Learning Sequential Skills for Bi-Manual Manipulation Tasks

Lernen von sequentiellen Fähigkeiten für beidhändige Manipulations-Aufgaben
Master-Thesis von Manuel Bied aus Horb
Tag der Einreichung:

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Heinz Koeppl
3. Gutachten: Simon Manschitz, M.Sc.

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Learning Sequential Skills for Bi-Manual Manipulation Tasks
Lernen von sequentiellen Fähigkeiten für beidhändige Manipulations-Aufgaben

Vorgelegte Master-Thesis von Manuel Bied aus Horb

1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Heinz Koeppl
3. Gutachten: Simon Manschitz, M.Sc.

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis 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 03.08.2017

(Manuel Bied)

Abstract

Bi-manual manipulation enhances the capability of robots to solve manipulation tasks. Compared to uni-manual manipulation, robots using bi-manual manipulation can solve more sophisticated and complex tasks. In order to have a robot solve a given task, the skills required for solving it have to be transferred to the robot. An intuitive process for transferring skills to robots is the approach to demonstrate tasks by a human teacher. The required skills to solve the demonstrated task are subsequently learned from the data. Learning these skills is a difficult problem. Since bi-manual manipulation tasks require coordination between the manipulators, the difficulty of coordination is added to the learning problem. Two important questions in respect to the coordination of the movements of the manipulators are *when-to-synchronise?* and *how-to-synchronise?* This thesis presents a novel approach for improving the teaching of bi-manual tasks through human demonstration to robots by addressing the questions *when-to-synchronise?* and how-to- synchronise?. These questions are addressed within the context of sequences of basic elementary movements called Movement Primitives. The main contribution of this thesis is an approach to explicitly synchronise the transitions of multiple flows of Movement Primitives. In order to achieve this explicit synchronisation the concept of Synchronisation Points is introduced. Synchronisation Points are the discrete points in time where a synchronisation of multiple flows of Movement Primitives is required. They are extracted from the demonstrated data by taking the statistics of the transitions into account. The Synchronisation Points are used to explicitly synchronise multiple flows of Movement Primitives by training classifiers for this transitions with the same data. Approaches which only consider synchronisation implicitly do not guarantee synchronisation and may fail to solve tasks which require synchronisation between the manipulators. In contrast to implicit approaches, the proposed approach ensures that transitions which are required to occur at the same time are synchronised and executed simultaneously. The approach is evaluated on two experiments. In the first experiment a *ROBOTIS-OP2* humanoid robot is used. A simple task is first kinesthetically demonstrated and successfully reproduced on the robot. In the second experiment a simulation of a two-armed robot is used. A box-picking task is evaluated in simulation. The task is demonstrated through preprogrammed behaviour added with random noise and successfully reproduced. The results indicate that the proposed approach successfully identifies the time points when synchronisation is required and ensures that this synchronisation is met during the reproduction of a task.

Keywords: Robotics, Learning from Demonstration, Kinesthetic Teaching, Movement Primitives, Sequential Skills, Bi-Manual Manipulation, Coordination, Synchronisation

Zusammenfassung

Die Fähigkeiten eines Roboters Manipulationsaufgaben zu lösen wird durch Beidhändigkeit verbessert. Die Verwendung beider Hände ermöglicht es Robotern bedeutungsvollere und komplexere Aufgaben zu lösen als mit nur einer Hand. Damit ein Roboter eine gestellte Aufgabe lösen kann, müssen dafür zunächst die benötigten Fähigkeiten auf den Roboter übertragen werden. Ein intuitiver Prozess um Fähigkeiten auf einen Roboter zu übertragen, ist der Ansatz, das Lösen der Aufgabe von einem Menschen demonstrieren zu lassen. Die benötigten Fähigkeiten um die demonstrierte Aufgabe lösen zu können werden dann aus den Daten gelernt. Das Lernen dieser Fähigkeiten ist ein schwieriges Problem. Da beidhändige Manipulation die Koordinierung beider Manipulatoren benötigt, wird das Lernproblem um die Schwierigkeit der Koordination erweitert. Unter den Fragen, die Koordination betreffen, lauten zwei wichtige Fragen: *Wann müssen die die Bewegungen der Manipulatoren synchronisiert werden?* und *Wie kann diese Sychronisation erreicht werden?* Diese Thesis präsentiert einen neuartigen Ansatz um das Lernen von Fähigkeiten zur Lösung von Aufgaben, die Beidhändigkeit erfordern zu verbessern. Diese Verbesserung wird dadurch erreicht, dass die oben genannten Fragen adressiert werden. Diese Fragen werden im Kontext von Sequenzen von Bewegungsprimitiven adressiert. Bei Bewegungsprimitiven handelt es sich um einfache elementare Bewegungen. Der Hauptbeitrag dieser Thesis ist ein Ansatz um mehrere Ablauffolgen von Bewegungsprimitiven in expliziter Art und Weise zu synchronisieren. Um diese explizite Synchronisation zu erreichen werden Synchronisationspunkte eingeführt. Synchronisations Punkte sind diskrete Zeitpunkte, an denen die Synchronisation mehrer Bewegungsfolgen von Bewegungsprimitiven erforderlich ist. Die Synchronisationspunkte werden unter Berücksichtigung von Statistiken über die auftretenden Übergänge aus den aufgenommenen Daten extrahiert. Die Synchronisationspunkte werden genutzt um die Bewegungsabfolgen explizit zu synchronisieren. Dies wird dadurch erreicht, dass die Klassifizierer, die die Übergänge zwischen Bewegungsprimitiven auslösen mit den selben Eingangsdaten trainiert werden. Ansätze, die Synchronisation nur implizit berücksitigen, garantieren keine Synchronisation. Daher besteht die Möglichkeit, dass Versuche eine Aufgaben zu lösen, die Synchronisation zwischen den Manipulatoren erfordern, misslingen. Im Gegensatz zu Ansätzen, die Synchronisation nur implizit berücksitigen, stellt der präsentierte Ansatz sicher, dass Übergange die zur selben Zeit auftreten müssen synchronisiert werden und simultan ausgeführt werden. Der Ansatz wird in zwei Experimenten evaluiert. Im ersten Experiment wird der humanoide Roboter *ROBOTIS-OP2* verwendet. Zunächst wird dem Roboter eine simple Aufgabe kinesthetisch demonstriert und anschließend erfolgreich vom Roboter reproduziert. Im zweite Experiment wird die Simulation eines zweiarmigen Roboters verwendet. In diesem Experiment wird das Lösen der Aufgabe, die darin besteht eine Box aufzuheben, in Simulation evaluiert. Die Aufgabe wird zunächst durch vorprogrammiertes Verhalten, das mit zufälligem Rauschen behaftet wird, demonstriert und anschließend reproduziert. Die Ergebnisse zeigen auf, dass der presentierte Ansatz erfolgreich die Zeitpunkte zu denen Synchronisation erforderlich ist erkennt und sicherstellt, dass diese Synchronisation beim Reproduzieren eingehalten werden.

Schlüsselwörter: Robotik, Lernen durch Demonstration, Kinesthetisches Demonstrieren, Bewegungsprimitive, Sequentielle Fähigkeiten, Beidhändige Manipulation, Koordination, Synchronisation

Contents

Figures and Tables

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Motivation

The capability of today's robots is still very limited and a lot of research needs to be done to achieve a level of full autonomy and high capability [6]. Enabling robots to solve sophisticated and meaningful tasks is an important step towards fully autonomous robots. Bi-manual manipulation enables robots to solve sophisticated and meaningful tasks which can not be solved uni-manual. Typical application areas for two-armed robots performing bi-manual manipulation are domestic and industrial environments. Since two-armed robots resemble, at least partially, humans they can be used in workspaces which are designed for humans without redesigning. The ability to use robots in human-centric environments is a key element to low-cost and flexible automation [58].

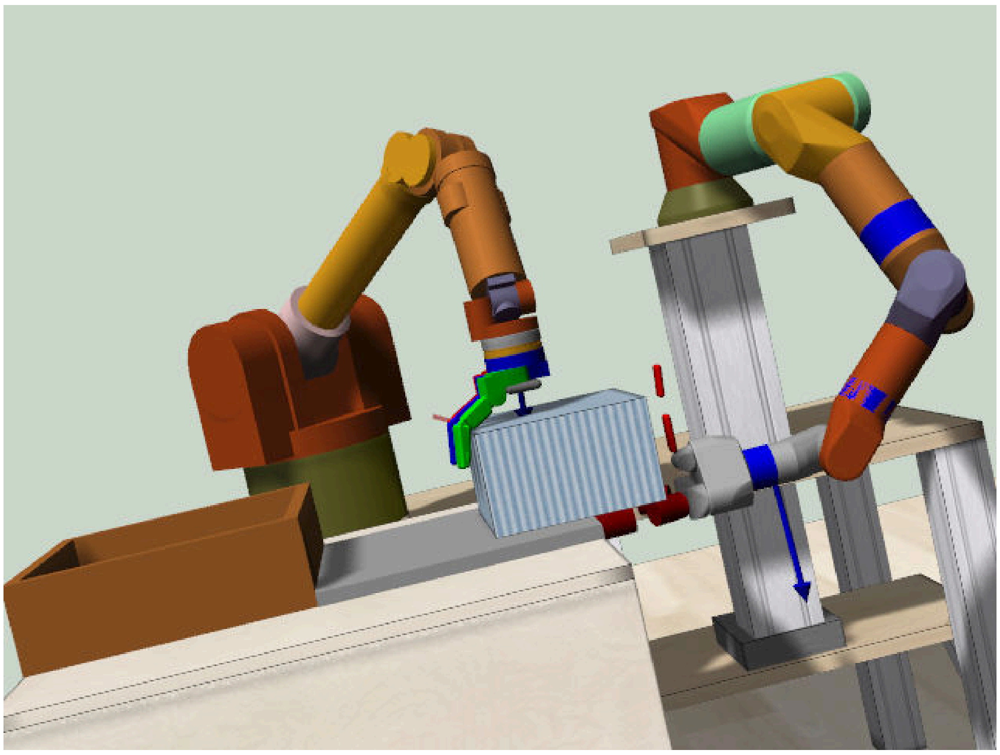


Figure 1.1: A two-armed robot is solving a bi-manual box-picking task in simulation. In order to lift up the box successfully, both robot hands need to start lifting simultaneously. The box may fall, if the movements are not coordinated between the two robot arms.

In order to have robots solving bi-manual manipulation tasks additional problems compared to the uni-manual case have to be solved. In order to illustrate these additional problems consider the task of picking up a box and placing it into the brown drawer as shown in Figure 1.1. First the robot hands need to move from their initial position to the box. Subsequently, the hands have to lift the box and place it into the drawer. Finally, the hands can move back into their initial position. In order to solve this task successfully the hands need to be coordinated. For this task there are two required coordination constraints which need to be met. On the one hand the hands need to keep a certain distance to properly grasp the box, on the other hand the lifting of the box has to start simultaneously to avoid that the box falls down. How these constraints can be ensured depends on the way the skills which solve the task are transferred to the robot (e.g. by programming).
Intuitive and easy usable tools to transfer skills to robots are another key element towards fully autonomous robots. These intuitive and easy usable tools can be used to transfer skills to robots in order to enable them to solve a wide range

of meaningful tasks. A promising way of skill transferring to robots is **Learning from Demonstration** [4]. Learning from Demonstration is done by showing a robot a certain task by a human expert. The demonstration can, for instance, be done in a way where the human teacher guides the end-effector of the robot to solve the task. Subsequently, the skills which are required to solve this task are extracted from the data. This extraction is done by learning an internal representation of the data which is able to make further predictions and decisions. The form of the representation depends on the chosen approach. A typical approach to represent skills are are basic elementary movements which are called **Movement Primitives (MPs)** [53]. One possibility to make the learning easier is to split the task into subtasks and to learn one Movement Primitive for each subtask. In addition to the Movement Primitives a representation for the sequence in which the Movement Primitives occur has to be learned. One possibility to represent the sequence is the use of directed graphs. A task can finally be reproduced by activating the learned Movement Primitives in the learned sequence.

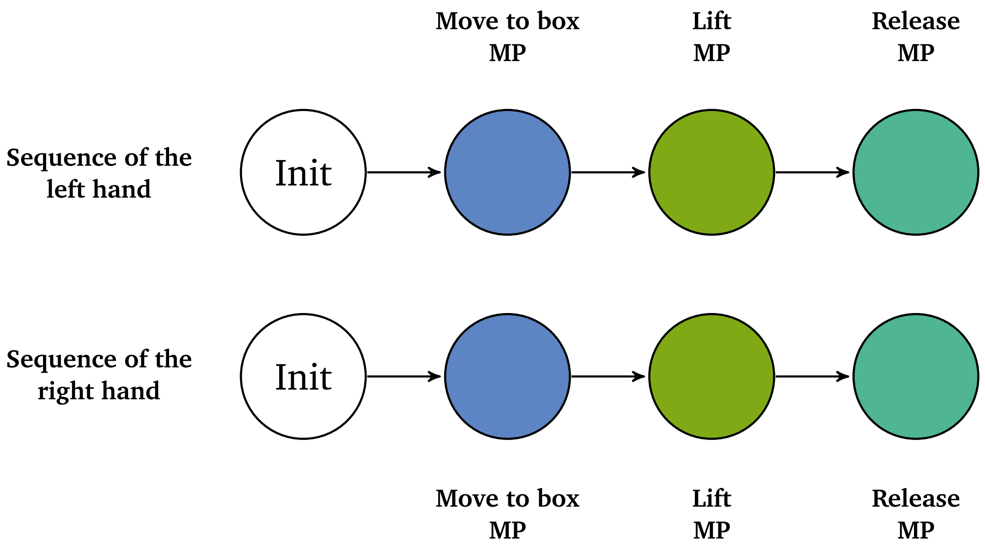


Figure 1.2: Concurrent Sequence Graph for a box-lifting task. The movements to solve the task are represented as sequences of MPs. Both hands start in its initial position. First, they move to to the box. Subsequently, the hands lift the box. Finally, when the hands reached their goal, they let go.

In order to make the connection between bi-manual manipulation tasks and sequences of Movement Primitives consider again the aforementioned example of the box-picking task. A possible representation of the Movement Primitives and their sufficient sequences to solve the box-picking task is shown in Figure 1.2. As previously stated in order to solve this task successfully it is required that both hands start lifting at the same time. In order to address this problem within the context of sequences of Movement Primitives the important questions are *Which Movement Primitives need to be activated at the same time?* and *How to ensure that they are activated at the same time during the reproduction?*.

The main contribution of this thesis is an approach to guarantee synchronisation between sequences of Movement Primitives in order to successfully solve bi-manual tasks which are taught by the Learning from Demonstration approach. Two important questions, namely *when-to-synchronise* and *how-to-synchronise* are addressed. Synchronisation is approached by the introduction of Synchronisation Points, which are extracted from the demonstrations by taking transition statistics into account. The Synchronisation Points are used to synchronise the flows of Movement Primitives explicitly by training classifiers with the same input data. Using the Synchronisation Points guarantees the necessary synchronisation between the end-effectors. Although the presented approach is introduced for the coordination of two end-effectors it can be used for an arbitrary number of sequences of Movement Primitives.

1.2 Overview of Chapters

This section gives an overview of the chapters to illustrate the structure of this thesis.

Chapter 2 introduces the necessary concepts and related work to put the contribution of this thesis into context. It explains the concepts of Learning from Demonstration, Movement Primitives and bi-manual manipulation. An overview of recent work within the field of these key concepts is given. Furthermore, the approach of Concurrent Sequence Graphs is explained in more detail.

Chapter 3 presents the main contribution of this thesis. The questions *when-to-synchronise* and *how-to-synchronise* are addressed. The concept of Synchronisation Points is introduced. It is explained how Synchronisation Points are identified in the data and how they are used to ensure synchronisation while reproducing a demonstrated task.

Chapter 4 gives an overview of the given software framework SeqLearn and the hardware platform *ROBOTIS-OP2*. It is described how the approach proposed in Chapter 3 is implemented and integrated into the given framework. Furthermore, adjustments to *ROBOTIS-OP2*'s hardware and software are presented.

Chapter 5 describes the two experiments which are conducted to evaluate the approach. As a comparison the experiments are repeated with two different approaches. Further are the results of the experiments presented and discussed.

Chapter 6 points out aspects of the approach which need further investigation. Additionally, suggestions on how to improve the approach are given.

Chapter 7 concludes this thesis. The presented approach and the obtained results are summarised. Furthermore, are the advantages of the proposed approach resumed.

2 Foundations and Related Work

Transferring skills to robots through learning approaches is an important and well researched topic within within the field of robotics. The contribution of this thesis builds upon widely spread concepts like Learning from Demonstration and Movement Primitives. Approaches which segment and sequence trajectories are often presented within the context of Learning from Demonstration and Movement Primitives. While these concepts and approaches stand on their own, they are closely related and are often used in interaction with each other. Although bi-manual manipulation is a well researched topic on its own, it is also approached within the context of learning.
This chapter presents the main ideas of the mentioned concepts, briefly explains how the concepts can be interconnected and presents related work done on these topics. Furthermore is the concept of Concurrent Sequence Graphs explained in more detail, because this concept is used for the approach presented in this thesis.

2.1 Learning from Demonstration as Method to Simplify Robot Programming

It is not possible to preprogramm robots to solve all possible tasks. For one thing the variety of thinkable tasks is large and for another thing it is difficult to cope with the uncertainty which follows from unstructured environments by programming explicit behaviour. One possible approach to deal with this problem is Learning from Demonstration, also referred to as **Programming by Demonstration** or **Learning by Imitation**, is a promising way to transfer skills to robots to solve new tasks in a user-friendly and intuitive manner [1, 4].
The questions to ensure a generic approach for transferring skills across various agents, which can for example be humans or robots, and situations can be stated as the following: *what-to-imitate*, *how-to-imitate*, *when-to-imitate* and *who-to-imitate* [41, 9]. The goal of Learning from Demonstration is to learn a (control) **Policy**. A policy represents a mapping between world state and desired actions.

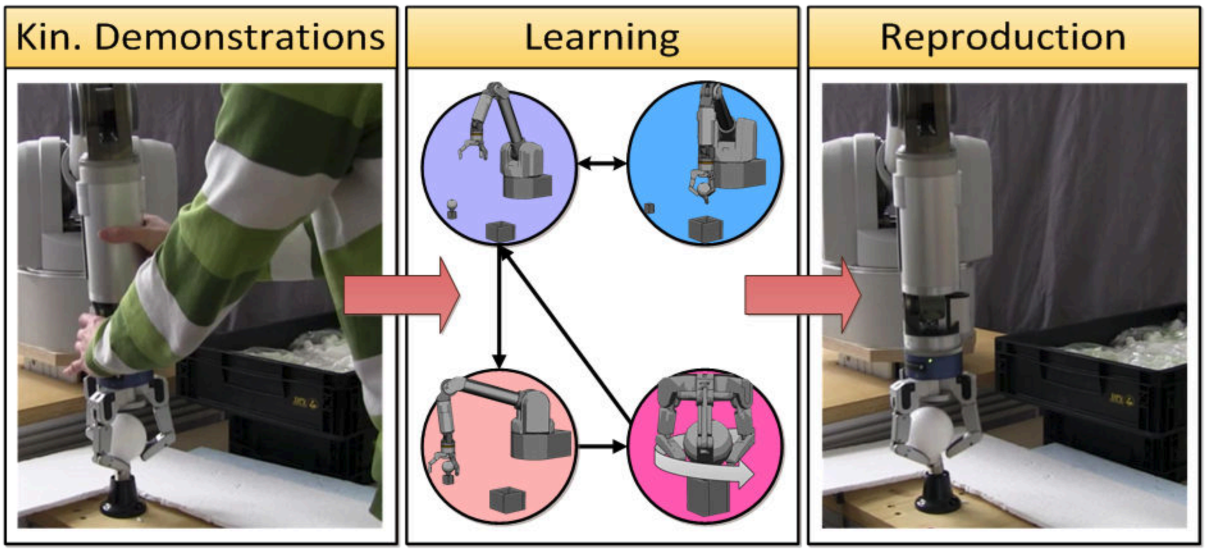


Figure 2.1: Illustration of the Learning from Demonstration procedure. Kinesthetic Teaching is used as method for demonstrating a task [35].

Pais Ureche and Billard [44] describe the typical procedure of Learning from Demonstration as follows. First, data is demonstrated and recorded. Subsequently, the data is analysed and encoded into a model of the task. The last step is to execute the task while using the learned model of the task. The procedure is shown in Figure 2.1.
Firstly, multiple examples or demonstrations are provided by a human teacher. The data capturing can be done in various forms, for example by a visual motion capture system (e.g. Lioutikov et al. [30]), using motion sensors (e.g. Steffen et al. [59]) or by **Kinesthetic Teaching** (e.g. Calinon and Billard [8]). Kinesthetic Teaching is a technique where a human directly guides the motions of the robot for solving a given task. Kinesthetic Teaching circumvents the **Correspondence**

Problem, namely the problem that the body of the teacher and the robot differ [13]. In the other aforementioned approaches a mapping from the body of the human to the body of the robot has to be found.
The next step, the analysing and encoding of the data into a model, is highly dependent on the chosen model. One possibility is to chose Movement Primitives. How the analysing and encoding of data can be done for Movement Primitives is described in Section 2.3. This step corresponds to the *what-to-imitate* problem. The last step of the process, namely the reproduction, corresponds to the *how-to-imitate* problem. For the reproduction a controller is needed which executes the desired trajectory coming from the model on the robot. Typical controllers which are used in the context of bi-manual manipulation are presented in Section 2.4.
Another prominent and important learning approach is **Reinforcement Learning**. This approach improves the behaviour of an agent like a robot in an iterative process by maximising a reward function. An approach which combines Reinforcement Learning with Learning from Demonstration is proposed by Pastor et al. [46].
The models to represent captured data and reproduce demonstrated behaviour are often biologically inspired. Schaal [52] states that Learning from Demonstration is on the one side a major step towards creating humanoid robots, but on the other side has become a new tool to investigate cognitive and biological questions.

2.2 Introducing Movement Primitives as Basic Elements

Movement Primitives can not only be used within the context of Learning from Demonstration. Many complex tasks can be divided into a sequence of simpler subtasks. The basic elementary movements to solve the subtasks are often referred to as Movement Primitives (MPs) in literature [53]. The concept of Movement Primitives is inspired by neuroscientific studies (e.g. Flash and Hochner [18] and Bizzi et al. [5]). Movement Primitives have been used to learn a variety of tasks, for example locomotion (Nakanishi et al. [40]) and the game 'ball in a cup' (Kober et al. [26]).
In order to achieve a higher autonomy of robots, it is desirable not only to have a robot which can solve one task, but to have a robot which masters a variety of skills and can learn new ones. Therefore, an important step towards autonomous robots is the automatic generation of skill libraries. Movement Primitives provide a good basis for that, since they offer a compact way of modular and reusable robot movement generators. An approach for automatic skill library creation are proposed by Chiappa et al. [11]. There exist a broad variety of approaches which incorporate different important aspects for solving complex tasks. Paraschos et al. [45] identify the following desirable properties which should be fulfilled by Movement Primitives:

1. **Co-Activation**: Multiple Movement Primitives can be activated at the same time.
2. **Modulation**: It is possible to modulate the goal of the Movement Primitive while keeping the shape of the trajectory.
3. **Optimality**: A certain optimality criterion is achieved.
4. **Coupling**: Coordination between the movement of the joints should be captured in the representation of the Movement Primitive.
5. **Learning**: The parameters of a single primitive should easily be acquired from demonstrations.
6. **Temporal Scaling**: It should be possible to execute the movement faster or slower.
7. **Rhythmic Movements**: The execution of rhythmic movements is possible.

There exist a variety of Movement Primitive approaches which implement some or all of the aforementioned desired properties, or focus on different aspects. In the following an election of different types of Movement Primitives is presented. One widely used approach is called **Dynamic Movement Primitives** [24, 54, 23]. Dynamic Movement Primitives combine Movement Primitives with **Dynamical Systems**. These Dynamical Systems are defined on the Movement Primitive's state space and incorporate state-feedback. Incorporating feedback yields the advantage of being robust against disturbances. Within the framework of Dynamic Movement Primitives the progression of desired trajectories is represented as canonical system. An approach which is proposed by Muelling et al. [39] is called **Mixture of Motor Primitives**. While adapting the approach of Mixture of Motor Primitives it is possible to select and generalise among different Movement Primitives. The work of Luksch et al. [31] introduce a framework based on a **Recurrent Neural Network**. The framework allows the activation of multiple Movement Primitives at the same time. Paraschos et al. [45] propose an approach which implements Movement Primitives in a probabilistic manner called **Probabilistic Movement Primitives**. Probabilistic Movement Primitives provide the possibility to combine and blend different Movement Primitives. Ben Amor et al. [3] propose an approach which is well suited for human-robot interaction called **Interaction Primitives**. The work of Maeda et al. [32] and Ewerton et al. [16] expands Interaction Primitives by combining them with Probabilistic Movement Primitives. The approach is called **Mixtures of Interaction Primitives**. Mixture of Interaction Primitives allow the generalisation to many different patterns instead of only one single pattern. Pastor et al. [47] add failure detection to Movement Primitives by augmenting expected sensory data. This approach is called **Associative Skill Memories**.

2.3 Sequences of Skills to Solve More Complex Tasks

Section 2.2 introduces Movement Primitives as basic elements to solve tasks. In order to extract Movement Primitives from demonstrations, the data has to be structured first. The segmentation of skills into sequences of subskills can be useful in two cases. The first case applies to tasks which can not be represented in a non-sequential manner. A reason can be that the required action is not identifiable by the global state of the system and it's environment alone. For these cases the history of actions needs to be taken into account. For example consider the situation where a robot stands in front of a door. Without any knowledge about the recent action history, it is not possible for the system to decide whether the robot just opened or just closed the door. Problems of this kind are often referred to as **Perceptual Aliasing** in literature [64]. The second case applies to more complex tasks. Even if these tasks can be solved by a single Movement Primitive, it might be advantageous to divide them into substasks first. Dividing tasks into subtasks reduces the complexity of each subtask and the resulting Movement Primitives might be easier to learn.
Learning sequences of Movement Primitives, which can be executed on a robot to solve a task, on demonstrated data is a difficult problem. There are different subproblems which have to be solved. One possibility to address the problem is to split the problem into two parts for simplifying the overall problem. The first step is the segmentation phase. In the segmentation phase the data is split into segments using certain criteria. Furthermore are the Movement Primitives learned. The second step is the sequencing phase. In the sequencing phase a model is learned which is able to reproduce the sequence structure in the data. An approach which combines both phases in one step is proposed by Daniel et al. [12], other approaches, which are presented in the following, focus on one of these two steps. The segmentation phase can either be done manually by a human who labels the data or in an automatic manner. Possible criteria to cut the data into segments are **Zero-Velocity-Crossings** proposed by Takano and Nakamura [61]. The work of Flanagan et al. [17] show the relevance of contact points for manipulation tasks, therefore also contact points can be used to segment the data. Various approaches have been proposed to segment the trajectories automatically. An important concept are **Hidden Markov Models**. A Hidden Markov Model is a directed graph. Every node in the graph corresponds to a state. The graph has two types of states: observable states and hidden states, which can be interpreted as the reason of the observable states. In the context of Movement Primitives often the hidden states correspond to the Movement Primitives. Every edge of the graph corresponds to a transition probability from one state to the next state. A common approach to estimate the number of hidden states is the **Bayesian Information Criterion** [55].
Niekum et al. [43] present an approach which uses an **Auto-Regressive Hidden Markov Model** for segmentation. Manschitz et al. [33] present an approach also based on a Hidden Markov Model, which additionally takes force interaction into account.
The sequencing phase is subsequently used to learn an upper-level layer which organises the Movement Primitives learned in the segmentation phase. While adapting this approach the switching behaviour between the Movement Primitives is interpreted as discrete events in a continuous system [48]. In the following an election of different representation forms to model the switching behaviour is given. One possibility to represent the behavioural structure are directed graphs, the events are represented as transitions in the directed graph (e.g. Kulic et al. [27]). An approach very similar to the use of directed graphs is the use of a **Finite State Machine**. Niekum and Chitta [42] proposes an approach which uses a Finite State Machine and learns with a **Beta Process Auto-Regressive Hidden Markov Model**, similar to the approach presented by Butterfield [7] which uses a **Hierarchical Dirichlet Process Hidden Markov Model** as classifier to decide whether to switch to the next Movement Primitive. Riano and McGinnity [50] presents an approach which also uses a Finite State Machine, the behaviour is learned with an evolutionary algorithm. Another approach is to use **Petri Nets** as representation (e.g. [10]). An approach which uses Petri Nets to learn a bi-manual manipulation tasks is presented by Zollner et al. [65] . An approach within the context of bi-manual manipulation which offers to teach both arms independently is proposed by Lioutikov et al. [29]. Manschitz et al. [38] propose an approach which is able to control an arbitrary number of control variables by the uses of multiple directed graphs.

2.4 Introducing Bi-manual Manipulation as Subspace of Dual-Arm Manipulation

Robots are increasingly used in environments which are originally designed for human use. Therefore anthropomorphic robots are needed. Smith et al. [58] identify the following factors as motivation for the use of a dual-arm setup:

1. **Similarity to operator**: Mainly in a teleoperated setup, it's easier to transfer the dual-arm skills from the operator to the robot.

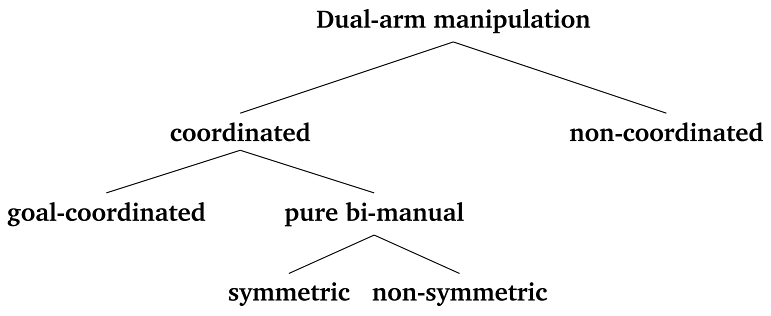


Figure 2.2: Dual-arm taxonomy according to Surdilovic et al. [60]

2. **Flexibility and stiffness**: The stiffness of a parallel manipulator can be combined with the task flexibility of a serial link manipulator by using two arms in a closed kinematic chain.

3. **Manipulability**: Tasks like peg-in-hole tasks might require to control two objects, this control can only be achieved by the use of two arms.

4. **Cognitive motivation**: Humans have an intuitive understanding of dual-arm manipulation. Therefore for setups where humans interact with robots, it is easier for the human observer to understand the action of the robot.

5. **Human form factor**: It is possible to use the robot in an environment which are designed for humans. Using robots in this environments offers the possibility to replace the human workers without redesigning the workspace.

The term dual-arm manipulation does not have an agreed upon definition. The term bi-manual manipulation is sometimes used synonymously to dual-arm manipulation. However, Surdilovic et al. [60] present a taxonomy on dual-arm operations, which is described as follows. Dual-arm operations may be divided into coordinated and non-coordinated operation. For the non-coordinated operation each arm realises an independent motion like drinking coffee with one hand and writing a text with the other hand, for the coordinated operation the motions are temporal and/or spatial coordinated. The coordinated operations can then be split into goal-coordinated motions (e.g. playing the piano) and pure bi-manual operations (e.g. lifting something with both hands). Bi-manual manipulation operations are mostly based on simple motions, which can be symmetric or asymmetric. Goal-coordinated operations are usually complicated manual activities which require long-term practice like playing the piano. This taxonomy is shown in Figure 2.2.
This taxonomy is useful to distinguish the nature of a task by the used term without knowing any more details about the task. For the concept of the proposed approach in this thesis the distinction between the goal-coordinated and the pure bi-manual does not matter. Therefore, within the course of this thesis the term bi-manual will refer to the pure bi-manual case as well as to the goal-coordinated case.
More important are the constraints which may be present in bi-manual task. These constraints can be pointed out as follows:

1. **Spatial constraints**: The positions and orientations of the arms are at least partially dependent on each other [2].

2. **Temporal constraints**: One arm can not continue executing its movements until the other arm has reached a certain subgoal [2].

3. **Force constraints**: One arm might be dominating regarding the force applied in the task. The dominating arm also might change during the task [63].

There exist various approaches how to extract and to ensure that these constraints are met. Pais Ureche and Billard [44] presents an approach which automatically extracts variables which influence other variables to determine a master-slave system. An approach which extracts pose constraints between the end-effectors is proposed by Silvério et al. [57].

A variety of approaches implement coordination on a controller level. In dual-arm manipulation tasks **Impedance Control** and **Hybrid Force/Position Control** are extensively used [58]. An impedance controller does not need a switching procedure to switch between contact and non-contact cases. The Hybrid Force/Position Controller can control the position of an object more precise. An approach which allows the coordination between multiple dual-arm robots using an Impedance Control scheme is proposed by Erhart et al. [15]. The work of Gams et al. [19] expands Dynamic Movement Primitives by force/torque feedback to achieve a desired force contact behaviour for a bi-manual task. The work of Umlauft et al. [62] propose **Cooperative Dynamic Movement Primitives**. Cooperative Dynamic Movement Primitives integrate an **Artificial Potential Field** to preserve a given formation while performing bi-manual manipulation. Gribovskaya and Billard [20] present an approach which explicitly models synchronisation on a control level. The synchronisation is achieved by adapting the velocity of one arm to the velocity of the other arm.

2.5 Representing Skills with Concurrent Sequence Graphs

The approach presented in this thesis builds upon the approach of **Concurrent Sequence Graphs** presented by Manschitz et al. [38], which will be explained in this section. Concurrent Sequence Graphs offer a form to represent skills as sequences of Movement Primitives and allow the control of an arbitrary number of control variables. Therefore concurrent Sequence Graphs can be used to represent the required skills to solve bi-manual manipulation tasks. The concept of concurrent Sequence Graphs is an extension of the concept of **Sequence Graphs**. Sequence Graphs are proposed by Manschitz et al. [35] and address the question: *When to execute which Movement Primitive?*
A Sequence Graph can be used to represent the sequence of Movement Primitives which is sufficient to solve a given task. Figure 2.3 shows the sequence which is sufficient for an one-armed robot to remove a light bulb from a socket as example.

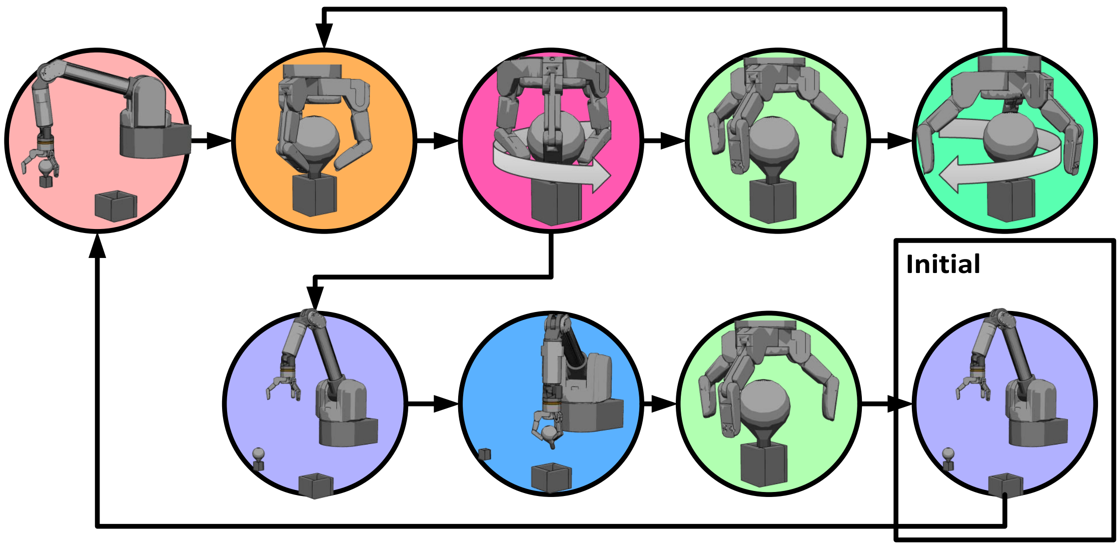
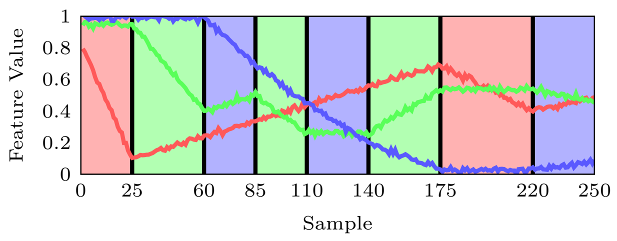
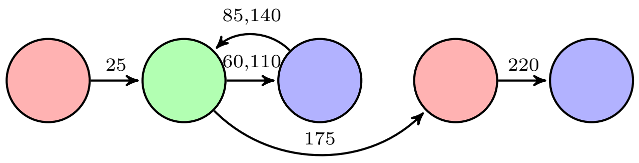


Figure 2.3: Sequence graph which is sufficient for an one-armed robot to remove a light bulb from a socket. The graphs show the sequence of Movement Primitives sufficient to solve the task. The robot starts in its initial position. First, the robot moves its hand to the light bulb. Next, it has to repeat an unscrewing sequence until the light bulb is loose. When the light bulb is loose the robot drops the light bulb into a box and returns to its initial position [35].

A Sequence Graph is created as follows. It is assumed that the Movement Primitives are already learned. The input data comes from the segmentation phase and consists of a hand-crafted feature set which is labelled for every time step with the Movement Primitive which are active during that time step. The activations of the Movement Primitives over time are called **Movement Primitive Activations**. The labelled data is subsequently used to create a Sequence Graph. A node is created for every Movement Primitive which is present in the data. Subsequently are the transitions added. For every transition from one Movement Primitive to the next Movement Primitive present in the data, a transition is added to the corresponding nodes in the graph. A set of labelled data and the corresponding Sequence Graph is shown in Figure 2.4.
As next step after the Sequence Graph has been created, one classifier is created for every node in the graph. The training input to the classifiers are the features of the associated Movement Primitive and the possible successors. The graph and the classifiers are used to decide which Movement Primitive are activated during reproduction. The reproduction is started with an initial node. Subsequently, the classifier belonging to the activated Movement Primitive has to decide at every time step on one of two options. The first option is to stay in the current activated node. The second option is to switch to one of the possible succeeding nodes in the graph. The training and the use of the classifier is illustrated in Figure 2.5.
The concept of concurrent Sequence Graphs extends the concept of Sequence Graphs to the case of an arbitrary number of control variables. In the remainder of this thesis this approach will therefore be referred to as **Concurrent Approach**.

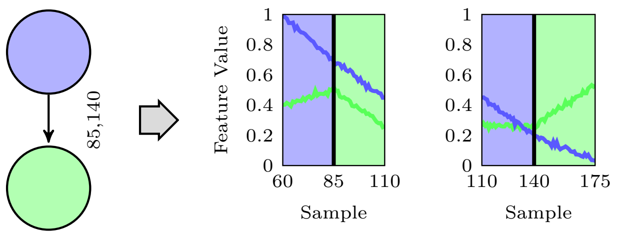


(a) "Sampled (labeled) data of one kinesthetic demonstration. The background color indicates the activated primitive, while the plot colors show which feature belongs to which primitive. In this simplified example each primitive has only one associated feature." [35]

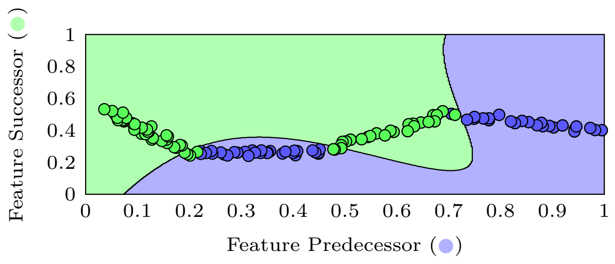


(b) "Based on the sequential order of the demonstrations, the skill is represented with a sequence graph. The labels correspond to the transitions points (TPs, see left figure). A TP is a point in time at which a switch between primitives occurs." [35]

Figure 2.4: Creation of a Sequence Graph. Every node corresponds to a Movement Primitive in the labelled data. A transition is added to the graph for every transition in the data.



(a) "One classifier is created for each node in the graph. Only the features of the previous primitive and its possible successors are used for training. In this exemplary transition from the upper sequence graph, the red primitive is not involved and hence its feature is not used." [35]



(b) "Classification result for a Support Vector Machine. Based on the training data (colored dots) the classifier finds a border separating both classes (background). During reproduction, this border is used to decide either to keep on executing the predecessor primitive or to switch to the successor." [35]

Figure 2.5: Classifiers are used to decide whether to stay in the actual node or to switch to a succeeding node. For training the classifiers the features of a Movement Primitive and its possible successors are used. The classifiers learn decision borders to decide to which node a certain feature state belongs.

Every flow of Movement Primitives controlling one variable is represented as one Sequence Graph. The Sequence Graphs are created and trained independently. Consider the task from Chapter 1 again. A two-armed robot has to lift up a box as example. The example assumes that in order to lift up the box the use of both hands is required. A possible sequence of Movement Primitives for both hands sufficient to master this task can be seen in Figure 1.2. The learning of the concurrent Sequence Graph happens as follows. The already segmented and with the associated Movement Primitives labelled data is given to the Concurrent Approach as input. First, the Concurrent Approach will learn a Sequence Graph which is able to represent the segmented data of the left hand. The transitions between the Movement Primitive is subsequently learned by training the classifiers of the left hand. Second, the Concurrent Approach will learn the Sequence Graph and train the classifiers of the right hand. Therefore, the learning process of both hands happens independently. The training input to the classifiers consists of the entire feature set representing the global state of the system. There is no explicit synchronisation between both hands. The synchronisation is ensured implicitly by training the classifiers with the global feature state of the system.

3 Learning a Synchronisation for a Bi-manual-Manipulation Task from Demonstration

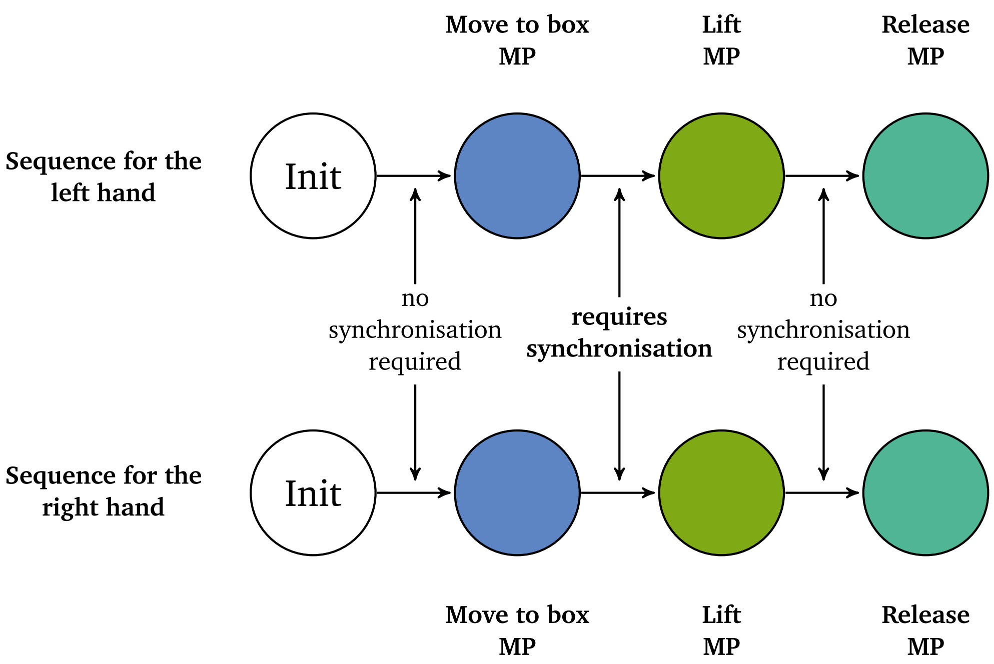
This chapter presents the main contribution of this thesis. It introduces the concept of Synchronisation Points. A Synchronisation Point indicates the start of a task phase in which the motions of two (or more) end-effectors are supposed to by synchronised with each other. When reproducing a task on a real robot, the Synchronisation Points can be used to ensure that temporal constraints between the end-effectors are met.

3.1 Problem Statement

In order to illustrate the problem consider as an example the task mentioned in Section 2.5 again. A two-armed robot has to lift up a box. The task can be described as two sequences of subtasks, one sequence for each hand. First, the left hand has to move to the box and the right hand has to move to the box. Each hand can solve its own subtask independent of the other hand. Second, both hands have to simultaneously move upwards in order to lift the box to another position. While each hand could start the lifting movement independent from the other hand the lifting of the box will fail if the hands do not start this movement simultaneously. Once the box has reached its destination, the next subtask for each hand is to move back into its initial position. This releasing from the box can be done independently. Each subtask can be solved by the activation of one Movement Primitive. Every sequence of Movement Primitives only controls one hand independent of the other hand. In this task, there is one pair of Movement Primitives which has to be started simultaneously. In order to successfully reproduce the task the lifting Movement Primitive for the left and the right hand have to started at the same time. When the hands do not start to lift the box at the same time, the box may flip or fall. In contrast to that it is not important which hand moves to the box first. Both hands can either move simultaneously to the box or consecutively in any order.
In correspondence to the questions introduced in Section 2.1 *what-to-imitate* and *how-to-imitate*, there are two important questions arising from the bi-manual nature of the task. These questions are stated as follows:

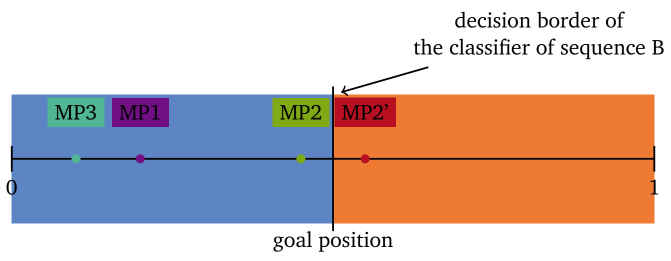
1. How to identify the Movement Primitives which need to be executed synchronously?
2. How to ensure a synchronous execution of the Movement Primitives when reproducing the task?

The proposed approach explicitly addresses these questions. The approach extends the Concurrent Sequence Approach presented by Manschitz et al. [38], which is explained in more details in Section 2.5. In order to point the problem of this approach out the learning for the Concurrent Sequence Graphs is briefly recapitulated. In the given box-picking task the learning can be described as follows. The already segmented and with the activated Movement Primitives labelled data is the input to the Concurrent Sequence Approach. First, the Concurrent Sequence Approach will learn a Sequence Graph which is able to represent the segmented data for one hand. The transitions between the Movement Primitive are subsequently learned by training classifiers. The classifiers are trained to decide on the transition by taking the entire system state as input. For example the classifiers for the sequence of the left hand take also the position of the right hand into account. Second, the Concurrent Sequence Approach will learn the Sequence Graph and train the classifiers for the other hand. Therefore the learning process of both hands happens independently. The Concurrent Sequence Approach provides no explicit synchronisation between both hands. The synchronisation is implicitly achieved by training the classifiers with the features representing the global state of the system. This implicit synchronisation may fail, because the classifiers are trained independently. The input to the classifier training is the segmented and labelled data. The set of Movement Primitives is assumed to be known. The quality of the performance of the classifiers is dependent on the accuracy of the segmentation and the learning process of the Movement Primitives. Errors during the segmentation and learning of the Movement Primitives increase error rate of the classifiers. Classification errors may lead to a failure of the reproduction of a task. Even for tasks which are represented with only one Sequence Graph the following classification errors may occur. The switching to a succeeding node can happen either to early or to late (or never). Furthermore, it is possible that a node switches to a wrong successor. In the case of bi-manual manipulation and multiple end-effectors in general there are additional error cases. In the first additional error case the reproduction fails because the switching to two Movement Primitives does not happen simultaneously. The second additional error case can occur because the classifiers are trained independently on the global feature state. This error may occur if the goal value of a Movement Primitive of one sequence does not lie within the decision border where the classifier of the other sequence trigger a transition. In this case the second sequence may get stuck. An example for such a case is shown in Figure 3.2.
The typical error cases can be summarised as follows:

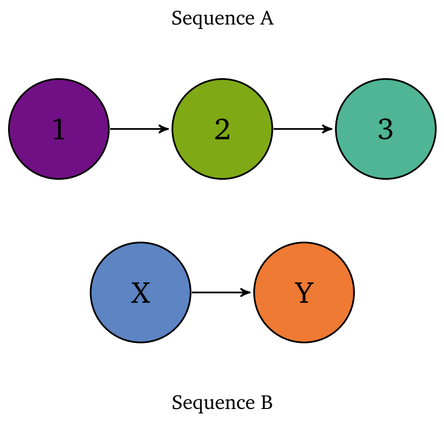


Move to box MP
Lift MP
Release MP
Sequence for the left hand
Init
no synchronisation required
requires synchronisation
no synchronisation required
Sequence for the right hand
Init
Move to box MP
Lift MP
Release MP

Figure 3.1: Concurrent Sequence Graph of an example box-picking task. The transition to the lifting Movement Primitives needs to occur synchronously in both graphs. For the other two pairs a synchronous transition is not required to successfully solve the task.



(a) MP1, MP2 and MP3 are the learned goals of Sequence A. The true underlying goal of MP2 is MP2'. The variable which is controlled by sequence A will never reach the orange region on the right.



(b) Sequence B switches from state A to state B when the variable controlled by Sequence A reaches the orange region(on the right). The learned decision border by sequence B and the learned goal by sequence A differ. Therefore, Sequence B will get stuck in state X.

Figure 3.2: Example for a sequence getting stuck while training all classifiers on the global feature set. For simplicity there is only one 1-dimensional feature which is the position of the variable controlled by Sequence A. Sequence B is assumed to be currently in the blue state (X). It switches to the orange state (Y) when the controlled variable reaches a value which lies in the orange region to the right. MP1, MP2 and MP3 belong to Sequence A. The learned goal of MP2 and the true underlying goal MP2' differ. The classifier of Sequence B has learned a decision border between MP2 and MP2'. The goal of the succeeding MP3 lies within the area which does not trigger a transition of Sequence B. Therefore, sequence B will get stuck in its current node.

1. The switching to a succeeding node occurs to early
2. The switching to a succeding node occurs to late
3. A node switches to the wrong successor
4. Transitions do not occur synchronously
5. One sequence gets stuck, because another sequence switches to early

additional for bi-manual manipulation tasks

In order to eliminate the two additional error cases it is important to explicitly address the aforementioned questions.

3.2 Introducing the Concept of Synchronisation Points

The previous section raised two important questions to be addressed in order to learn skills for solving bi-manual tasks. This section addresses the first aforementioned question: *How to identify the Movement Primitives which need to be executed synchronously?*
In order to ensure synchronisation of transitions, the information when to synchronise has to be extracted first from the data. Tasks which can be represented as independent Movement Primitives, e.g. the bi-manual box-picking task, require synchronisation at distinct points in the overall sequence. In further sections of this thesis, these distinct points are referred to as **Synchronisation Points**. Since Synchronisation Points are a key feature of the approach presented in this thesis, it is therefore referred to as **Synchronisation Point Approach**. The following describes the criteria which are used to extract the Synchronisation Points from the data.
The work of Kelso et al. [25] shows that the largest difference for the initiation of hand movements performed by humans amounts to eight milliseconds. The first property of a Synchronisation Point is proposed based on this work. The assumption is that Movement Primitives which are required to be started synchronously are demonstrated in such a way. On that condition the difference between the starting times of two Movement Primitives is close to zero. Therefore the mean of the starting time differences over all demonstrations is below a certain (small) threshold. In order to simplify the overall problem, this threshold is considered to be known, as it can be hand chosen for example.
There are other cases where the mean does not exceed the certain threshold that are not suitable to be Synchronisation Points. Consider the following case, where the set of demonstrations can be split into two subsets. In first subset transition A is demonstrated before transition B. In the second subset transition B is demonstrated before transition A. For simplicity of the example it is assumed that the subsets consists of the same number of demonstrations. Furthermore, it is assumed that in the first subset transition A is always demonstrated two seconds before transition B and for the second subset transition B is always demonstrated two seconds before transition A. In this example the mean of the starting time differences over the full set of demonstrations also calculates to zero. Therefore there are cases where the criterion of a mean close to zero is not sufficient alone. In the remainder of this thesis these cases are referred to as **Variable Transition Order** cases. Variable Transition Order cases have a larger variance than the cases which require synchronisation. In order to exclude Variable Transition Order cases from the extracted set of Synchronisation Points the variance of the starting time differences over all demonstrations must be below a certain threshold. For simplification purpose this threshold is considered to be known.
To summarise, it can be noted that, when comparing the starting times of Movement Primitives to extract Synchronisation Points, the differences of the starting times need to fulfil two properties. These two properties are:

1. **The mean of the differences is close to zero**
2. **The variance of the differences lies below a certain threshold**

If both of these properties are met the pair of transitions is extracted as a Synchronisation Point. For sequences of an arbitrary number of control variables the set of Synchronisation Points can be extracted by comparing all combinations of all transitions of one sequence to all transitions of all other sequences. The Synchronisation Points can now be used to ensure synchronous execution of Movement Primitives as described in Section 3.3.

3.3 Learning the Transition Behaviour Between Movement Primitives

The previous section describes the extraction of the Synchronisation Points as answer to the first question raised in Section 3.1. This section will describe how to use the Synchronisation Points in order to address the second previously stated question: *How to ensure a synchronous execution of the Movement Primitives when reproducing the task?*
Within this thesis the transitions from one active Movement Primitive to the next Movement Primitive are triggered

by classifiers. Therefore possible answers to the aforementioned question will stay within the scope of classifiers. The concept will be kept general without deciding on a certain type of classifier. The behaviour of the classifiers can be influenced by the feature set the classifiers are trained on. Without using a more advanced selection method to chose features there are two simple possibilities to choose the feature set from. In the first possibility the classifiers of a sequence of Movement Primitives use only the features which are directly controlled by the sequence. For example, for the sequence which controls the position of the left end-effector, the controlled variables are the X, Y and Z coordinate of the left end-effector. While adapting this approach there will be no synchronisation between the Movement Primitives. The reproduction of a task will only succeed by chance. The other possibility is to include the whole feature set representing the entire state of the system. This possibility goes along with the problems which are pointed out in Section 3.1. Therefore the approach presented in this thesis addresses the problems which arise from the bi-manual nature of a task, namely that the switching does not happen synchronously and that one sequence may get stuck. In order to achieve that the classifiers associated with a Synchronisation Point behave the same they are trained with the same training data. For every demonstration there are three possibilities to chose the used data from. The first possibility is to use the data of the transition which switches first. The second possibility is to use the data of the transition which switches last. The third possibility is to modify the data to represent a mixture form of both cases. While the choice which of these three options to use does influence the behaviour, it does not influence, whether the required classifiers switch synchronously and is not important for the main concept of the algorithm. This choice can therefore be seen as an implementation detail. The implementation of the approach is explained in Section 4.2.2. The classifiers are subsequently trained on the chosen data. Addressing the problem in this way ensure a synchronous execution of the Movement Primitives when reproducing a task.

Therefore both questions which are stated in Section 3.1 are answered within this chapter. The following chapter presents the given framework and how the presented approach is implemented into the framework.

4 Integration into Skill Learning Framework

The proposed approach is implemented within the context of a given learning framework. In addition to the implementation of the approach, modifications to the hardware platform are done. This chapter provides an overview over the given hardware and software framework. The different system components and their interaction is explained. Subsequently, the implementations and modifications are explained.

4.1 System Overview

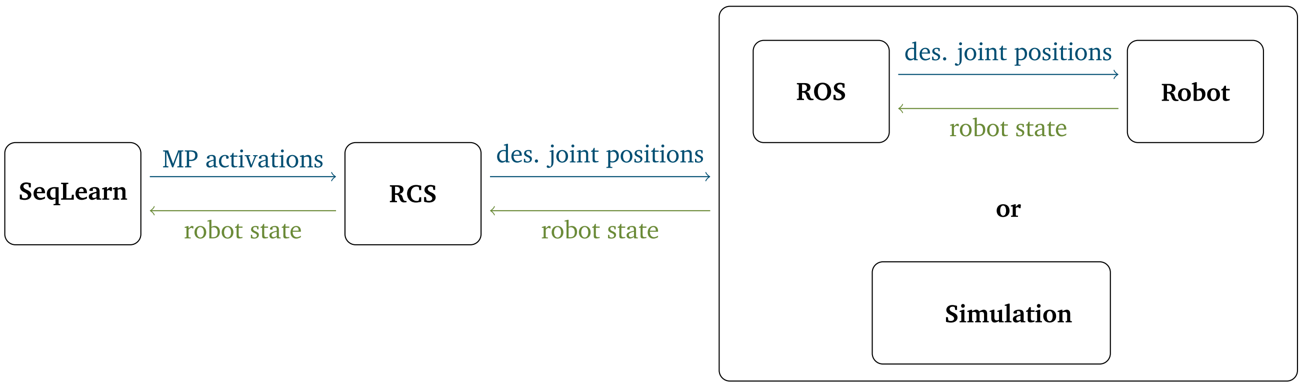


Figure 4.1: Overview of the system components and their interaction. The information which flows from the robot towards the software framework is shown in green, the information which flows from the software part towards the robot is shown in blue. In order to reproduce a task SeqLearn activates a set of Movement Primitives based on the state of the robot. RCS calculates the desired joint positions from the activated Movement Primitives. The desired joint positions are send either to a simulation or to ROS. ROS passes the desired joint positions on to the robot. The movements to reach the desired joint positions are executed on the robot or in simulation. The state of the robot is communicated backwards from the robot or the simulation to SeqLearn.

An overview of the system which is used during this thesis can be seen in Figure 4.1. The software part of the system is shown on the left, the hardware part, which can either be real hardware or simulated hardware, is shown on the right. **SeqLearn** is a high level framework which is responsible for learning skills to solve demonstrated tasks. For the reproduction of a task SeqLearn activates a set of Movement Primitives according to the given state of the robot and the learned models. The **Robot Control System (RCS)** provides an implementation of Movement Primitives based on the work of Luksch et al. [31]. RCS calculates the desired joint positions for a given set of activated Movement Primitives. RCS also provides a simulator for different robots. The calculated desired joint positions are subsequently send to the robot. For the *ROBOTIS-OP2* the desired joint positions are first send to the middleware **Robot Operating System (ROS)**, which is widely used within the robotics community. ROS provides a variety of features like implementations of computer vision and navigation algorithms, but within this thesis only the modules which provide communication features are used. The required movements to reach the desired joint positions are subsequently executed in simulation or on the robot. The (simulated) state of the robot, which consists of the current joint positions and additional sensor values from the actuators, is communicated back from the robot over RCS to SeqLearn.

4.2 Introducing the SeqLearn Framework

This section introduces the high-level learning framework **SeqLearn**, which is used to learn skills from demonstrations. A brief overview of the pipeline is given. Furthermore it is explained how the presented approach is implemented and integrated into the existing framework.

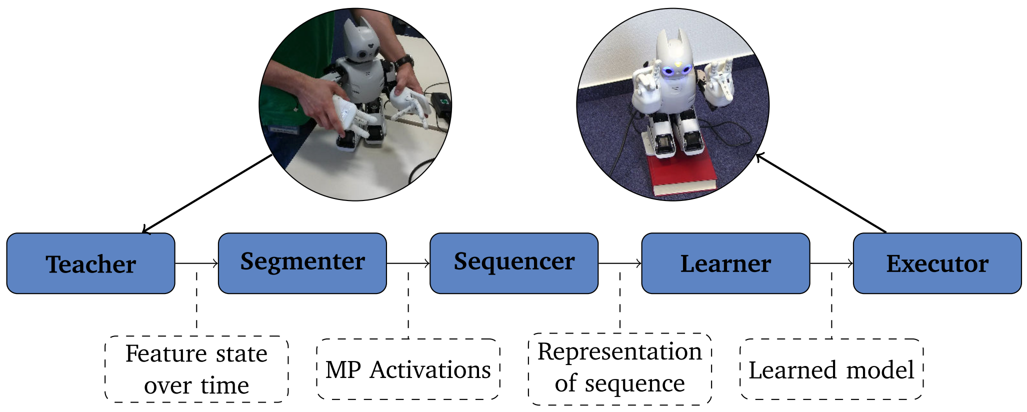


Figure 4.2: The pipeline of the SeqLearn Framework. Firstly, a task is taught through Kinesthetic Teaching, the Teacher records the raw data and converts it into a set of hand-crafted features. The Segmenter segments the data and adds the Movement Primitive Activations to data set. The Sequencer creates a representation of the sequence present in the data. The Learner trains a model which is able to reproduce the data. The Executor uses the learned model to finally reproduce the demonstrated task.

4.2.1 From Demonstration to Reproduction: the SeqLearn Framework

The SeqLearn pipeline is an implementation of the process of demonstrating a task to its reproduction (cf. Chapter 2). The process can be split into five submodules: **Teacher**, **Segmenter**, **Sequencer**, **Learner** and **Executor**. Figure 4.2 illustrates the process of the pipeline. The Teacher is used to capture data which is acquired by Kinesthetic Teaching. The used features are usually not the raw joint angles and sensor values, but rather a representation in task-level space like position and orientation of the end-effectors for example. Alternatively, simulation can be used to capture the data. In the case of simulation a Finite State Machine is used to define the sequence of the activation of Movement Primitives. The next module is the Segmenter. The input to the Segmenter are the features which are the output of the Teacher. First potential cuts between successive segments are detected by using contact events and Zero-Velocity-Crossings (cf. Section 2.3). Each segment is assigned to a Movement Primitive using a Hidden Markov Model (cf. Section 2.3). The output of the Segmenter are features which are labelled with the Movement Primitive Activations. More details on the segmentation are given in [33]. The input of the Sequencer are the features and the associated labels. The Sequencer is used to learn directed graphs as representation of the possible Movement Primitive sequences. The Learner learns a model for the transitions from one Movement Primitive to the succeeding Movement Primitives. This learning can for example be done by the use of classifiers. SeqLearn provides a variety of classifier implementations, for example Support Vector Machines, Sparse Logistic Regression and Progress Prediction (an approach proposed by Manschitz et al. [33]). The sequencing and learning phase are closely related and influence each other. Further details on the sequencing and classifier training can be found in [35] and [37]. The last module is the Executor. The Executor is instantiated with the learned model, its input is the current state of the robot in real-time. The Executor decides based on this state which Movement Primitives are to be activated. The decision is based on the output of the learned model.

4.2.2 Integrating Synchronisation into SeqLearn

The Synchronisation Point Approach, which is proposed within this work (cf. Chapter 3), is implemented within the scope of the Sequencer and Learner of the SeqLearn framework. The implementation of the Synchronisation Point Approach consists of three algorithms. The first algorithm is used to create the independent graph sequences. The second algorithm is used to extract the Synchronisation Points. The last algorithm integrates the Synchronisation Points into the training of the classifiers. The first algorithm to create the independent graph sequences is shown in Algorithm 1 as pseudo code. The input of the algorithm is the output of the Segmenter. The input consists of the feature set $F(j)$ and the label set $L(j)$ with $j = 1, ..., M$ for a task which has been demonstrated M times. The number of graphs which will be created is C. This number corresponds to the number of the controlled variables in the task. Be m the index of the demonstration, d the dimension of the feature set and n_m the the number of data points for demonstration m. The dimensions for the data sets of one demonstration are $F(m) \in \mathbb{R}^{d \times n_m}$ and $L(m) \in \mathbb{R}^{c \times n_m}$. The graphs are created independently from each other and added to the graph set one after another. Each graph is created with an initial node without transitions. A node represents the activation of a Movement Primitive and for every Movement Primitive which occurs in the labels a node is created. A transition is created for every observed transition between Movement Primitives in $L(m)$. The trajectory (in

Algorithm 1: Generation of Independent Graph Sequences

```
Data: Features $F(M)$, Labels $L(M)$
Result: Returns the Set of Independent Graph Sequences $G$
1  $G \leftarrow createEmptySetOfGraphs()$;
2  for $c = 1 : C$ do
3      $g_c \leftarrow createEmptyGraph()$;
4      for $m = 1 : M$ do
5          if not $g_c.hasNodeWithLabel(L(m)_{(c,1)})$ then
6              $node \leftarrow g_c.addInitialNode(L(m)_{(c,1)})$ ;
7          $trajectory \leftarrow createEmptyTrajectory()$ ;
8          $trajectory.addData(F_1(m))$ ;
9          for $i = 2 : n_m$ do
10             if $L(m)_{(c,i)} \neq L(m)_{(c,i-1)}$ then
11                 if not $g_c.hasNodeWithLabel(L(m)_{(c,i)})$ then
12                     $node \leftarrow g_c.addInitialNode(L(m)_{(c,i)})$ ;
13                     $g_c.addTransition(L(m)_{(c,i-1)}, L(m)_{(c,i)}$ ;
14                 $node.addTrajectory(trajectory)$ ;
15                 $trajectory.setToEmpty()$ ;
16             $trajectory.addData(F_i(m))$
17     $G.addGraph(g_c)$;
18 return $G$ ;
```

terms of the recorded features) which is associated with the activation of a Movement Primitive is added to the node for every demonstration. The output of the algorithm is the set of independent graph sequences G.
Algorithm 2 shows the pseudo code which is implemented to extract the Synchronisation Points. The input to the algorithm is the set of independent graph sequences G. The output of the algorithm is the set of Synchronisation Points SP. T_i denotes the number of transitions in the graph with index i. A transition $transition^{(M)}$ consists of the timings $timings^{(M)}$ of the M occurrences of the transition. Every Synchronisation Point has two transitions associated with it. These two transitions are denoted as $transition^{(M)}_{(a)}$ and $transition^{(M)}_{(b)}$. The thresholds $threshold_{mean}$ and $threshold_{var}$ denote the manually set threshold for the mean and variance for accepting a pair of two transitions as Synchronisation Point. In order to extract the Synchronisation Points every transition in a graph is compared to all transitions in the other graphs. For one comparison of two transitions the difference of the the occurrence times is calculated for each demonstration. As a next step, the mean and the variance of these differences are calculated. If the mean and the variance is below the chosen threshold the set of these two transitions is marked as Synchronisation Point and added to the set of Synchronisation Points SP.
The pseudo code for the training of the classifiers in consideration of the Synchronisation Points is shown in Algorithm 3. The input to the algorithm are the set of independent graph sequences G, the set of Synchronisation Points SP and a chosen classifier type CT. N denotes the number of Synchronisation Points and sp_n denotes the Synchronisation Point with index n. In addition to the occurring timings, which are introduced in Algorithm 2 a transition now consists of a associated classifier and for each demonstration a copy of the trajectory of the Movement Primitive which is active before the transition and after the transition. For every demonstration and every Synchronisation Point, the trajectory with the earlier occurring transition time is replaced by the trajectory of the later occurring transition time. Once the replacement process is done for all Synchronisation Points, a classifier is learned for every transition by using the the copies of the trajectories before and after the transition as input.

Algorithm 2: Extraction of Synchronisation Points

```
Data: Set of Independent Graph Sequences $G$
Result: Returns the Set of Synchronisation Points $SP$
SP ← createEmptySetOfSynchronisationPoints();
for $c$ = 1 : C-1 do
    for $a = 1 : T_c$ do
        $transition^{(M)}_{(a)} \leftarrow g_c.getTransition(a)$ ;
        $timings^{(M)}_{(a)} \leftarrow transition^{(M)}_{(a)}.getTimings()$ ;
        for $k$ = c+1 : C do
            for $b = 1 : T_k$ do
                $transition^{(M)}_{(b)} \leftarrow g_c.getTransition(b)$ ;
                $timings^{(M)}_{(b)} \leftarrow transition^{(M)}_{(b)}.getTimings()$ ;
                $mean \leftarrow calculateMeanOfTransitionDifferences(timings_{(a)}, timings_{(b)})$ ;
                $variance \leftarrow calculateMeanOfTransitionDifferences(timings_{(a)}, timings_{(b)})$ ;
                if $mean < threshold_{mean}$ AND $var < threshold_{var}$ then
                    $sp \leftarrow createSynchronisationPoint(transition^{(M)}_{(a)}, transition^{(M)}_{(b)})$ ;
                    $SP.add(sp)$ ;
return $SP$;
```

Algorithm 3: Classifier Training

```
Data: $G$, $SP$, Type of Chosen Classifier $CT$
Result: Trains the Classifier in a Synchronised Manner
for $n = 1 : N$ do
    $sp_n \leftarrow SP.getSynchronsiationPoint(n)$ ;
    $transition^{(M)}_{(a)}, transition^{(M)}_{b} \leftarrow sp_n.getTransitions()$ ;
    $timings^{(M)}_{(a)} \leftarrow transition^{(M)}_{(a)}.getTimings$ ;
    $timings^{(M)}_{(b)} \leftarrow transition^{(M)}_{(b)}.getTimings$ ;
    for $m = 1 : M$ do
        if $timings^{(m)}_{(b)} > timings^{(m)}_{(a)}$ then
            $transition^{(m)}_{(a)}.replaceTrajectory(transition^{(m)}_{(b)})$
        else
            $transition^{(m)}_{(b)}.replaceTrajectory(transition^{(m)}_{(a)})$
for $c = 1 : C$ do
    for $t = 1 : T_c$ do
        $transition^{(M)} \leftarrow g_c.getTransition(t)$ ;
        $dataBeforeTransition \leftarrow transition.getDataBeforeTransition()$ ;
        $dataAfterTransition \leftarrow transition.getDataAfterTransition()$ ;
        $classifier \leftarrow createClassifierOfType(CT)$ ;
        $classifier.train(dataBeforeTransition, dataAfterTransition)$ ;
        $transition.add(classifier)$ ;
return;
```

4.3 Integration with *ROBOTIS-OP2*

The platform which is used for evaluating the proposed approach is a *ROBOTIS-OP2*. It is a small sized (approximately 50 centimetres) humanoid robot with 20 degrees of freedom. *ROBOTIS-OP2* is manufactured by the Korean manufacturer *Robotis* and is developed in collaboration with *Virginia Tech*, *Purdue University*,and *University of Pennsylvania*. It is developed as expandable, modifiable research platform by Ha et al. [22]. The CAD models and the software framework are open source. In order to evaluate the proposed approach on the *ROBOTIS-OP2* several modifications on the hardware and the corresponding software framework are done. These modifications can be summarised into two categories. The first category is the enabling of Kinesthetic Teaching for *ROBOTIS-OP2*. The second category is the installation of a pair of additional hands. The modified version of *ROBOTIS-OP2* with the newly installed pair of hands is shown in Figure 4.3. For a better understanding of the modifications a brief overview of the lower level software framework is given. Subsequently are the modification described.

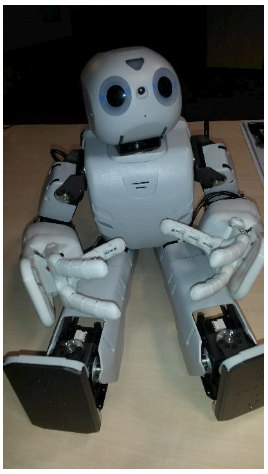


Figure 4.3: The small humanoid *ROBOTIS-OP2* with its new pair of hands.

4.3.1 Overview of the Lower Level Framework

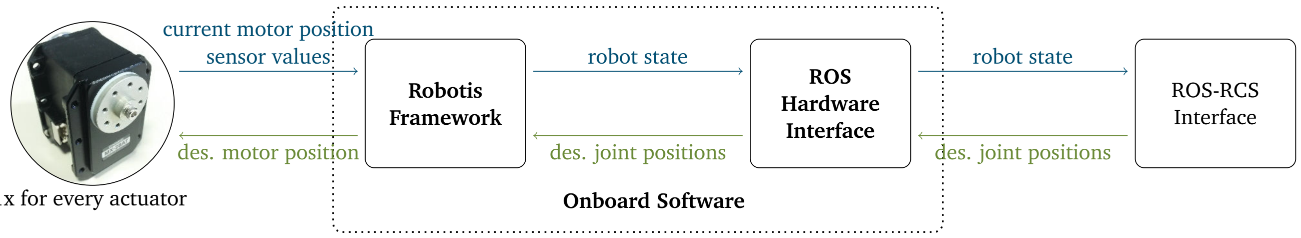


Figure 4.4: Overview of the lower level framework. The RCS module is split into one interface to RCS and one interface to the onboard software framework which controls the robot. The necessary motor positions to reach the desired joint positions coming from RCS are calculated by the Robotis Framework. The desired positions for each motor are send to the associated actuator. The actuator sends the current motor position and other sensor values back to the Robotis Framework.

This section gives an overview of the lower level framework as it is shown in Figure 4.4. The software framework that originally runs on the robot is called **Robotis Framework**. It is used to control the robot on a level of desired joint positions. The Robotis Framework calculates the corresponding target motor positions and sends them to the actuators. The actuators use a PID controller to control the motor positions. The Robotis Framework also reads the current motor positions and other sensor values (e.g. camera image). The current joint positions are then derived from current motor positions. The ROS module, which is already presented in Section 4.1 consists of two submodules. The first submodule is referred to as **ROS-Robotis Interface**. It runs on the robot and provides a ROS interface to the Robotis Framework. The second submodule is referred to as runs on the side of RCS, and provides an interface to RCS and is called the ROS-RCS Interface. The desired joint positions are then propagated from RCS to the Robotis Framework over the two ROS modules. Vice versa are the current joint positions and other sensor values propagated to RCS from the Robotis Framework.

4.3.2 Enabling of Kinesthetic Teaching for *ROBOTIS-OP2*

While similar robots as *ROBOTIS-OP2* are being used for Kinesthetic Teaching (e.g. the *Nao* small humanoid robot [49]), to the best of the author's knowledge, *ROBOTIS-OP2* has not been used for Kinesthetic Teaching before. The research

ROBOTIS-OP2 is used for seems to focus on gait patterns and walking algorithms [21, 28, 51]. The possibility to use *ROBOTIS-OP2* for Kinesthetic Teaching is implemented as follows. It is implemented that the ROS-Robotis Interface is able to receive messages from a ROS module (e.g. the ROS-RCS module) to enable Kinesthetic Teaching. On receiving this message certain motors, namely the motors corresponding to the arms, are switched off. The off switching of the motors is implemented within the Robotis Framework by sending a switch off message to a certain register of the corresponding actuator. Additional to the off switching it is implemented that the current motor positions are still send to the ROS-Robotis module even if the motors are switched off. In the default case, the Robotis Framework does communicate the true current value back because it publishes the desired joint angles back to ROS. This behaviour is modified in a way that the true current values are communicated back to ROS.

4.3.3 Installation of Pair of New Hands

ROBOTIS-OP2's original pair of hands is a set of stiff bars. The original set is shown in Figure 4.5. There are three degrees of freedom for each arm. The manipulation capability is strongly limited by by this number. Therefore, the original hands are replaced by a set of new hands manufactured by the company *Seed Robotics*. These hands have three fingers and eleven (under-actuated) degrees of freedom. Only four of the degrees of freedom are actuated: one for the orientation of the hand, one for the position of the thumb and two for the position of the index fingers. In order to use the new hands with the given framework the internal representation of the robot is changed for ROS, RCS and the Robotis Framework by adding the new joints to these submodules. These modifications are necessary that the desired joint values and the current values for these values can be propagated within the system. For the time being at which the research for this thesis is conducted the firmware of the actuators of the hands are not compatible to features which are used in the Robotis Framework. It is therefore not possible to fully integrate the hands with the framework. Section 5.1 describes a simple task which is taught to *ROBOTIS-OP2*, because of the limitations in the degrees of freedom it is not possible to demonstrate a more meaningful than this simple task.

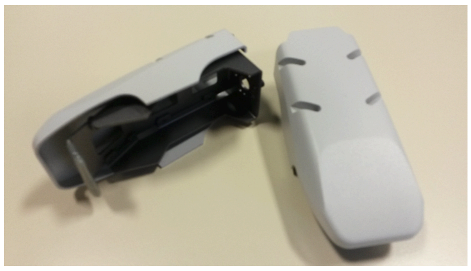


Figure 4.5: *ROBOTIS-OP2*'s original set of hands. This set of hands is replaced by a new one.

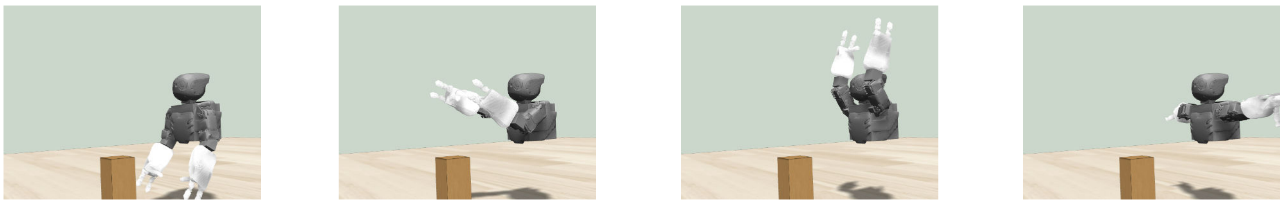
5 Evaluation of the Approach

In order to evaluate the Synchronisation Point Approach it is tested on two experiments. In the first experiment *ROBOTIS-OP2* is used. A simple toy task is taught through Kinesthetic Teaching. The task is successfully reproduced in simulation and on the real robot. In the second experiment a two-armed robot is used in simulation. A Finite State Machine is used to teach a box-picking task. Subsequently, the task is successfully reproduced in simulation. In order to compare the Synchronisation Point to other approaches two different approaches are tested on the experiments in simulation. For the first approach the implementation of the Synchronisation Point Approach is used without using the Synchronisation Points for training the classifiers. This approach is similar to the Concurrent Sequence Approach and relies on implicit synchronisation, therefore it is referred to as **Implicit Approach** in the following. The second approach which is used is presented by Manschitz et al. [34]. The structure consists of a higher level control graph and Concurrent Sequence Graphs on a lower level. The higher level graph controls the switching of the Concurrent Sequence Graph. Due to the hierarchical nature of this approach, it is referred to as **Hierarchical Approach** in the following.
For conducting the experiments the SeqLearn pipeline (described in Section 4.2) is used. For the Sequencer, Learner and Executor submodules, implementations of the three aforementioned approaches are used. This section describes the conducted experiments and presents the results.

5.1 Evaluation on *ROBOTIS-OP2*

In the first experiment *ROBOTIS-OP2* is used. First, the simple task is demonstrated by Kinesthetic Teaching. Subsequently, the captured data is processed by the SeqLearn pipeline. All three approaches are first tested and evaluated in simulation. Furthermore, after checking the results, the Synchronisation Point Approach is executed on the real *ROBOTIS-OP2*.

5.1.1 Demonstration of a Simple Task



(a) Goal positions of the Init MPs.
(b) Goal positions of the Front of Chest MPs.
(c) Goal positions of the Above Head MPs.
(d) Goal positions of the Sideways MPs.

Figure 5.1: Key positions of the demonstrated toy task for both hands. Each position corresponds to the goal position of one Movement Primitive.

The motivation behind the task can be imagined as picking up a ball from a table, moving it above the head and then removing one hand after the other from the ball. However, in the demonstrated task contact with an object is not taken into account. Only the position of the hands is considered. The task consists of four key positions for each hand. The key positions of one hand are symmetrical to the key positions of the other hand. Those four positions are the initial position beside the hip, a position in front of the chest, a position above the head and a position sideways to the body. These key positions are the goals of the learned Movement Primitives which are therefore referred to as **Init, Front of Chest, Above Head** and **Sideways**. The key positions are shown in Figure 5.1.
One goal of the experiment is to test the behaviour of the Synchronisation Point Approach on transitions which are required to switch synchronously. Additionally, the experiment is designed to test the behaviour on the Variable Transition Order case (cf. Section 3.2 for a more detailed explanation of this case). Therefore, the task is taught in two variations as described in the following. In the first set of demonstrations one hand, which can either be the left or the right hand, moves from the initial position to the front of the chest. The hand which started does not move to the next position until the other hand also reached the position in front of the chest. Next, the second hand moves to the position in front of the chest. Subsequently, both hands move synchronously to the position above the head. Next, the right hand

moves to the position beside the body. Finally, the left hand moves to the position beside the body. In the second set of demonstrations the other hand starts to move to the position in front of the chest and waits there for the hand which started the movements in the first set of demonstrations. The rest of the demonstration is done in the same way as in the first set of demonstrations. It is worth pointing out that the right hand starts with the movement from the position above the head to the position beside the body in both demonstration sets.

5.1.2 Identifying the Synchronisation Points

The results of the comparisons of the transition differences are shown in Table 5.1. The comparison of one Init Movement Primitive to the other Init Movement Primitive is meaningless, because these Movement Primitives are always started simultaneously when the reproduction is started. Therefore this comparison is ignored for the evaluation. The comparison pair Above Head - Above Head has the lowest mean and variance. Compared to the other mean values also the mean of the pair Front of Chest-Front of Chest is small, but the variance of this pair is in an order of two magnitudes higher than the variance of the Above Head-Above Head pair. Therefore only the Above Head-Above Head pair is selected as Synchronisation Point.
The comparison of one Above Head Movement Primitive to the other Above Head Movement Primitive has the lowest mean and variance (marked in green in the table). Compared to the other mean values also the comparisons of the Front of Chest Movement Primitives to each other has a small mean. The variance of this comparison is in an order of two magnitudes higher than the variance of the Above Head to Above Head comparison. Compared to the other comparison pairs the variance is in the same order. Therefore only the Above Head pair is selected as Synchronisation Point.

	Init	Front of Chest	Above Head	Sideways
Init	0	-3.486	-5.816	-8.775
Front of Chest	3.327	-0.159	-2.489	-5.447
Above Head	5.882	2.395	0.066	-2.8932
Sideways	9.188	5.702	3.372	0.413

(a) mean in s

	Init	Front of Chest	Above Head	Sideways
Init	0	2.823	1.164	1.344
Front of Chest	0.466	2.182	0.720	1.408
Above Head	1.213	0.541	0.003	0.588
Sideways	2.739	0.938	0.744	2.441

(b) variance in s^2

Table 5.1: Mean and variance of the comparisons of the transition pairs present in the demonstrated task. The comparison of the Above Head Movement Primitives has the lowest mean and variance.

5.1.3 Reproduction in Simulation

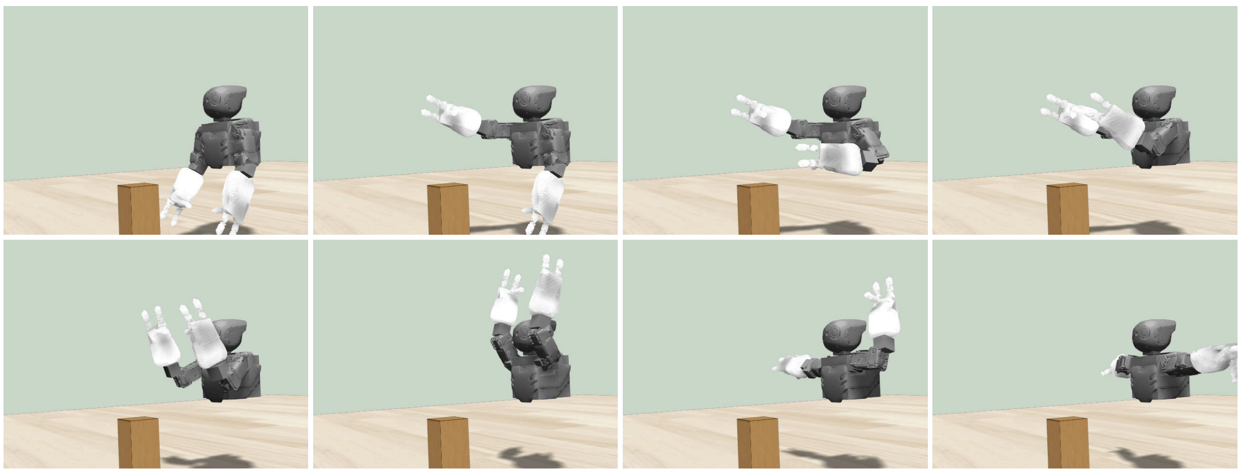
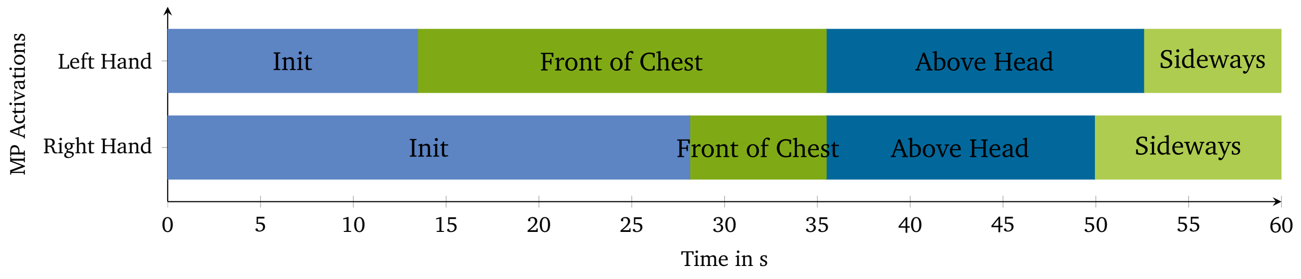
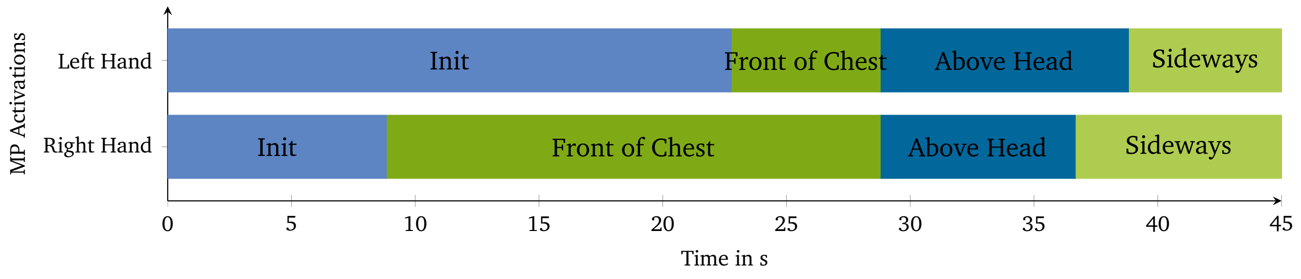


Figure 5.2: Successful reproduction of the demonstrated task in simulation using the Synchronisation Point Approach.

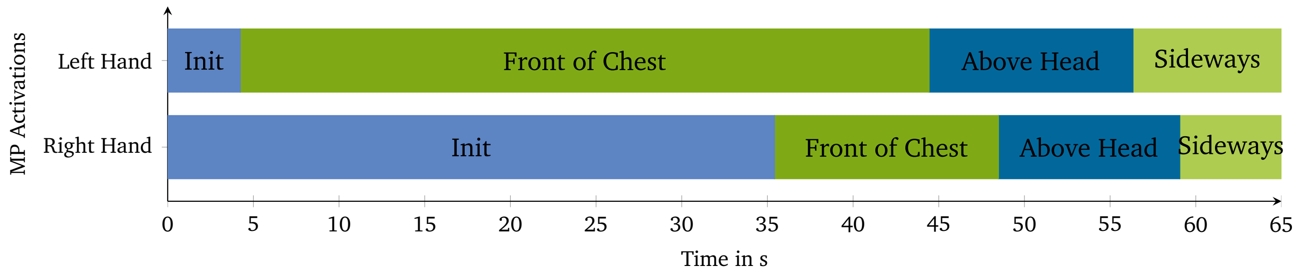
While using the Implicit Approach and the Synchronisation Point Approach the sequences of Movement Primitives can be started independent of each other. Therefore the reproduction is executed two times for both approaches. In one



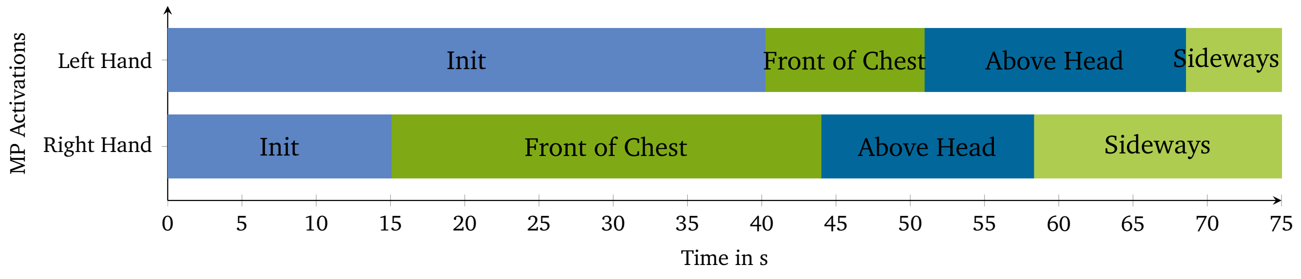
(a) MP Activations while using the Synchronisation Point Approach for reproduction in simulation. The left hand starts to execute the demonstrated movements.



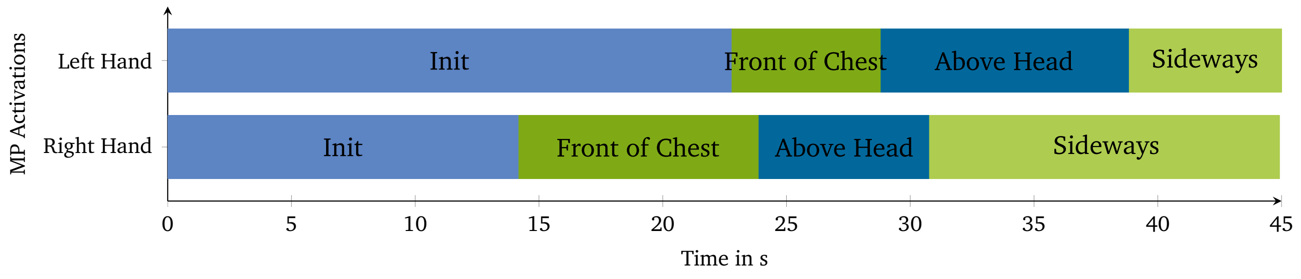
(b) MP Activations while using the Synchronisation Point Approach for reproduction in simulation. The right hand starts to execute the demonstrated movements.



(c) MP Activations while using the Implicit Approach for reproduction in simulation. The left hand starts to execute the demonstrated movements.



(d) MP Activations while using the Implicit Approach for reproduction in simulation. The right hand starts to execute the demonstrated movements.



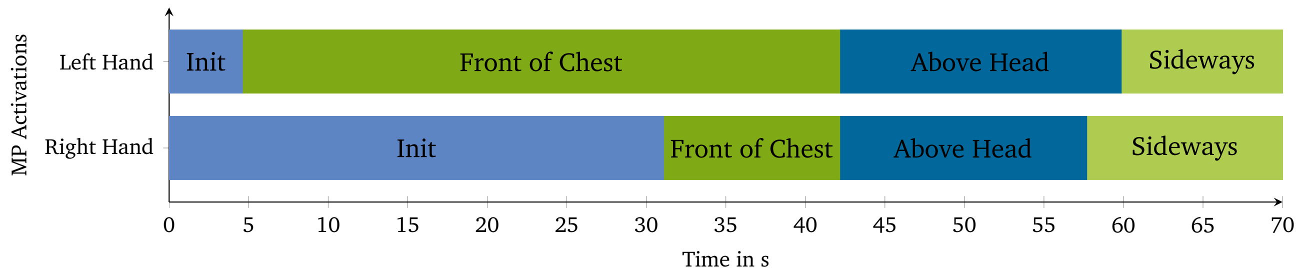
(e) MP Activations while using the Hierarchical Approach for reproduction in simulation.

Figure 5.3: Movement Primitive Activations while using different approaches to reproduce the demonstrated simple task in simulation.

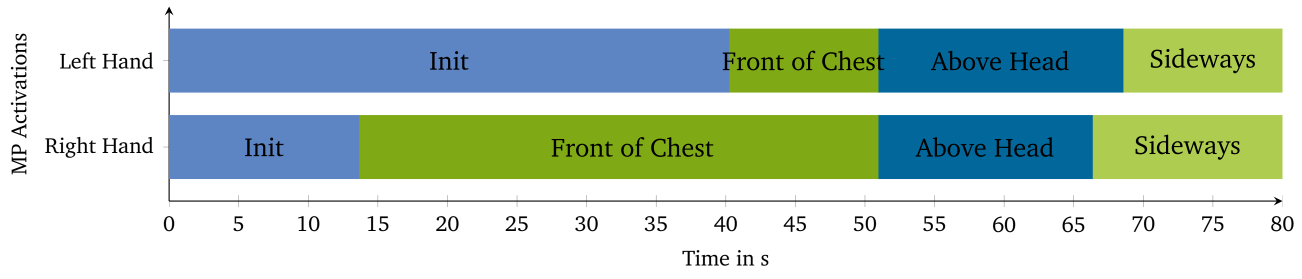
execution the sequence of the left hand is started first and in the other execution the sequence of the right hand is started first. While using the Hierarchical Approach it not possible to start the sequence independent of each other, since these are controlled by a higher level graph. Therefore the execution is done without variations.
Only the Synchronisation Point Approach is able to successfully reproduce the task. The successful reproduction while starting with the right hand is shown in Figure 5.2. The Movement Primitive Activations for all approaches are shown in Figure 5.3. As it can be seen in Figure 5.3a and Figure 5.3b the reproduced behaviour by using the Synchronisation Point Approach matches the demonstrated behaviour. None of the sequences switches to early and the switching to the Above Head Movement Primitives occurs synchronously. Furthermore the switching to the Sideways Movement Primitives occurs in the right order.
In the reproduction while using the Implicit Approach the sequence of the hand which is started first switches to early to the Above Head Movement Primitive. The activations for the Movement Primitives are shown in Figure 5.3c and Figure 5.3d. While using the Hierarchical Approach the right hand switches to early the Above Head.

5.1.4 Reproduction on the Real Robot

Since the Implicit Approach and the Hierarchical Approach are not able to reproduce the demonstrated behaviour in simulation they are not used for a reproduction on the real robot. The Synchronisation Point Approach is used to successfully reproduce the demonstrated task on the real robot. The Movement Primitive Activations for the reproduction on the real robot are shown in Figure 5.4. The reproduction while starting the sequence of the left hand first can be seen in Figure 5.5, the reproduction while starting the sequence of the right hand first is shown in Figure 5.6. In the Movement Primitive Activations of the reproduction it can be seen that the switching to the Above - Head Movement Primitives occurs synchronously.



(a) MP Activations while using the Synchronisation Point Approach for reproduction on the real robot. The left hand starts to execute the demonstrated movements.



(b) MP Activations while using the Synchronisation Point Approach for reproduction on the real robot. The right hand starts to execute the demonstrated movements.

Figure 5.4: Movement Primitive Activations while using different approaches to reproduce the demonstrated simple task on *ROBOTIS-OP2*.

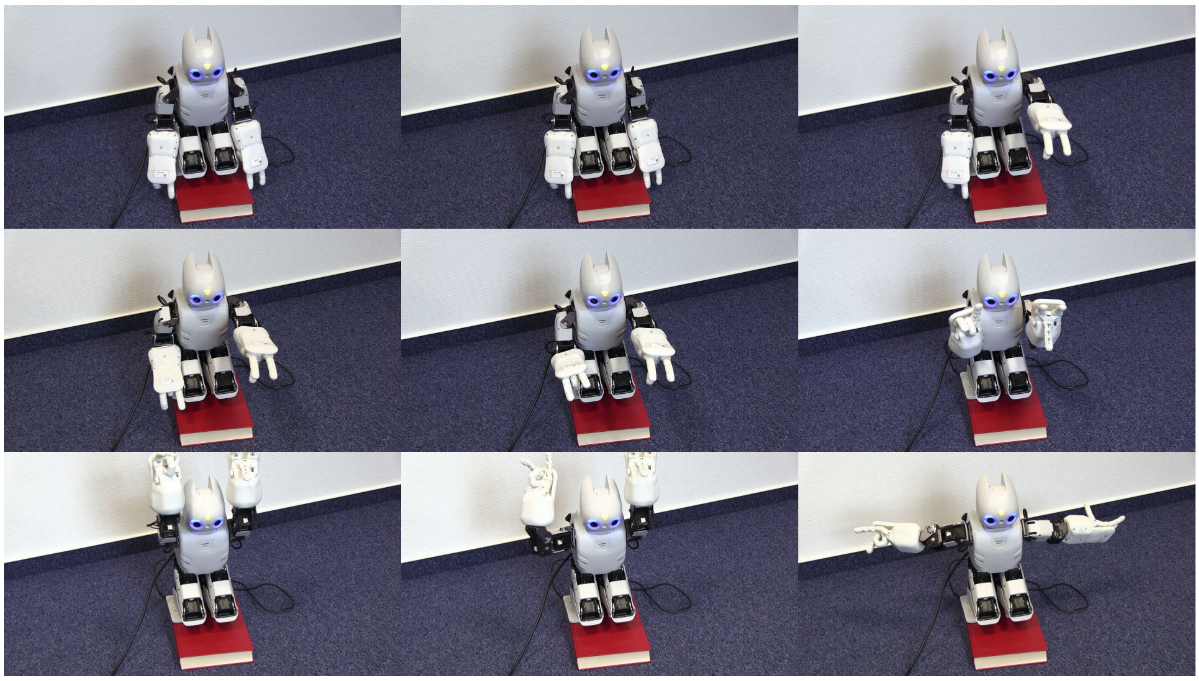


Figure 5.5: Reproduction of a demonstrated simple task on *ROBOTIS-OP2* using the Synchronisation Point Approach. The left hand starts to execute the demonstrated movements.

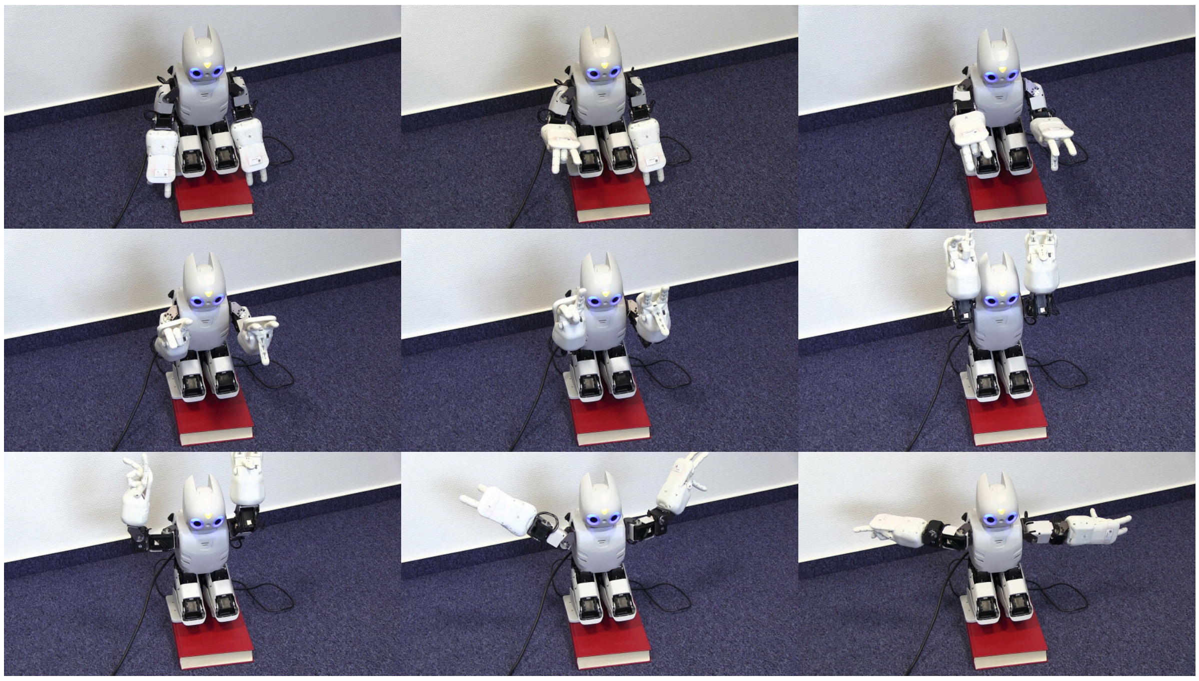


Figure 5.6: Reproduction of a demonstrated simple task on *ROBOTIS-OP2* using the Synchronisation Point Approach. The right hand starts to execute the demonstrated movements.

5.2 Evaluation On the Simulated Two-Armed Robot

In order to evaluate the approach on a more complex task, a second experiment using a two-arm setup is conducted. The demonstration and reproduction are only executed in simulation. First, the behaviour is taught using a Finite State Machine. Subsequently, the captured data is processed by the SeqLearn pipeline. All three approaches are tested and evaluated on the simulation.

5.2.1 Demonstration of a Box-Picking Task

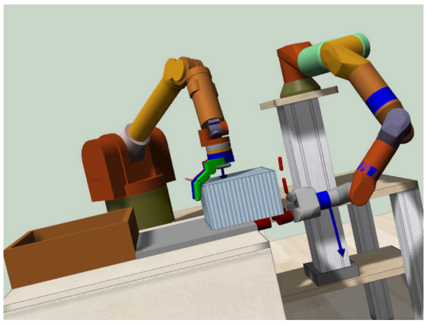


Figure 5.7: A two-armed robot consisting of a *WAM* (on the left) and a *SCHUNK* (on the right) solving the box-picking task.

The setup which is used is shown in Figure 5.7. The robot to the left side is a *WAM* robot, the robot to the right side is a *SCHUNK* robot. The goal of the demonstrated task is to pick up the box on top of the table and drop it into a drawer. In this task there are five key positions for the *WAM* and four key positions for the *SCHUNK*. These key positions are furthermore the goals of the Movement Primitives. In order to demonstrate this task a Finite State Machine is used. The Finite State Machine activates the Movement Primitives in the order as described in the following. The original position of the box is on top of a table. Beside the box there is a drawer. The task for the two-arm setup is to grab the box and let it fall into the drawer. First, the *SCHUNK* moves its hand beside the table (Init Movement Primitive), the *WAM* moves its hand to the box (Init Movement Primitive) and pushes the box onto the hand of the *SCHUNK* (Push Movement Primitive). As soon as the box is on the *SCHUNK* hand, the *WAM* and the *SCHUNK* simultaneously lift the box (Lift Movement Primitives). After reaching a certain position above the table, they simultaneously move to a position above the drawer to the left (Move Left Movement Primitives). After reaching this position the *WAM* moves upwards and the *SCHUNK* sideways (Let Go Movement Primitives). The result of these movements is that the box falls into the drawer. The transition pairs which need to occur synchronously are the pairs Lift - Lift, Move Left - Move Left and Let Go - Let Go. In order to ensure variance within the demonstrations a random time delay is introduced between the transitions of Movement Primitives which are not required to switch synchronously. The Movement Primitives which are required to switch synchronously are demonstrated by the Finite State Machine this way.

5.2.2 Identifying the Synchronisation Points

The results of the comparisons of the transition differences are shown in Table 5.2. As it can be seen in this table the pairs Lift - Lift, Move Left - Move Left and Let Go - Let Gohave the smallest mean, furthermore the variance of these pairs is small compared to the other pairs. Therefore these three pairs are selected as Synchronisation Points.

	Init	Lift	Move Left	Let Go
Init	0	-23.700	-35.570	-92.511
Push	13.300	-10.400	-22.270	-79.212
Lift	23.605	-0.095	-11.965	-68.906
Move Left	35.120	11.420	-0.450	-57.391
Let Go	92.829	69.129	57.259	0.318

(a) mean in s

	Init	Lift	Move Left	Let Go
Init	0	0.404	1.340	6.951
Push	0.109	0.129	1.011	6.522
Lift	0.619	0.121	0.726	7.135
Move Left	0.893	0.396	0.054	3.990
Let Go	6.035	4.611	2.594	0.253

(b) variance in s^2

Table 5.2: Mean and variance of the comparisons of the transition pairs present in the demonstrated box-picking task. The pairs Lift - Lift, Move Left - Move Left and Let Go - Let Go have the smallest mean and variance.

5.2.3 Reproduction in Simulation

The reproduction is repeated for every of the three approaches. Movement Primitive Activations for all three approaches are shown in Figure 5.10. Only the Synchronisation Point Approach is able to successfully reproduce the demonstrated behaviour. The successful reproduction using the Synchronisation Point Approach is shown in Figure 5.8. The corresponding Movement Primitive Activations can be seen in Figure 5.10a.

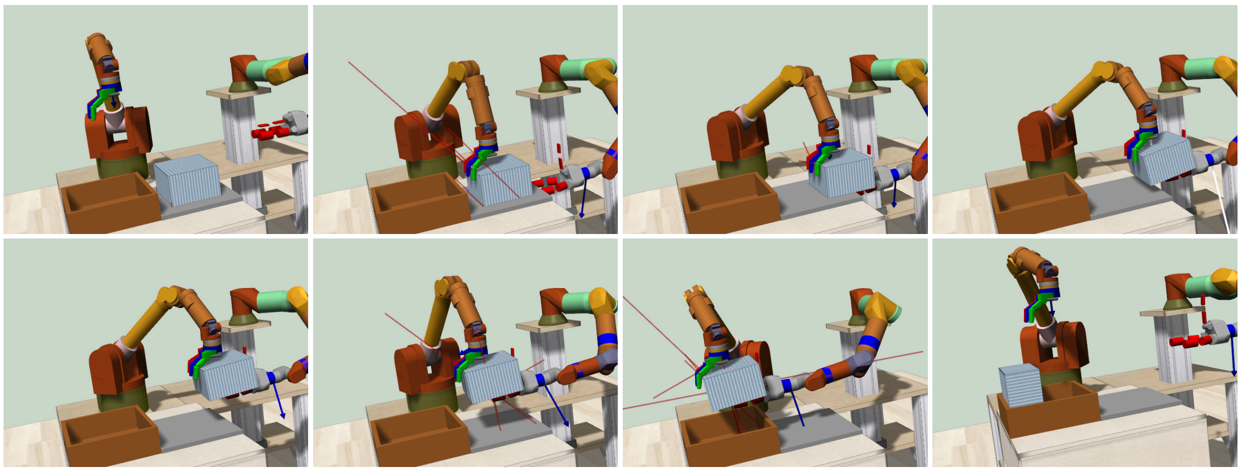


Figure 5.8: Successful reproduction of the demonstrated task while using the Synchronisation Point Approach. The the lifting of the box, the moving to the left and the letting occurs synchronously. The robot is able to successfully solve the task.

The reproduction of the task while using the Implicit Approach is shown in Figure 5.9. The corresponding Movement Primitive Activations are shown in Figure 5.10b. In the Movement Primitive Activations it can be seen that the *WAM* switches approximately one second before the *SCHUNK* to its Lift Movement Primitive. At this point both arms still lift the box together, but when the *SCHUNK* and the *WAM* reach the position above the table, the *SCHUNK* switches to the next Movement Primitive (Move Left) and the *WAM* does not. The *SCHUNK* pushes to the left, but the *WAM* does not move. Therefore the box drops to the ground and the reproduction fails. The sequence of the *WAM* only switches to the Lift when it is already to late.

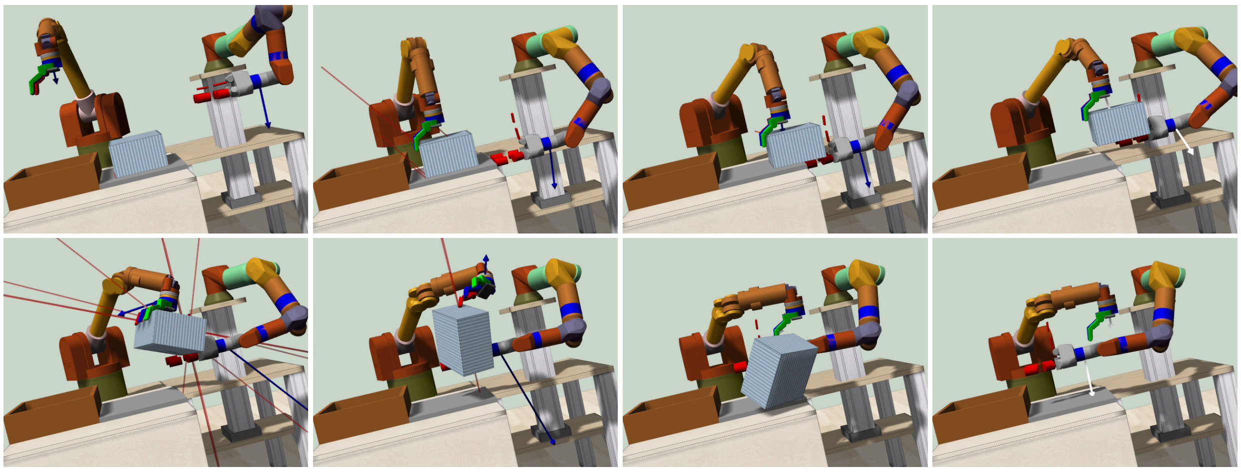
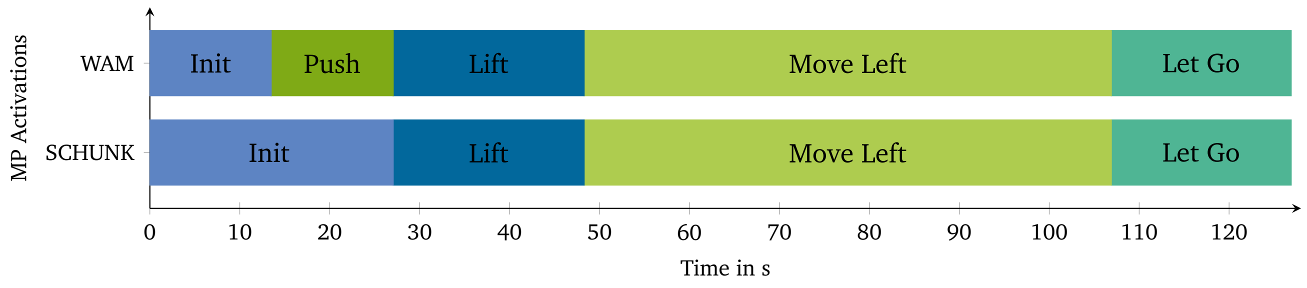
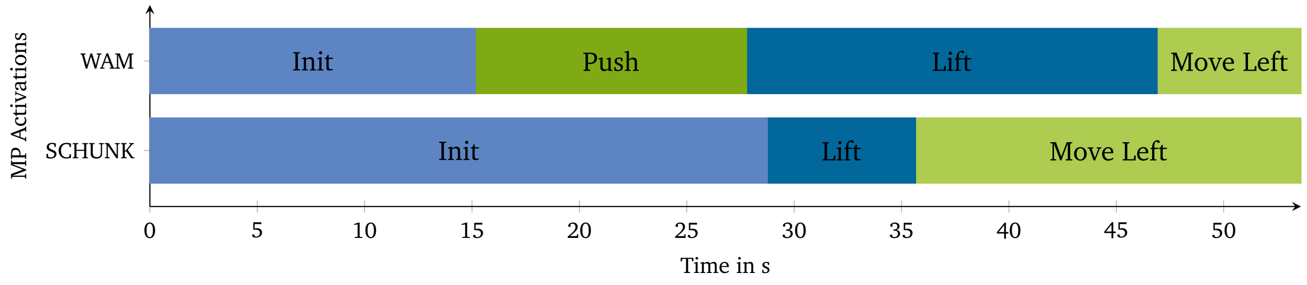


Figure 5.9: Failed attempt to reproduce the demonstrated task while using the Implicit Approach. The right arm starts to move to the left, but the left arm does not follow the movement. The box falls to the ground and the attempt to solve the task fails.

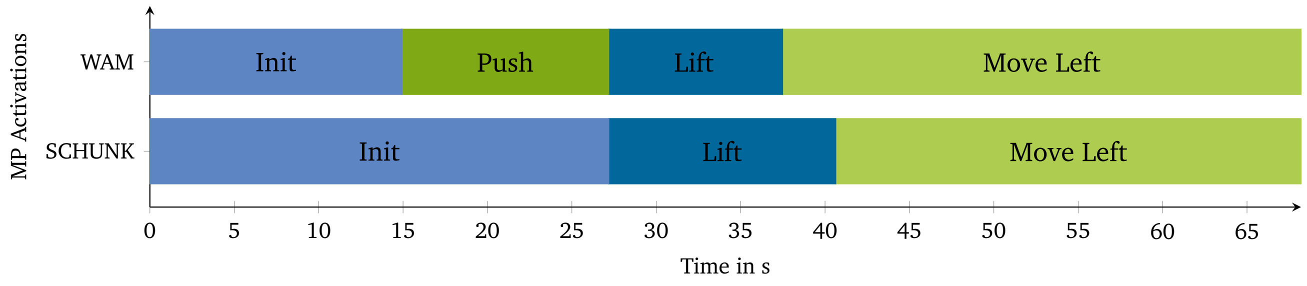
The reproduction of the task while using the Hierarchical Approach is shown in Figure 5.11. The Movement Primitive Activations are shown in Figure 5.10c. This time, the transitions to the Lift Movement Primitives occur at the same time.



(a) MP Activations while using the Synchronisation Point Approach for reproduction of the box-picking task.



(b) MP Activations while using the Implicit Approach for reproduction of the box-picking task.



(c) MP Activations while using the Hierarchical Approach for reproduction of the box-picking task.

Figure 5.10: Movement Primitive Activations while using different approaches to reproduce the demonstrated box-picking task.

The cause why the Hierarchical Approach fails is different compared to the Implicit Approach. When the *SCHUNK* and the *WAM* reach the position above the table, the *WAM* switches to the next Movement Primitive (Move Left), but the *SCHUNK* does not. The *WAM* moves to the left but the *SCHUNK* does not follow and the box falls on the table. The sequence of the *SCHUNK* only switches to the Lift Movement Primitive when the box lies already on the ground.

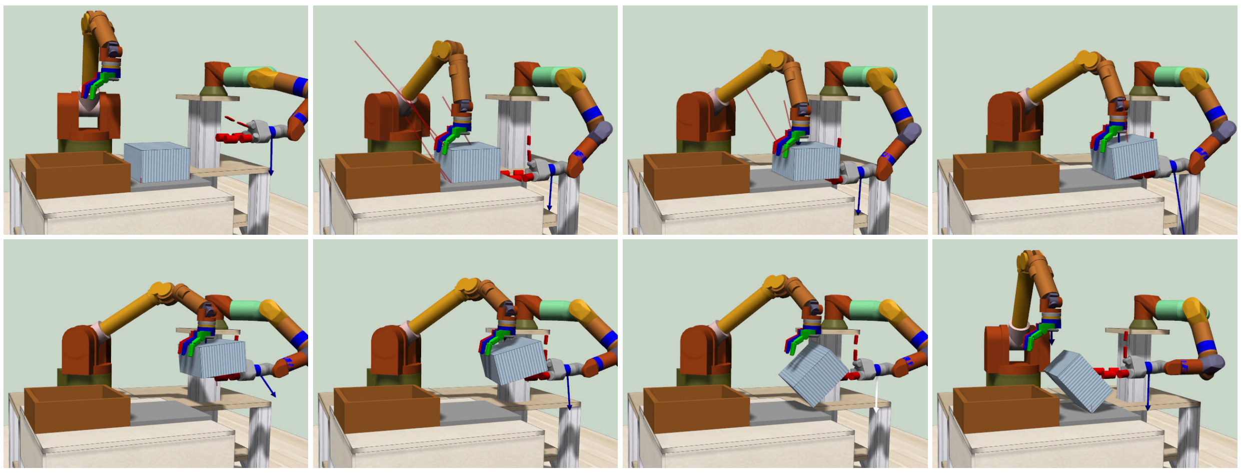


Figure 5.11: Failed attempt to reproduce the demonstrated task while using the Hierarchical Approach. The left arm starts to move to the left, but the right arm does not follow the movement. The box falls on the tables and the attempt to solve the task fails.

5.3 Discussion of the Experiments Results

The experiments which are presented in this chapter show that it is possible to learn bi-manual manipulation tasks by using the proposed Synchronisation Point Approach. Both experiments show that the Synchronisation Point Approach ensures synchronicity for cases where both other approaches fail. The first experiment shows that the Variable Transition Order cases (cf. Section 3.2) are excluded from the set of Synchronisation Points. Additionally, the first experiment shows that it is feasible to use the Synchronisation Point Approach on a real robot and not only on a simulation. The second experiment shows that Synchronisation Point Approach can furthermore successfully be used for a more meaningful task and not only on a simple task. The reasons of why the reproduction attempts fail are different for the Implicit Approach and the Hierarchical Approach. While using the Implicit Approach the reproduction fails because the sequence of the *SCHUNK* switches to the Lift Movement Primitive before the sequence of the *WAM*. For the Hierarchical Approach the reproduction fails, because the order of these transitions is switched. Since all three approaches are trained on the same input data, these two different error cases indicate that the Synchronisation Point Approach solves this problem in a general way and not only because the input data might be biased in a certain way.
The threshold which is used for the Synchronisation Point in the first experiment is with 66 milliseconds reasonably small. For the box-picking task the largest mean which is still accepted as Synchronisation Point is with 450 milliseconds only arguably small. The next larger mean is with approximately ten seconds by a factor of 20 larger than the mean for the largest Synchronisation Point. Therefore accepting it as a Synchronisation Point seems like a reasonable choice. The mean of approximately half a second must have been introduced by the SeqLearn pipeline, because in the demonstration the transitions occurred at the same time point. Nevertheless, this raises the question how to choose the the thresholds for the mean and variance to identify the Synchronisation Points in a general way.

6 Future Work

In Chapter 5 the theoretical considerations of Chapter 3 are brought to experimental validation. In order to focus on the main properties of the proposed approach simplifications in the demonstrated task are made. These simplifications do not result in a loss of generality of the proposed approach. In order to be applicable to a wide range of classes of tasks adaptions to the approach are required. Within this chapter the properties of the approach which can be further improved upon are shown. Furthermore are basic considerations on directions to go presented.

6.1 Evaluation on a Real Robot with a Complex Task

The experiments conducted in Chapter 5 show that the proposed approach can be used to learn skills to solve a simple bi-manual task. It is also shown that a more complex task can be solved in simulation. Both of these results combined indicate that it might be feasible to learn skills to solve a more complex bi-manual task and execute it on a real robot. The next step is to conduct such an experiment. This is not done within this thesis, because in the current state of the *ROBOTIS-OP2* it is not possible to demonstrate a more complex task. Furthermore, the two-arm setup used in simulation is not ready to be used in a real experiment.

6.2 Extraction of Statistics before Segmentation

The Synchronisation Point Approach is dependent on the quality of the input data coming from the previous phases of the learning process. For each flow of Movement Primitives the learning in the previous phases happens independent of the other flows. Coordination constraints are only after the segmentation and learning of the Movement Primitive goals taken into account. The sequencing as a later step in the pipeline suffers from errors or inaccuracy which are made in a preceding step. Information about the coordination constraints can get lost during the process. This information loss can be seen in the second experiment, where the difference between two transitions amounts 450 ms after the segmentation, even though it is demonstrated simultaneously. Therefore, it might be better to extract these constraints before or during the segmentation phase. Taking relative relationships between end-effectors or high-level features into account, as for example the distance between two end-effectors, as proposed in the work of Gribovskaya and Billard [20] might already improve the segmentation. Extracting coordination constraints before the segmentation would require alignment of the demonstration data. In order to align the data approaches like **Dynamic Time Warping** ([56]) or **Spatio-Temporal Feature Chain** ([14]) could be used.
Changing the segmentation process is out of the scope of this thesis. However, taking the information about the Synchronisation Points into account in an iterative process could improve the segmentation. In this iterative process the segmentation could be redone after the Synchronisation Points are extracted. The Synchronisation Points are subsequently taken into account for a second segmentation phase.

6.3 Incorporation of Cyclic Behaviour and Sequence Variations

In the demonstrations of the tasks presented in Chapter 3, the order of Movement Primitives within one flow stayed the same across all demonstrations. There is no cyclic behaviour demonstrated. Furthermore, there are no transitions which only occur in some demonstrations, but do not occur in the other demonstrations. However, without including cyclic behaviour a broad class of tasks is excluded. Solving a task like unscrewing a light bulb as presented by Manschitz et al. [36] are meaningful and it is desirable to include them for bi-manual manipulation. The difficulty is to find a general approach to compare transitions if they occur multiple times within one demonstration or occur only in some demonstrations. It needs to be addressed how a transition is processed if it does not occur in all demonstrations, but occurs synchronously to another transition for all demonstrations where it does appear.
Manschitz et al. [37] presented three different approaches how to build graphs with different degrees of information included in them. These approaches represent cyclic behaviour differently. It needs to be further evaluated how to include the information about Synchronisation Points into different graph structures.

6.4 Considerations on Constraints on the Sequential Order

The Synchronisation Point Approach only takes the case where simultaneous switching is required into account. Cases where the activation of one Movement Primitive has to occur before the other are not taken into account. An example for such a case is a task where first a drawer has to be opened with one hand and subsequently something has to be put into it with the other hand. This behaviour can be modelled with the concepts introduced with the approach. For a Movement Primitive which may only be activated after another Movement Primitive reached its goal. The first Movement Primitive will be called **Constrained Movement Primitive**, the second one will be called **Condition Movement Primitive**. In order to model this behaviour, a dummy node for the Condition Movement Primitive could be introduced into the graph. The dummy node does not activate a Movement Primitive. The dummy node could then be used to synchronise the transition of the Constrained Movement Primitive with the transition of the dummy node to the succeeding Movement Primitive. The problem with this approach is, that it results the flow of the Condition Movement Primitive to wait for the flow of the Constrained Movement Primitive, which is a undesired behaviour. This again could be solved by introducing additional flow sequences which are used for synchronisation, but resulting in a more complex graph structure. The feasibility of this approach for tasks with a high number of Movement Primitive flows needs to be further investigated. Another problem is how to extract the constraints on sequential order, one possibility is to create a constraint for every transition which always occurs before another transition. The downside of this approach is that it will create a lot of unnecessary constraints and complicate the graph structure.

6.5 Adding More Sophisticated Feature Selection

There are two simple options to chose the set of features which are used for training the classifiers. The first option is to train the classifiers belonging to a sequence only on the features which are directly controlled by this sequence. The second simple option is the global feature set. A third simple option is possible by the use of the Synchronisation Points. In this third possibility only the classifiers which are associated with a Synchronisation Point are trained on the global feature set, all other classifiers are trained on the features which are controlled by the sequence the classifiers belong to. Another possibility is to take the variance for each feature in the transition points into account. Features which are relevant for the transitions are likely to have a smaller variance than features which are not relevant for the decision to trigger to the next Movement Primitive. Using the variance to decide whether a feature is used to train a classifier could probably increase the performance of the classifiers. However, only relying on the variance will probably not work and requires further investigation how this information can be used.

6.6 Scalability: Chaining of Multiple Synchronisation Points

The approach allows not only two, but multiple controlled variables. The question in regards to scalability is how can sets which consist of more than two Movement Primitive flows be synchronised. For most cases it should be possible, if there is a pair of transitions which intersects with another pair of transitions to train the union of all transitions on the same input data. However, approaching the problem like this might fail in some cases. For these cases a more sophisticated approach is required which provides a metric on how to compare sets of transitions with more than two elements.

6.7 Integrating Additional Logical Operators

Extracting the Synchronisation Points is equivalent to searching for transitions which are connected to each other via a logical **AND**. For more sophisticated tasks other logical operators for example **XOR** or **NAND** are possible. The work of Zou and Roos [66] proposes an approach to learn logical combinations of features based on Sparse Logistic Regression. Whether it is feasible to integrate this logical combination learning with the approach presented in this thesis needs further investigation.

7 Conclusion

Approaches like the Learning from Demonstration approach are important, because they offer an intuitive and easy way to teach robots how to solve a broad variety of tasks. An important class of tasks are bi-manual manipulation tasks. Learning the required skills to master bi-manual manipulation tasks adds the problem of coordination to the learning problem.

This thesis focuses on the temporal constraints of the coordination problem. The two important questions when-to-sychronise and how-to-synchronise are addressed and the problems, which go along with these questions within the context of multiple flows of independent Movement Primitives, are identified. In order to explicitly solve the identified problems an approach which is called Synchronisation Point Approach is proposed in Chapter 3. The proposed approach extends the approach of Concurrent Sequence Graphs, which deal with the aforementioned questions in an implicit manner. In order to explicitly synchronise Concurrent Sequence Graphs the concept of Synchronisation Points is presented. The Synchronisation Points are extracted by considering the mean and the variance of transition comparisons. Subsequently, the Synchronisation Points are used to train classifers which are associated with the transitions. In the Synchronisation Points the same training data is used to train the classifiers ensuring that they switch simultaneously.

The implementation and integration of the Synchronisation Point Approach is presented in Chapter 4. Additionally, the necessary changes in hardware and software to evaluate the approach on a *ROBOTIS-OP2* robot are explained. In order to evaluate the presented approach two experiments are designed and conducted. The results of the conducted experiments validate the presented theoretical results. A simple task is demonstrated and reproduced on *ROBOTIS-OP2* proving that the presented approach can successfully be used on a real robot. The box-picking task, which is evaluated in simulation, demonstrates that it is possible to solve meaningful tasks. The comparisons with two other approaches indicate that the proposed approach solves the synchronisation problem in a general way and does not only favour a certain bias in the input data.

Applying the proposed concepts not only to extract temporal constraints, but also to spatial and force constraints is promising. Integrating cyclic behaviour and considering coordination constraints already in the segmentation phase offer potential future work as presented in Chapter 6.

To conclude, it can be said that, this thesis contributes an important improvement to the teaching of bi-manual tasks through human demonstration by explicitly extracting and integrating information about synchronisation.

References

[1] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*.

[2] Asfour, T., Azad, P., Gyarfas, F., and Dillmann, R. (2008). Imitation learning of dual-arm manipulation tasks in humanoid robots. *International Journal of Humanoid Robotics*.

[3] Ben Amor, H., Neumann, G., Kamthe, S., Kroemer, O., and Peters, J. (2014). Interaction primitives for human-robot cooperation tasks. *International Conference on Robotics and Automation*.

[4] Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Survey: Robot programming by demonstration. Handbook of Robotics, Chapter 59.

[5] Bizzi, E., D'Avella, A., Saltiel, P., and Tresch, M. (2002). Modular organization of spinal motor systems. *Neuroscientist.*

[6] Brock, O. (2011). Berlin Summit on Robotics: Conference Report.

[7] Butterfield, J. (2010). Learning from Demonstration using a Multi-valued Function Regressor for Time-series Data. *International Conference on Humanoid Robots*.

[8] Calinon, S. and Billard, A. (2007). Active teaching in robot programming by demonstration. *International Workshop on Robot and Human Interactive Communication*.

[9] Calinon, S., Guenter, F., and Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*.

[10] Chang, G. and Kulic, D. (2013). Robot task learning from demonstration using Petri nets. *International Symposium on Robot and Human Interactive Communication*.

[11] Chiappa, S., Kober, J., and Peters, J. (2009). Using Bayesian Dynamical Systems for Motion Template Libraries. *Advances in Neural Information Processing Systems*.

[12] Daniel, C., Neumann, G., Kroemer, O., and Peters, J. (2013). Learning sequential motor tasks. *International Conference on Robotics and Automation*.

[13] Dautenhahn, K. and Nehaniv, C. L. (2002). *Imitation in Animals and Artifacts*. MIT Press, Cambridge, MA, USA.

[14] Ding, W., Liu, K., Cheng, F., and Zhang, J. (2015). STFC: Spatio-temporal feature chain for skeleton-based human action recognition. *Journal of Visual Communication and Image Representation*.

[15] Erhart, S., Sieber, D., and Hirche, S. (2013). An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation. *IEEE International Conference on Intelligent Robots and Systems*.

[16] Ewerton, M., Neumann, G., Lioutikov, R., Amor, H. B., Peters, J., and Maeda, G. (2015). Learning multiple collaborative tasks with a mixture of Interaction Primitives. *International Conference on Robotics and Automation*.

[17] Flanagan, J. R., Bowman, M. C., and Johansson, R. S. (2006). Control strategies in object manipulation tasks. *Current Opinion in Neurobiology*.

[18] Flash, T. and Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*.

[19] Gams, A., Nemec, B., Ijspeert, A. J., and Ude, A. (2014). Coupling movement primitives: Interaction with the environment and bimanual tasks. *Transactions on Robotics*.

[20] Gribovskaya, E. and Billard, A. (2008). Combining Dynamical Systems control and programming by demonstration for teaching discrete bimanual coordination tasks to a humanoid robot. *Human-Robot Interaction*.

[21] Ha, I., Tamura, Y., and Asama, H. (2011a). Gait Pattern Generation and Stabilization for Humanoid Robot Based on Coupled Oscillators. *International Conference on Intelligent Robots and Systems*.

[22] Ha, I., Tamura, Y., Asama, H., Han, J., and Hong, D. W. (2011b). Development of open humanoid platform DARwIn-OP. *SICE Annual Conference*.

[23] Ijspeert, A., Nakanishi, J., and Schaal, S. (2002a). Learning Attractor Landscapes for Learning Motor Primitives. *Advances in Neural Information Processing Systems*.

[24] Ijspeert, A., Nakanishi, J., and Schaal, S. (2002b). Movement imitation with nonlinear dynamical systems in humanoid robots. *International Conference on Robotics and Automation*.

[25] Kelso, J. a., Southard, D. L., and Goodman, D. (1979). On the coordination of two-handed movements. *Journal of experimental psychology. Human perception and performance*.

[26] Kober, J., Mohler, B., and Peters, J. (2008). Learning perceptual coupling for motor primitives. *International Conference on Intelligent Robots and Systems*.

[27] Kulic, D., Ott, C., Lee, D., Ishikawa, J., and Nakamura, Y. (2012). Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*.

[28] Li, X., Li, Y., and Cui, X. (2017). Kinematic analysis and gait planning for a DARwIn-OP Humanoid Robot. *International Conference on Robotics and Biomimetics*.

[29] Lioutikov, R., Kroemer, O., Maeda, G., and Peters, J. (2016). Learning manipulation by sequencing motor primitives with a Two-Armed robot. *Advances in Intelligent Systems and Computing*.

[30] Lioutikov, R., Neumann, G., Maeda, G., and Peters, J. (2015). Probabilistic segmentation applied to an assembly task. *International Conference on Humanoid Robots*.

[31] Luksch, T., Gienger, M., Muhlig, M., and Yoshiike, T. (2012). Adaptive movement sequences and predictive decisions based on hierarchical dynamical systems. *International Conference on Intelligent Robots and Systems*.

[32] Maeda, G., Ewerton, M., Lioutikov, R., Ben Amor, H., Peters, J., and Neumann, G. (2015). Learning interaction for collaborative tasks with probabilistic movement primitives. *International Conference on Humanoid Robots*.

[33] Manschitz, S., Gienger, M., Kober, J., and Peters, J. (2016). Probabilistic Decomposition of Sequential Force Interaction Tasks into Movement Primitives. *International Conference on Intelligent Robots and Systems*.

[34] Manschitz, S., Gienger, M., Kober, J., and Peters, J. (2017). Learning sequential force interaction skills [submitted]. *Robotics and Autonomous Systems*.

[35] Manschitz, S., Kober, J., Gienger, M., and Peters, J. (2014a). Learning to sequence movement primitives from demonstrations. *International Conference on Intelligent Robots and Systems*.

[36] Manschitz, S., Kober, J., Gienger, M., and Peters, J. (2014b). Learning to Unscrew a Light Bulb from Demonstrations. *International Symposium on Robotics Research*.

[37] Manschitz, S., Kober, J., Gienger, M., and Peters, J. (2015a). Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. *Robotics and Autonomous Systems*.

[38] Manschitz, S., Kober, J., Gienger, M., and Peters, J. (2015b). Probabilistic progress prediction and sequencing of concurrent movement primitives. *International Conference on Intelligent Robots and Systems*.

[39] Muelling, K., Kober, J., and Peters, J. (2010). Learning table tennis with a mixture of motor primitives. *International Conference on Humanoid Robots*.

[40] Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., and Kawato, M. (2004). Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*.

[41] Nehaniv, C. L. and Dautenhahn, K. (1999). Of hummingbirds and helicopters: an algebraic framework for interdisciplinary studies of imitation and its applications. *Interdisciplinary Approaches to Robot Learning*.

[42] Niekum, S. and Chitta, S. (2013). Incremental Semantically Grounded Learning from Demonstration. *Robotics: Science and Systems IX*.

[43] Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., and Barto, A. G. (2015). Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*.

[44] Pais Ureche, A.-L. and Billard, A. (2015). Learning bimanual coordinated tasks from human demonstrations. *Human-Robot Interaction*.

[45] Paraschos, A., Daniel, C., Peters, J., and Neumann, G. (2013). Probabilistic Movement Primitives. *Neural Information Processing Systems*.

[46] Pastor, P., Kalakrishnan, M., and Chitta, S. (2011). Skill Learning and Task Outcome Prediction for Manipulation. *International Conference on Robotics and Automation*.

[47] Pastor, P., Kalakrishnan, M., Righetti, L., and Schaal, S. (2012). Towards associative skill memories. *International Conference on Humanoid Robots*.

[48] Peters, J. (2007). *Machine Learning of Motor Skills for Robotics*. PhD thesis, University of Southern California, Los Angeles, CA, USA.

[49] Pierris, G. and Dahl, T. S. (2010). Compressed Sparse Code Hierarchical SOM on learning and reproducing gestures in humanoid robots. *International Workshop on Robot and Human Interactive Communication*.

[50] Riano, L. and McGinnity, T. M. (2012). Automatically composing and parameterizing skills by evolving Finite State Automata. *Robotics and Autonomous Systems*.

[51] Rudy, J. (2014). ZMP walking controller for humanoid walking using darwin-op. *Degree Thesis*.

[52] Schaal, S. (1999). Is Imitation Learnig the Route to Humanoid Robots? *Trends in Cognitive Sciences*.

[53] Schaal, S., Kotosaka, S., and Sternad, D. (2001). Nonlinear Dynamical Systems as Movement Primitives. *International Conference on Humanoid Robotics Cambridge MA*.

[54] Schaal, S., Mohajerian, P., and Ijspeert, A. (2007). Dynamics systems vs. optimal control - a unifying view. *Progress in Brain Research*.

[55] Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*.

[56] Senin, P. (2008). Dynamic Time Warping Algorithm Review. *Science*.

[57] Silvério, J., Rozo, L., Calinon, S., and Caldwell, D. G. (2015). Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. *International Conference on Intelligent Robots and Systems*.

[58] Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D. V., and Kragic, D. (2012). Dual arm manipulation - A survey. *Robotics and Autonomous Systems*.

[59] Steffen, J., Elbrechter, C., Haschke, R., and Ritter, H. (2010). Bio-inspired motion strategies for a bimanual manipulation task. *International Conference on Humanoid Robots*.

[60] Surdilovic, D., Yakut, Y., Nguyen, T. M., Pham, X. B., Vick, A., and Martin Martin, R. (2010). Compliance control with dual-arm humanoid robots: Design, planning and programming. *International Conference on Humanoid Robots*.

[61] Takano, W. and Nakamura, Y. (2006). Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. *International Conference on Humanoid Robots*.

[62] Umlauft, J., Sieber, D., and Hirche, S. (2014). Dynamic Movement Primitives for cooperative manipulation and synchronized motions. *International Conference on Robotics and Automation*.

[63] Ureche, A. L. P. and Billard, A. (2014). Encoding bi-manual coordination patterns from human demonstrations. *International Conference on Human-Robot Interaction*.

[64] Whitehead, S. D. and Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*.

[65] Zollner, R., Asfour, T., and Dillmann, R. (2004). Programming by demonstration: dual-arm manipulation tasks for humanoid robots. *International Conference on Intelligent Robots and Systems*.

[66] Zou, Y. and Roos, T. (2016). Sparse logistic regression with logical features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.