likely attractor goal $\boldsymbol{g}$. Note that the controller parameters α and β are predefined in this chapter, as learning them is out of the scope of this thesis. We consider it an interesting research problem to learn them from the demonstrations as well. Our control framework allows the activation of multiple MPs at the same time. As, position, orientation and fingers can be controlled largely independently, we usually activate three MPs at the same time when controlling the robot. The advantage is that this co-activation greatly simplifies our MPs. For instance, we can reach different positions with a constant orientation by changing only the MP controlling the position of the end-effector. To integrate this property into the decomposition, we split the task-space data into three distinctive sets according to what the controlled entity is (position, orientation or fingers). We apply our task-decomposition method to each of these sets, as illustrated in Figure 3.1. Therefore, the task-decomposition results in three independent sets of MPs, one for each controlled entity. The decomposition method is introduced in the following Section 3.2. The core of the method is the Directional Normal Distribution, a probability distribution which is introduced in detail in Section 3.3.

After decomposing a task into MPs, our system learns to sequence the MPs in order to master the overall task. When executing a learned skill on a real robot, the MPs are activated in a reactive manner

based on a feature set $\boldsymbol{f} \in \mathbb{R}^{N_F \times 1}$. Here, N_F is the dimension of the feature space. In theory, the feature space can be comprised of arbitrary measurable values (e.g., camera images). However, in this chapter, it will be usually comprised of the robot's state in task-space coordinates. As such, the feature state implicitly contains the positions and orientations of all objects in the scene, as each object is associated with a coordinate frame. In order to learn a mapping from features to MP activations, we record the features during the demonstrations $\boldsymbol{f}^{(m)}_{1:N_m} \in \mathbb{R}^{N_F \times 1}$, as well. The sequencing method is based on sequence graph concept presented in the previous chapter. In Section 3.4, we adopt the concept to account for the required synchronization of the otherwise independently controlled MPs of the three different entities. One important difference to the previous chapter is that the task-decomposition can be used for labeling the demonstrations with the most likely MPs over time. Therefore, the decomposition method replaces the tedious process of manually labeling the demonstrations.

After learning to sequence the MPs, the resulting skill can be executed on a real robot. In Section 3.5, we evaluate the proposed approach on three different real robot experiments: box stacking, box flipping and light bulb unscrewing. Finally, we discuss the results and some remaining open problems in Section 3.6.

3.2 Proposed Task-Decomposition Approach

The task-decomposition approach consists of three steps. First, the demonstrations are split into smaller segments. Second, a HMM is trained to cluster the segments. Segments are assigned to the same cluster if they can be represented by the same attractor movement in one of the task-spaces. Finally, the MPs are extracted from the parameters of the resulting clusters. The three steps will be explained in detail in the following subsections. The explanations are accompanied by a simple toy example that is using only position and force data. The reader can imagine the toy task as follows. The task is to first approach an object, subsequently push against the object and then move to a fixed final position. Figure 3.2 depicts an overview of the toy example in the context of our task-decomposition method. As discussed earlier, the same method is used for the individual decompositions of the three controlled entities (position, orientation and fingers)[1].

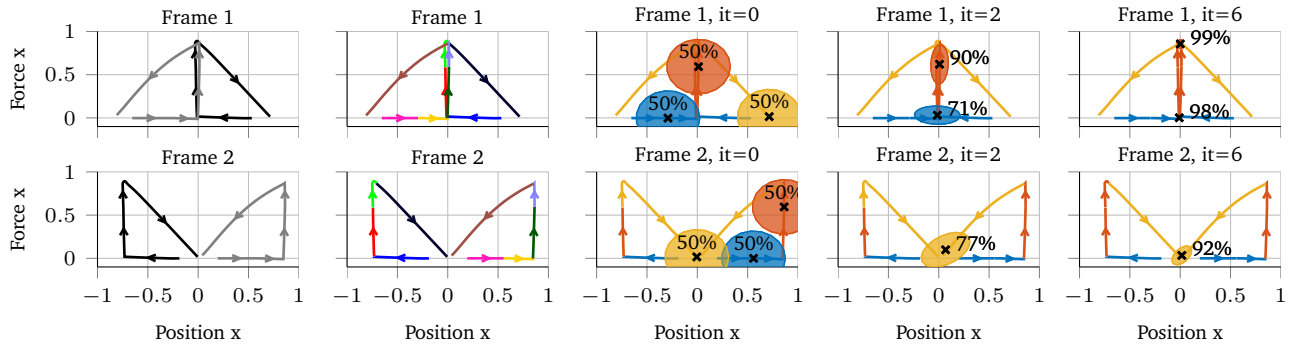
3.2.1 Segmentation

As a first step, we split the demonstrations into smaller segments by finding the zero-velocity crossings (ZVCs) of the individual task-spaces. Hence, a segment is added if the robot gets into or looses contact with an object or if a controlled entity stopped and starts to move again. For each demonstration m, the time-indices of the ZVCs are stored in a time-ordered vector $\boldsymbol{z}^{(m)}$. Based on $\boldsymbol{z}$, the demonstrations are split into sequences of segments $\boldsymbol{x}^{(m,k)}_{z_l:z_{l+1}}$, where l corresponds to the lth segment. Figure 3.2b depicts the segmentation of the toy example.

3.2.2 Clustering the Segments

After the segmentation, we assume the data is over-segmented, as multiple segments may correspond to the same movement. Therefore, we cluster the individual segments based on a similarity measure. As we concentrate on learning skills for tasks that can be represented by a sequence of point-to-point movements, we argue that segments converging to the same target (in one of the task-spaces) should be assigned to the same cluster. This convergence property of the segments can be measured by a probability distribution which we call Directional Normal Distribution (DND). This distribution is based on the normal distribution and has two parameters, a mean $\boldsymbol{\mu}$ and a Covariance matrix $\boldsymbol{\Sigma}$. The difference to a

[1] Handling orientations needs some adaptions that are explained in Chapter 3.3.2



(a) Demonstrations **(b)** Segmentation **(c)** Hidden Markov Model

Figure 3.2.: Overview of the presented task-decomposition approach using a simple 1D toy example. Plot (a) shows two demonstrations (black and gray) in two different coordinate frames. Our approach first segments the data by finding zero-velocity crossings and contact changes (b). The different segments are illustrated using different colors. Subsequently, the segments are clustered by using a Hidden Markov Model (HMM) (c). Here, we use mixtures of Directional Normal Distributions (DNDs) as state emissions of the HMM. DNDs allow for clustering the segments based on their convergence properties. If two segments are converging to the same attractor in one of the task-spaces corresponding to the coordinate frames, they are more likely assigned to the same cluster. After training, a MP is defined for each resulting cluster, and its coordinate frame, control variables and attractor target can be inferred from the parameters of the cluster. The attractor goals of the MPs are marked in the plot and the uncertainty about their position is shown with ellipsoids. The goals are only shown in the most likely coordinate frame of the MP. For instance, the ●-MP will be position-controlled in the second frame (with a certainty of 92%), as the target force is close to zero and the segments assigned to the MP converge best in this frame.

Normal Distribution is the meaning of the $\boldsymbol{\mu}$ parameter. While $\boldsymbol{\Sigma}$ can also be interpreted as an uncertainty parameter, the mean is defined as an attractor goal. For a given segment, the distribution yields a probability that indicates how well the segment is converging towards the attractor goal. Therefore, the distribution can for instance be utilized for determining if two or more segments converge to the same attractor goal. In Section 3.3, we present an algorithm which allows for learning the parameters of the distribution from data. For this section, it is sufficient to know that we can estimate the most likely target to which a set of segments converge in a probabilistic manner by using DNDs.

For clustering the segments, we integrate the DND as state distribution into a Hidden Markov Model (HMM). More precisely, as our data is represented in different task-spaces and we want to measure the convergence for each task-space and segment, we use a mixture of DNDs for each HMM state. One advantage of using such a probabilistic model is that we can perform model selection to choose the optimal number of clusters for the demonstrations. For this step, we use the Bayesian Information Criterion (BIC), a standard model selection criterion for probabilistic models [Schwarz, 1978].

The HMM training process is illustrated in Figure 3.2c. We initialize the parameters of the model by setting each DND mean to an end-point of a randomly selected segment. The covariance matrices are initialized with that of an isotropic Gaussian $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$. The transition matrix is initialized randomly with a small bias on self-transitions. Subsequently, the segments are assigned to the clusters based on their convergence properties by maximizing the log-likelihood of the model with the Baum-Welch algorithm [Welch, 2003].

3.2.3 Extraction of MPs

The HMM training assigns segments to the same cluster if they converge to the same attractor goal in one of the task-spaces. We argue that these segments can be represented by the same MP. The parameters

of each resulting MP can be inferred from the parameters of the corresponding cluster. For each cluster, the largest mixture weight of the DND mixture indicates in which of the task-spaces the MP can be represented best. The target of the MP is then defined as the mean of the mixture with the largest weight. Finally, if the assigned task-space of an MP is composed of forces and positions along the same axis, we explicitly decide whether force or position are chosen as control variable. Here, we use a simple heuristic. Force is only chosen if the desired attractor force of the MP is higher than a threshold and the velocity mean of the segments assigned to this MP is below a threshold. Thus, if a force was measured along a certain axis and the robot was not moving in this direction, we assume the teacher wanted the robot to apply a force. For an illustrative example of the parameter extraction, the reader is referred to Figure 3.2c.

3.3 Measuring Convergence with the Directional Normal Distribution

We introduce the DND as an instrument for measuring the convergence of the individual segments. The distribution has two parameters $\Theta = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. The mean $\boldsymbol{\mu}$ corresponds to an attractor goal, while the covariance matrix $\boldsymbol{\Sigma}$ indicates the uncertainty about the location of $\boldsymbol{\mu}$. In order to be used for our purpose, the distribution has to fulfill two properties. First, given a segment $\boldsymbol{x}_{1:N}$ the probability $p(\boldsymbol{x}_{1:N}|\Theta)$ should be large if the segment is converging to the target $\boldsymbol{\mu}$, and small otherwise. Second, given a single or multiple segments, we need a method for estimating the most likely parameters of the distribution. In this section, we present the idea behind the distribution and its density function. In Section 3.3.1, we then derive a method for estimating the parameters from data with a maximum likelihood approach.

For deriving the density function, we assume the following statement to hold. If a segment is converging to a given target, then on average, the velocity vector $\boldsymbol{v}_i$ corresponding to a data point $\boldsymbol{x}_i$ from the segment should roughly point towards the target. Thus, the probability of a segment is computed as joint probability of the points from the segment

$$p\left([\boldsymbol{x}_{1:N}, \boldsymbol{v}_{1:N}]|\Theta\right) = \prod_{i=1}^{N} p\left([\boldsymbol{x}_i, \boldsymbol{v}_i]\,\middle|\Theta\right).$$

Here, N is the number of points of the segment and $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ are the individual points from the segment. We usually compute the velocity by approximating the gradient $\boldsymbol{v}_i \approx (\boldsymbol{x}_i - \boldsymbol{x}_{i-1})/h$ with step-size h. In that case, the equation can also be written as

$$p\left(\boldsymbol{x}_{1:N}|\Theta\right) = p\left(\boldsymbol{x}_1\right) \prod_{i=2}^{N} p\left(\boldsymbol{x}_i\middle|\boldsymbol{x}_{i-1}, \Theta\right).$$

In the following, we will stick to the notation which makes use of the velocity vector $\boldsymbol{v}$. Given a single data point $\boldsymbol{x}$ and its velocity vector $\boldsymbol{v}$, the density function of a DND is defined as

$$p\left([\boldsymbol{x}, \boldsymbol{v}]\,\middle|t, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{e^{-\frac{1}{2}(\boldsymbol{x}+t\boldsymbol{v}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}+t\boldsymbol{v}-\boldsymbol{\mu})}}{\sqrt{(2\pi)^d\,|\boldsymbol{\Sigma}|}}, \tag{3.3}$$

The scalar parameter t projects the point along its velocity vector. The idea is to choose the parameter so that the distance between the projection and the mean of the distribution becomes minimal. If pointing away from the mean, a data point should be assigned a low probability. Therefore, the parameter is restricted to $t \geq 0$. By rearranging (3.3), it can also be written as a normal distribution of t

$$p\left(t\middle|\mu_t, \sigma_t^2\right) = Z_c e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}}, \tag{3.4}$$

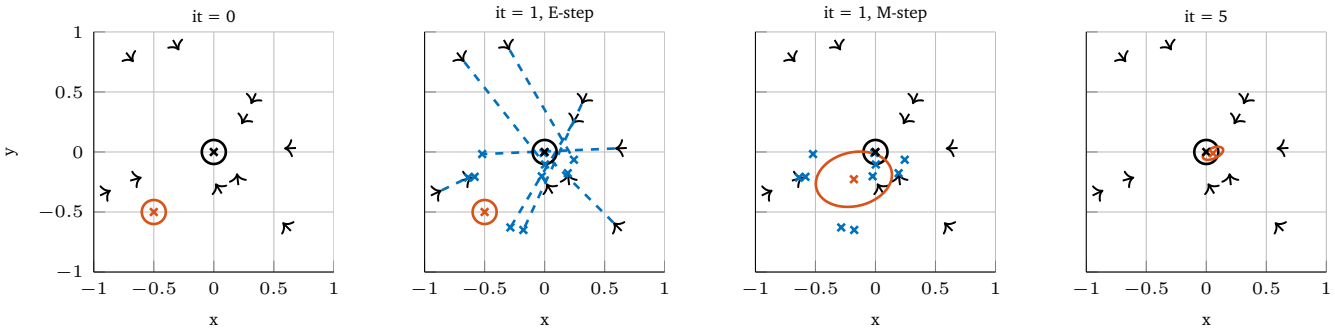


Figure 3.3.: Illustration of the EM algorithm. The black arrows correspond to data points and their velocity vectors. The data points are drawn uniformly from the shown grid. The velocity vectors are determined by drawing points from the black normal distribution and scaling the difference vector to the data points to a fixed value. The EM algorithm iteratively finds a target on the grid that fits best to all data points and the corresponding velocity vectors. The red ellipsoid shows the initial and current guesses for the target. In the E-step of the algorithm, for each data point a projection along the velocity vector is computed that fits best to the current estimate of the target (blue dashed lines). In the M-step, a new estimate for the distribution parameter is found.

with normalization constant Z_c. Due to the constraint $t \geq 0$, we can assume that the posterior of t is distributed to a truncated normal distribution

$$p\left(t \middle| \left[\boldsymbol{x}, \boldsymbol{v}\right], \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \frac{1}{\sqrt{(2\pi)}\sigma_t} \frac{1}{1 - \Phi(-\frac{\mu_t}{\sigma_t})} e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}}, \tag{3.5}$$

$$\mu_t = \frac{\boldsymbol{v}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{x})}{\boldsymbol{v}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{v}}, \tag{3.6}$$

$$\sigma_t = (\boldsymbol{v}^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{v})^{-\frac{1}{2}}. \tag{3.7}$$

Here, Φ is the cumulative distribution function of a normal distribution with standard deviation one and mean zero. To avoid running into numerical issues, we set μ_t to zero and σ_t to one if the norm of the velocity is below a threshold. The joint probability of the data point and its projection is defined as

$$p\left(\left[\boldsymbol{x}, \boldsymbol{v}\right], t | \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = p\left(\left[\boldsymbol{x}, \boldsymbol{v}\right] \middle| t, \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) p\left(t \middle| \boldsymbol{\mu}, \boldsymbol{\Sigma}\right). \tag{3.8}$$

When learning the parameters of the distribution, we are not interested in the projection itself. Instead, we want to integrate it out, as it depends on the (unknown) parameters of the distribution. Given a segment $[\boldsymbol{x}_{1:N}$ and $\boldsymbol{v}_{1:N}]$, the complete log-likelihood function is defined as

$$\log p\left(\left[\boldsymbol{x}_{1:N}, \boldsymbol{v}_{1:N}\right] \middle| \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) = \sum_{i=1}^{N} \int_0^{\infty} \log p\left(\left[\boldsymbol{x}_i, \boldsymbol{v}_i\right], t_i \middle| \boldsymbol{\mu}, \boldsymbol{\Sigma}\right) dt_i. \tag{3.9}$$

Here, we assumed that the values t_i are independent of each other. We justify this assumption by the fact that given a fixed $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the t_i's can be computed independently of each other. We would also like to point out that the individual points $\boldsymbol{x}_i$ all come from a continuous trajectory with a continuously changing velocity vector $\boldsymbol{v}_i$. Therefore, the projections $\boldsymbol{x}_i + t_i \boldsymbol{v}_i$ will automatically be close nearby for succeeding data points. In order to infer the parameters of our distribution, we want to maximize the log-likelihood function. The required integrations over t_i is intractable as the variables are restricted to the range $[0, \infty]$. Therefore, in the following section, we derive an Expectation-Maximization (EM) algorithm to get an iterative update scheme for the parameters of the distribution.

3.3.1 Parameter Learning

In the context of the task-decomposition, the aim is to find the best attractor goal for a given set of segments when learning the parameters of the distribution. As the probability is defined as the joint probability of the individual points from the segments, it does not matter if the parameters are learned for a single or multiple segments. Therefore, the notion of a segment is omitted in this section.

The principle of the EM algorithm is depicted in Figure 3.3. In the Expectation step of the algorithm, we use the current parameter values $\boldsymbol{\theta}^{\text{old}} = \{\boldsymbol{\mu}^{\text{old}}, \boldsymbol{\Sigma}^{\text{old}}\}$ for estimating the posterior distribution of the latent variables $p(\boldsymbol{t}\big|\boldsymbol{x}_{1:N}, v_{1:N}, \boldsymbol{\theta}^{\text{old}})$. Here, $\boldsymbol{t} = [t_1, t_2, \ldots, t_N]^T$ is the concatenation of all projections. As the t_i's are independent of each other, the estimation can be done for each data point separately as in (3.5). Intuitively, the E-step projects each data point as close as possible to the current mean of the distribution. In the Maximization step, we get new estimates for our parameters $\boldsymbol{\theta}^{\text{new}}$ by maximizing the expectation of the complete-data log-likelihood under the posterior of the latent variables

$$\begin{aligned}\boldsymbol{\theta}^{\text{new}} &= \arg\max_{\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{t}}\left[\log p\left(\left[\boldsymbol{x}_{1:N}, v_{1:N}\right], \boldsymbol{t}\middle|\boldsymbol{\theta}, \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\right)\right], \\ &= \arg\max_{\boldsymbol{\Theta}} \sum_{i=1}^{N} \int_0^{\infty} p\left(t_i\middle|\left[\boldsymbol{x}_i, v_i\right], \boldsymbol{\theta}\right) \log p\left(\left[\boldsymbol{x}_i, v_i\right], t_i\middle|\boldsymbol{\theta}, \mu_t^{(i)}, \sigma_t^{(i)}\right) dt_i. \end{aligned} \tag{3.10}$$

As the integral is evaluated for each data point separately, we omit the indices for the solution of the integral in the following. The integral evaluates to

$$\begin{aligned}&\int_0^{\infty} p\left(t\middle|\left[\boldsymbol{x}, v\right], \boldsymbol{\theta}\right) \log p\left(\left[\boldsymbol{x}, v\right], t\middle|\boldsymbol{\theta}, \mu_t, \sigma_t\right) dt \\ &= \int_0^{\infty} c_1 e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}} \left(c_2(\boldsymbol{\Theta})t^2 + c_3(\boldsymbol{\Theta})t + c_4(\boldsymbol{\Theta})\right) dt \\ &= c_1\left(c_2(\boldsymbol{\Theta})d_2 + c_3(\boldsymbol{\Theta})d_3 + c_4(\boldsymbol{\Theta})d_4\right). \end{aligned} \tag{3.11}$$

The values of the constants can be found in Appendix B.1. Note that c_2, c_3, and c_4 are constants only for the integration, as they depend on the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ which we want to estimate. Upon evaluating this integral, the parameters can be obtained using (3.10). As the constants c_2 to c_4 depend on the parameters, we compute the derivatives of (3.11) with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ and set them to zero. Now we work again on the entire data set, so we will use the indices again.

$$\begin{aligned}&\frac{\partial \mathbb{E}_{\boldsymbol{t}}\left[\log p\left(\left[\boldsymbol{x}_{1:N}, v_{1:N}\right], \boldsymbol{t}\middle|\boldsymbol{\theta}, \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\right)\right]}{\partial \boldsymbol{\mu}} \\ &= \sum_{i=1}^{N} c_1^{(i)} \left(d_2^{(i)} \frac{\partial c_2^{(i)}}{\partial \boldsymbol{\mu}} + d_3^{(i)} \frac{\partial c_3^{(i)}}{\partial \boldsymbol{\mu}} + d_4^{(i)} \frac{\partial c_4^{(i)}}{\partial \boldsymbol{\mu}}\right) \\ &= \boldsymbol{\Sigma}^{-1} \sum_{i=1}^{N} c_1^{(i)} \left(d_3^{(i)} v_i - d_4^{(i)} \boldsymbol{\mu} + d_4^{(i)} \boldsymbol{x}_i\right) = \mathbf{0}. \end{aligned}$$

Multiplying by $\boldsymbol{\Sigma}$ from the left and rearranging leads to the solution

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^{N} c_1^{(i)} d_4^{(i)} \boldsymbol{x}_i + c_1^{(i)} d_3^{(i)} v_i}{\sum_{i=1}^{N} c_1^{(i)} d_4^{(i)}}. \tag{3.12}$$

Note that the cumulative distribution function Φ is closely related to the error function erf by the relation

$$\Phi\left(-\frac{\mu_t}{\sigma_t}\right) = \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right).$$

Algorithm 3 EM-algorithm for DND

Require: data $\boldsymbol{x}_{1:N}$, velocities $\boldsymbol{v}_{1:N}$, initial mean $\boldsymbol{\mu}$, initial cov $\boldsymbol{\Sigma}$

```
while not converged do
    for all data points x_i, v_i do // Start E-step
        compute constants c_1^(i) and d_2^(i), ..., d_4^(i) (B.1)
    compute new mean μ(c_1, d_3, d_4) (3.12) // Start M-step
    compute new covariance matrix Σ(c_1, d_2, d_3, d_4) (3.13)
return mean μ, covariance matrix Σ
```

Therefore, the constants $c_1^{(i)} d_3^{(i)}$ and $c_1^{(i)} d_4^{(i)}$ could be further simplified. For instance, $c_1^{(i)} d_4^{(i)}$ equals one (proof skipped here) and so (3.12) can also be written as

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \left(\boldsymbol{x}_i + c_1^{(i)} d_3^{(i)} \boldsymbol{v}_i \right).$$

For estimating the covariance matrix $\boldsymbol{\Sigma}$, we compute the derivative with respect to its inverse $\boldsymbol{\Sigma}^{-1}$.

$$\begin{aligned}
&\frac{\partial \mathbb{E}_{\boldsymbol{t}} \left[\log p \left(\left[\boldsymbol{x}_{1:N}, \boldsymbol{v}_{1:N} \right], \boldsymbol{t} \middle| \boldsymbol{\theta}, \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t \right) \right]}{\partial \boldsymbol{\Sigma}^{-1}} \\
&= \sum_{i=1}^{N} c_1^{(i)} \left(d_2^{(i)} \frac{\partial c_2^{(i)}}{\partial \boldsymbol{\Sigma}^{-1}} + d_3^{(i)} \frac{\partial c_3^{(i)}}{\partial \boldsymbol{\Sigma}^{-1}} + d_4^{(i)} \frac{\partial c_4^{(i)}}{\partial \boldsymbol{\Sigma}^{-1}} \right) \\
&= \frac{1}{2} \sum_{i=1}^{N} c_1^{(i)} \Big(- d_2^{(i)} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} + d_3^{(i)} \boldsymbol{v}_i (\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}} \\
&\quad + d_3^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i) \boldsymbol{v}_i^{\mathrm{T}} - d_4^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i)(\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}} + d_4^{(i)} \boldsymbol{\Sigma} \Big) = \boldsymbol{0}.
\end{aligned}$$

Rearranging then leads to the solution

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} c_1^{(i)} \left(d_2^{(i)} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} - d_3^{(i)} \boldsymbol{v}_i (\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}} - d_3^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i) \boldsymbol{v}_i^{\mathrm{T}} + d_4^{(i)} (\boldsymbol{\mu} - \boldsymbol{x}_i)(\boldsymbol{\mu} - \boldsymbol{x}_i)^{\mathrm{T}} \right). \tag{3.13}$$

In summary, the parameters of the distribution can be estimated iteratively by first computing the c and d constants for each data point according to Appendix B.1 (E-step). Second, a new estimate for the parameters of the distribution can be found by computing $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ according to (3.12) and (3.13) (M-step). The algorithm is summarized in Algorithm 3.

3.3.2 Extension for Orientations

The Euclidean distance measure used in (3.3) is not suitable for 3D orientations. In this section, we show that by making some reasonable approximations, we can make our algorithm applicable also for orientations.

In order to be applicable, we have to define a projection $\boldsymbol{x} + t\boldsymbol{v}$ and a difference vector for orientations that can be written in the form $\boldsymbol{x} + t\boldsymbol{v} - \boldsymbol{\mu}$. We define the target orientation in the inertia frame with the rotation matrix $\boldsymbol{R}_{\mathrm{FI}}$. The matrix $\boldsymbol{R}_{\mathrm{VI}}(t) = \boldsymbol{R}_{\mathrm{VS}}(t) \boldsymbol{R}_{\mathrm{SI}}$ represents the start orientation rotated with the angle t around the angular velocity axis. The rotation matrix is also given in the inertia frame. Hence, the projection parameter t has a different meaning than for Euclidean task-spaces. Instead of projecting

a position along its velocity vector, it rotates an orientation by an angle around the angular velocity axis. By writing the matrices as vectors $\boldsymbol{r} \in \mathbb{R}^{9\times 1}$, we can define a difference vector with

$$\begin{aligned}
\boldsymbol{r}_{\mathrm{VI}}(t) - \boldsymbol{r}_{\mathrm{FI}} &= \boldsymbol{d}\cos(t) + \boldsymbol{e}\sin(t) + \boldsymbol{f} - \boldsymbol{r}_{\mathrm{FI}} \\
&\approx \boldsymbol{d}(\cos(\alpha) - \sin(\alpha)(t-\alpha)) \\
&\quad + \boldsymbol{e}(\sin(\alpha) + \cos(\alpha)(t-\alpha)) + \boldsymbol{f} - \boldsymbol{r}_{\mathrm{FI}}, \\
&= \boldsymbol{x} + t\boldsymbol{v} - \boldsymbol{\mu}, \\
\boldsymbol{x} &= \alpha\cos(\alpha)(-\boldsymbol{d} - \boldsymbol{e}) \\
&\quad + \alpha\sin(\alpha)(-\boldsymbol{d} + \boldsymbol{e}) + \boldsymbol{f}, \\
\boldsymbol{v} &= \cos(\alpha)(d+e) + \sin(\alpha)(-d+e), \\
\boldsymbol{\mu} &= \boldsymbol{r}_{\mathrm{FI}}.
\end{aligned} \tag{3.14}$$

The values of the constants $\boldsymbol{d}$, $\boldsymbol{e}$ and $\boldsymbol{f}$ can be found in Appendix B.2. Now our difference vector can be written in the form $\boldsymbol{x} + t\boldsymbol{v} - \boldsymbol{\mu}$ and we can estimate the target orientation with the same algorithm as in the previous section. Still, some details have to be taken care of. First, we approximated the sine and cosine terms with a first-order Taylor approximation around α. We therefore have to evaluate how good the approximation is. Second, the approximation introduced the parameter α and it is not clear how to choose it. Third, estimating the parameters of the distribution with our standard algorithm will result in a value for $\boldsymbol{\mu}$ which does not describe a proper orientation. Therefore, after applying our standard decomposition method, we reshape the target vector to a matrix and find the closest true rotation matrix to our matrix using the approach of Higham [1989].

In the following, we introduce an intuitive way of choosing α. A natural way of measuring the difference between two orientations is the axis angle $\theta(t)$ of the relative transformation $\boldsymbol{R}_{\mathrm{FV}} = \boldsymbol{R}_{\mathrm{FI}}(\boldsymbol{R}_{\mathrm{VI}})^{\mathrm{T}}$

$$\begin{aligned}
\theta(t) &= \cos^{-1}\left(\frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{R}_{\mathrm{FV}}) - 1\right)\right) \\
&= \cos^{-1}\left(\frac{1}{2}\left(a\cos(t) + b\sin(t) + c - 1\right)\right).
\end{aligned} \tag{3.15}$$

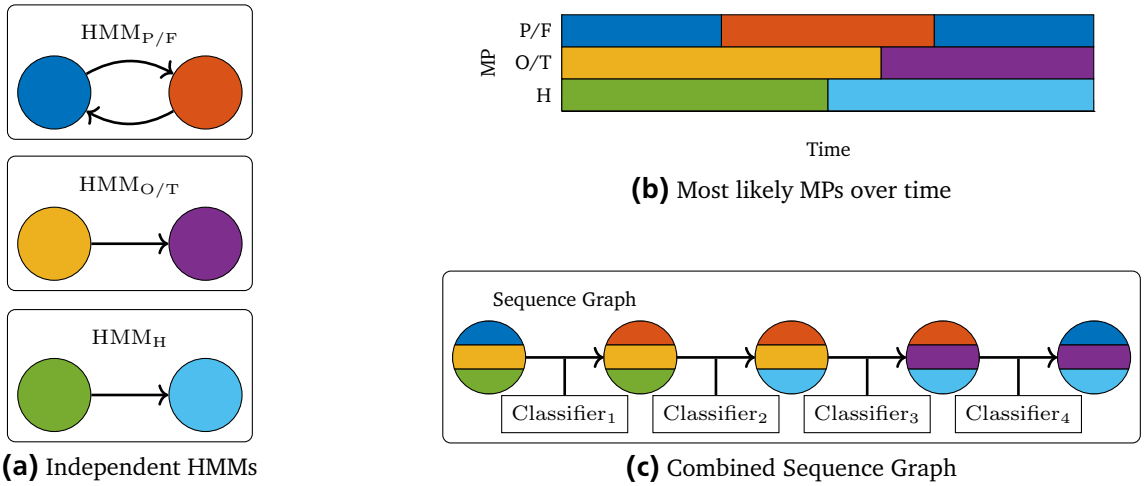
The constants a, b and c can be derived straightforwardly. For further details, see B.2.1. In order to project the current orientation as closely as possible to the target orientation, we compute the derivative of (3.15) with respect to t and set it to zero

$$\begin{aligned}
\frac{\partial\theta}{\partial t} &= \frac{0.5a\sin(t) - 0.5b\cos(t)}{\sqrt{1 - \left(\frac{1}{2}\left(a\cos(t) + b\sin(t) + c - 1\right)\right)^2}} \\
&= 0 \quad \Rightarrow \quad t = \mathrm{atan2}(b, a).
\end{aligned} \tag{3.16}$$

Hence, if we knew the target orientation without any uncertainty, we could compute a distance measure between the current orientation and the target in closed form. For the Taylor approximation, we can therefore make use of a trick. If we neglect the uncertainty $\boldsymbol{\Sigma}$, we can set the target orientation to the current estimate of the mean $\boldsymbol{\mu}$. Subsequently, we compute $\tilde{t}$ according to (3.16) and use it for the Taylor approximation $\alpha = \tilde{t}$. Therefore, we get a good approximation by developing the Taylor series about a value close to the true t. The approximation allows us to integrate orientations in our method in a straightforward way.

3.4 Movement Primitive Sequence Learning

In the previous sections, we introduced our task-decomposition method. Applying the method results in a set of MPs and a HMM which reflects potential sequential orderings of the MPs. In fact, we apply the



(a) Independent HMMs

(b) Most likely MPs over time

(c) Combined Sequence Graph

Figure 3.4.: Illustration of the sequence learning. The three HMMs (a) resulting from the task-decomposition are used for labeling the demonstrations with the most likely MPs (different colors) over time (b). The labeled demonstrations are compiled into a single sequence graph representing potential MP orders (c). Local classifiers are responsible for transitioning between successive MPs when executing the skill on a real robot.

method three times with data from different task-spaces, as the MPs controlling the position/force of the end-effector, its orientation and the joint angles of the fingers are learned independently of each other. While extracting the MPs of these three different entities independently of each other simplifies learning and leads to simple reusable MP, the MPs cannot be treated independently of each other when executing a skill on a real robot. For instance, the end-effector may only be allowed to move after an object has been grasped. Therefore, the decision when to execute which MP for one entity has to be conditioned on the state of the two other entities.

In order to account for the necessary synchronization of the MPs, we utilize the sequence graph concept presented in the previous chapter and adapt it for our purpose. Our overall aim is to learn a skill that has a reactive behavior. Instead of following a pre-computed path, the system is supposed to decide online when to stop a movement and which movement to start next. Thus, it has to decide when to activate which of the MPs based on the current state of the robot and the state of the environment (e.g., current positions of objects in the scene). For a successful execution of a learned skill, the MPs have to be activated at the correct time points and in the correct order. The decision has to be made based on the feature state $\boldsymbol{f} \in \mathbb{R}^{N_f \times 1}$, where N_f is the dimension of the feature space. In theory, this feature state can be comprised of arbitrary values, but in this chapter, it will be equivalent to the task-space data $\boldsymbol{x}$. Thus, it implicitly encodes the state of the robot in relation to all objects in the scene (including the forces) and contains the joint angles of the fingers as well as the measured forces.

In the standard formulation of the sequence graph, a node corresponds to a MP and each node is associated with a local classifier. When executing a skill, one node in the graph is considered active and the associated classifier decides when to transition to a succeeding node, leading to the activation of another MP. In order to synchronize the activations of the MPs of the three different entities, we use a single sequence graph where the active node activates three MPs, one for each controlled entity. This process is depicted in Figure 3.4. First, we use the HMMs resulting from the task-decomposition for labeling the demonstrations with the most likely MPs over time. From these labels, we generate the sequence graph representing the sequential structure of the MPs. Finally, we train the local classifiers for each node, where a classifier maps the feature state to three activation vectors $\boldsymbol{f} \rightarrow \{\boldsymbol{a}_{\mathrm{P/F}}, \boldsymbol{a}_{\mathrm{O/T}}, \boldsymbol{a}_{\mathrm{H}}\}$, one for each controlled entity. Therefore, when executing a learned skill on a robot, a transition leads to the activation of (up to) three new MPs. We use Logistic Regression for training, but in general

any classifier can be used. Note that there are also alternative more sophisticated ways to synchronize the sequences (e.g., generate three graphs and couple their transitions), but finding the best way to synchronize the controlled entities is beyond the scope of this thesis.

3.5 Evaluation of the Approach

For evaluating our approach, we performed kinesthetic demonstrations on a gravity compensated seven degree of freedom (DoF) Barrett WAM robot with a four DOF hand. We performed three different tasks: box flipping, box stacking and unscrewing a light bulb. For all tasks, we evaluate the decomposition and reproduction of the task. The purpose of the first task is to evaluate if our method is able to distinguish between position and force control. For the box stacking task, the focus is on finding the correct coordinate frames. The light bulb unscrewing task is a longer and more complex task which requires both, i.e., performing movements relative to multiple objects as well as distinguishing force from position control. For the box stacking task, we also compare our decomposition method to a baseline method and a state-of-the-art method.

For simplicity, we predefined the MPs controlling the fingers of the robot for all tasks, despite that teaching grasps is possible with our approach. During the demonstrations, we activated these MPs manually by pressing a key on a keyboard. Note that we predefined the MPs only for simplifying the teaching process. We still applied our task-decomposition to the resulting finger joint angles data. The kinesthetic demonstrations were recorded with 200Hz. As the force-torque measurements are quite noisy, we filtered the data using a Hann filter with a window size of 100. Due to the noisy force sensor and the different units (Newton vs. meters) the force data was additionally scaled by the factor $1/40$, so that $4\,\mathrm{N}$ difference correspond to a position difference of $10\,\mathrm{cm}$. For all experiments, force was chosen as control variable for a dimension of a task-space if the target force was above the threshold of $5.0\,\mathrm{N}$ and the effector was not moving. An effector was considered to be not moving if the average velocity was below the threshold of $10^{-3}\,\mathrm{m/s}$ (see Section 3.2.3).

3.5.1 Box Flipping

For the box flipping task, the sequence of subtasks demonstrated by the teacher is shown in Figure 3.5. In order to flip the box, the teacher first pushed it against the obstacle. Then, he flipped it by pushing the box against the obstacle while rotating it. We demonstrated the task five times with slightly varying starting positions of the box. Two coordinate frames relative to the obstacle were defined for the task-decomposition. A Cartesian frame and a cylindrical coordinate frame whose axis was aligned with the long side of the obstacle, as shown in Figure 3.6. Kober et al. [2015] evaluated their approach on a similar box flipping task. They also used a cylindrical coordinate frame as it works very well for rotatory movements such as opening a door. Additionally, they stated that it is very robust with respect to inaccuracies in the rotation axis which is important when using force control. In contrast to their approach, we did not have to align the demonstrations in time to decompose the task. The orientation of the end-effector and the fingers were held constant throughout the demonstrations and are therefore neglected here.

Our task-decomposition algorithm resulted in four MPs. Three of the five demonstrations and the corresponding results are depicted in Figure 3.7. The results show that the decomposition is consistent over all demonstrations, even though the demonstrations are slightly over-segmented. The MPs closely resemble the behavior of the teacher (see Table 3.1). Initially, the ●-MP approaches the box. Subsequently, the ●-MP pushes the object against the box. Then, the ●-MP flips the box and the ●-MP moves the end-effector to its final position. For the first two MPs, the algorithm chose the Cartesian frame, while for the latter two, it chose the Cylindrical frame. The frame choices can be better understood when looking at the demonstrations plotted spatially (see Figure 3.7b). The box flipping resulted in a (blue) curved

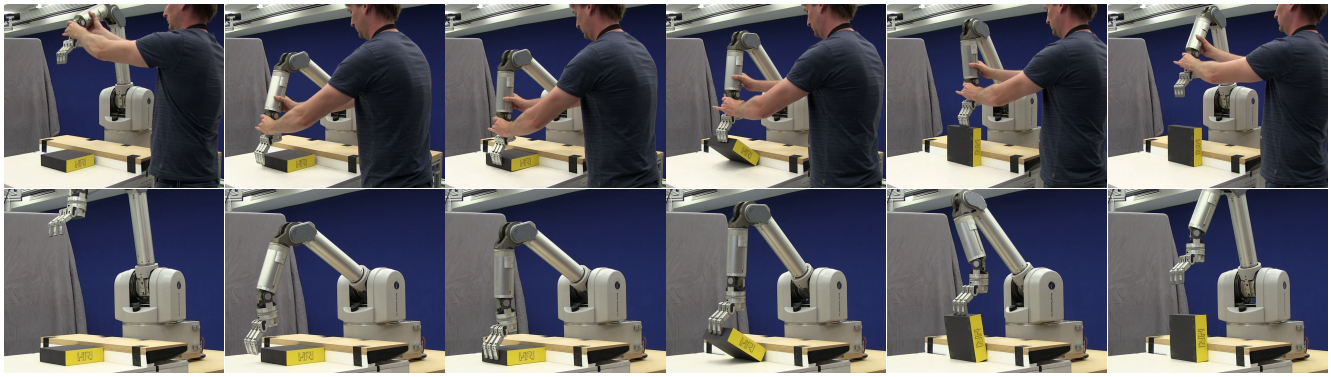
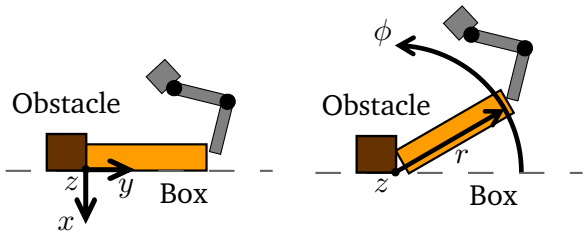


Figure 3.5.: Illustration of teaching and reproduction of the box flipping task. The task was to initially move the end-effector to a position close to the box. Subsequently, the box was pushed against the obstacle. Then, the box was flipped. Finally, the end-effector was moved to its final position.



(a) Cartesian Frame **(b)** Cylindrical Frame

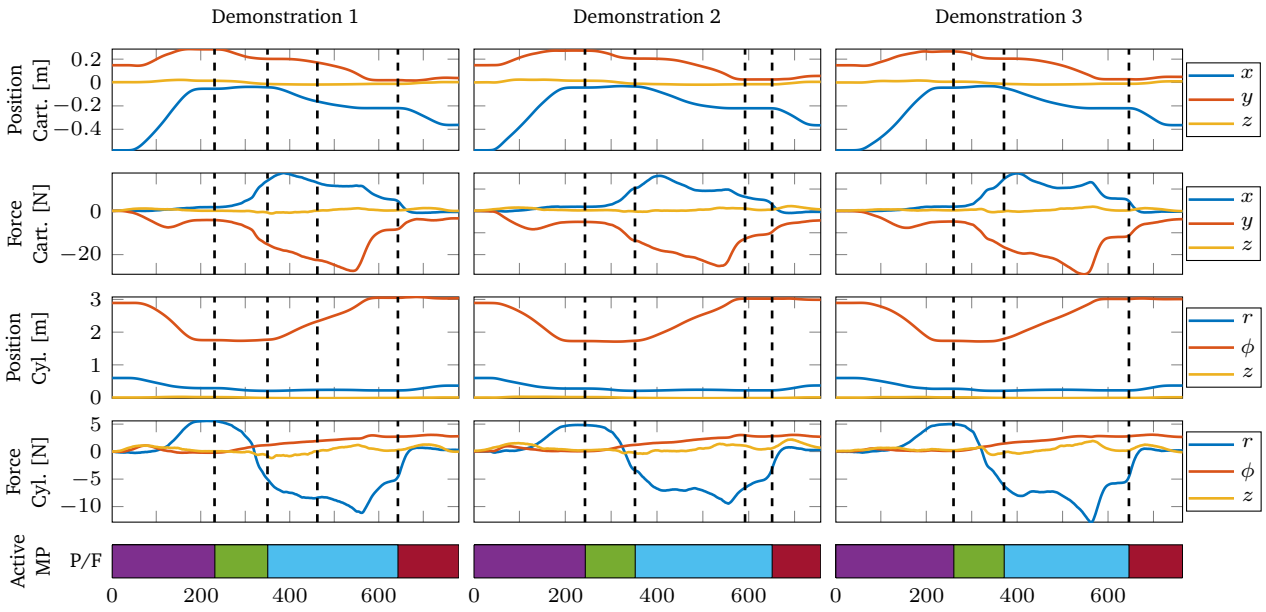
Figure 3.6.: The two coordinate frames of the box flipping task shown from the side. The origin of the Cartesian frame is at the lower edge of the obstacle. The z-axes of both frames are identical and correspond to the lower edge of the obstacle (indicated by a black dot). Here, the origin is at the center of the lower edge. Note that the frames are attached to the obstacle and not to the box.

line in Cartesian space and a straight line in the cylindrical space, which is why the algorithm choose the cylindrical frame for the ●-MP. For the remainder of the task, the teacher tried to move the end-effector in a straight line. Still, the algorithm chose the cylindrical frame for the final ●-MP. As shown in the plots, the final positions slightly varied over the demonstrations and the teacher did not really move the end-effector in a straight line. The low confidence of 51.5% shows that the demonstrations seem to converge slightly better to a desired attractor goal in the Cylindrical coordinate frame.

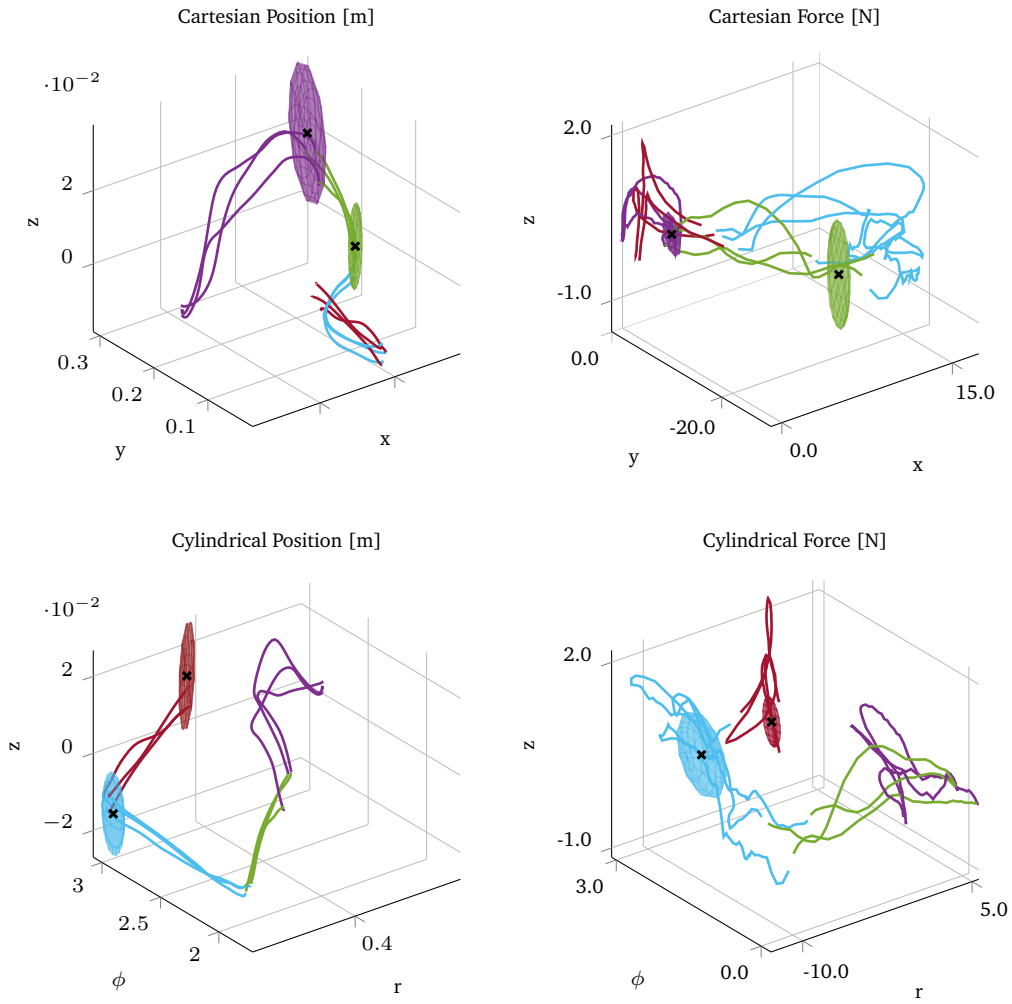
Only for the box flipping MP (●), the algorithm chose force control for one dimension of the coordinate frame. The reason is that while this MP was active, the radial distance to the origin of the coordinate frame was constant and hence the corresponding task coordinates did not change. As the target force of the MP along the radial axis of the coordinate frame exceeded the predefined threshold of $5.0\,\mathrm{N}$ and no velocity was measured, force control was chosen for this dimension of the task-space.

MP	Description	Frame	Confidence	Force
●	Approach Box	Cartesian	95.2%	-
●	Push Box	Cartesian	58.3%	-
●	Flip Box	Cylindrical	98.4%	Radial
●	Approach Final Position	Cylindrical	51.5%	-

Table 3.1.: Resulting description, coordinate frame and force selection for each MP of the box flipping task. Only the MP that is active when flipping the box is applying a force along the radial axis of the cylindrical coordinate frame. The confidence corresponds to the mixture weight of the winner frame.



(a) Demonstrations and task-decomposition over time.



(b) Demonstrations and task-decomposition in $3D$ space.

Figure 3.7.: Experimental results for the box flipping task. The MP plot shows the active MP in different colors. It can be seen from the plots that the decomposition of the task is consistent throughout all demonstrations.

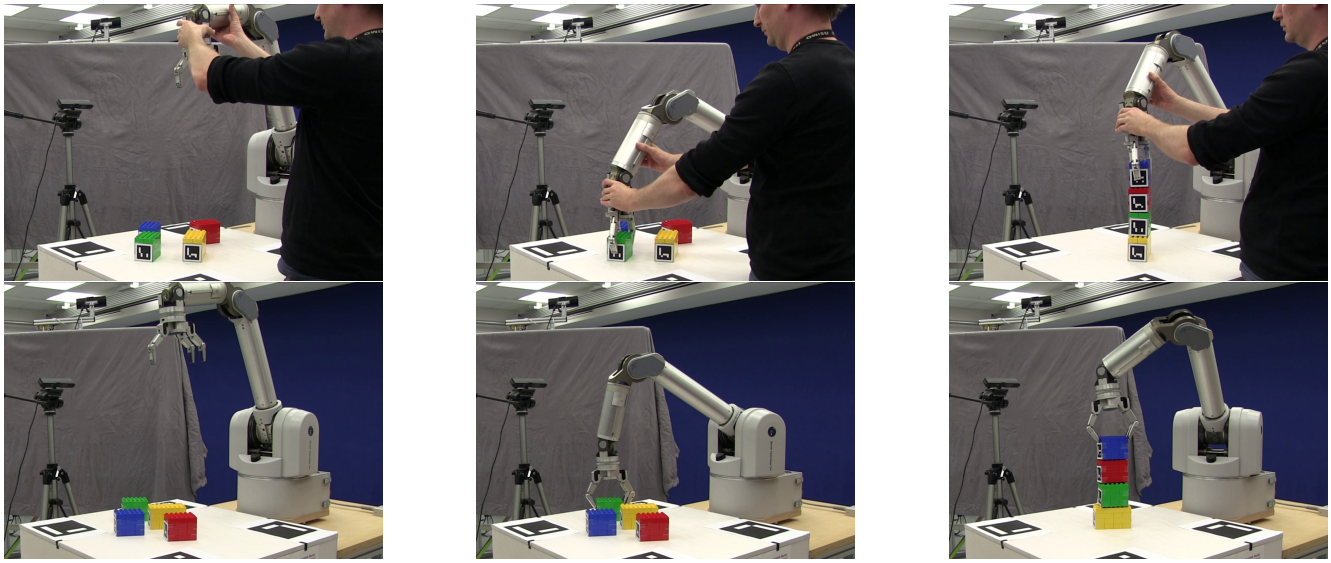


Figure 3.8.: Experimental setup for the box stacking task (top) and pictures from the reproduction (bottom). The four boxes were put to random initial positions for each demonstration and the reproduction. They were tracked using AR markers.

After decomposing the task and learning to sequence the MPs, the learned skill was successfully reproduced on the real robot. The found MPs were sufficient for performing the necessary movements and the transitions between MPs were triggered at the correct states. For the sequence learning, the following features were used: The end-effector position in the cylindrical frame and the Cartesian frame as well as the data from the force sensor at the wrist of the robot.

3.5.2 Box Stacking

The goal of the second task is to stack four boxes on top of each other. Position and orientation of the boxes are tracked with Augmented Reality (AR) markers using the ArUco library [Garrido-Jurado et al., 2014]. We performed three demonstrations of the task. For each demonstration, the initial positions and orientations of the boxes were chosen randomly. The stacking sequence was the same for all demonstrations. First, the green box was put on the yellow box. Then, the red box was put on the green box and finally, the blue box was put on the red box. Figure 3.8 shows a demonstration of the task. For a successful stacking of the boxes, it is important that all MPs use the correct coordinate frames. The task does not require any force control and therefore the force data is omitted here.

As an example, Figure 3.9 depicts the data of one demonstration and the corresponding sequence of most likely MP activations resulting from the task-decomposition. In total, our algorithm extracted seven position MPs, four orientation MPs and the two hand MPs corresponding to the ones we defined for grasping and opening the fingers. The MPs can be described as follows. In the beginning, the end-effector approaches the green box (●) with opened fingers (●). During this phase, the orientation is also controlled relative to the green box. After grasping the box (●), the end-effector is moved to a position above of the yellow box (●), with the orientation also aligned to the yellow box. Subsequently, it is lowered a few centimeters (●) and the box is released. To avoid overturning the box stack, the end-effector is then moved up again (●) before moving on to the next box. This scheme is continued until the task is complete. Three of the seven position MPs approach the different boxes that are supposed to be stacked on the yellow box. The remaining four MPs all represent positions above of the yellow box with different heights. The different heights result from the growing size of the box stack and are consistent with the data.

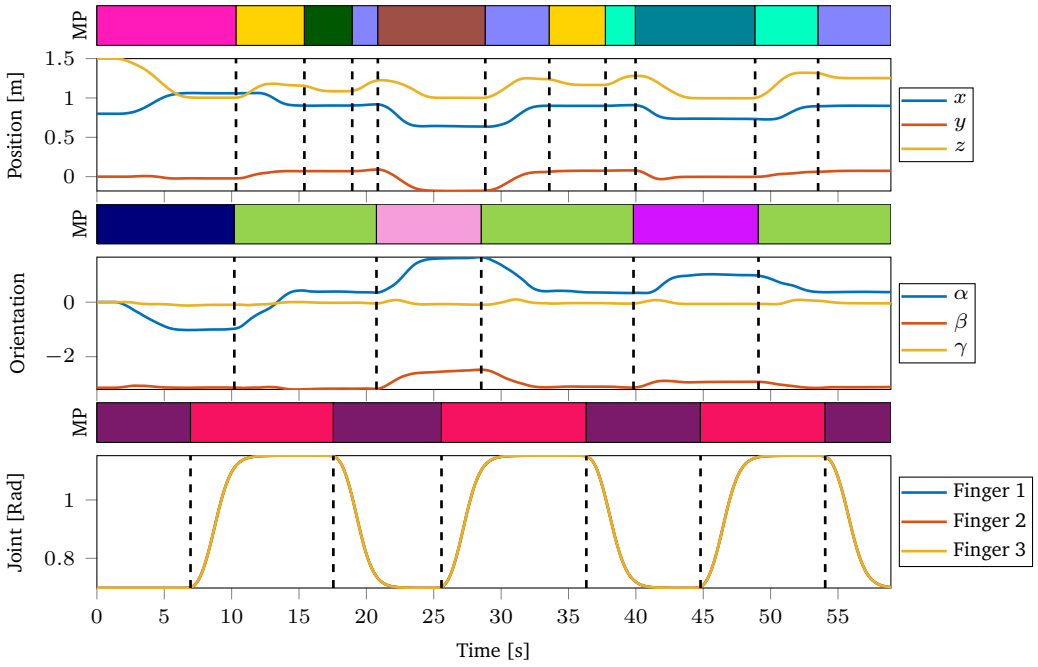


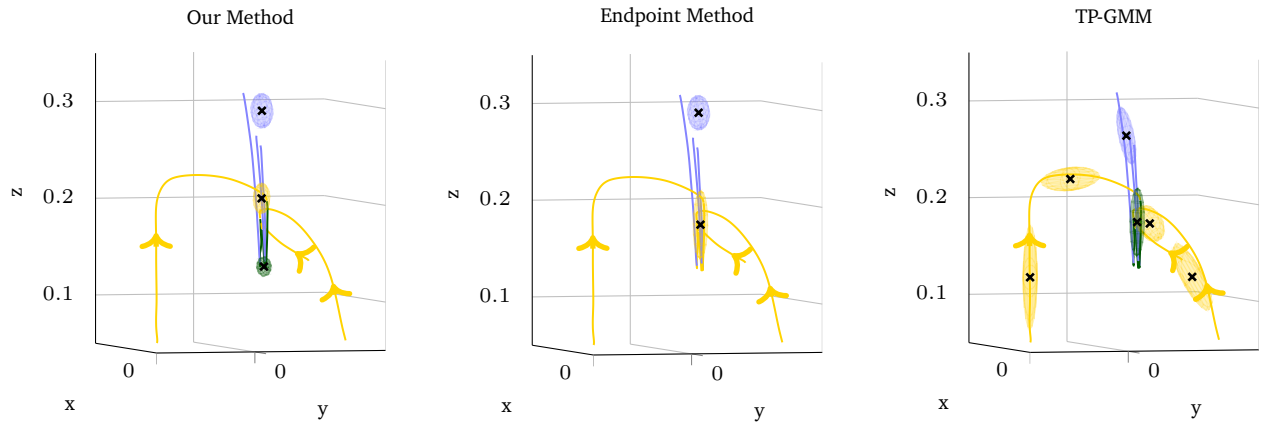
Figure 3.9.: One demonstration (in the world frame) and resulting task-decomposition for the box stacking task.

For the box stacking task, we also compared our method against three other approaches. As baseline, we clustered the end-points of the segments using a HMM with GMMs as state emissions. Thus, in contrast to our approach, this approach does not take into account the entire segment and its direction. The second approach is the TP-GMM, introduced by Calinon et al. [2012] and for instance used by Rozo et al. [2016]. The TP-GMM is a hierarchical Gaussian Mixture Model with two layers. On the higher layer, each mixture represent an MP. On the lower layer, the mixtures represent the trajectory in the different task-spaces. The main difference to our approach is the semantic of an MP. While in our approach an MP describes a goal-directed movement, the TP-GMM models a joint distribution of the position and velocities. By conditioning on the current position, the model can also be used to generate desired velocities, allowing the model to be used for the reproduction directly. The third approach is the BP-AR-HMM utilized by Niekum et al. [2015b] for decomposing a task into a set of MPs. Their approach uses the Beta-Process Autoregressive HMM for segmenting the demonstrations in the world frame. After training, they cluster the end-points of all segments assigned to a single MP in each coordinate frame. The cluster with the minimum variance is chosen as coordinate frame for the MP. In contrast to our approach, the baseline and the TP-GMM, the authors first segment the demonstrations in the world frame and later assign coordinate frames to the resulting MPs. For both approaches, we did our best to tune the parameters of the corresponding methods. We did not implement the methods ourselves, but instead used the well-tested code the authors provide on their website. The TP-GMM is implemented in the PbDlib[2]. The Matlab code for the BP-AR-HMM is provided by Scott Niekum[3].

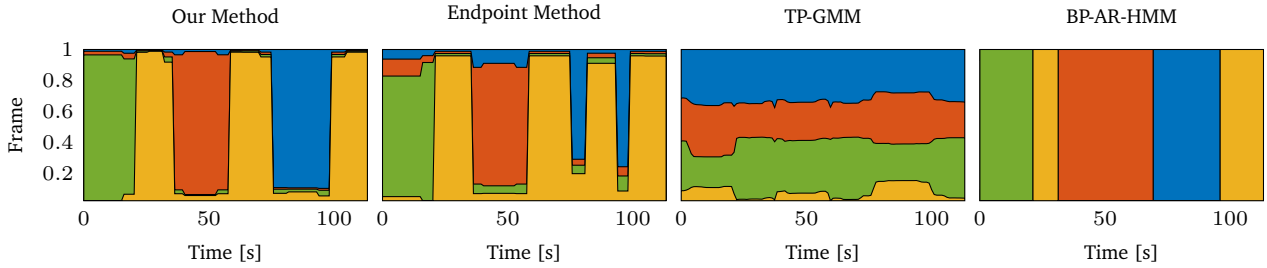
The decomposition resulted in a total of 13 MPs for our approach and the baseline approach, 9 MPs for the BP-AR-HMM and 65 MPs for the TP-GMM (for position, orientation and hand). The number of MPs for our approach, the baseline method and the TP-GMM has been determined using the Bayesian Information Criterion. For the BP-AR-HMM, we changed the hyperparameters of the method until we were satisfied with the results. A comparison of the resulting decompositions for all approaches is shown in Figure 3.10. While our approach clearly discriminates the steps the teacher performed for stacking the boxes (e.g., align green box over yellow box before stacking), the baseline approach merges some of these steps to a single MP. Even though the resulting MPs are similar to the MPs of our approach,

[2] http://www.idiap.ch/software/pbdlib

[3] http://www.cs.utexas.edu/~sniekum/code.php



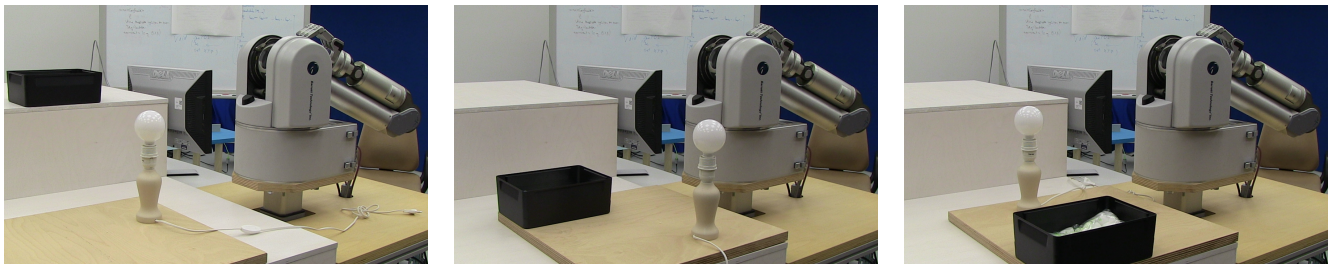
(a) In this snippet of the box stacking task decompositions, the green box was stacked on the yellow box. The decompositions are shown spatially in the coordinate frame of the yellow box. The colors correspond to the colors from Figure 3.9. Compared to our approach, the baseline approach merges the red and green MP to a single MP. The TP-GMM in general needs more MPs, as it models a path instead of MP targets. As it is difficult to visualize the parameters of the auto-regressive model, the BP-AR-HMM is not shown here.



(b) For one full demonstration, the most likely coordinate frames over time are shown in the lower plots. The colors indicate the active coordinate frame (frame of the yellow, green, red or blue box). Our approach clearly distinguishes between phases of getting the different boxes and stacking them on the yellow box. The other approaches either do not separate the phases in such a clear way or (TP-GMM) or pick the wrong coordinate frame in some task phases (baseline and BP-AR-HMM).

Figure 3.10.: Comparison of our method with a baseline approach and two state-of-the-art approaches. For the baseline approach, we clustered the end-points of each segment using a HMM with GMMs as state emissions. Further discussions of the results can be found in Section 3.5.2.

they cannot be used for reproducing the task properly. The green box will not be stacked on the yellow box, but will rather be released above the yellow box. In addition, it may not be aligned properly. The green box may land correctly on the yellow box, but success will be rather random. The results show that it is not sufficient to cluster the end-points of the segments, as it becomes harder to assign the proper coordinate frames to the different segments. By incorporating the convergence properties of the segments, our approach yields better results compared to the baseline approach. The TP-GMM results in significantly more MPs compared to our approach for two reasons. First, as it models the whole trajectory instead of only the attractor goals, more MPs are needed in general. Second, if for instance an object is approached from two different directions, the two trajectories will be very different (even in the coordinate frame of the object), but their target position will be the same. Therefore, our approach is able to model such movements with fewer MPs compared to the TP-GMM. While the TP-GMM has the advantage of being able to model arbitrarily non-linear trajectories, the coordinate frame assignments made by our model better resemble the natural structure of the demonstrated task (see Figure 3.10b)). We consider it future work to combine the advantages of both models. The BP-AR-HMM results in the fewest number of MPs. While the resulting coordinate frame assignments in Figure 3.10b look promising at first, the phase where the red box is stacked on the yellow box is represented in the wrong coordinate frame. The reason is that the demonstrations are segmented in the world frame before assigning the



(a) (b) (c)

Figure 3.11.: The three different experimental setups for the light bulb unscrewing task. For each setup, the light bulb holder and box were put to different locations on the two tables.

MP	Description	Frame
	Approach Light Bulb	Light Bulb
	Pull Bulb (Force in z)	Light Bulb
	Lift Bulb	Light Bulb
	Approach Box	Box
	Approach Final Position	Box
	Screw	World
	Unscrew	World
	Open Fingers	-
	Close Fingers	-

Table 3.2.: Resulting coordinate frame and description for each MP. Only the MP that is active during unscrewing applies a force, as the teacher was pulling the light bulb during this phase of the task. Two MPs control the orientation of the end-effector and the fingers, respectively.

coordinate frames to the individual MPs. As a result, segments which could be assigned to the same MP may be assigned to different MPs, as they are dissimilar in the world frame. Therefore, we believe that the coordinate frame assignment should be integrated into the segmentation process.

As a proof of concept, we reproduced the task on the real robot on a setup that was not demonstrated to the robot (see Figure 3.8). The reproduction showed that the robot can reproduce the task correctly. All MPs are represented in the correct coordinate frame and sequenced in the correct order. The boxes were successfully stacked on top of each other, even though they were not stacked as accurately as in the demonstrations. Most likely, the reason for the small displacement (approx. $0.5\,\mathrm{cm}$ to $1.0\,\mathrm{cm}$) was due to the imprecision of the Kinect sensor used for tracking the objects.

3.5.3 Light Bulb Unscrewing

For the unscrewing task, the human teacher demonstrated the following sequence of subtasks. First, he approached the light bulb with the end-effector. Subsequently, the robot's hand was closed. Next, the teacher rotated the wrist of the robot arm in order to turn the bulb. During this movement, he also pulled the light bulb (by applying a force along the z-axis), to test whether it is still in its holder. After turning the light bulb, the fingers were opened and the wrist was rotated back. This unscrewing cycle was repeated until the light bulb got loose. Next, the light bulb was pulled out of the holder and the end-effector was moved to a box where the hand was opened again. For the demonstrations, we put the light bulb holder and the box to three different positions and performed three kinesthetic demonstrations for each setup. The setups are depicted in Figure 3.11. Two objects were tracked in the scene, the light bulb holder and the box.

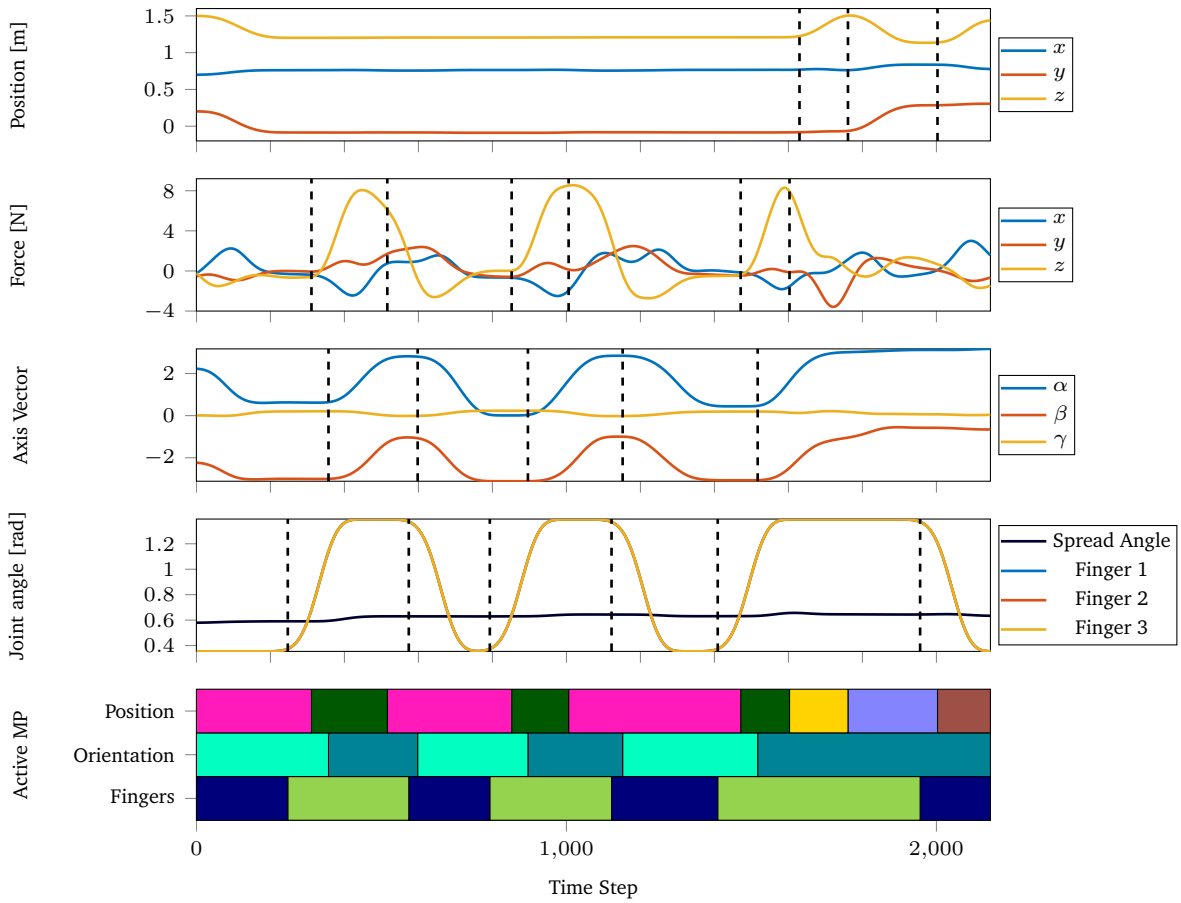
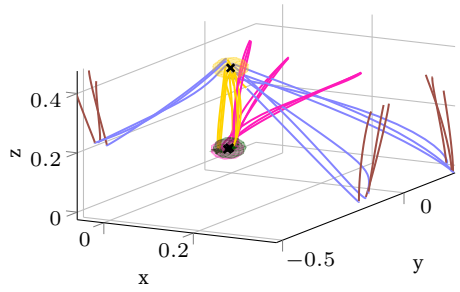


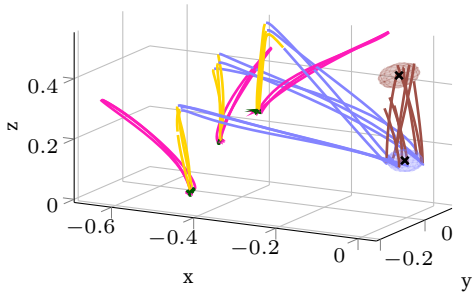
Figure 3.12.: Task-decomposition over time for one of the nine demonstrations. From the top, the plots show the position of the end-effector, the measured forces at the wrist, the orientation of the end-effector and the finger joint angles, respectively. The dashed lines indicate the zero-velocity crossings and contact changes. The plot on the bottom shows the most likely MP for each point in time. All MPs can be associated with a meaningful description (see Table 3.2).

Our method extracted nine MPs from the demonstrations, as shown in Table 3.2. Four MPs control the position of the end-effector and one MP controls the position of the end-effector along the x- and y-axes, while applying a force along the z-axis. Additionally, two MPs control the orientation of the end-effector and two MPs the finger configuration. Figure 3.12 depicts the most likely MP for each point in time for one of the nine demonstrations. Except for small time gaps between changes of position, orientation and/or finger MPs, the MP sequences exactly reflect the subtask sequence the teacher performed and thus we can associate each MP with a meaningful description in Table 3.2. Note that the time gaps are not incorrect but instead reflect the behavior of the teacher. For instance, it is not possible to unscrew the bulb before closing the fingers. Therefore, the teacher usually waits for a short moment until he or she is sure the fingers are closed before starting to unscrew, causing the aforementioned time gaps in the data.

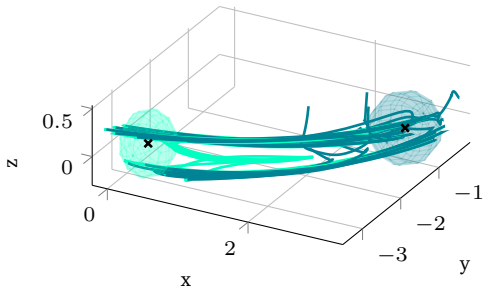
Even though the number of unscrewing repetitions varied over the demonstrations, all demonstrations were decomposed in a similar way. We illustrated this in Figure 3.13, which shows all demonstrations spatially. As all subtasks performed by the teacher can be described relative to either the light bulb or the box, the world frame was not chosen for any MP controlling the position of the end-effector. During the demonstrations, the teacher was standing at a fixed position and aligned the orientations equally for all demonstrations. However, the objects and thus also their corresponding coordinate frames were rotated for the different setups. As a result, our algorithm chose the world frame for all orientation MPs.



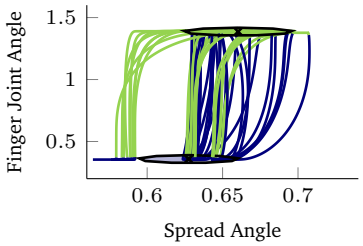
(a) Positions Light Bulb Frame



(b) Positions Box Frame



(c) Orientations World Frame



(d) Fingers World Frame

Figure 3.13.: Task-space trajectories for all nine demonstrations of the light bulb unscrewing task. Plots (a) and (b) show the end-effector positions in the light bulb coordinate frame and box coordinate frame, respectively. The colors indicate to which MP each segment is assigned to. The markers correspond to the mean $\boldsymbol{\mu}$ of the MP targets and the ellipsoids indicate their covariance matrices $\boldsymbol{\Sigma}$. Note that the position target of the blue MP is hard to see because it coincides with the target of the red MP. The targets are only plotted in the coordinate frame that was assigned to each MP. Plots (c) and (d) show the orientations of the end-effector and the finger configurations, respectively (both in the world frame). As for this task all three fingers were aligned equally, only the joint angle of one finger is shown in (d).

The MPs controlling the DoFs of the hand are not associated with a coordinate frame as they directly control the joints.

For all position MPs, the resulting target forces are depicted in Table 3.3. Only for the ●-MP the target force along the z-axis is large enough so that the MP is force-controlled along this axis. The MP corresponds to the teacher pulling the light bulb while unscrewing. When reproducing the task, this force leads to an acceleration of the robot's arm as soon as the light bulb gets loose, enabling the system to detect the loose bulb without any external sensors. The drawback is that we cannot detect whether the light bulb is still in the hand of the robot or if it slipped during reproduction.

For the reproduction of the task, we set the light bulb holder to five different positions and performed two trials for each position. In all ten trials, the robot was able to unscrew the light bulb and successfully put it in the container box.

3.5.4 Discussion of the Experiments

The experimental results show that our method segments the demonstrations properly and finds a meaningful decomposition for all tasks. The reproduction showed that our system can learn to perform the tasks from scratch in an unsupervised fashion. Still, the method has some properties that need to be discussed and some limitations that need further research.

MP	x	y	z
●	0.08	-1.43	-0.51
●	-1.08	0.30	**7.12**
●	1.08	-1.35	0.02
●	0.06	0.01	-0.17
●	-0.24	-0.85	-0.88

Table 3.3.: Resulting target forces in Newton in the corresponding frames of the position MPs. Only the ●-MP is force-controlled along the z-axis because the desired force is large enough.

As already mentioned, we scaled the force data before training to compensate for the noisy sensor and the different units (Newton vs. meters). Additionally, the scale factor reflects an important issue occurring in kinesthetic demonstrations. While it is easy to guide a gravity compensated robot to a desired position, applying desired forces is difficult. For instance, in the light bulb task we tried to pull the bulb always with the same force. Still, the forces were roughly in a range of 5 to $15\,\mathrm{N}$. Without scaling the force data, the method may incorrectly assign two pulling segments to different MPs, as they converge to very different force targets. We evaluated different scale factors and observed that the decomposition is the same for a broad range of values ($1/20$ to $1/80$ yield the same results). Hence, there does not seem to be a need for fine-tuning the scale parameter for different tasks. Still, we state that from the force data of a kinesthetic demonstration, it is usually only clearly distinguishable whether the teacher wanted to push or pull in a certain direction or did not apply a force at all.

One limitation of the system is that the teacher has to know at least a little bit about the assumptions we make about the demonstrations. The teacher should be aware that he or she should pause between successive movements instead of co-articulating between them. The reason is that the decomposition relies on the segmentation, which uses ZVCs. Hence, no segments will be found if the end-effector does not stop between successive movements. Second, our MPs represent point-to-point movements in task-space. There is no need to move the end-effector in completely straight lines, but the teacher should at least roughly perform point-to-point movements. While this requirement seems to be very restrictive, a broad range of tasks can be performed using our framework.

3.6 Conclusion

To conclude this chapter, we will briefly summarize our proposed approach. Then, an epilogue will discuss some open problems and cover suggestions for future work.

3.6.1 Summary of this Chapter

In this chapter, we presented an approach for learning sequential skills from unlabeled kinesthetic demonstrations of a task. In order to learn a skill, the proposed approach decomposes the task into a set of MPs. Subsequently, the system learns to sequence these MPs such that the task can be reproduced on a real robot. While the sequence learning is based on the sequence graph idea presented in the last chapter, the focus of this chapter was on finding a set of MPs which adequately represent the demonstrations.

We assume a MP has a goal in task space coordinates that should be reached if it is activated. A goal can be a desired position of a robot body, joint angle, force or a combination thereof and can be defined relative between bodies using coordinate frames. From the demonstrations, these parameters (control variables, coordinate frames, attractor goal) are found without supervision for each MP. The core of the decomposition method is the Directional Normal Distribution (DND). The probability distribution allows to simultaneously determine the most likely sequence of MPs as well as their composition, i.e., their

coordinate frames, control variables and target coordinates from the demonstrations. Determining the control variables allows to distinguish phases of force from position control, enabling a robot to explicitly apply forces only when needed. The resulting MP sequences resemble very closely the natural structure of the task.

We evaluated our approach on three different tasks: box flipping, box stacking and light bulb unscrewing. The evaluation showed that our method successfully learns to perform these tasks from very few demonstrations. Additionally, we compared our method to a baseline method, the Task-Parametrized Gaussian Mixture Model (TP-GMM, [Calinon et al., 2012]) and the Beta-Process Autoregressive Hidden Markov Model (BP-AR-HMM, [Niekum et al., 2015b]). In contrast to these approaches, our method correctly learned in which coordinate frame the robot should be controlled throughout all task phases. This property is essential as it ensures that the learned skill can be generalized to object positions and orientations which differ from the demonstrations.

3.6.2 Epilogue

Most sections of this chapter are based on a paper which is currently under review in the Robotics & Autonomous Systems Journal [Manschitz et al., 2017a,submitted]. This paper in turn is based on an earlier conference paper [Manschitz et al., 2016]. Certain aspects of the proposed approach could be improved or extended, as discussed in this section.

In order to model the segmented demonstrations, we make use of HMMs and learn the parameters of the model by using the Baum Welch algorithm. This algorithm only finds a local maximum of the log-likelihood function and therefore relies on a good initialization of the parameters. In our experiments, we ran the algorithm multiple times with different initializations and then picked the resulting model which yielded the highest likelihood. For more complex tasks, many initializations may be necessary for finding a good parameter set, resulting in a time-consuming optimization process. Spectral learning algorithms for HMMs (e.g., [Song et al., 2010]) might be a remedy for this issue, as they do not rely on heuristics for finding a good initial estimate of the parameters. Another idea is to evaluate the quality of all possible segmentations instead of just a few of them. While this idea sounds computationally expensive, Lioutikov et al. [2015, accepted] proposed an approach which reduces the algorithmic complexity and makes the problem tractable.

Pre-segmenting demonstrations by finding the ZVCs or by using other heuristics can help to reduce the complexity of the learning problem. Additionally, a pre-segmentation often leads to more meaningful segments in the context of robotics. Still, such heuristics may not be appropriate in all situations and might lead to wrong segmentations. Therefore, it is an important research problem to integrate robotic priors such as the ZVCs into the segmentation in a purely data-driven way [Jonschkowski and Brock, 2015]. Ideally, the priors should bias the system towards segmentations which are more likely in the context of robotics. If the data contradicts the prior knowledge, the segmentation should be adapted such that it still results in meaningful segments which allow for reproducing the task.

Our approach is specifically tailored to point-to-point movements. The evaluations showed that a wide range of tasks can be performed with these kind of movements. If the teacher is aware of this property and performs the demonstrations accordingly, our approach finds a meaningful set of MPs which can be utilized for reproducing the task on a real robot. Still, due to this constraint, our system can for instance not follow a desired force profile or generate movements of arbitrary shape. An important problem therefore is to extend the approach to support these kind of movements. Similar to TP-GMMs, one idea for such an extension is to mix the activations of multiple MPs instead of explicitly deciding which MP to activate for each point in time. In fact, this idea is picked up in the following chapter and will lead to a novel MP representation we call Mixture of Attractors.

4 Mixture of Attractors: A Novel Movement Primitive Representation for Learning Complex Object-Directed Movements

In the previous two chapters, we presented a framework for learning sequential skills from unlabeled kinesthetic demonstrations of a task. In order to be able to learn a skill successfully, the teacher had to consider two constraints of our approach when demonstrating a task. First, he or she had to pause between successive movements so that the algorithm could later identify them as individual movements. Demonstrating a task in this way sometimes feels unnatural. As an example, consider the task of writing the word "hello". Here, there is no need to stop between writing the individual letters. In fact, it even feels more natural to blend smoothly between writing the letters. The phenomenon of blending smoothly between successive movements will be referred to as co-articulation throughout this chapter. Second, the teacher was only allowed to perform point-to-point movements. Due to these constraints, a skill could not be learned if a task involved movements of arbitrary shape. In addition, the dynamics of the demonstrations were ignored. Hence, no skills could be learned for tasks that for instance require the robot to slow down or speed up in different task phases.

In this chapter, we introduce Mixture of Attractors, a novel movement primitive (MP) representation, which allows for learning such skills from a few demonstrations. The representation allows for learning object-directed movements of arbitrary shape and smooth blending between successive movements. Additionally, it inherently supports multiple coordinate frames, enabling the system to generalize a skill to unseen object positions and orientations. In contrast to most existing approaches, neither a heuristic nor a good initialization of parameters is needed to choose the coordinate frames. Instead, a skill is learned by solving a convex optimization problem. The approach is evaluated and compared to other MP representations on data from the Omniglot handwriting data set and on real demonstrations of a handwriting task. The evaluations show that the presented approach outperforms other state-of-the-art concepts in terms of generalization capabilities and accuracy. This chapter is mainly based on the work presented in Manschitz et al. [2017b,accepted].

4.1 Introduction

Despite impressive results in the recent years, some of the main challenges in the Learning from Demonstration domain remain unsolved. For instance, learning complex tasks usually requires more demonstrations than a user would be willing to provide. One reason for the large number of required demonstrations is that learning a skill requires finding a mapping of a potentially large input space (e.g., camera input), to a potentially large output space (e.g., desired joint positions). Depending on the task and robot, such a mapping can become highly non-linear and almost arbitrarily complex. If only a few demonstrations of a task are performed, then only a small fraction of the input space is covered with data. Learning an input to output mapping from this sparse data often results in skills that are able to perform a task if the environmental conditions are about the same as in the demonstrations. Yet, it is often not clear how the system will generalize to unseen situations.

Different approaches exist for improving the data efficiency of the algorithms and to increase the generalization capabilities of a skill. Many approaches have in common, that they either aim at reducing the dimension of the input and/or output space or change the parametrization of the input to output mapping. A third option is to limit an approach to a certain class of problems. This limitation allows for using task or problem knowledge to bias a system towards decisions which are beneficial for the desired class, but may prevent the method from being applicable in a more general context.

Our method aims at learning skills for tasks that require handling multiple objects. We assume a task is demonstrated a few times with varying object positions and orientations, as depicted in Figure 4.1.

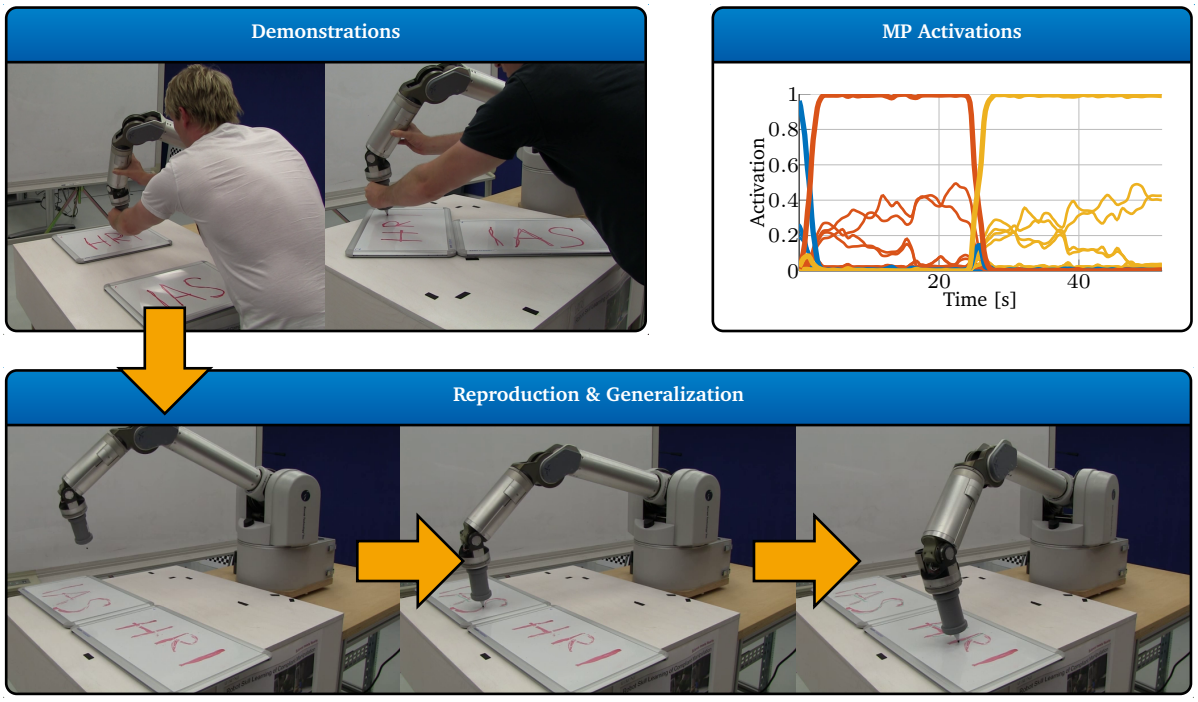


Figure 4.1.: The system learns a handwriting skill from kinesthetic demonstrations by learning a set of attractors and their continuous activations over time. The attractors can be defined in different coordinate frames, enabling the system to generalize the learned skill to unseen whiteboard positions. The plot shows the learned attractor activations for the handwriting task (thin lines). Blue lines correspond to attractors represented in the world frame, red in the IAS frame and yellow the HRI frame. The thick lines show the sum of the attractors defined in the individual coordinate frames.

From these demonstrations, the system is supposed to learn a skill which generalizes to unseen positions and orientations of the involved objects.

The main contribution of the chapter is a novel MP representation, which we call Mixture of Attractors (MoA). MoA represents movements in multiple coordinate frames. When learning a skill, a weighting of the coordinate frames is learned that explains the demonstrations well. For instance, in a task phase where the robot is supposed to manipulate an object, the weights of the coordinate frame attached to this object will be large, allowing the robot to manipulate the object at arbitrary positions. Moreover, a continuous representation of the weights is learned, allowing the robot to smoothly blend between successive movements. The proposed learning algorithm for MoA is formalized as convex optimization problem. Therefore, it does not rely on a good initialization of the parameters, nor on any heuristics on choosing the coordinate frames.

4.1.1 Related Work

MPs are a tool for increasing the data efficiency of skill learning algorithms. They are basic reusable building blocks that can be used for generating complex movements. By using MPs, the movement generation can be parametrized which leads to a complexity reduction of the learning problem. Over the last decade(s), many different MP representations have been proposed. Among the most prominent ones are Dynamic Movement Primitives (DMPs, [Ijspeert et al., 2002, 2013]), Gaussian Mixture Models (GMMs, [Calinon and Billard, 2007]), Stable Estimator of Dynamical Systems (SEDS, [Khansari-Zadeh and Billard, 2011]), Interaction Primitives [Amor et al., 2014] or Probabilistic Movement Primitives (ProMPs, [Paraschos et al., 2013]). In the following section, we discuss some of the approaches utilizing MPs for learning skills from demonstrations. With the approach presented in this chapter, we aim at con-

tributing to three important aspects in the learning from demonstration domain: learning from few demonstrations, coordinate frame selection and co-articulated movements.

Movement Primitives

Discussing the pros and cons of all or even only the most important MP representations is out of the scope of this thesis. Still, we would like to briefly review DMPs and GMMs, as they are closely related to our approach. A benchmark on some of the aforementioned MP representations is provided by Lemme [2014]. In the benchmark, the author evaluates the generalization capabilities and robustness against perturbations of DMPs, GMMs, ProMPs, Control Lyapunov Function-based Dynamic Movements (CLF-DMs, [Khansari-Zadeh and Billard, 2014]) and Neural imprinted Vector Fields (NiVFs, [Lemme et al., 2013, 2014a]).

DMPs represent a one-dimensional movement with the Dynamical System (DS)

$$\tau \ddot{x}(t) = \alpha_x(\beta_x(g - x(t)) - \dot{x}(t)) + f(z), \tag{4.1}$$

where τ is a time constant and α_x and β_x are control parameters. If the forcing term f equals zero, the DS corresponds to a stable spring-mass-damper system with an unique attractor g. The forcing term can be used for shaping the movement. The movement is driven via the canonical system

$$\tau \dot{z}(t) = -\alpha_z z(t), \tag{4.2}$$

where z is a phase variable which is set to one at the start of a movement and converges to zero for $t \to \infty$. Given K basis functions Ψ_k, the forcing term is expressed as

$$f(z) = \frac{\sum_{k=1}^{K} \Psi_k(z) w_k}{\sum_{k=1}^{K} \Psi_k(z)} x(g - x_0), \tag{4.3}$$

$$\Psi_k(z) = \exp\left(-\frac{1}{2\sigma_k^2}(z - c_k)\right), \tag{4.4}$$

where σ_k is the width of the basis function and c_k its center. If a movement is more than one-dimensional, each dimension is modeled with a DMP and the individual movements are coupled with each other by driving them with the same canonical system. The parameters $\boldsymbol{w} = [w_1, w_2, \ldots, w_K]$ can be learned from data using locally weighted regression. We would like to point out one interesting property of DMPs here. First, the movement is conditioned on the difference between the start and desired goal position of a movement. Hence, the MP representation allows for adapting a movement to varying initial and final positions. If start and end position are the same, the forcing term vanishes and the DMP degrades to a point attractor. This problem occurs for instance if a rhythmic should be performed. In that case, a different type of forcing function has to be used, as shown by Ijspeert et al. [2013]. DMPs have been used for learning skills for tasks such as table-tennis [Muelling et al., 2013], pancake flipping [Kormushev et al., 2010] or the game of ball-in-a-cup [Kober and Peters, 2009]. MoA also represents a movement with a DS and uses a formulation similar to (4.1). In contrast to DMPs, MoA can represent arbitrary movements without changing the formulation of the DS (e.g., rhythmic and non-rhythmic movements). In addition, the different dimensions of a movement are coupled directly with each other.

GMMs have also been extensively used as MP representation, for instance to learn skills for mini-golf [Kronander et al., 2011], floor sweeping [Silvério et al., 2015] or ball hitting [Calinon et al., 2010]. The underlying idea is to model the data with a joint distribution over positions $\boldsymbol{x}$ and velocities $\boldsymbol{v}$

$$\begin{aligned} \boldsymbol{z} &= \left[\boldsymbol{x}^T, \boldsymbol{v}^T\right]^T, \\ p(\boldsymbol{x}, \boldsymbol{v}) &= \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}, \boldsymbol{v} | \theta_k) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\boldsymbol{z} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right). \end{aligned} \tag{4.5}$$

Subsequently, a movement can be generated via Gaussian Mixture Regression, by conditioning the velocity on the current position

$$\hat{v} \sim p\left(v|\boldsymbol{x}\right). \tag{4.6}$$

The parameters of the GMM, the mixture prior $\boldsymbol{\pi}$ and the means and covariance matrices of the individual Gaussian distributions $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ with $k = 1, \ldots, K$, can be learned with an Expectation-Maximization (EM) algorithm which maximizes the log-likelihood of the model. GMMs have been mainly used for learning reactive time-invariant behaviors where the velocity is directly conditioned on the current position. If the state $\boldsymbol{x}$ is augmented with time information, a movement can also be modulated temporally (e.g., [Calinon and Billard, 2008, Mühlig et al., 2012]). GMMs allow for generating smooth movements and due to their probabilistic formulation an optimal number of mixtures can be estimated directly from the data via model selection methods. One drawback of GMMs is that the EM algorithm only finds a local optimum of the parameters. For complex tasks, the algorithm often has to be run multiple times with different initializations in order to find a good set of parameters. In contrast, MoA learns a skill by solving a convex optimization problem and therefore does not rely on a proper initialization of the parameters. In addition, we do not assume that the individual data points from a trajectory are independent. Instead, we integrate the temporal dependency of succeeding data points directly into the cost function.

Learning from few Demonstrations

One of the main challenges in learning skills from demonstrations is to extract as much information as possible from the demonstrations. A human teacher usually is not willing to provide tens or hundreds of demonstrations. Ideally, only a handful demonstrations should be sufficient to learn a skill. Lee et al. [2015] use Principal Component Analysis for reducing the dimensionality of the input data before segmenting the demonstrations into a set of MPs. A similar approach is proposed by Drumwright et al. [2004] who use Isomaps for reducing the dimensionality. This idea is picked up by Melchior and Simmons [2010]. The authors propose a method for embedding robot trajectories in a low-dimensional space by extending Isomap with a neighbor-finding technique which is better suited for the time-series data resulting from human demonstrations. Chebotar et al. [2014] learn an expected tactile trajectory from demonstrations and integrate it into the execution of a skill. Reducing the dimensionality of the tactile feedback allows for learning from fewer demonstrations.

Extracting a set of MPs from the demonstrations is a frequently used technique for decomposing the overall learning problem into smaller parts which might be easier to learn (e.g., [Akgun and Thomaz, 2016, Hangl et al., 2016, Huang et al., 2016, Manschitz et al., 2016]). Kappler et al. [2015] learn to activate MPs based on sensory input which may be high-dimensional. Yet, they assume the MPs are learned at a previous stage. Kim et al. [2013] learn a skill from few potentially inaccurate demonstrations. Tanwani and Calinon [2016] introduce the semi-tied GMM and use it for learning a skill from demonstrations. One main difference compared to a normal GMM is the decomposition of the covariance matrix into two terms: a common latent feature matrix shared by all mixtures and a mixture-specific diagonal matrix. As not a full covariance matrix is estimated for each mixture component, the overall dimensionality is reduced. Colomé and Torras [2014] learn skills with DMPs and introduce a coordination matrix which allows for coupling the otherwise independent dimensions of the representation. As the authors show, the coordination matrix increases the data-efficiency as fewer independent parameters have to be learned.

The movements generated by many of the aforementioned MP representations are usually modulated temporally (e.g., DMPs or ProMPs). In that case, the input dimension is inherently reduced to one, which simplifies the learning process. In this chapter, we also modulate the movements temporally. The reason why our approach is able to learn complex tasks from only a handful demonstrations is the way coordinate frames are integrated into the learning process.

When controlling a robot in task-space, the generalization capabilities of a system can be greatly improved if the MPs operate in task-spaces which are defined relative to objects. If each object is associated with a coordinate frame and a task-space which controls the robot relative to this coordinate frame, the system is inherently able to generalize the movements to setups which have not been seen in the demonstrations. As the teacher usually does not want to specify which MP performs a movement relative to which object, this parameter also has to be learned from the demonstrations. As we also dealt with coordinate frame selection methods in the previous chapter, the following paragraphs will be similar to those of Section 3.1.1.

In the approach presented by Cederborg et al. [2010], a movement can be performed relative to an object, relative to a starting position or in a robot frame. A movement is modeled with a GMM which is trained incrementally. One GMM is trained in each of these coordinate frames and the robot is controlled in the frame which yields the lowest training error. The coordinate frame is fixed throughout task execution and therefore it is not possible to learn skills which require switching between the frames such as picking up an object and moving it to another location. A more general approach is proposed by Dong and Williams [2011, 2012]. Here, demonstrations are first segmented into smaller parts based on contact information. Next, the segments are clustered based on the similarity of relevant movement variables at the beginning and end of the segments (e.g., distance between end-effector and an object). The segments corresponding to a single cluster are aligned in time using Dynamic Time Warping and the corresponding movement is then represented using a probabilistic flow field. When executing a task, a movement is selected based on the similarity between the current and expected state of the movement variables at the beginning of the segments assigned to a cluster. Akin to this approach, Niekum et al. [2015b] also extract a set of MPs from the demonstrations and select an appropriate coordinate frame for each MP. The end-points of the segments assigned to an individual MP are clustered in each frame and the coordinate frame leading to the lowest inter-cluster distance is chosen as winner.

One common idea shared by the aforementioned approaches is that the importance of a variable of interest correlates with its precision (inverse variance) across different demonstrations. For instance, if a teacher grasps an object, the distance between his or her hand and the object will be consistently approximately zero in all demonstrations. Kober et al. [2015] picked up this idea and assign segments to MPs based on a score function which rewards convergence (similar end-points but different starting points) and penalizes divergence of the individual segments. The authors do not only pick proper coordinate frames for all MPs, but also choose the control variables for them. For instance, a MP may control the velocity or position of an end-effector or control the applied force. Ureche et al. [2015] propose a similar score function which compares the variance of a variable over a time-window with the variance over all trials. They postulate a variable is significant for a task if it changes its value significantly within a single demonstration and systematically across demonstrations. Mühlig et al. [2009] also take additional criteria other than the variance into account. For instance, they consider a kinetic criterion which incorporates effort and discomfort into the selection of a task-space. They show the advantage of this criterion in a bi-manual manipulation task where the teacher demonstrates one phase of the task only with one arm. In this phase, the second arm is not moving, yielding low variance. The authors state that the movement of the arm is irrelevant for this task phase and their criterion ensures that it is not considered as important. Such task-space selection mechanisms are out of the scope of this chapter. Instead, we concentrate on selecting the proper coordinate frames throughout task execution.

The aforementioned approaches explicitly decide in which coordinate frame the robot should be controlled for each segment or point in time. An exception is the approach proposed by Mühlig et al. [2009]. Here, a task blending matrix continuously weights the frames over time. The advantage of such a representation is that a smooth transition can be ensured when switching between different coordinate frames. A similar behavior is achieved by the Task-Parametrized Gaussian Mixture Model TP-GMM, which was introduced by Calinon et al. [2012]. It generalizes the GMM to support multiple coordinate

frames and is for instance used by Rozo et al. [2015]. Here, a probability distribution over the coordinate frame weights is learned which varies over time. Our approach also does not select the coordinate frames explicitly. Instead, it learns to activate a set of attractors represented in different coordinate frames over time in a way which is most consistent with the demonstrations. Estimating the activations is formulated as convex optimization problem which does not rely on a good initialization of the parameters, which for instance is the case for the TP-GMM.

Co-Articulated Movements

When performing a sequence of movements, humans tend to co-articulate between successive movements, in particular if a task has to be performed at high speed or with low accuracy [Sosnik et al., 2004]. In the context of skill learning, co-articulation renders the problem of movement identification more difficult, as the start and end of movements becomes less obvious. Approaches which rely on heuristics such as zero velocity crossings for detecting transitions between MPs may suffer from such co-articulated movements. If no transition is detected between two successive movements, such approaches would interpret them as a single movement. The resulting MP may be unusable in different situations, especially if the two movements were supposed to be performed relative to different objects.

In the latter case, also the movement generation becomes more difficult. If two MPs are defined in different coordinate frames and are supposed to be executed in a sequence, it is often not clear how to transition between them. The same is true if the MPs come from a library and for instance their initial conditions (e.g., acceleration) are different. Nemec and Ude [2012] extend the DMP formulation such that two consecutive MPs can be joined together in a continuous way. Muelling et al. [2010, 2013] use a gating network which allows for mixing a set of MPs. They do not use their approach for sequencing MPs, but for generalizing table-tennis strikes to unknown situations. Ewerton et al. [2015] present the Mixture of Interaction Primitives framework, which can also be utilized for generalizing from a set of MPs.

Two MP representations which explicitly support co-articulated movements are GMMs and SEDS. Calinon and Billard [2007] encode a movement as joint probability distribution over the positions and velocities by using GMMs. A movement is generated by conditioning the velocity on the current position. SEDS also makes use of GMMs, but additionally guarantees convergence to a desired target at the cost of a larger computational effort. Both representations support co-articulated movements as the movements generated by them are generally smooth. ProMPs can also blend between two successive MPs. Yet, it is not clear how to learn a blending factor from demonstrations. Our approach learns to activate a set of attractors so that the generated movement follows the demonstrated behavior. If the human teacher transitioned smoothly between two successive movements, the resulting MP activations will change slowly, leading to the same behavior.

4.1.2 Properties of the Mixture of Attractors Representation

MoA combines many of the advantages of other existing representations. The system learns to continuously activate a set of attractors over time by solving a convex optimization problem. The attractors can be represented in different coordinate frames and therefore allow for generalizing a movement to arbitrary object locations. The learned attractor activations closely resemble the demonstrations. If a human teacher co-articulates between successive movements, the activations will reflect this behavior and will lead to a smooth transition when executing the skill on a real robot. Altogether, our system is able to learn complex skills from only a few demonstrations and is able to generalize a skill to novel setups. To the best of our knowledge, MoA is the first MP representation which can represent complex object-directed movements and does not rely on a good initial parameter estimation.

The remainder of the chapter is organized as follows. In Section 4.2, the MoA MP representation is introduced formally. Next, Section 4.3 shows how the representation can be used for robot control. The

approach is then evaluated in Section 4.4 before concluding and giving a short outlook on future work in Section 4.5.

4.2 Mixture of Attractors

The basic idea behind MoA is to represent a movement as composition of very simple Dynamical Systems (DS). We refer to an attractor as a spring-mass-damper system of the form

$$\ddot{\boldsymbol{x}}(t) = \alpha(\beta(\boldsymbol{g} - \boldsymbol{x}(t)) - \dot{\boldsymbol{x}}(t)), \tag{4.7}$$

where α and β are positive controller parameters. The parameters can be set to guarantee the stability of the DS ($\beta = \alpha/4$, see Ijspeert et al. [2013]). In that case, the DS converges to its attractor goal $\boldsymbol{g}$ for $t \to \infty$. In this chapter, $\boldsymbol{x}$ corresponds to the Cartesian position of the robot's end-effector. Instead of having a single attractor, we assume that a movement is generated by a linear combination of K attractors

$$\ddot{\boldsymbol{x}}(t) = \sum_{k=1}^{K} a_k(t)\alpha(\beta(\boldsymbol{g}_k - \boldsymbol{x}(t)) - \dot{\boldsymbol{x}}(t)). \tag{4.8}$$

Hence, MoA defines a movement by a set of K attractor goals $\boldsymbol{g}_k$ and their activations $a_k(t)$. Note that the activations explicitly depend on the time which allows for shaping the trajectory and generate complex movements. We assume the activations of each time-step sum up to one and therefore, the equation can also be written as

$$\ddot{\boldsymbol{x}}(t) = \alpha(\beta(\boldsymbol{G}\boldsymbol{a}(t) - \boldsymbol{x}(t)) - \dot{\boldsymbol{x}}(t)), \tag{4.9}$$

where the attractor goals are summarized in the matrix $\boldsymbol{G} = [\boldsymbol{g}_1, \ldots, \boldsymbol{g}_K]$ and the attractor activations of one time step are summarized in the vector $\boldsymbol{a}(t)$. In the following sections, we show how the attractor goals and activations can be learned from demonstrations and discuss some of the properties of the MP representation.

4.2.1 Trajectory Tracking

We introduce the learning method by assuming we want to follow a desired trajectory. Later, we focus on learning skills which go beyond pure replaying of a demonstration and extend the approach to support multiple coordinate frames. The first step is to time-discretize (4.9) using

$$\ddot{\boldsymbol{x}}_i \approx \frac{\boldsymbol{x}_i - 2\boldsymbol{x}_{i-1} + \boldsymbol{x}_{i-2}}{h^2} \quad \text{and} \quad \dot{\boldsymbol{x}}_i \approx \frac{\boldsymbol{x}_i - \boldsymbol{x}_{i-1}}{h}, \tag{4.10}$$

where h is the step-size and $i \in \mathbb{N}$ is the time step. Inserting (4.10) in (4.9) leads to

$$\begin{aligned} \boldsymbol{x}_i &= \underbrace{\frac{2 + \alpha h}{1 + \alpha h + \alpha\beta h^2}}_{c_1} \boldsymbol{x}_{i-1} \underbrace{- \frac{1}{1 + \alpha h + \alpha\beta h^2}}_{c_2} \boldsymbol{x}_{i-2} \underbrace{+ \frac{\alpha\beta h^2}{1 + \alpha h + \alpha\beta h^2}}_{c_3} \boldsymbol{G}\boldsymbol{a}_i \\ &= c_1\boldsymbol{x}_{i-1} + c_2\boldsymbol{x}_{i-2} + c_3\boldsymbol{G}\boldsymbol{a}_i, \end{aligned} \tag{4.11}$$

where c_1, c_2 and c_3 are constants to keep the equation compact. We want to learn the parameters $\boldsymbol{G}$ and $\boldsymbol{a}_i$, so that we track a demonstrated trajectory $\tau = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_N\}$ as closely as possible. Here, N is the length of the demonstration. In this section, we assume to know the goals $\boldsymbol{G}$ and want to estimate

the movement primitive activations $\boldsymbol{a}_i$. In order to track the trajectory, we minimize the mean squared error (MSE) over time between the demonstrated trajectory and the trajectory generated by MoA

$$J = \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{y}_i)^T (\boldsymbol{x}_i - \boldsymbol{y}_i). \tag{4.12}$$

The aim of this section is to minimize the cost function with respect to the activations $\boldsymbol{a}_i$. Note that our system is fully determined by the attractor goals, their activations over time and the start points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Therefore, as a first step $\boldsymbol{x}_i$ is expressed in terms of these variables

$$\boldsymbol{x}_i = \lambda_t \boldsymbol{x}_2 + \mu_t \boldsymbol{x}_1 + \sum_{j=1}^{N} \gamma_{i,j} \boldsymbol{G} \boldsymbol{a}_j. \tag{4.13}$$

The scalar constants λ_i, μ_i and $\gamma_{i,j}$ can be specified recursively

$$\begin{aligned} \lambda_i &= c_1 \lambda_{i-1} + c_2 \lambda_{i-2}, & \lambda_1 &= 0, \quad \lambda_2 = 1, \\ \mu_i &= c_1 \mu_{i-1} + c_2 \mu_{i-2}, & \mu_1 &= 1, \quad \mu_2 = 0, \\ \gamma_{i,j} &= c_1 \gamma_{i-1,j} + c_2 \gamma_{i-2,j} + c_3 \delta_{i,j}, & \gamma_{1,j} &= 0, \quad \gamma_{2,j} = 0, \end{aligned}$$

where δ is the Kronecker delta with $\delta_{i,j} = 1$ for $i = j$ and $\delta_{i,j} = 0$ otherwise. Given a trajectory τ, we can compute the constants, set $\boldsymbol{x}_1 = \boldsymbol{y}_1$, $\boldsymbol{x}_2 = \boldsymbol{y}_2$ substitute $\hat{\boldsymbol{y}}_i = \boldsymbol{y}_i - \lambda_i \boldsymbol{x}_2 - \mu_i \boldsymbol{x}_1$ and plug this term into the cost function

$$J = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} \gamma_{i,j} \boldsymbol{G} \boldsymbol{a}_j - \hat{\boldsymbol{y}}_i \right)^T \left(\sum_{j=1}^{N} \gamma_{i,j} \boldsymbol{G} \boldsymbol{a}_j - \hat{\boldsymbol{y}}_i \right). \tag{4.14}$$

If we concatenate all movement primitive activations in a single vector $\boldsymbol{a} = \left[\boldsymbol{a}_1^T, \ldots, \boldsymbol{a}_N^T\right]^T$ and concatenate $\hat{\boldsymbol{G}}_i = [\gamma_{i,1}\boldsymbol{G}, \ldots, \gamma_{i,N}\boldsymbol{G}]$, then (4.14) can be rearranged to

$$\begin{aligned} J &= \sum_{i=1}^{N} \left(\hat{\boldsymbol{G}}_i \boldsymbol{a} - \hat{\boldsymbol{y}}_i \right)^T \left(\hat{\boldsymbol{G}}_i \boldsymbol{a} - \hat{\boldsymbol{y}}_i \right) \\ &= \boldsymbol{a}^T \underbrace{\left(\sum_{i=1}^{N} \hat{\boldsymbol{G}}_i^T \hat{\boldsymbol{G}}_i \right)}_{0.5\boldsymbol{H}} \boldsymbol{a} - 2 \underbrace{\left(\sum_{i=1}^{N} \hat{\boldsymbol{y}}_i^T \hat{\boldsymbol{G}}_i \right)}_{-\boldsymbol{f}^T} \boldsymbol{a} + \underbrace{\sum_{i=1}^{N} \hat{\boldsymbol{y}}_i^T \hat{\boldsymbol{y}}_i}_{const} \\ &= \frac{1}{2} \boldsymbol{a}^T \boldsymbol{H} \boldsymbol{a} + \boldsymbol{f}^T \boldsymbol{a} + const, \end{aligned} \tag{4.15}$$

which can be minimized via Quadratic Programming (QP). For the optimization, constraints have to be added to ensure the individual activations are in the range $[0, 1]$ and the activations for each time step i sum up to one. Therefore, the overall minimization problem is

$$\min_{\boldsymbol{a}} \frac{1}{2} \boldsymbol{a}^T \boldsymbol{H} \boldsymbol{a} + \boldsymbol{f}^T \boldsymbol{a}, \text{ such that } \begin{cases} \|\boldsymbol{a}_i\|_1 = 1, \\ \boldsymbol{0} \leq \boldsymbol{a} \leq \boldsymbol{1}. \end{cases} \tag{4.16}$$

The optimization problem (4.16) can now be solved using an out of the box standard QP solver. Please note that the matrix $\boldsymbol{H}$ is a sum of the form $\sum \hat{\boldsymbol{G}}_i^T \hat{\boldsymbol{G}}_i$. Therefore, the matrix is positive semi-definite

and a standard QP solver is guaranteed to find a global minimum. In order to proof this, we show that $\boldsymbol{a}^T \boldsymbol{H} \boldsymbol{a} \geq 0$ is true for all $\boldsymbol{a}$

$$\boldsymbol{a}^T \left(\sum_{i=1}^{N} \hat{\boldsymbol{G}}_i^T \hat{\boldsymbol{G}}_i \right) \boldsymbol{a} = \sum_{i=1}^{N} \boldsymbol{a}^T \hat{\boldsymbol{G}}_i^T \hat{\boldsymbol{G}}_i \boldsymbol{a} = \sum_{i=1}^{N} \left(\hat{\boldsymbol{G}}_i \boldsymbol{a} \right)^T \hat{\boldsymbol{G}}_i \boldsymbol{a} \tag{4.17}$$

$$= \sum_{i=1}^{N} \boldsymbol{z}_i^T \boldsymbol{z}_i = \sum_{i=1}^{N} \sum_{d=1}^{NK} z_{i,d}^2 \geq 0, \tag{4.18}$$

where we used $\boldsymbol{z}_i = \hat{\boldsymbol{G}}_i \boldsymbol{a}$ and NK is the length of vector $\boldsymbol{a}$.

4.2.2 Parametrizing the Activations

So far, the optimizer could freely choose the activations for each point in time. This freedom may lead to jumps in the activations, potentially resulting in jerky movements. In order to generate more natural, smooth movements, the activations can be parametrized. In that case, the optimizer is only allowed to change the activations at fixed points in time (e.g., every 50ms). We call these points support points. In between support points, the activations are interpolated. If we summarize all activations of the support points in a matrix $\boldsymbol{S} \in \mathbb{R}^{K \times N_S}$, where N_S is the number of support points and K the number of attractors, the activations at time-step i are interpolated according to

$$\boldsymbol{a}_i = \boldsymbol{S} \boldsymbol{w}_i. \tag{4.19}$$

Here, $\boldsymbol{w}_i$ is a weight vector for time-step i. It determines how the activations will be interpolated. In order to generate smooth movements, we use normalized Radial Basis Functions (RBFs)

$$w_{i,s} = \frac{e^{-\gamma^2 (t_i - t_s)^2}}{\sum_{j=1}^{N_S} e^{-\gamma^2 (t_i - t_j)^2}}, \tag{4.20}$$

where $w_{i,s}$ is the sth value of vector $\boldsymbol{w}_i$, t_i is the time at time step i and t_s is the (temporal) center of the RBF. The bandwidth γ of the RBF is a hyperparameter that determines how smoothly the weights change over time.

Our aim is now to reformulate the cost function (4.15), so that we can find the support point activations by again solving a QP. In order to do so, we first rewrite the matrix of the support point activations $\boldsymbol{S}$ as a vector $\boldsymbol{s}$ by concatenating the rows of the matrix. Reformulating (4.19) in terms of $\boldsymbol{s}$ results in

$$\boldsymbol{a}_i = \begin{pmatrix} \boldsymbol{w}_i^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\ \boldsymbol{0}^T & \ddots & \boldsymbol{0}^T \\ \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{w}_i^T \end{pmatrix} \boldsymbol{s} = \boldsymbol{W}_i \boldsymbol{s}. \tag{4.21}$$

Next, we plug (4.21) into (4.14)

$$\hat{J} = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} \gamma_{i,j} \boldsymbol{G} \boldsymbol{W}_j \boldsymbol{s} - \hat{\boldsymbol{y}}_i \right)^T \left(\sum_{j=1}^{N} \gamma_{i,j} \boldsymbol{G} \boldsymbol{W}_j \boldsymbol{s} - \hat{\boldsymbol{y}}_i \right) \tag{4.22}$$

$$= \sum_{i=1}^{N} \left(\boldsymbol{\Lambda}_i \boldsymbol{s} - \hat{\boldsymbol{y}}_i \right)^T \left(\boldsymbol{\Lambda}_i \boldsymbol{s} - \hat{\boldsymbol{y}}_i \right), \tag{4.23}$$

$$\boldsymbol{\Lambda}_i = \sum_{j=1}^{N} \gamma_{i,j} \boldsymbol{G} \boldsymbol{W}_j, \tag{4.24}$$

and rewrite the equation to emphasize the similarity to (4.15)

$$\begin{aligned}\hat{J} &= \boldsymbol{s}^T \underbrace{\left(\sum_{i=1}^{N} \boldsymbol{\Lambda}_i^T \boldsymbol{\Lambda}_i\right)}_{0.5\hat{\boldsymbol{H}}} \boldsymbol{s} - \underbrace{2\left(\sum_{i=1}^{N} \hat{\boldsymbol{y}}_i^T \boldsymbol{\Lambda}_i\right)}_{-\hat{\boldsymbol{f}}^T} \boldsymbol{s} + \underbrace{\sum_{i=1}^{N} \hat{\boldsymbol{y}}_i^T \hat{\boldsymbol{y}}_i}_{const}, \\ &= \frac{1}{2}\boldsymbol{s}^T \hat{\boldsymbol{H}} \boldsymbol{s} + \hat{\boldsymbol{f}}^T \boldsymbol{s} + const. \end{aligned} \tag{4.25}$$

The modified cost function can now be minimized via Quadratic Programming

$$\min_{\boldsymbol{s}} \frac{1}{2}\boldsymbol{s}^T \hat{\boldsymbol{H}} \boldsymbol{s} + \hat{\boldsymbol{f}}^T \boldsymbol{s}, \text{ such that } \begin{cases} \|\boldsymbol{s}_j\|_1 = 1 \quad \forall \quad j = \{1, \ldots, N_s\}, \\ \mathbf{0} \leq \boldsymbol{s} \leq \mathbf{1}. \end{cases} \tag{4.26}$$

where $\boldsymbol{s}_j$ are the activations of the jth support point (jth column of $\boldsymbol{S}$). Note that these activations are spread over the activation vector $\boldsymbol{s}$ as we constructed it by concatenating the matrix $\boldsymbol{S}$ row-wise. The size of the matrix $\hat{\boldsymbol{H}}$ is $KN_S \times KN_S$, while the matrix $\boldsymbol{H}$ from the original formulation in (4.16) has a size of $KN \times KN$. As $N_S < N$, parametrizing the activations does not only lead to smoother movements, but also reduces the computational costs and memory requirements of the optimization. As the form of $\hat{\boldsymbol{H}} = \sum_{i=1}^{N} \hat{\boldsymbol{y}}_i^T \boldsymbol{\Lambda}_i^T \boldsymbol{\Lambda}_i \hat{\boldsymbol{y}}_i$ is equivalent to that of $\boldsymbol{H}$, proof (4.18) still holds and the QP is convex.

4.2.3 Support for Multiple Coordinate Frames

In the previous sections, the goal matrix $\boldsymbol{G}$ was constant over time. In order to support multiple coordinate frames, we can relax this assumption. An attractor can be defined in a coordinate frame which is not the world frame. In that case, its attractor goal can be transformed into the world frame for each time-step. Hence, the goal matrix at time-step i becomes $\boldsymbol{G}_i$, where the columns of the matrix correspond to the attractor goals transformed into the global world frame. The only change in the formulation of the QP has to be made for (4.24), where $\boldsymbol{G}$ is replaced with the time-indexed matrix $\boldsymbol{G}_j$

$$\boldsymbol{\Lambda}_i = \sum_{j=1}^{N} \gamma_{i,j} \boldsymbol{G}_j \boldsymbol{W}_j. \tag{4.27}$$

Changing the fixed goal matrix $\boldsymbol{G}$ to a matrix which varies over time does not change the fact that the quadratic matrix $\hat{\boldsymbol{H}}$ is positive semi-definite. The reason is that it is still constructed by the term

$$\hat{\boldsymbol{H}} = \sum_{i=1}^{N} \boldsymbol{\Lambda}_i^T \boldsymbol{\Lambda}_i. \tag{4.28}$$

Therefore, the problem can still be solved in a globally optimal manner. Support for multiple coordinate frames can also be added to the non-parametrized formulation of MoA. Here, only the time-indexed goal matrices have to be inserted into $\hat{\boldsymbol{G}}_i = [\gamma_{i,1}\boldsymbol{G}_1, \ldots, \gamma_{i,N}\boldsymbol{G}_N]$. We would like to point out that the QP has to be solved only once. In a real-world task, if a coordinate frame is associated with an object, the attractor goals defined in this frame automatically move together with this object. Therefore, a movement which is defined relative to these attractors is automatically adapted to changing object positions and orientations without solving the QP again.

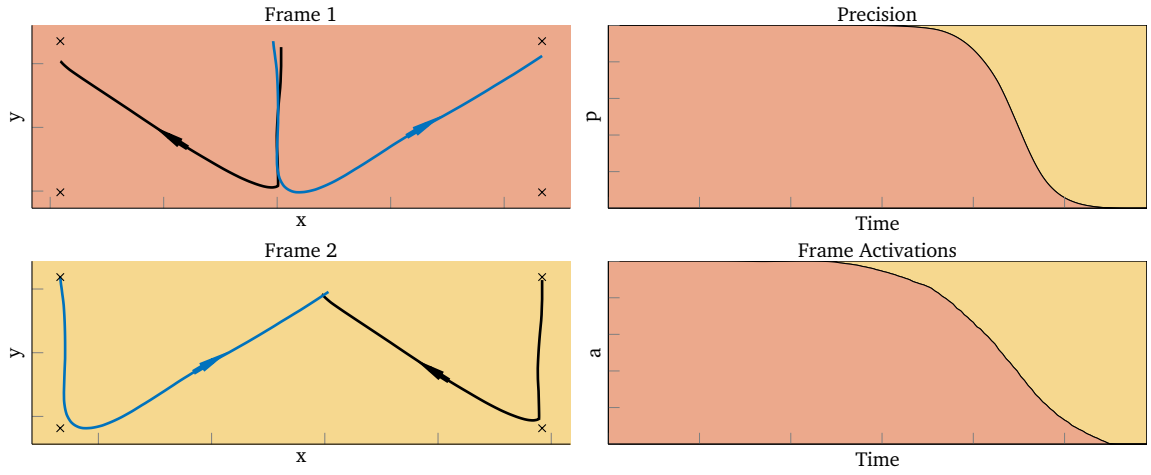


Figure 4.2.: Toy example to illustrate the MoA optimization. On the left, two demonstrations (black and blue) are shown in two different coordinate frames. The demonstrations start at the same position in the first frame, approach a target in this frame and then approach a target in the second frame. The attractor goals are marked with an x. On the top right, the ratio of the precision (reciprocal of the variance) in the two coordinate frames is shown over time. The frame activations resulting from the MoA optimization (bottom right) are akin to the precision.

4.3 Using Mixture of Attractors for Robot Control

So far, we introduced the MoA framework and showed how the activations can be learned to track a demonstrated trajectory. In this section, we show how MoA can be used for learning a skill from demonstrations of a task. To do so, we discuss two aspects that were not covered in this chapter so far. First, we show how the number of attractors and their goals can be learned from demonstrations of the task. Second, we discuss how many attractors should be defined in which coordinate frame. At the end of the section, we present the final skill learning algorithm.

First of all, we would like to point out that a skill is learned using multiple demonstrations of a task. So far, the attractor activations were learned by minimizing the MSE between a trajectory generated by MoA and a single demonstration. For learning a skill, the MSE of M demonstrations add up to

$$J = \sum_{m=1}^{M} \sum_{i=1}^{N} \left(\boldsymbol{x}_i^{(m)} - \boldsymbol{y}_i^{(m)} \right)^T \left(\boldsymbol{x}_i^{(m)} - \boldsymbol{y}_i^{(m)} \right), \tag{4.29}$$

where we assumed that all demonstrations have the same length N. As the MSE of the different demonstrations simply add up, the form of the QP does not change when minimizing J. Therefore, for M demonstrations, we can compute M independent QPs and then compute the sum of all $\hat{\boldsymbol{H}}$'s and $\hat{\boldsymbol{f}}$'s to form a single QP which has the same form as (4.26).

4.3.1 Choosing the Number of Attractors and their Goals

No matter how many attractors are chosen, the points $\boldsymbol{Ga}$ generated by MoA will always lie within the convex hull of their attractor goals, as the activations sum up to one. For two linearly independent attractors, the system would generate points on a line. For three linearly independent attractors points within a triangle. A generic solution to estimate the number of attractors in a D dimensional space is to choose it so that any trajectory within this space can be generated by the system. Therefore, we

propose to use 2^D attractors, where D is the dimension of the space the MPs operate on[1]. We propose to choose the attractor goals by computing the minimum and maximum values for each dimension of the demonstrations. The attractor goals then build the corner points of the bounding box of the demonstrations. If multiple coordinate frames are used, 2^D attractors are chosen for each coordinate frame independently. The attractor goals are illustrated in Figure 4.2 for a simple $2D$ toy example.

4.3.2 Learning the Importance of the Coordinate Frames

When learning a skill, we associate the world and each object in the scene with one coordinate frame. The attractor activations should be learned in a way that yields the best generalization performance. For instance, if a task requires approaching an object, the activations of the attractors which control the robot in the coordinate frame of this object should be large during this phase of the task. The approach presented in this section inherently ensures this property. A common approach for choosing coordinate frames is to compare the variance of the data over multiple demonstrations in the individual coordinate frames (e.g., Kober et al. [2015], Ureche et al. [2015]). For a certain task phase, the movement will be represented in the frame that has the lowest variance. Our approach leads to similar results without relying on a heuristic to decide which coordinate frame to choose for which phase of a task. Instead, the optimization converges to a solution where the activations of the attractors represented in a coordinate frame will be large if the variance is low.

Our main assumption is that the demonstrations are aligned in time and the movement will be modulated temporally. For aligning the demonstrations, we use Dynamic Time Warping (DTW). As the demonstrations are represented in different coordinate frames, it is not straightforward to align them temporally. As cost function to measure the distance between two points $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$ from different demonstrations we measure the Euclidean distance in each frame k and use the overall minimum as cost term

$$\mathrm{DTW}(i,j) = \min_k \left\| \boldsymbol{x}_i^{(k)} - \hat{\boldsymbol{x}}_j^{(k)} \right\|_2, \tag{4.30}$$

where $\boldsymbol{x}_i^{(k)}$ is the ith data point of a demonstration represented in the kth coordinate frame. After aligning the demonstrations in time, the attractor goals are computed for each frame separately according to the previous section and subsequently transformed into the world frame for all non-world frames. The 2^D attractor goals of frame k at time step i will be noted as $\boldsymbol{G}_i^{(k)}$. The attractor goals of the frames are stacked together, resulting in a single goal matrix

$$\boldsymbol{G}_i = \left[\boldsymbol{G}_i^{(1)}, \ldots, \boldsymbol{G}_i^{(K)} \right] \tag{4.31}$$

for each time step. Now, the attractor activations can be computed according to (4.26). The process of estimating the activations is illustrated in Figure 4.2 by using a simple toy example. The optimizer can freely choose the attractor activations over time, but has to explain all demonstrations with the same sequence of activations. As it is not possible to do so in a single coordinate frame, the optimizer converges to a solution where the frame activations (sum of attractor activations represented in the same frame) vary over time. Therefore, when learning a skill with MoA, our system never explicitly selects coordinate frames for different phases of a task. Instead, by defining the attractors in different coordinate frames, the importance of the individual coordinate frames emerges automatically as a result of the optimization process.

[1] In general, a set $D+1$ attractors can be found which covers the demonstrations. For instance, for a $2D$ space, a triangle can be found which cover the demonstrations. We use a 2^D bounding box as it is more intuitive and we represent movements only in $2D$ and $3D$ space in this chapter.

Algorithm 4 MoA Learning Algorithm

Require: M Trajectories $\tau^{(m)} = \left\{ \boldsymbol{x}_1^{(m)}, \ldots, \boldsymbol{x}_N^{(m)} \right\}$, K Coordinate Frames
Hyperparameters: Number of support points N_S, bandwidth γ, DS parameters α and β
Align trajectories using DTW (4.30)
Compute support point weights $\boldsymbol{w}_{1:N}$ (4.20)
for each Frame k **do**
    Compute attractor goals $\boldsymbol{G}^{(k)}$ (Section 4.3.1)
$\hat{\boldsymbol{H}} = \boldsymbol{0}, \hat{\boldsymbol{f}} = \boldsymbol{0}$
for each Trajectory m **do**
    for each Frame k **do**
        Convert goals to world frame $\boldsymbol{G}_{1:N}^{(k)}$
    Concatenate goal matrices to single matrix for each time-step $\boldsymbol{G}_{1:N}$ (4.31)
    $[\hat{\boldsymbol{H}}, \hat{\boldsymbol{f}}]$ += generateQP$\left(\boldsymbol{x}_{1:N}^{(m)}, \boldsymbol{w}_{1:N}, \boldsymbol{G}_{1:N}\right)$ (4.26)
Solve QP($\hat{\boldsymbol{H}}, \hat{\boldsymbol{f}}$) to find support point activations $\boldsymbol{S}$
return Attractor goals $\boldsymbol{G}$, Support point activations $\boldsymbol{S}$

4.3.3 Choosing the Hyperparameters

When using the suggested bounding box method for choosing the positions and numbers of the attractors goals, the only hyperparameters a user has to choose are the number of support points and the bandwidth γ of the corresponding RBF. In all experiments presented in this chapter, the bandwidth of each RBF was set so that the function evaluates to a value $\alpha = 0.1$ at the center's of the neighboring RBFs. The numbers of support points were chosen by increasing the number until we were satisfied with the results. In future work, we plan to develop a method for optimizing the hyperparameters in a principled manner.

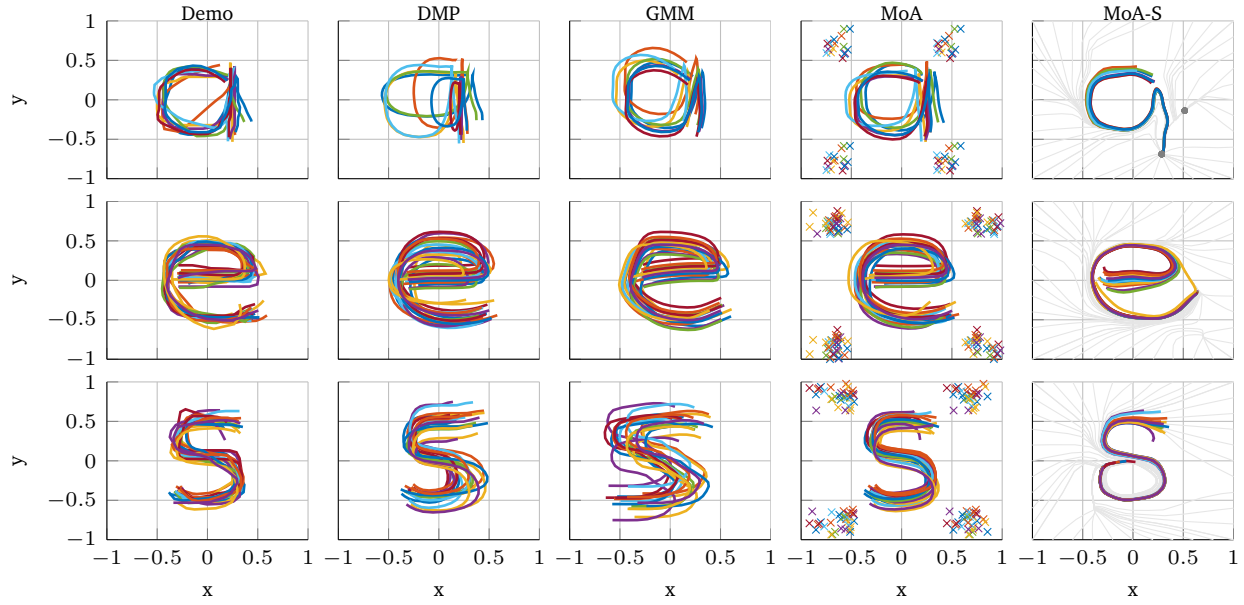
4.3.4 Final Algorithm

The MoA learning algorithm is summarized in Algorithm 4. Input to the algorithm are M demonstrations and the coordinate frames. The only hyperparameters that have to be set are the parameters of the Dynamical System α, β, the number of support points N_S and the bandwidth of the RBFs. Usually, α is set to achieve a desired stiffness of the system and β is set to $\alpha/4$ so that the DS is stable. First, the demonstrations are aligned in time using DTW. Next, the weights of the support points are computed for each time-step. The attractor goals are computed for each frame separately. Then, the goals of the frames are transformed into the world frame for each time-step and demonstration. Finally, the QP is generated and solved. The learned skill is composed of the resulting support point activation matrix $\boldsymbol{S}$, and the attractor goals $\boldsymbol{G}$.

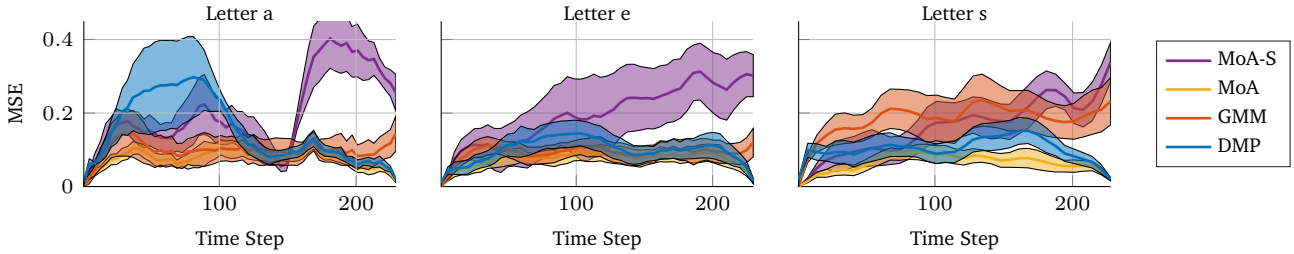
4.4 Evaluation of the Approach

For evaluating MoA, we performed two experiments.

The aim of the first experiment was to compare some of its properties to Dynamic Movement Primitives (DMPs) and Gaussian Mixture Models (GMMs). These MP representations have similar properties to MoA, which allows for a fair comparison. The comparison is carried out on letters from the Omniglot handwriting data set. In a second experiment, we evaluated the generalization capabilities of the system on a real robot handwriting task. The task is demonstrated kinesthetically and later reproduced on a real seven degrees of freedom (DoF) Barrett WAM robot. Additionally, we compare the generalization capabilities with two state-of-the-art approaches.



(a) Written letters.



(b) Mean squared error over time.

Figure 4.3.: Results on the Omniglot handwriting data set. The upper plots show the demonstrations and reproduced trajectories in $2D$ space for two letters of the data set. For MoA, additionally the attractor goals are shown. All methods except for MoA-S are time-dependent. For the time-independent MoA-S, the gray lines show the attractor landscape. Here, the movements converge to the gray dots. The lower plots show the corresponding mean squared error over time (average and $\pm$ one standard deviation).

4.4.1 Handwriting Evaluation

The Omniglot data set was introduced by Lake et al. [2015] as a $2D$ data set for one-shot learning. It contains over 1500 different handwritten characters from 50 different alphabets. Each character was drawn online using Amazon's Mechanical Turk by 20 different people. The images come with stroke data as sequences of $2D$ coordinates associated with time information t. We use the data set to compare MoA with DMPs. Both MP representations are trained on characters from the Latin alphabet. Before training, we removed all characters that were not drawn continuously (e.g., more than one stroke was used as the participant lifted the pen). The remaining characters were preprocessed as follows. First, we shifted each character so that the mean of the image is at $[0, 0]$. For each character, we computed the average standard deviation from the mean and subsequently scaled each image to have the same standard deviation. As a last preprocessing step, we aligned the trajectories in time using DTW. The preprocessing of the data was the same for all methods.

For a fair comparison, the movements generated by all methods were modulated temporally. For the GMM approach, a joint probability $p(t, \nu)$ was learned. The movements were then generated by conditioning the current velocity on the time $p(\nu|t)$. For MoA and DMPs, the basis functions were activated temporally. We did our best to tune the hyperparameters of all methods. For the GMM approach, we in-

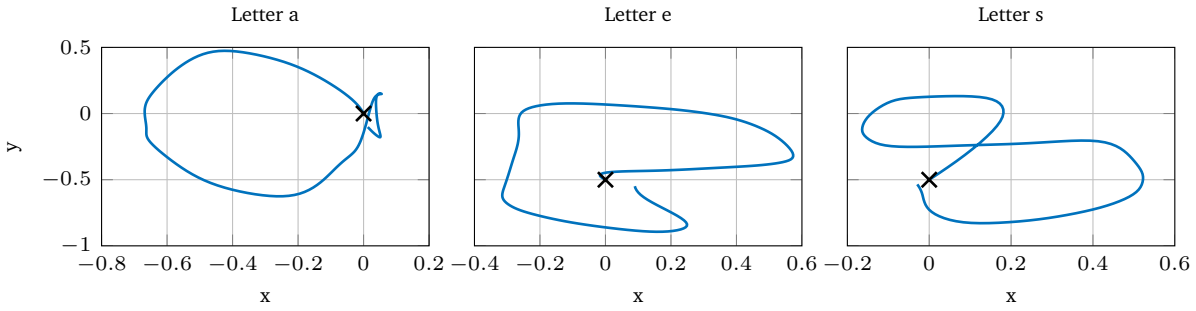
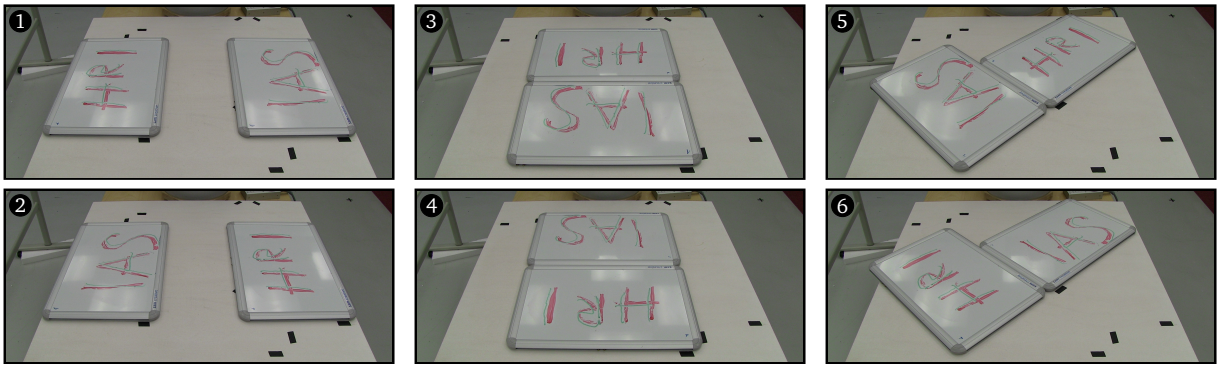


Figure 4.4.: Letters drawn by MoA if start and desired end point are set to the same value. In contrast to DMPs, MoA allows for generating such movements without any special treatment.

creased the number of Gaussians to 15 until the Mean Squared Error (MSE) between the generated and demonstrated movements did not increase anymore. For MoA and DMPs we used the method suggested in Section 4.3.3, which resulted in 20 support points. In contrast to MoA, the movements generated by DMPs are conditioned on a desired start and end point. To add this property to our MP representation, we associated a coordinate frame with the start and end points of each demonstration, respectively. As these points vary over the demonstrations, the attractor goals (transformed into the world frame) are automatically shifted together with the origins of the coordinate frames. As the MPs represent movements in $2D$ space, four attractors where used for each coordinate frame. The locations of the attractor goals were set according to the bounding box method suggested in Section 4.3.1. In addition to the aforementioned MP representations, we also trained a second variant of MoA, where the weights of the basis functions were not conditioned on time, but on the current spatial position $\boldsymbol{x}_t$. We will refer to this variant as MoA-S. Here, we also used 20 support points. The centers of the basis functions were found by clustering the demonstrations with kMeans and choosing the centers of the clusters as centers of the basis functions.

The results for three exemplary characters from the Latin alphabet are shown in Figure 4.3. The MSE between the drawn characters and the ones generated by the different time-dependent MP representations are in the same range with MoA slightly outperforming the other representations. DMPs perform worst for the letter 'a'. The reason is that the letter is not drawn very consistently. As DMPs depend linearly on the difference between start and end point, they do not seem to be robust against movements that have similar start and end points, but different shapes. The time-independent variant MoA-S has the largest average error of all MP representations. While the generated movements reflect the general shape of the letters, they either converge to a spurious attractor ('a') or enter a cycle ('e' and 's'). Please note, however, that the focus of this chapter are time-dependent movements. Therefore, the intention of MoA-S was to evaluate if it is in principle possible to learn time-independent movements with our MP representation. We consider it future work to investigate in more detail MoA's applicability for learning and representing time-independent movements, for instance by analyzing important properties such as asymptotic stability (e.g., Medina and Billard [2017], Perrin and Schlehuber-Caissier [2016]). The conclusion from the Omniglot experiment is that the sequences of MP activations resulting from the MoA optimization process lead to movements which closely follow the demonstrations.

One interesting property of MoA which was not discussed so far is that it allows for generating movements where start and point are equivalent such as drawing a circle. DMPs shape a movement with a forcing function of the form $f(z) = g(x, \boldsymbol{w})(g - x_0)$ (see (4.3)), where $\boldsymbol{w}$ are the parameters, x is the state of the system, x_0 is the initial state and g is the attractor goal. If attractor goal and initial state are the same, this forcing function evaluates to zero. In that case, another forcing function can be used [Ijspeert et al., 2013], but the type of forcing function has to be chosen beforehand. Hence, it is not possible to train a MP with the standard forcing function and then perform a movement with equivalent start and end points. To illustrate that MoA does not have this limitation, we took the activations resulting from the MoA-F variant, placed the coordinate frames of both start and point at the same arbitrary



(a) Demonstration Setups

(b) Reproduction Results

Figure 4.5.: Pictures of the six setups used for demonstrating the handwriting task (left). On the right, the setup for the reproduction is shown. For the demonstrations, we used a red pen and for the reproduction a green pen. The differences between the demonstrations and reproduction can be explained with the utilized controller and are not a result of the learning process.

location and simulated the resulting movement. Figure 4.4 depicts the movements generated by MoA. The main conclusion from the Omniglot experiment is that the sequences of MP activations resulting from the MoA optimization process lead to movements which closely follow the demonstrations.

4.4.2 Robot Handwriting Evaluation

In a second experiment, we demonstrated a handwriting task on a Barrett WAM robot via kinesthetic teaching. As end-effector, a pen was attached to the robot, as shown in Figure 4.1. The task was to first write "IAS" on one whiteboard and subsequently write "HRI" on a second whiteboard. The intention of the experiment was to evaluate the generalization capabilities of our method. Therefore, the whiteboards were placed at different locations on the table for each demonstration (see Figure 4.5). The learned skill was then reproduced on a setup which was not seen in the demonstrations. Each whiteboard was associated with a coordinate frame. During the demonstrations, we recorded the $3D$ position of the tip of the pen in world coordinates and relative to each whiteboard. In order to generalize the skill to unseen whiteboard positions, the system has to learn to control the tip of the pen in the correct coordinate frame in each phase of the task. For instance, when writing "IAS", the pen has to be controlled in the coordinate frame of the corresponding whiteboard.

Overall, we performed six demonstrations of the task. The data was recorded with a frequency of $40\,\mathrm{Hz}$. Before training, the demonstrations were aligned in time using DTW. The movement generated by our system was modulated temporally by the activation of 75 equally distributed support points. First, we trained our system using all six demonstrations. Subsequently, we put the whiteboards to positions which were different from the demonstrations and executed the skill on the real robot. The results are shown in Figure 4.5b. The robot was able to generalize to the unseen setup and mastered the task for the new whiteboard positions. The system learned to control the pen in the correct coordinate frame for each phase of the task and therefore was able to generalize the skill to the new situation. During execution, the orientation of the end-effector was determined by a predefined null-space configuration, as we focus on learning Cartesian positions in this chapter.

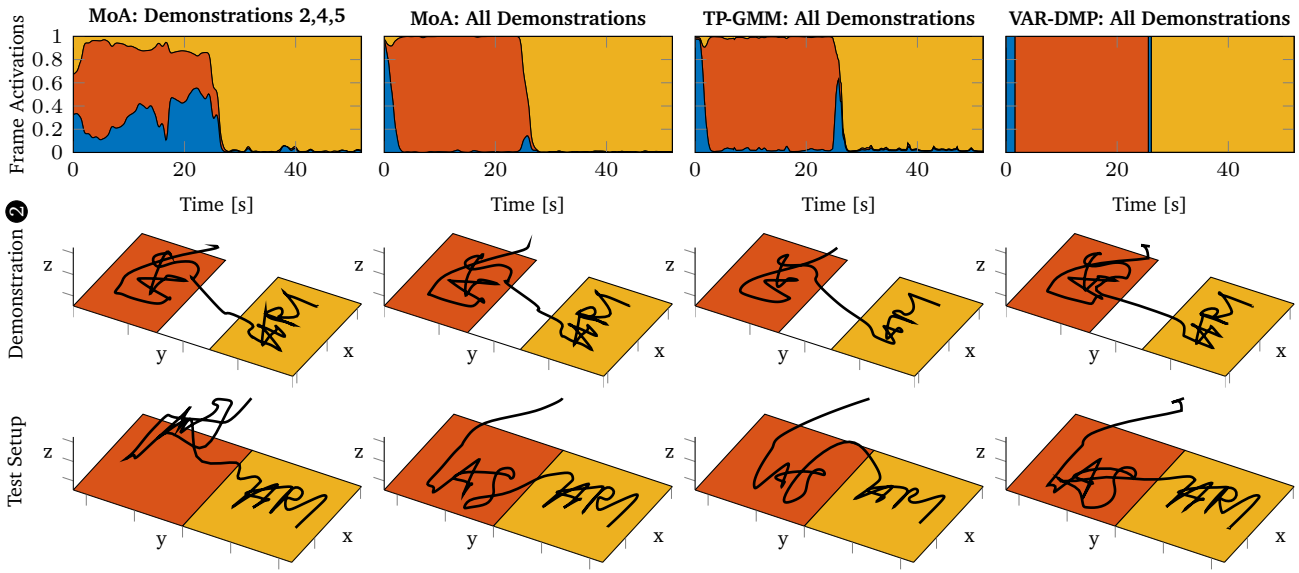


Figure 4.6.: Comparison of MoA with two other approaches. The upper plots show the frame activations over time. The colors correspond to the individual coordinate frames (● world frame, ● first whiteboard, ● second whiteboard). The remaining plots show the predicted $3D$ paths for the one demonstration (second row) and the test setup (third row). The colored rectangles correspond to the two whiteboards. For the plots on the left, we trained MoA with a combination of demonstrations which did not lead to a proper discrimination of the coordinate frames. Therefore, the skill cannot be generalized to the test setup. The other representations were able to generalize the skill to the test setup. Compared to VAR-DMP, MoA and TP-GMM also learn to blend smoothly between successive movements.

Next, we trained two state-of-the-art methods on all demonstrations and evaluated their generalization capabilities in simulation. The first method is the TP-GMM [Calinon et al., 2012]. The method uses Gaussian distributions to model the data spatially in the different coordinate frames. For a fair comparison with our approach, we augmented the state-space with time. When reproducing a skill, we conditioned the desired position on the current time, so that the movement was also modulated temporally. We used 75 Gaussians for training, as more Gaussians did not lead to an improvement of the results anymore. The second method is an approach by Ureche et al. [2015]. Here, the authors explicitly choose the coordinate frames over time based on the variance of the demonstrations. For a fixed time window and each frame, they compare the variance in this time window to the variance of the entire demonstration. The frame with the lowest value of the corresponding cost function is chosen as winner. For each resulting segment, we use DMPs to represent the movements in the corresponding frames. In the following, this approach will be referred to as VAR-DMP. Figure 4.6 shows the resulting coordinate frame selections of all three approaches. Additionally, the generated trajectories for one demonstration setup and the test setup are shown. The resulting frame activations of all three approaches look quite similar and all approaches generalize the learned skill to the test setup. VAR-DMP successfully writes the two words, but does not learn the transition phase when changing the coordinate frame. As a consequence, the generated movement is less smooth and sometimes points in the wrong direction for a short period of time (e.g., in the beginning or after writing the S of IAS). TP-GMM performs slightly worse compared to MoA. One reason why MoA follows the shape of the movement more accurately is that it takes the attractor activation recursively into account, whereas TP-GMM treats successive data points as they were independent.

In order to evaluate if our system can learn the handwriting skill from fewer demonstrations, we additionally trained it individually on all 63 possible subsets of demonstrations (e.g., demonstration 1, 2, and 4) and evaluated the generalization capabilities of the learned skill in simulation. Figure 4.6

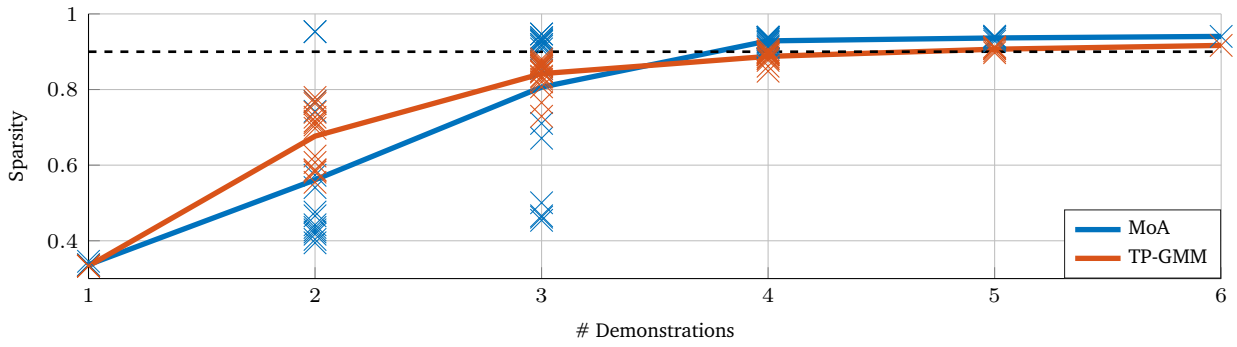


Figure 4.7.: Comparison of the sparsity of the frame activations for MoA and TP-GMMs. The solid lines show the mean of the sparsity, while the marks show the results for the individual combinations of the demonstrations. With an increasing number of demonstrations, the discrimination of the coordinate frames becomes better. We observed that a sparsity of at least 0.9 (dashed line) is required to generalize to the test setup.

shows the frame activations and generated trajectory for one exemplary subset. While some of the learned skills were able to reproduce the task for the test setup, others were not. All skills were able to reproduce the task on the setups they were trained on. For the handwriting task, a good metric for quantifying the generalization capabilities is to measure the sparsity of the individual coordinate frame activations. Ideally, the pen should be controlled in the world frame in the beginning of the task, as the robot always started from the same initial joint configuration. When writing the two words IAS and HRI, the pen has to be controlled in the corresponding coordinate frames of the whiteboards. Except for the transition phases between these three task phases, the activations of the coordinate frames should be constant. Therefore, we measure the average sparsity of the activations as metric for the generalization capabilities for the handwriting task

$$J_{\text{sparsity}} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \left\| \boldsymbol{a}_i^{(k)} \right\|_1^2, \tag{4.32}$$

where $\boldsymbol{a}_i^{(k)}$ are the activations of the attractors which are represented in coordinate frame k. In our case, $K = 3$. A fully sparse solution would result in $J_{\text{sparsity}} = 1$, while an equally distributed solution would result in $J_{\text{sparsity}} = 1/K$. Figure 4.7 shows the average sparsity for all training instances. The solutions become sparser if the skill is trained on more demonstrations. Depending on how dissimilar the whiteboard positions are in each demonstration, two demonstrations can be sufficient for learning a solution which is as sparse as the solution for all six demonstrations. We consider it future work to investigate why certain combinations lead to more sparse solutions than others and why the variance of the sparsity is larger compared to the TP-GMM.

4.5 Conclusion

This section summarizes this chapter and highlights its main contributions. The chapter will be concluded by an epilogue which discusses some open problems and gives suggestions for future work.

4.5.1 Summary of this Chapter

In this chapter, we presented the Mixture of Attractors (MoA) movement primitive representation. Due to its integration of multiple coordinate frames, MoA can be utilized for learning complex object-directed

skills from only a handful demonstrations. The resulting skills generalize well to unseen object positions and orientations. In addition, the system blends smoothly between successive movements. Learning a skill is formalized as convex optimization problem. Therefore, in contrast to most other approaches, no heuristic is needed for choosing the coordinate frames and the quality of the skill does not depend on an initial estimate of parameter values.

We evaluated our approach in simulation and on a real robot. In a first experiment, we trained MoA on handwritten letters from the Omniglot dataset. Here, the attractor activations resulting from the MoA optimization process led to movements which closely resemble the demonstrations. MoA outperformed DMPs, a widely used state-of-the-art MP representation, in terms of the MSE. In a second experiment, we evaluated the generalization capabilities of MoA on a kinesthetically demonstrated handwriting task. The learned skill was executed on a real Barrett WAM robot on a setup which was not demonstrated. From only a few demonstrations, MoA learned the important aspects of the task and outperformed two state-of-the-art approaches in terms of accuracy and/or generalization capabilities.

4.5.2 Epilogue

As shown in this chapter, MoA is a promising MP representation with many advantageous benefits compared to other representations. The evaluation on the Omniglot handwriting dataset revealed that the movements generated by MoA are very similar to those generated by DMPs. Due to MoA's support of multiple coordinate frames, movements can be conditioned on a desired start or end point or can be performed relative to objects in a scene, resulting in better generalization abilities compared to DMPs. In addition, MoA is able to represent cyclic or rhythmic movements where start and end points are equivalent without the need of changing the formulation. On the other hand, DMPs have other advantages such as convergence to a desired goal position for $t \to \infty$. While this property could also be enforced for MoA by adding a constraint to the QP (4.26), a detailed comparison of both approaches might reveal other properties which demarcate both representations. More general, a comparison of other prominent MP representations with MoA is yet to be made.

Certain aspects of the proposed method could be improved or extended in future work to scale to larger domains or improve learning performance. For example, A manipulation task is often comprised of phases where a teacher has to closely follow a desired path (e.g., when writing a letter), as well as phases where such an accurate tracking is not required (e.g., when lifting the pen between writing two letters). So far, this characteristic is not reflected in the DS we use for generating a movement. One interesting direction for future work therefore is to integrate noise into the formulation. If we add Gaussian distributed noise with mean zero and variance $\boldsymbol{\Sigma}_i$ for time step i to (4.9), we can assume a movement is generated by

$$\boldsymbol{x}_i = c_1\boldsymbol{x}_{i-1} + c_2\boldsymbol{x}_{i-2} + c_3\boldsymbol{G}_i\boldsymbol{a}_i + \boldsymbol{\epsilon}_i, \quad \boldsymbol{\epsilon}_i \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}_i\right). \tag{4.33}$$

Hence, the noise introduces a stochastic dependency of data point $\boldsymbol{x}_i$ on the previous two data points and the attractor activations

$$p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}, \boldsymbol{x}_{i-2}, \boldsymbol{a}_i) \sim \mathcal{N}\left(c_1\boldsymbol{x}_{i-1} + c_2\boldsymbol{x}_{i-2} + c_3\boldsymbol{G}_i\boldsymbol{a}_i, \boldsymbol{\Sigma}_i\right). \tag{4.34}$$

Instead of minimizing the MSE between the generated movement and the demonstrations, the activations can now be estimated by maximizing the log-likelihood

$$\log \mathcal{L}(\boldsymbol{a}_{1:N}|\boldsymbol{x}_{1:N}) = \log p(\boldsymbol{x}_{1:N}|\boldsymbol{a}_{1:N}) = \log\left(p(\boldsymbol{x_1})p(\boldsymbol{x_2})\prod_{i=3}^{N} p(\boldsymbol{x}_i|\boldsymbol{x}_{i-1}, \boldsymbol{x}_{i-2}, \boldsymbol{a}_i)\right) \tag{4.35}$$

$$= const - \frac{1}{2}\sum_{i=3}^{N}\left(\boldsymbol{x}_i - \boldsymbol{\mu}_i(\boldsymbol{a}_i)\right)^T \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{x}_i - \boldsymbol{\mu}_i(\boldsymbol{a}_i)\right), \tag{4.36}$$

where we used $\boldsymbol{\mu}_i(\boldsymbol{a}_i) = c_1\boldsymbol{x}_{i-1} + c_2\boldsymbol{x}_{i-2} + c_3\boldsymbol{G}_i\boldsymbol{a}_i$. The similarity of the log-likelihood (4.36) with the original MSE cost function (4.12) is apparent. The log-likelihood formulation (4.36) has two interesting properties compared to the original MSE formulation. First, the variance of the demonstrations $\boldsymbol{\Sigma}_i$ can be considered in a natural way. Second, it may pave the way for future research. For instance, placing a prior on the attractor activations might allow for learning the importance of the coordinate frames from a single demonstration.

We mentioned before that the memory footprint of the QP optimization is mainly determined by the quadratic matrix $\hat{\boldsymbol{H}}$ in (4.26) whose size depends on the number of support points and the number of attractor goals. While the number of attractor goals depends on the dimension of the task-spaces and is therefore task-independent, the number of support points grows with the length and complexity of a task. In order to scale MoA to larger domains, we therefore consider it future work to find a different optimization method which is less memory demanding. One idea would be to replace the activations with a softmax function. As the cost function is differentiable, gradient descent could be used for finding the activations. Introducing the softmax function also allows for removing the hard constraints on the activations, as the outcome of the function is between zero and one for all attractors and the activations inherently sum up to one. Another idea would be to exploit the convex hull of the attractor goals. By applying Radon's theorem [Tverberg, 1966], the problem may be decomposed into multiple smaller problems that can be solved more efficiently.

One downside of MoA is that in order to learn a weighting of the coordinate frames, the demonstrations have to be aligned in time using DTW. Such an alignment is only possible if the sequence of movements is the same for all demonstrations, which is not necessarily the case for a wide range of tasks. For instance, in repetitive tasks a movement may be repeated a couple of times until a a goal is achieved (e.g., unscrewing a light bulb). Another example are tasks where the subtask order is not important. When preparing a salad, it does not matter if the cucumber or the tomatoes are chopped first. If the teacher is not aware of the underlying assumptions we made, he or she might demonstrate such a task in a way which makes the time-alignment impossible. It is notable here that this issue is not specific to MoA. It applies to all approaches which take the variance of the data in different coordinate frames into account (e.g., [Ureche et al., 2015]). We consider it future work to either extend DTW to support such demonstrations or to make the required time alignment superfluous.

Even though we did not use MoA for directly controlling the joints of the robot in this chapter, we would like to mention one interesting property of MoA when using it for joint control. For a robot with one degree of freedom, two attractors are sufficient to represent any movement. If these attractors are set within the joint limits of the robot, the movements generated by the system never violate these joint limits if the dynamical system is stable. This statement is also true for an arbitrary number of degrees of freedom. In addition, it may be possible to add constraints to the optimization process which ensure that certain undesired joint configurations are not reached. We consider it future work to evaluate MoA's suitability for robot joint-control.

5 Conclusion

This thesis contributes to the state-of-the-art in the domain of Learning from Demonstration. Our specific focus was on learning sequential skills for robot manipulation tasks. In the following section, we summarize our contributions to the state-of-the-art and discuss open problems which were either out of the scope of the thesis or are directly related to the proposed methods presented in the thesis.

5.1 Summary of the Contributions

The individual chapters of this thesis treat different aspects of sequential skill learning. In Chapter 2 we looked at the skill learning problem from a high-level perspective. We dealt with the question of how to coordinate a set of single MPs in order to perform a sequential task. For this purpose, we introduced the concept of a sequence graph. In a sequence graph, each node is associated with a MP and a classifier. When executing a learned skill on a robot, the task of a classifier is to decide when to transition to a succeeding node in the graph, leading to the activation of a different MP. Hence, the task of the classifiers is to orchestrate the individual MPs. In the chapter, we evaluated different sequence graph structures and evaluated the performance of various types of linear and non-linear classifiers in simulation and in real robot experiments. One important insight from the experiments was that finding a good graph structure for a task is crucial for the overall quality of a skill. While in general a compact structure is beneficial, it also increases the chances of perceptual aliasing. This phenomenon occurs if it is not possible to find a unique mapping from the perceptual input to a corresponding movement. The evaluations showed that our approach can learn fairly complex tasks such as light bulb unscrewing from labeled demonstrations. In addition, the approach can also be utilized for teaching an error recovery strategy to the robot.

In Chapter 2, we concentrated on the coordination of the individual MPs. For this purpose, we assumed the individual MPs were known beforehand. In Chapter 3, this assumption was relaxed. We presented our probabilistic task-decomposition method which extracts a set of MPs from kinesthetic demonstrations of a task. For each MP, the method chooses an appropriate coordinate frame and the control variables (e.g., position or force) for all dimensions of the task-space associated with the coordinate frame. The number of MPs is not known beforehand and estimated via model-selection. At the core, the method is based on a probability distribution we call Directional Normal Distribution (DND). In the chapter, we presented the distribution and an Expectation-Maximization algorithm for inferring its parameters from data. In the context of skill learning, the mean of a DND can be interpreted as the most likely attractor goal of a MP and the covariance matrix as the uncertainty about the mean. For learning a skill, we combined the task-decomposition method with the sequence graph concept presented in the previous chapter. Instead of labeling the demonstrations manually with the active MPs, the task-decomposition method allows for computing the most likely sequence of MPs. Hence, the demonstrations are labeled automatically with the most likely MPs and subsequently fed into the sequence learning approach, resulting in the final skill. We evaluated the approach on three different tasks: box stacking, box flipping and light bulb unscrewing. The evaluations showed that our approach can efficiently learn skills from only two to six demonstrations and can generalize them to setups which were not demonstrated to the robot. They also showed that the resulting MPs are meaningful in a sense that each of them can be associated with a meaningful description (e.g., grasp an object). This insight is supported by the fact that the sequences of most likely MP activations for the light bulb unscrewing task are very similar to the manual labels used for the light bulb experiment in Chapter 2. In contrast to many other state-of-the-art methods, our task-decomposition approach does not require a time-alignment of the individual demonstrations. Therefore, it allows for learning skills for tasks where the demonstrated movement sequences differ between the demonstrations.

In the first chapters of the thesis, movements were represented by simple point attractors. While the sequencing graph concept is independent of the underlying MP representation, the task-decomposition approach presented in Chapter 3 can only be applied to those tasks which can be represented with point-to-point movements. Therefore, in Chapter 4, we presented the Mixtures of Attractors (MoA) MP representation. In contrast to the MP representation used in the preceding chapters, MoA allows for learning complex object-directed movements of arbitrary shape. The basic idea is to generate a movement by continuously changing the activations of a set of attractors over time. Object-directed movements can be generated by defining these attractors in different coordinate frames. We showed that learning the attractor activations with MoA can be formalized as convex optimization problem. Therefore, the optimization does not depend on a good initial estimate of the parameters. It is a unique characteristic of our approach that the importance of the individual coordinate frames emerges automatically from the optimization process without relying on a heuristic for choosing the frames. The sequence of attractor activations resulting from the optimization lead to smooth movements and smooth transitions between different object-directed movements. We evaluated MoA on a handwriting dataset and with a real robot experiment. The evaluations on the handwriting dataset showed that the movements generated by MoA closely resemble the demonstrations. We compared MoA to Dynamic Movement Primitives (DMPs) and found that the movements generated by MoA lead to smaller errors and better generalization capabilities compared to DMPs. For the real robot experiment, we demonstrated a handwriting task to a seven degrees of freedom Barrett WAM robot with a pen as end-effector. The evaluations showed that our system was able to learn the task from only two to six demonstrations and to generalize the learned skill to a setup which was not demonstrated to the robot. We compared our approach to two other approaches. Here, MoA outperformed one method in terms of generalization capabilities. The second method could be utilized for successful skill learning, but generated movements which were less smooth compared to those generated by MoA.

In summary, we tackled different skill learning aspects in a top-down manner. Starting from the high-level coordination of single movements, we moved on to learning the composition of the individual movements (e.g., coordinate frame and control variables). Finally, we also learned the shape of the movements in relation to objects in the scene. Altogether, the methods presented in this thesis contribute to bringing us closer to the dream of having autonomous robots in our homes.

5.2 Open Problems for Future Research

While we tackled many aspects of skill learning for sequential robot manipulation tasks, some other aspects were left untouched and can be considered open research problems. In addition, our research also raised new interesting questions that will be briefly described in the following section. Please note that some open problems which are directly related to our approaches were already discussed in the epilogue sections of the preceding chapters.

5.2.1 Extracting Relevant Task-Spaces from Demonstrations

In all the experiments presented in this thesis, we controlled the robot's end-effector in task-space coordinates. The reason is that for many tasks, the exact position of the other body parts of the robot are not relevant for task success. Therefore, they are usually not consistent over the demonstrations and considering them would render the learning problem more difficult. For the task-decomposition approach presented in Chapter 3, we predefined a pool of possible task-space candidates from which the algorithm could choose the best match for each MP. These candidates controlled either the position/force or the orientation of the end-effector, or the degrees of freedom of the robot's hand. While this pool of task-spaces was sufficient for our experiments, other tasks may for instance require moving the robot's elbow to a desired position. We consider it future work to expand the pool of possible task-spaces. In

addition, a method is needed which allows for deciding if certain task-spaces are relevant for a task or if they should be removed from the possible pool of task-spaces.

5.2.2 Transferring Knowledge to new Tasks

So far, new skills were learned completely from scratch and no knowledge was transferred between different tasks. Transferring such knowledge may help to reduce the necessary number of demonstrations for a new skill or may even improve learning performance in general. As an example, consider the handwriting task from Chapter 4. If we demonstrated a similar task where we wrote two different words on the two whiteboards, the knowledge about the importance of the individual coordinate frames of the handwriting task might be transferable to the new task. Transferring knowledge between tasks might also allow for refining skills if necessary. If only one small aspect of a task changes, it is conceivable that either only this aspect is demonstrated or that a full demonstration is provided, but the existing skill is reused and adapted.

5.2.3 Improving Performance over Time

Currently the behavior of the system is fixed after learning. Such a system is problematic for two reasons. First, learning a skill perfectly from demonstrations is often impossible. Even humans have to practice skills first, e.g., for instance to find out what forces to apply. Especially the direct interaction with objects is difficult, as important properties such as friction or mass distributions cannot be visually observed. Kinesthetic demonstrations alleviate this issue but do not resolve it, as the data (e.g., forces) resulting from an interaction are rather noisy. One logical next step would be applying reinforcement learning (RL) methods, as they allow for learning or improving a skill via trial and error. Even though RL methods were out of the scope of this thesis, we believe that it should be at least relatively straightforward to use the Mixtures of Attractors MP representation for RL. MoA's attractor activations are inherently in the range $[0, 1]$ and, therefore, can be approximated using softmax functions. The parameter of these softmax functions are differentiable, enabling for instance the use of policy gradient methods. In fact, Kormushev et al. [2010] already applied RL to a Dynamical System formulation which is very similar to that of MoA and therefore their work might serve as a basis for further research. Another possible application of RL is the improvement of a learned sequence graph, e.g., by adding or removing connections between the individual nodes or by refining the parameters of the individual classifiers which determine the transition behavior between MPs. Applying RL methods in this context is less straightforward and requires hierarchical models [Daniel et al., 2012, 2013, Sutton et al., 1999].

5.2.4 Recovering from Bad Demonstrations

Learning skills from kinesthetic demonstrations does not only require good algorithms for learning from data. It also requires the teacher to perform good demonstrations. Ideally, the demonstrations should provide the system a desired behavior under certain environmental conditions. The more the demonstrations deviate from what the teacher expects the robot to do, the more likely the system is to reproduce an undesired behavior. In order to be more robust against bad demonstrations, a method for detecting and discarding outliers might be helpful. It is also conceivable that the method can be combined with an approach for interactive corrections we mentioned earlier.

5.2.5 Integration of Transition Learning into Task-Decomposition

In Chapter 3, a skill was learned by first applying the proposed task-decomposition method to extract a set of MPs. Subsequently, the sequence graph concept presented in Chapter 2 was applied for learning

when to transition between the individual MPs. One issue we observed in our experiments was that the task-decomposition resulting from maximizing the log-likelihood of our model is not necessarily optimal for learning the transition behavior. For instance, the task-decomposition method may indicate a transition between two MPs prematurely, in which case the classifier resulting from the transition learning may not find a proper separating border between the two classes in the feature space anymore. In order to improve the overall skill quality, one idea would be to feed the quality of the sequence graph back into the task-decomposition and to optimize both methods in alternating order until convergence. We consider it an important research problem to develop such a method.

5.2.6 Planning Ahead

The purely reactive behaviors learned in the first part of the thesis are not necessarily optimal. Often, it is beneficial to decide which movement to perform based on potential succeeding movements. For instance, if an object has to be grasped and moved somewhere else, a human would pick a grasp and orientation dependent on the desired final position of the object and not just based on the object's current position. Such a decision requires to plan ahead at least a few steps. In order to equip our system with such an ability, planning methods have to be taken into account. While it is usually not possible to plan an entire behavior in advance, some planning might help for learning skills with better generalization capabilities. Therefore, two key questions to answer are how to balance learning and planning, and how far to plan ahead.

5.2.7 Bi-Manual Manipulation

Throughout the thesis, our focus was on learning skills for single arm robots. In order to use one of the presented methods for bi-manual manipulation tasks, further considerations have to be taken into account. In Chapter 3, we learned the MPs controlling the position and orientation of the end-effector independently of each other. One benefit of the independence is that the individual MPs are reusable in more situations. For instance, it is possible to approach the same object using the same position MP but with different orientation MPs. We already picked up this independence idea and proposed an approach where a set of MPs is learned independently for each arm, resulting in two sequence graphs [Bied, 2017]. In a succeeding step, the sequence graphs are coupled with each other such that MPs from the different graphs can be started at the same time if necessary. While the independence assumption is a straightforward way of extending the task-decomposition approach to bi-manual manipulation tasks it clearly neglects the fact that the arms have to interact with each other during some task phases. Therefore, it can be seen only as one step in the direction of learning bi-manual skills and we consider it future work to fully support learning such skills with our framework.

5.2.8 Cause and Effect of Robot Interaction

Many problems in robot learning arise from the fact that robots directly interact with the environment. Decisions such as activating or deactivating a MP are usually made based on the robot's state and that of the environment, which in turn is directly affected by the robot's actions. Without further context (just by inspecting the data without knowledge about the task), it is often unclear if a change in the environment is caused by the robot or by an external stimulus. As an example, consider an increasing magnitude of a force signal measured at the robot's wrist during a kinesthetic demonstration. Without knowledge about the task or the intention of the human teacher, it is not clear if the force is actively generated by the teacher or for instance by a second person which is pushing an object against the robot. Resolving such causality issues requires a fundamental understanding of task and environment, which is out of the scope of this thesis. In order to acquire such an understanding, physics simulations may be of importance, as they could allow for rejecting physically implausible decisions.

5.3 Publications

Excerpts of the research presented in this thesis have led to the following publications.

5.3.1 Journal Papers

S. Manschitz, M. Gienger, J. Kober, and J. Peters. Mixture of attractors: A novel movement primitive representation for learning motor skills from demonstrations. *IEEE Robotics and Automation Letters (RA-L)*, 2017b,accepted

S. Manschitz, M. Gienger, J. Kober, and J. Peters. Learning sequential force interaction skills. *Robotics and Autonomous Systems*, 2017a,submitted

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. *Robotics and Autonomous Systems*, 74:97–107, 2015b. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2015.07.005

5.3.2 Conference Papers

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Probabilistic decomposition of sequential force interaction tasks into movement primitives. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2016

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Probabilistic progress prediction and sequencing of concurrent movement primitives. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2015a

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to sequence movement primitives from demonstrations. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014a

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to unscrew a light bulb from demonstrations. In *ISR/ROBOTIK 2014*, 2014b

A Curriculum Vitae

Research Interests
Learning from Demonstrations
Learning of Sequential Skills
Movement Generation

Research Description
The particular focus of my Ph.D. project is to learn sequential skills for robot manipulation tasks. By coordinating basic elementary movements, complex sequential and parallel movement behaviors can be achieved. An illustrative example is the replacement of a light bulb: The robot's movement skill can be composed of elementary primitives, such as reaching towards the lamp, aligning the fingers with the bulb, grasping the bulb or turning it in the thread. The sequential skill is coordinating these primitives with a flexible arbitration scheme: It needs to maintain the causal order of the primitives (e.g., reach, pre-shape, grasp), while coordinating the timing of primitives that are active in parallel (co-articulation of left and right hand for bi-manual skills). In case of larger disturbances, the skill needs to adapt the sequential flow to account for the changed situation (e.g., pick up the bulb if it drops out of the hand).

Educational Background
Since 2014 Technische Universität Darmstadt / Honda Research Institute Europe, Offenbach
Ph.D. student, Department Intelligent Autonomous Systems (Prof. Jan Peters)

2011 - 2014 Master of Science in Information Systems Technology
Technische Universität Darmstadt
Final Grade: 1.0
Master's thesis: "Learning Sequential Skills for Robot Manipulation Tasks"

2007 - 2011 Bachelor of Science in Information Systems Technology
Technische Universität Darmstadt
Final Grade: 1.2
Bachelor's thesis: "Automated Conversion of Matlab Simulink Models into a Hardware Synthesizeable Form"

1997 - 2006 Abitur (A-Levels) at Ernst-Göbel-Schule, Höchst im Odenwald
Final Grade: 1.4
Major subjects: Mathematics, English, Computer Science

B Derivation of Constants

This chapter contains the derivations of some constants used within the task-decomposition approach presented in Chapter 3.

B.1 Constants for EM-algorithm

For deriving the EM-algorithm, the integral

$$\int_0^\infty p\left(t\middle|\left[\boldsymbol{x}, v\right], \boldsymbol{\theta}\right) \log p\left(\left[\boldsymbol{x}, v\right], t\middle|\boldsymbol{\theta}, \mu_t, \sigma_t\right) dt$$

has to be evaluated, which leads to

$$\begin{aligned}
&= \int_0^\infty c_1 e^{-\frac{(t-\mu_t)^2}{2\sigma_t^2}} \left(c_2(\boldsymbol{\Theta})t^2 + c_3(\boldsymbol{\Theta})t + c_4(\boldsymbol{\Theta})\right) dt \\
&= c_1 \left(c_2(\boldsymbol{\Theta})d_2 + c_3(\boldsymbol{\Theta})d_3 + c_4(\boldsymbol{\Theta})d_4\right),
\end{aligned}$$

where we used the constants

$$\begin{aligned}
c_1 &= \frac{1}{\sqrt{(2\pi)}\sigma_t} \frac{1}{1-\Phi(-\frac{\mu_t}{\sigma_t})}, \quad c_2 = -\frac{1}{2} v^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} v - \frac{1}{2\sigma_t^2}, \\
c_3 &= \frac{1}{2} v^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{x}) + \frac{1}{2} (\boldsymbol{\mu} - \boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} v + \frac{\mu_t}{\sigma_t^2}, \\
c_4 &= -\frac{1}{2} \log\left((2\pi)^{d+1} \sigma_t^2 \left|\boldsymbol{\Sigma}\right|\right) - \log\left(1 - \Phi\left(-\frac{\mu_t}{\sigma_t}\right)\right) \\
&\quad - \frac{1}{2} (\boldsymbol{\mu} - \boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu} - \boldsymbol{x}) - \frac{\mu_t^2}{2\sigma_t^2} \\
d_2 &= \mu_t \sigma_t^2 e^{-\frac{\mu_t^2}{2\sigma_t^2}} + \sqrt{\frac{\pi}{2}} \sigma_t \left(\mu_t^2 + \sigma_t^2\right) \left(1 + \operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right), \\
d_3 &= \sigma_t^2 e^{-\frac{\mu_t^2}{2\sigma_t^2}} + \sqrt{\frac{\pi}{2}} \mu_t \sigma_t \left(1 + \operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right), \\
d_4 &= \sqrt{\frac{\pi}{2}} \sigma_t \left(1 + \operatorname{erf}\left(\frac{\mu_t}{\sqrt{2}\sigma_t}\right)\right).
\end{aligned}$$

For more information, the reader is referred to Section 3.3.1.

B.2 Constants for Orientations

The start orientation rotated with an angle t around the angular velocity axis $\boldsymbol{a} = [a_x, a_y, a_z]^T$, given in the inertia frame can be defined as

$$\boldsymbol{R}_{\mathrm{VI}}(t) = \boldsymbol{R}_{\mathrm{VS}}(t) \boldsymbol{R}_{\mathrm{SI}},$$

where the entries of the matrix $\boldsymbol{R}_{\mathrm{VS}}(t)$ are

$$
\begin{aligned}
\boldsymbol{R}_{\mathrm{VS}}^{(0,0)}(t) &= (1-a_x^2)\cos(t) + a_x^2,\\
\boldsymbol{R}_{\mathrm{VS}}^{(0,1)}(t) &= -a_x a_y \cos(t) + a_z \sin(t) + a_x a_y,\\
\boldsymbol{R}_{\mathrm{VS}}^{(0,2)}(t) &= -a_x a_z \cos(t) - a_y \sin(t) + a_x a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(1,0)}(t) &= -a_x a_y \cos(t) - a_z \sin(t) + a_x a_y,\\
\boldsymbol{R}_{\mathrm{VS}}^{(1,1)}(t) &= (1-a_y^2)\cos(t) + a_y^2,\\
\boldsymbol{R}_{\mathrm{VS}}^{(1,2)}(t) &= -a_y a_z \cos(t) + a_x \sin(t) + a_y a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(2,0)}(t) &= -a_x a_z \cos(t) + a_y \sin(t) + a_x a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(2,1)}(t) &= -a_y a_z \cos(t) - a_x \sin(t) + a_y a_z,\\
\boldsymbol{R}_{\mathrm{VS}}^{(2,2)}(t) &= (1-a_z^2)\cos(t) + a_z^2,
\end{aligned}
$$

with $\boldsymbol{R}_{\mathrm{VS}}^{(i,j)}(t)$ being the entry of the ith row and jth column. The matrix can be written in the form $\boldsymbol{R}_{\mathrm{VS}}(t) = \boldsymbol{D}\cos(t) + \boldsymbol{E}\sin(t) + \boldsymbol{F}$ with

$$
\begin{aligned}
\boldsymbol{D} &= \begin{pmatrix} 1-a_x^2 & -a_x a_y & -a_x a_z \\ -a_x a_y & 1-a_y^2 & -a_y a_z \\ -a_x a_z & -a_y a_z & 1-a_z^2 \end{pmatrix},\\
\boldsymbol{E} &= \begin{pmatrix} 0 & a_z & -a_y \\ -a_z & 0 & a_x \\ a_y & -a_x & 0 \end{pmatrix},\\
\boldsymbol{F} &= \begin{pmatrix} a_x^2 & a_x a_y & a_x a_z \\ a_x a_y & a_y^2 & a_y a_z \\ a_x a_z & a_y a_z & a_z^2 \end{pmatrix}.
\end{aligned}
$$

Therefore, we can write

$$
\begin{aligned}
\boldsymbol{R}_{\mathrm{VI}}(t) &= \boldsymbol{R}_{\mathrm{VS}}(t)\boldsymbol{R}_{\mathrm{SI}}\\
&= (\boldsymbol{D}\cos(t) + \boldsymbol{E}\sin(t) + \boldsymbol{F})\\
&= \boldsymbol{D}\boldsymbol{R}_{\mathrm{SI}}\cos(t) + \boldsymbol{E}\boldsymbol{R}_{\mathrm{SI}}\sin(t) + \boldsymbol{F}\boldsymbol{R}_{\mathrm{SI}}.
\end{aligned}
$$

The values of the constant vectors $\boldsymbol{d}$, $\boldsymbol{e}$ and $\boldsymbol{f}$ used in (3.14) can then be found by multiplying the matrices and vectorizing the resulting matrices.

B.2.1 Axis Angle Derivation

The axis angle between the rotated coordinate system and the target coordinate system is defined as

$$
\theta(\phi_\omega) = \cos^{-1}\left(\frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{R}_{FV}) - 1\right)\right) \tag{B.1}
$$

$$
= \cos^{-1}\left(\frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{R}_{FS}(\boldsymbol{R}_{VS})^T) - 1\right)\right) \tag{B.2}
$$

We are only interested in the trace of the resulting transformation matrix $\boldsymbol{R}_{FV} = \boldsymbol{R}_{FV}^{(0,0)} + \boldsymbol{R}_{FV}^{(1,1)} + \boldsymbol{R}_{FV}^{(2,2)}$, which can be written as follows

$$\begin{aligned}
\boldsymbol{R}_{FV}^{(0,0)} &= \boldsymbol{R}_{FS}(0,0)(a_x^2(1-\cos(t)) + \cos(t)) \\
&\quad + \boldsymbol{R}_{FS}(0,1)(a_x a_y(1-\cos(t)) - a_z \sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(0,2)(a_x a_z(1-\cos(t)) + a_y \sin(t)), \\
\boldsymbol{R}_{FV}^{(1,1)} &= \boldsymbol{R}_{FS}(1,0)(a_x a_y(1-\cos(t)) + a_z \sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(1,1)(a_y^2(1-\cos(t)) + \cos(t)) \\
&\quad + \boldsymbol{R}_{FS}(1,2)(a_y a_z(1-\cos(t)) - a_x \sin(t)), \\
\boldsymbol{R}_{FV}^{(2,2)} &= \boldsymbol{R}_{FS}(2,0)(a_x a_z(1-\cos(t)) - a_y \sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(2,1)(a_y a_z(1-\cos(t)) + a_x \sin(t)) \\
&\quad + \boldsymbol{R}_{FS}(2,2)(a_z^2(1-\cos(t)) + \cos(t)).
\end{aligned}$$

The terms can be sorted and rearranged

$$\begin{aligned}
\boldsymbol{R}_{FV}^{(0,0)} &= a_0 \cos(t) + b_0 \sin(t) + c_0, \\
\boldsymbol{R}_{FV}^{(1,1)} &= a_1 \cos(t) + b_1 \sin(t) + c_1, \\
\boldsymbol{R}_{FV}^{(2,2)} &= a_2 \cos(t) + b_2 \sin(t) + c_2.
\end{aligned}$$

Here, we used

$$\begin{aligned}
a_0 &= \boldsymbol{R}_{FS}^{(0,0)}(1-a_x^2) - \boldsymbol{R}_{FS}^{(0,1)} a_x a_y - \boldsymbol{R}_{FS}^{(0,2)} a_x a_z, \\
a_1 &= \boldsymbol{R}_{FS}^{(1,1)}(1-a_y^2) - \boldsymbol{R}_{FS}^{(1,0)} a_x a_y - \boldsymbol{R}_{FS}^{(1,2)} a_y a_z, \\
a_2 &= \boldsymbol{R}_{FS}^{(2,2)}(1-a_z^2) - \boldsymbol{R}_{FS}^{(2,0)} a_x a_z - \boldsymbol{R}_{FS}^{(2,1)} a_y a_z, \\
b_0 &= \boldsymbol{R}_{FS}^{(0,1)} a_z - \boldsymbol{R}_{FS}^{(0,2)} a_y, \\
b_1 &= \boldsymbol{R}_{FS}^{(1,2)} a_x - \boldsymbol{R}_{FS}^{(1,0)} a_z, \\
b_2 &= \boldsymbol{R}_{FS}^{(2,0)} a_y - \boldsymbol{R}_{FS}^{(2,1)} a_x. \\
c_0 &= \boldsymbol{R}_{FS}^{(0,0)} a_x^2 + \boldsymbol{R}_{FS}^{(0,1)} a_x a_y + \boldsymbol{R}_{FS}^{(0,2)} a_x a_z \\
c_1 &= \boldsymbol{R}_{FS}^{(1,0)} a_x a_y + \boldsymbol{R}_{FS}^{(1,1)} a_y^2 + \boldsymbol{R}_{FS}^{(1,2)} a_y a_z \\
c_2 &= \boldsymbol{R}_{FS}^{(2,0)} a_x a_z + \boldsymbol{R}_{FS}^{(2,1)} a_y a_z + \boldsymbol{R}_{FS}^{(2,2)} a_z^2
\end{aligned}$$

Using $a = a_0 + a_1 + a_2$, $b = b_0 + b_1 + b_2$ and $c = c_0 + c_1 + c_2$, the axis angle becomes

$$\theta(t) = \cos^{-1}\left(\frac{1}{2}\left(a\cos(t) + b\sin(t) + c - 1\right)\right).$$

This axis angle representation is used in Section 3.3.2 to compute a minimum angle t for the current estimate of the parameters.

List of Figures

List of Algorithms

List of Tables

Bibliography

F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude. Adaptation of manipulation skills in physical contact with the environment to reference force profiles. *Autonomous Robots*, 39(2):199–217, 2015. ISSN 1573-7527. doi: 10.1007/s10514-015-9435-2.

M. A. R. Ahad, J. K. Tan, H. S. Kim, and S. Ishikawa. Human activity recognition: Various paradigms. In *Int. Conf. Control, Automation and Systems*, pages 1896–1901, Oct 2008. doi: 10.1109/ICCAS.2008.4694407.

B. Akgun and A. Thomaz. Simultaneously learning actions and goals from demonstration. *Autonomous Robots*, 40(2):211–227, 2016. ISSN 1573-7527. doi: 10.1007/s10514-015-9448-x.

M. Alvarez, D. Luengo, and N. Lawrence. Latent force models. In *Artificial Intelligence and Statistics*, pages 9–16, 2009.

M. A. Alvarez, J. Peters, B. Schölkopf, and N. D. Lawrence. Switched latent force models for movement segmentation. In *Advances in Neural Information Processing Systems*, pages 55–63, 2011.

H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters. Interaction primitives for human-robot cooperation tasks. In *IEEE Int. Conf. Robotics and Automation*, pages 2831–2837, 2014. doi: 10.1109/ICRA.2014.6907265.

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

A. Baisero, Y. Mollard, M. Lopes, M. Toussaint, and I. Lutkebohle. Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2015.

J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface*, pages 185–194, 2004.

M. Bied. Learning sequential skills for bi-manual manipulation tasks. Master's thesis, TU Darmstadt, 2017. URL http://www.ausy.tu-darmstadt.de/uploads/Site/EditPublication/Bied_Msc_2017.pdf.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. ISBN 0387310738.

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

J. Butterfield, S. Osentoski, G. Jay, and O. Jenkins. Learning from demonstration using a multi-valued function regressor for time-series data. In *IEEE-RAS Int. Conf. Humanoid Robots*, 2010.

S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, volume 3, pages 2769–2774, Sept 2004. doi: 10.1109/IROS.2004.1389828.

S. Calinon and A. Billard. Active teaching in robot programming by demonstration. In *IEEE Int. Symp. on Robot and Human*, 2007.

S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling skill constraints in joint space and task space. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 367–372, September 2008.

S. Calinon, E. Sauser, A. Billard, and D. Caldwell. Evaluation of a probabilistic approach to learn and reproduce gestures by imitation. In *IEEE Int. Conf. Robotics and Automation*, pages 2381–2388, Anchorage, Alaska, USA, May 2010.

S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *IEEE Int. Conf. Humanoid Robots*, 2012.

R. J. Carroll and D. Ruppert. *Transformation and Weighting in Regression*. Chapman & Hall, Ltd., 1988.

T. Cederborg, M. Li, A. Baranes, and P.-Y. Oudeyer. Incremental local online gaussian mixture regression for imitation learning of multiple tasks. *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 267–274, 2010.

G. Chang and D. Kulić. Robot task error recovery using petri nets learned from demonstration. In *IEEE Int. Conf. Advanced Robotics*, 2013a.

G. Chang and D. Kulić. Robot task learning from demonstration using petri nets. In *IEEE Int. Symp. Robot and Human Interactive Communication*, 2013b.

Y. Chebotar, O. Kroemer, and J. Peters. Learning robot tactile sensing for object manipulation. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014.

S. Chernova and A. L. Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014. doi: 10.2200/S00568ED1V01Y201402AIM028.

S. Chiappa and J. Peters. Movement extraction by detecting dynamics switches and repetitions. In *Advances in Neural Information Processing Systems*, 2010.

S. Chiappa, J. Kober, and J. Peters. Using bayesian dynamical systems for motion template libraries. In *Advances in Neural Information Processing Systems*, 2009.

A. Colomé and C. Torras. Dimensionality reduction and motion coordination in learning trajectories with dynamic movement primitives. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. ISSN 0885-6125.

C. Daniel, G. Neumann, and J. Peters. Learning concurrent motor skills in versatile solution spaces. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3591–3597, 2012.

C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Learning sequential motor tasks. In *IEEE Int. Conf. Robotics and Automation*, 2013.

N. Dantam and M. Stilman. The motion grammar: Analysis of a linguistic method for robot control. *IEEE Transactions on Robotics*, 29(3):704–718, 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2239553.

M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2):1–142, 2013. ISSN 1935-8253. doi: 10.1561/2300000021.

S. Dong and B. Williams. Motion learning in variable environments using probabilistic flow tubes. In *IEEE Int. Conf. Robotics and Automation*, pages 1976–1981, May 2011. doi: 10.1109/ICRA.2011.5980530.

S. Dong and B. Williams. Learning and recognition of hybrid manipulation motions in variable environments using probabilistic flow tubes. *International Journal of Social Robotics*, 4(4):357–368, Nov 2012. ISSN 1875-4805. doi: 10.1007/s12369-012-0155-x.

E. Drumwright, O. C. Jenkins, and M. J. Mataric. Exemplar-based primitives for humanoid movement classification and control. In *IEEE Int. Conf. Robotics and Automation*, volume 1, pages 140–145, 2004.

M. Elbanhawi and M. Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014. ISSN 2169-3536. doi: 10.1109/ACCESS.2014.2302442.

D. Endres, A. Christensen, L. Omlor, and M. A. Giese. Emulating human observers with bayesian binning: Segmentation of action streams. *ACM Trans. Applied Perception*, 8(3):16:1–16:12, 2011. ISSN 1544-3558.

K. Erol, J. Hendler, and D. S. Nau. Htn planning: Complexity and expressivity. In *Nat. Conf. Artificial Intelligence*, pages 1123–1128, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. ISBN 0-262-61102-3.

M. Ewerton, G. Neumann, R. Lioutikov, H. Ben Amor, J. Peters, and G. Maeda. Learning multiple collaborative tasks with a mixture of interaction primitives. In *Int. Conf. Robotics and Automation*, pages 1535–1542, 2015.

R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *Int. Joint Conf. Artificial Intelligence*, pages 608–620. Morgan Kaufmann Publishers Inc., 1971.

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

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

E. Fox, M. I. Jordan, E. B. Sudderth, and A. S. Willsky. Sharing features among dynamical systems with beta processes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, pages 549–557. Curran Associates, Inc., 2009.

S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2014.01.005.

K. Gräve and S. Behnke. Incremental action recognition and generalizing motion generation based on goal-directed features. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012.

D. Grollman and O. Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2010.

G. Guerra-Filho and Y. Aloimonos. A language for human action. *Computer*, 40(5):42–51, May 2007. ISSN 0018-9162. doi: 10.1109/MC.2007.154.

S. Hangl, E. Ugur, S. Szedmak, and J. Piater. Robotic playing for hierarchical complex skill learning. In *Int. Conf. Intelligent Robots and Systems*, 2016.

B. Hayes and B. Scassellati. Discovering task constraints through observation and active learning. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014.

N. Higham. Matrix nearness problems and applications. In *Applications of Matrix Theory*, pages 1–27. Oxford University Press, 1989.

R. Houmanfar, M. Karg, and D. Kulić. Movement analysis of rehabilitation exercises: Distance metrics for measuring patient progress. *IEEE Systems Journal*, 10(3):1014–1025, Sept 2016. ISSN 1932-8184. doi: 10.1109/JSYST.2014.2327792.

B. Huang, M. Li, R. L. De Souza, J. J. Bryson, and A. Billard. A modular approach to learning manipulation strategies from human demonstration. *Autonomous Robots*, 40:903–927, 2016. ISSN 1573-7527. doi: 10.1007/s10514-015-9501-9.

A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 50(2):21:1–21:35, 2017. ISSN 0360-0300. doi: 10.1145/3054912.

A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE Int. Conf. Robotics and Automation*, pages 1398–1403, 2002. doi: 10.1109/ROBOT.2002.1014739.

A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00393.

International Federation of Robotics. 31 million robots helping in households worldwide by 2019. In *Press Release*, 2016. URL https://ifr.org/downloads/press/02_2016/2016-DEC_20_IFR_press_release_service_robots_2019_FINAL_QS.pdf.

R. Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015. ISSN 0929-5593. doi: 10.1007/s10514-015-9459-7.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

M. Kallmann, R. Bargmann, and M. Mataric. Planning the sequencing of movement primitives. In *Int. Conf. Simulation of Adaptive Behavior*, 2004.

D. Kappler, P. Pastor, M. Kalakrishnan, M. Wuthrich, and S. Schaal. Data-driven online decision making for autonomous manipulation. In *Proceedings of Robotics: Science and Systems*, 2015.

M. Karg, W. Seiberl, F. Kreuzpointner, J. P. Haas, and D. Kulić. Clinical gait analysis: Comparing explicit state duration hmms using a reference-based index. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 23(2):319–331, March 2015. ISSN 1534-4320. doi: 10.1109/TNSRE.2014.2362862.

S. Khansari-Zadeh and A. Billard. Learning stable non-linear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

S. M. Khansari-Zadeh and A. Billard. Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752 – 765, 2014.

O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

B. Kim, A.-m. Farahmand, J. Pineau, and D. Precup. Learning from limited demonstrations. In *Int. Conf. Neural Information Processing Systems*, pages 2859–2867, 2013.

J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems*, volume 22. MIT Press, 2009.

J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.

J. Kober, M. Gienger, and J. Steil. Learning movement primitives for force interaction tasks. In *IEEE Int. Conf. Robotics and Automation*, 2015.

G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *Int. Journal of Robotics Research*, 31(3):360–375, 2012.

G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez. Symbol acquisition for probabilistic high-level planning. In *Int. Conf. Artificial Intelligence*, pages 3619–3627. AAAI Press, 2015. ISBN 978-1-57735-738-4.

P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3232–3237, Oct 2010. doi: 10.1109/IROS.2010.5649089.

N. C. Krishnan, D. Colbry, C. Juillard, and S. Panchanathan. Real time human activity recognition using tri-axial accelerometers. In *Sensors, signals and information processing workshop*, pages 3337–3340, 2008.

O. Kroemer, H. van Hoof, G. Neumann, and J. Peters. Learning to predict phases of manipulation tasks as hidden states. In *IEEE Int. Conf. Robotics and Automation*, pages 4009–4014, 2014. doi: 10.1109/ICRA.2014.6907441.

K. Kronander, M. S. M. Khansari-Zadeh, and A. Billard. Learning to control planar hitting motions in a minigolf-like task. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 710–717, Sept 2011. doi: 10.1109/IROS.2011.6094402.

D. Kulić, W. Takano, and Y. Nakamura. Combining automated on-line segmentation and incremental clustering for whole body motions. In *ICRA*, pages 2591–2598, 2008.

D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *Int. Journal of Robotics Research*, 31(3):330–345, 2012.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Int. Conf. Machine Learning*, 2001.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050.

O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys Tutorials*, 15(3):1192–1209, 2013. ISSN 1553-877X. doi: 10.1109/SURV.2012.110112.00192.

S. H. Lee, I. H. Suh, S. Calinon, and R. Johansson. Autonomous framework for segmenting robot trajectories of manipulation task. *Autonomous Robots*, 38(2):107–141, 2015. ISSN 1573-7527. doi: 10.1007/s10514-014-9397-9.

A. Lemme. *Bootstrapping movement primitives from complex trajectories*. PhD thesis, Bielefeld University, 2014.

A. Lemme, K. Neumann, F. Reinhart, and J. J. Steil. Neurally imprinted stable vector fields. In *European Symposium on Artificial Neural Networks*, pages 327–332, 2013.

A. Lemme, K. Neumann, R. Reinhart, and J. Steil. Neural learning of vector fields for encoding stable dynamical systems. *Neurocomputing*, 141(Supplement C):3–14, 2014a. ISSN 0925-2312. doi: 10.1016/j.neucom.2014.02.012.

A. Lemme, R. F. Reinhart, and J. J. Steil. Self-supervised bootstrapping of a movement primitive library from complex trajectories. In *IEEE-RAS Int. Conf. Humanoid Robots*, 2014b. doi: 10.1109/HUMANOIDS.2014.7041443.

S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. In *ArXiv*, 2016.

Z. Li, Z. Wei, W. Jia, and M. Sun. Daily life event segmentation for lifestyle evaluation based on multi-sensor data recorded by a wearable device. In *Int. Conf. IEEE Engineering in Medicine and Biology Society*, pages 2858–2861, July 2013. doi: 10.1109/EMBC.2013.6610136.

J. F. S. Lin, M. Karg, and D. Kulić. Movement primitive segmentation for human motion modeling: A framework for analysis. *IEEE Transactions on Human-Machine Systems*, 46(3):325–339, 2016. ISSN 2168-2291. doi: 10.1109/THMS.2015.2493536.

R. Lioutikov, G. Neumann, G. Maeda, and J. Peters. Probabilistic segmentation applied to an assembly task. In *Int. Conf. Humanoid Robots*, 2015.

R. Lioutikov, G. Neumann, G. Maeda, and J. Peters. Learning movement primitive libraries through probabilistic segmentation. *International Journal of Robotics Research (IJRR)*, accepted.

M. L. Littman. *Algorithms for Sequential Decision-making*. PhD thesis, Brown University, Providence, RI, USA, 1996.

T. Luksch, M. Gienger, M. Muehlig, and T. Yoshiike. Adaptive movement sequences and predictive decisions based on hierarchical dynamical systems. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012.

G. Maeda, M. Ewerton, R. Lioutikov, H. Amor, J. Peters, and G. Neumann. Learning interaction for collaborative tasks with probabilistic movement primitives. In *Int. Conf. Humanoid Robots*, 2014.

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to sequence movement primitives from demonstrations. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014a.

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to unscrew a light bulb from demonstrations. In *ISR/ROBOTIK 2014*, 2014b.

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Probabilistic progress prediction and sequencing of concurrent movement primitives. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2015a.

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. *Robotics and Autonomous Systems*, 74:97–107, 2015b. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2015.07.005.

S. Manschitz, J. Kober, M. Gienger, and J. Peters. Probabilistic decomposition of sequential force interaction tasks into movement primitives. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2016.

S. Manschitz, M. Gienger, J. Kober, and J. Peters. Learning sequential force interaction skills. *Robotics and Autonomous Systems*, 2017a,submitted.

S. Manschitz, M. Gienger, J. Kober, and J. Peters. Mixture of attractors: A novel movement primitive representation for learning motor skills from demonstrations. *IEEE Robotics and Automation Letters (RA-L)*, 2017b,accepted.

D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. In *Technical Report CVC TR98003/DCS TR1165, Yale Center for Computational Vision and Control*, 1998.

J. R. Medina and A. Billard. Learning stable task sequences from demonstration with linear parameter varying systems and hidden markov models. In *Conference on Robot Learning*, 2017.

F. Meier, E. Theodorou, F. Stulp, and S. Schaal. Movement segmentation using a primitive library. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3407–3412, 2011.

F. Meier, E. Theodorou, and S. Schaal. Movement segmentation and recognition for imitation learning. *Journal of Machine Learning Research*, 22:761–769, 2012.

N. A. Melchior and R. Simmons. Dimensionality reduction for trajectory learning from demonstration. In *IEEE Int. Conf. Robotics and Automation*, pages 2953–2958, 2010.

A. Meltzoff and M. Moore. Imitation of facial and manual gestures by human neonates. *Science*, 198 (4312):75–78, 1977. ISSN 0036-8075.

B. Michini. *Bayesian Nonparametric Reward Learning from Demonstration*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, August 2013.

K. Muelling, J. Kober, and J. Peters. Learning table tennis with a mixture of motor primitives. In *IEEE-RAS Int. Conf. Humanoid Robots*, pages 411–416, 2010.

K. Muelling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *Int. Journal of Robotics Research*, 32(3):263–279, 2013.

M. Mühlig, M. Gienger, J. J. Steil, and C. Goerick. Automatic selection of task spaces for imitation learning. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2009. doi: 10.1109/IROS.2009.5353894.

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

J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008. doi: 10.1177/0278364908091463.

C. L. Nehaniv and K. Dautenhahn. Like me? measures of correspondence and imitation. *Cybernetics & Systems*, 32(1-2):11–51, 2001.

C. L. Nehaniv and K. Dautenhahn. The correspondence problem. *Imitation in animals and artifacts*, 41, 2002.

B. Nemec and A. Ude. Action sequencing using dynamic movement primitives. *Robotica*, 30:837–846, 9 2012.

S. Niekum, S. Osentoski, G. Konidaris, and A. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012.

S. Niekum, S. Chitta, A. Barto, B. Marthi, and S. Osentoski. Incremental semantically grounded learning from demonstration. In *Robotics Science and Systems*, 2013.

S. Niekum, S. Osentoski, C. G. Atkeson, and A. G. Barto. Online bayesian changepoint detection for articulated motion models. In *IEEE Int. Conf. Robotics and Automation*, 2015a.

S. Niekum, S. Osentoski, G. D. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *International Journal of Robotics Research*, 34 (2):131–157, 2015b.

A. L. Pais, K. Umezawa, Y. Nakamura, and A. Billard. Learning robot skills through motion segmentation and constraints extraction. HRI Workshop on Collaborative Manipulation, 2013.

J. Pajarinen and V. Kyrki. Robotic manipulation of multiple objects as a pomdp. *Artificial Intelligence*, 247(Supplement C):213 – 228, 2017. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2015.04.001. Special Issue on AI and Robotics.

A. Paraschos, C. Daniel, J. Peters, and G. Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems*, 2013.

M. Pardowitz, R. Zollner, and R. Dillmann. Learning sequential constraints of tasks from user demonstrations. In *IEEE-RAS Int. Conf. Humanoid Robots*, 2005.

P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal. Towards associative skill memories. In *IEEE-RAS Int. Conf. Humanoid Robots*, 2012.

V. Pavlovic, J. M. Rehg, and J. Maccormick. Learning switching linear models of human motion. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

N. Perrin and P. Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96(Supplement C):51 – 59, 2016. ISSN 0167-6911.

J. Peters. Machine learning of motor skills for robotics. *USC Technical Report*, pages 05–867, 2005.

P. Ranchod, B. Rosman, and G. Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2015.

R. P. N. Rao, A. P. Shon, and A. N. Meltzof. A bayesian model of imitation in infants and robots. *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*, pages 217–247, 2007. doi: 10.1017/CBO9780511489808.016.

B. Reiner, W. Ertel, H. Posenauer, and M. Schneider. Lat: A simple learning from demonstration method. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2014.

L. Riano and T. M. McGinnity. Automatically composing and parameterizing skills by evolving finite state automata. *Robotics and Autonomous Systems*, 60(4):639–650, 2012.

L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell. Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints. In *Int. Conf. Intelligent Robots and Systems*, 2015.

L. Rozo, S. Calinon, D. G. Caldwell, P. Jiménez, and C. Torras. Learning physical collaborative robot behaviors from human demonstrations. *IEEE Transactions on Robotics*, 32(3):513–527, 2016. ISSN 1552-3098. doi: 10.1109/TRO.2016.2540623.

J. Schwartz and M. Sharir. A survey of motion planning and related geometric algorithms. *Artificial Intelligence*, 37(1):157 – 169, 1988. ISSN 0004-3702. doi: 10.1016/0004-3702(88)90053-7.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. ISSN 00905364. doi: 10.2307/2958889.

J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell. Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In *Int. Conf. Intelligent Robots and Systems*, pages 464–470, 2015.

L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and A. J. Smola. Hilbert space embeddings of hidden markov models. In *Int. Conf. Machine Learning*, 2010.

R. Sosnik, B. Hauptmann, A. Karni, and T. Flash. When practice leads to co-articulation: the evolution of geometrically defined movement primitives. *Experimental Brain Research*, 156(4):422–438, 2004.

F. Steinmetz, A. Montebelli, and V. Kyrki. Simultaneous kinesthetic teaching of positional and force requirements for sequential in-contact tasks. In *IEEE-RAS Int. Conf. Humanoid Robots*, 2015. doi: 10.1109/HUMANOIDS.2015.7363552.

K. Sullivan and S. Luke. Learning from demonstration with swarm hierarchies. In *Int. Conf. Autonomous Agents and Multiagent Systems*, pages 197–204, 2012.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

W. Takano and Y. Nakamura. Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. In *IEEE-RAS Int. Conf. Humanoid Robots*, pages 425–431, 2006.

A. K. Tanwani and S. Calinon. Learning robot manipulation tasks with task-parameterized semi-tied hidden semi-markov model. *IEEE Robotics and Automation Letters*, 2016.

G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In P. B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1345–1352. MIT Press, 2007.

M. Tenorth and M. Beetz. A unified representation for reasoning about robot actions, processes, and their effects on objects. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012.

H. Tverberg. A generalization of radon's theorem. *Journal of the London Mathematical Society*, s1-41(1): 123–128, 1966. doi: 10.1112/jlms/s1-41.1.123.

A. Ureche, K. Umezawa, Y. Nakamura, and A. Billard. Task parameterization using continuous constraints extracted from human demonstrations. *IEEE Transactions on Robotics*, 31(6):1458–1471, 2015. ISSN 1552-3098. doi: 10.1109/TRO.2015.2495003.

L. R. Welch. Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4):10–13, 2003.

S. D. Whitehead and D. H. Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7 (1):45–83, 1991. ISSN 0885-6125. doi: 10.1023/A:1022619109594.

A. M. Wing. Motor control: Mechanisms of motor equivalence in handwriting. *Current Biology*, 10(6): R245 – R248, 2000. ISSN 0960-9822. doi: https://doi.org/10.1016/S0960-9822(00)00375-4.

M. Yamamoto, H. Mitomi, F. Fujiwara, and T. Sato. Bayesian classification of task-oriented actions based on stochastic context-free grammar. In *Int. Conf. Automatic Face and Gesture Recognition*, pages 317–322, April 2006. doi: 10.1109/FGR.2006.28.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *Journal of Computational and Graphical Statistics*, pages 1081–1088, 2001.