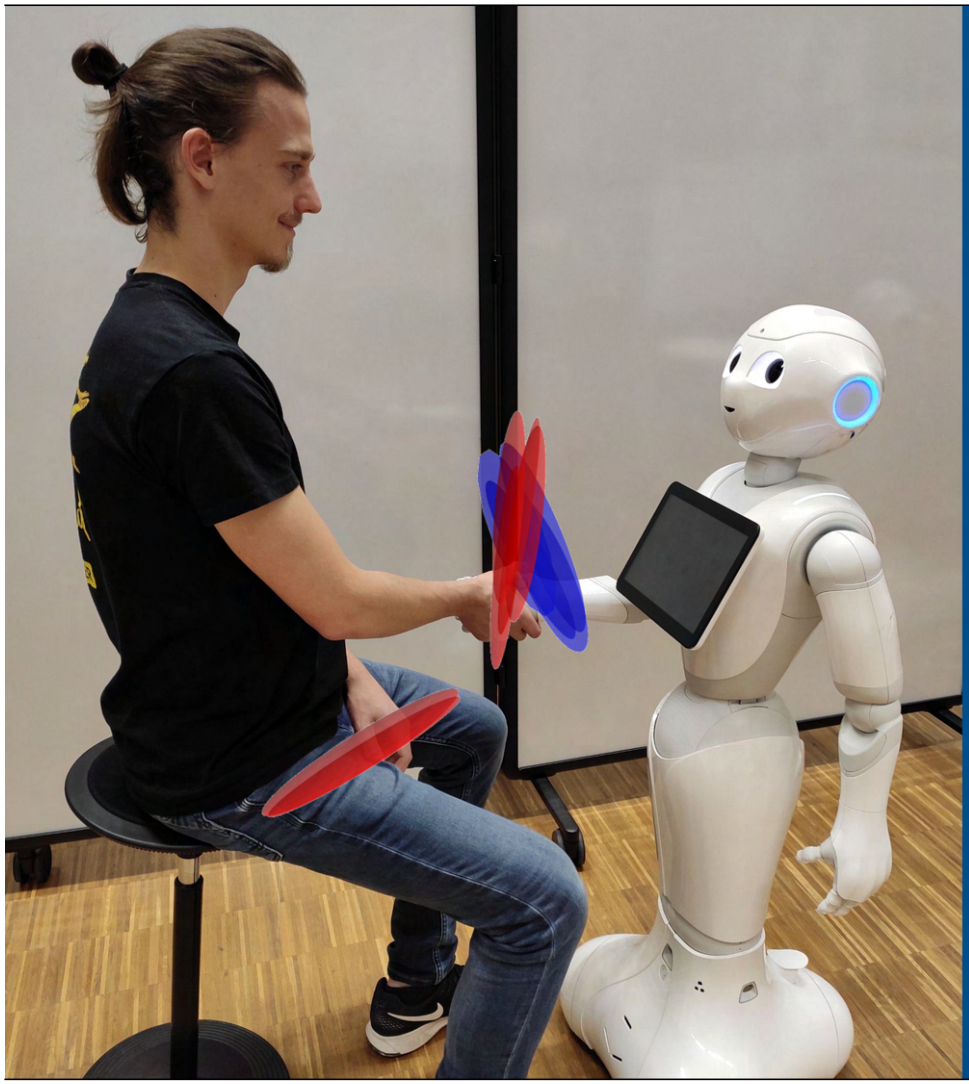
Social Interaction Segmentation and Learning using Hidden semi-Markov Models

Soziale Interaktionen Segmentieren und Lernen mittels Hidden semi-Markov Modellen
Bachelor thesis by Louis Sterker
Date of submission: October 16, 2022

1. Review: Vignesh Prasad
2. Review: Prof. Dr. Jan Peters
3. Review: Prof. Dr. Dr. Ruth Stock-Homburg
Darmstadt



TECHNISCHE UNIVERSITÄT DARMSTADT

IAS

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

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

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 16. Oktober 2022

L. Sterker

Abstract

Robots, especially humanoid robots, will become more and more present in our everyday life. A crucial factor for them to be accepted by society is their overall social behavior, which mainly includes the ability to properly interact with humans in collaborative Human-Robot Interactions (HRI) like handshaking or hand waving. To raise the social acceptance even more, such interactions should appear as natural and human-like as possible.
Learning from Demonstration (LfD) methods from Human-Human Interactions (HHI) are effective in learning interaction dynamics for generating robot trajectories in a responsive and timely manner, which is of significant importance for the interaction to appear natural. To enhance a better generalization, these interactions can naturally be broken down into multiple more primitive segments that can be learned individually. A great way to implement such an approach is through Hidden semi-Markov Models (HSMMs), which are capable of performing such kind of segmentation in an unsupervised and modular manner.
In the proposed framework we are using HSMMs for jointly learning the interaction dynamics between a human and a robot as multiple sequential segments. We represent the observations of each segment as a Multivariate Normal Distribution (MVN) to sufficiently encode the joint probabilities of the human-human demonstrations. The performance of our framework is tested in simulation as well as in real scenarios. For the latter, we conduct a user study showing the importance of an algorithm being able to properly react and adapt to human users to be well perceived.

Zusammenfassung

Roboter, insbesondere humanoide Roboter, werden in unserem Alltag immer präsenter. Ein entscheidender Faktor für ihre Akzeptanz in der Gesellschaft ist ihr allgemeines soziales Verhalten, welches vor allem die Fähigkeit umfasst mit Menschen in kollaborativen Mensch-Roboter Interaktionen, wie Händeschütteln oder Winken, zu interagieren. Um die soziale Akzeptanz noch weiter zu erhöhen, sollten solche Interaktionen so natürlich und menschenähnlich wie möglich erscheinen.
Learning from Demonstration Methoden von Mensch-Mensch Interaktionen sind effektiv beim Erlernen von Interaktionsdynamiken zur Generierung von Robotertrajektorien in einer reaktionsschnellen und zeitnahen Art und Weise, was von großer Bedeutung ist, damit die Interaktion natürlich erscheint. Um eine bessere Generalisierung zu erreichen, können diese Interaktionen auf natürliche Art und Weise in mehrere primitivere Segmente unterteilt werden, welche einzeln erlernt werden können. Ein hervorragender Weg zur Umsetzung eines solchen Ansatzes sind verdeckte semi-Makrov Modelle (HSMMs), die in der Lage sind, eine solche Segmentierung auf unüberwachte und modulare Weise vorzunehmen.
In dem vorgestellten Framework verwenden wir HSMMs für das gemeinsame Lernen von Interaktionsdynamiken zwischen einem Menschen und einem Roboter als mehrere aufeinander folgende Segmente. Wir stellen die Beobachtungen jedes Segments als eine mehrdimensionale Normalverteilung (MVN) dar, um die gemeinsamen Wahrscheinlichkeiten der Mensch-Mensch Demonstrationen hinreichend zu kodieren. Die Performanz unseres Frameworks wird sowohl in einer Simulation als auch in realen Szenarien getestet. Für letzteres führen wir eine Nutzerstudie durch, die zeigt, wie wichtig es ist, dass ein Algorithmus in der Lage ist, richtig auf den menschlichen Nutzer zu reagieren und sich entsprechend anpassen, um positiv wahrgenommen zu werden.

Contents

1. Introduction

The following introduces the idea of this thesis and in Section 1.1, this Chapter motivates the concepts used for implementation. Our main contributions are listed out in Section 1.2. Finally, in Section 1.3, the remainder of this thesis is outlined.

Social interactions in the context of Human-Human Interaction (HHI) refers to all kinds of gestures used to carry out social conventions which, among other things, includes greeting someone with a handshake or celebrating something by giving a high five. Such kinds of social rituals are of great importance in our society [1]. They help with establishing trust among others and can incorporate the belonging to a group. Thus, when it comes to the interactions between humans and robots, non-verbal communication is a key feature for a (social) robot to be accepted by society. So in order for robots to get integrated into our everyday life, they should be capable of knowing and performing social interactions.
Another crucial fact in raising the social affiliation for robots is the way the robot moves when executing the gestures. It needs to be as natural and human-like as possible to not appear uncanny to the interaction partners [2].

Having a robot learn social interactions is not an easy task. Every human is individual and therefore will perform interactions slightly different [3]. Even the same human will not always perform an interaction in the same manner. Furthermore, the robot needs to adapt to different body heights or shapes, which also results in interactions being performed slightly differently. In addition, social rituals vary between different cultures [4]. Hence all that can not be pre-programmed, it is of great importance for the robot to be able to adapt and generalize to these different situations on its own.
All that should be employable in realistic scenarios by just observing the interaction partner and not having any additional knowledge about the context. Within real-time, the robot needs to classify the observation and respond to it appropriately.

These preliminaries bring us to the Human-Robot Interaction (HRI) approach proposed in this thesis. We rely on an imitation learning method based on HHIs captured by motion

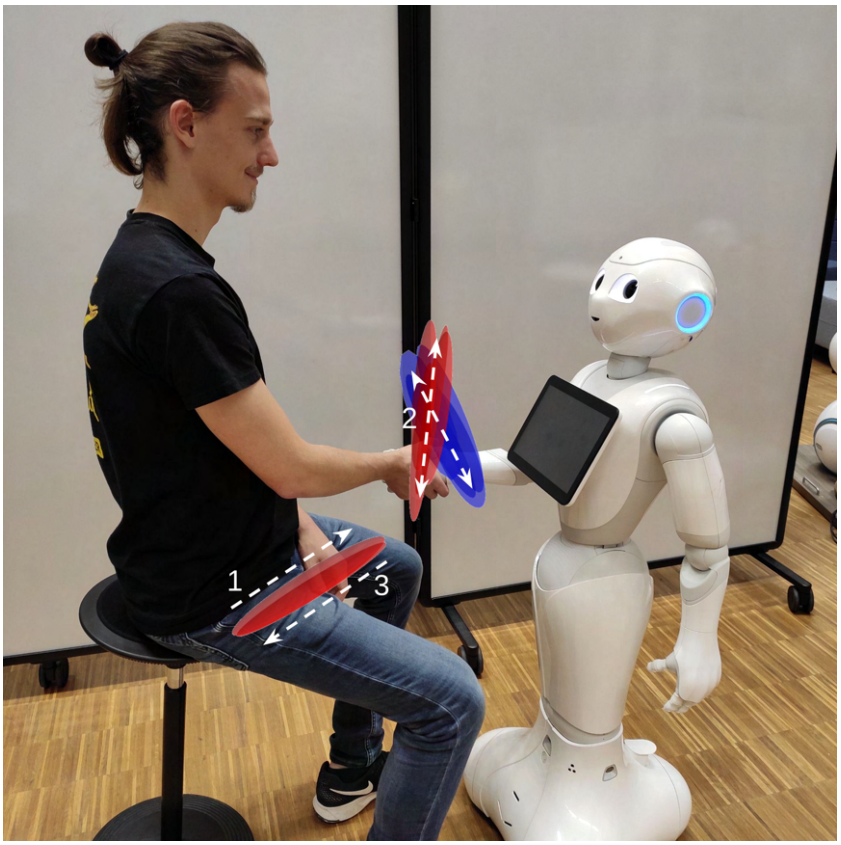


Figure 1.1.: This figure gives a general overview of the approach proposed in this thesis. It shows the joint distributions over the segments of the learnt interaction of both the human (red) and the robot (blue), which are represented as Gaussian's. The arrows along with their numbering symbolize the sequential order of the segments.

tracking systems. We learn the joint probabilities of both human actors with Hidden semi-Markov Models (HSMMs) consisting of Multivariate Normal Distributions (MVN) as the underlying state distributions for encoding the observation probabilities. While learning, the demonstrated trajectories of an action first get segmented into more primitive motions like the reaching of the hands during a handshake or fist bump. Each segment then corresponds to one of the states of the HSMM, which further learns the temporal sequencing of those. While reproduction, the learned model gets conditioned on the

observed trajectory and emits the corresponding observations. An overview of the proposed framework can be seen in Figure 1.1, which shows an exemplary interaction between a human and a robot with the corresponding learned joint distributions encoded as MVNs, along with their sequential order.

1.1. Motivation

In Imitation Learning - also referred to as Learning from Demonstration (LfD) [5, 6, 7, 8] - the goal of the agent is to extract information from a (human) teacher and learn a mapping between the situation and the demonstrated behavior. In contrast to direct programming, the agent does not deterministically learn to perform a task or a skill. With that, the robot learns ways to circumvent unknown scenarios on its own and therefore does not need to be explicitly pre-programmed to every possible scenario.
Moreover, the robot also learns a skill in the way the teacher demonstrates it, including the way the single body parts move, which prevents the robot from finding his "own" - for example more efficient - solution. That results in the skill looking natural and human-like when being reproduced and accordingly encounters the aforementioned problem of robots quickly causing a feeling of uncanniness to humans [2].
Another advantage of Imitation Learning is the fact that it is accessible to non-expert users. Once an algorithm is provided, the teacher simply needs to perform the task and does not need to come up with a method to express the situation in a way that the agent understands it.

HSMMs [9, 10] are a great way for learning such data. They allow for segmenting the demonstrated trajectories into smaller more primitive motions, like the reaching phase of a handshake or fist bump, which brings a few beneficial features along. Firstly, it makes it easy to generalize between the points where the primitives connect, amplifying the ability to adapt to unseen movements. Secondly, the single segments can be reused within the same action (e.g. for repeated movements like the "waving part" in a hand wave), but also along actions that have related sub-actions, like the reaching phase of a handshake or fist bump.
The HSMMs themselves learn to properly sequence those primitives. In contrast to regular HMMs [11, 12], they bring the advantage of not only modeling spacial but also temporal dependencies. The MVNs as the underlying distributions allow for modeling the observations in a probabilistic way.

1.2. Contributions

The general idea of utilizing HSMMs in Human-Robot Interaction (HRI) scenarios has already been explored quite a bit [13, 14, 15]. Most of these approaches however vary from our one by relying on different underlying architectures. Also, a lot of these approaches are not evaluated in real scenarios, and even if tested on real robots, they mostly rely on kinesthetic teaching for training the model, which is tedious and only works on the robot that the model is trained on.
The approach proposed in this thesis has the power of learning a sufficient model just from Human-Human Interaction (HHI) data, which can easily be recorded. Moreover, once a model is learned on such data, the framework can even be applied to all different kinds of humanoid robots.

1.3. Thesis Structure

The remainder of this thesis is structured as follows. Chapter 2 provides detailed insights into HMMs and HSMMs and also discusses related works. The actual approach is described in Chapter 3. In Chapter 4, we are evaluating the proposed framework by first giving insights into the datasets we used. Further, the performance of the approach is evaluated in a more theoretical and general way, after which testing in simulation and real scenarios is discussed. The latter also includes the conducted user study along with its results. Finally, in Chapter 5, we are first concluding the results achieved in this thesis and secondly, we are discussing the shortcomings and giving an outlook for future work.
Moreover, Appendix A provides further details into the theory of Hidden Markov Models (HMMs) and Hidden semi-Markov Models (HSMMs). Appendix B is discussing the differences between HMMs and HSMMs more thoroughly in terms of their performance regarding our use case.

2. Foundations and Related Work

The following chapter provides a broad overview of the technologies and basic concepts we use for our approach. Section 2.1 introduces Hidden Markov Models (HMMs) [11, 12] and Hidden semi-Markov Models (HSMMs) [9, 10], which are fundamental to the approach proposed in this thesis. Section 2.2 discusses related works that inspired and motivated this thesis.

2.1. Hidden (semi-)Markov Models

Hidden semi-Markov Models (HSMMs) [9, 10] are a special class of Hidden Markov Models (HMMs) [11, 12], where the probability of changing a (hidden) state does not only depend on the previous state, but also on the duration for which the systems stays in that state. Thus, in contrast to HMMs, this duration can be variable in HSMMs. The duration is associated with the number of observations produced while in the state.
A HMM/HSMM is a common mathematical stochastic model that developed a broad field of applications not only in computer science but also in natural sciences, physiology, economy, and so on [16, 9]. The most popular usage however lies within pattern recognition such as speech or optical characters, but also gesture recognition for Human-Robot Interaction (HRI) which is the topic of this thesis.

In general, a HMM is a stochastic process based on a Markov-Chain with unobservable states - hence the term *hidden*. It is defined by a 5-tuple $\lambda = (\mathcal{S}, \mathcal{O}, \mathcal{D}, \boldsymbol{A}, \boldsymbol{\pi})$, where

$\mathcal{S} = \{s_1, s_2, ..., s_n\}$	is the set of all possible states
$\mathcal{O} \subset \mathbb{R}^m$	is the set of observations (sometimes also referred to as emissions)
$\mathcal{D} = \{d_1, d_2, ..., d_n\}$	is the set of the underlying distributions, which characterize the emission probabilities of each state. Each state s_i is associated with a distribution d_i with $i \in \{1, 2, ..., n\}$
$\boldsymbol{A} \in [0, 1]^{n \times n}$	defines the probabilities of the state transitions
$\boldsymbol{\pi} \in [0, 1]^n$	is the initial state distribution

Some additional definitions are:

1. A sequence of states is defined as $s_{1:T} \triangleq s_1, s_2, ..., s_T$ where $s_t \in \mathcal{S}$ describes the state at time step t with $t \in \{1, 2, ..., T\}$.
2. A sequence of observations is defined as $\boldsymbol{o}_{1:T} \triangleq \boldsymbol{o}_1, \boldsymbol{o}_2, ..., \boldsymbol{o}_T$ where $\boldsymbol{o}_t \in \mathcal{O}$ describes the observation at time step t with $t \in \{1, 2, ..., T\}$.
3. The probability $p(s_t|s_{t+1})$ to transition from a state s_i at time step t to a state s_j at time step $t + 1$ is denoted as $a_{t,t+1}$ or equivalently $a_{i,j}$ with $a_{t,t+1}, a_{i,j} \in \boldsymbol{A}$.
4. HSMMs have the additional feature of having a variable duration of staying in a state. For that a variable $d \in \{1, 2, ..., D\}$ that specifies this duration, is introduced. $p_s(d)$ defines a distribution that is fitted over the state $s \in \mathcal{S}$, which describes the duration of staying in a certain state s.

With that, in a HMM, a sequence of observations $\boldsymbol{o}_{1:T}$ is modeled as a sequence of hidden latent states $s_{1:T}$ that emit observations $\mathcal{O}$ with probabilities according to the underlying distributions $\mathcal{D}$. For the first time step, the hidden latent states are distributed according to the initial probability distribution $\boldsymbol{\pi}$. The following transitions between the states emerge from the matrix $\boldsymbol{A}$.
In the case of this thesis the underlying distribution d_i of a state s_i is a Multivariate Normal Distribution (MVN) with mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ with $i \in 1, 2, ..., n$. It characterizes the emission probabilities of the observations $\mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. That, in essence, can be seen as learning a Gaussian Mixture Model (GMM) over the observations and learning the temporal sequencing between the Gaussian components. More on that in Chapter 3.

HMMs are based on Markov chains with the states being unobservable. Thus it fulfills the (first order) Markov property, which states that the probability of getting into an arbitrary

state s_{t+1} only depends on the current state s_t, but not on previous ones.
As already mentioned, HSMMs bring the addition of each state having a duration distribution $p_s(d)$ that defines the time the model stays in that particular state s. Along with that, the length of time spent in a state also determines the number of observations produced in that state.

With those preliminaries set, there are three standard problems to be solved for HMMs/ HSMMs to make use of them in real-world applications:

1. **The Evaluation Problem**
   The evaluation problem asks for the probability of observing a (partial) observation sequence $\boldsymbol{o}_{1:t}$, given a model λ, i.e. $p(\boldsymbol{o}_{1:t}|\lambda)$. In other words, how likely is it that the observation sequence $\boldsymbol{o}_{1:t}$ is produced by a given model λ.
   This problem is solved by the Forward-Backward algorithm specified in Appendix A.1.

2. **The Decoding Problem**
   The decoding problem asks for the most probable sequence of states $s_{1:t'}$ emitting an observation sequence $\boldsymbol{o}_{1:t}$, given a model λ, i.e. $p(s_{1:t'}, \boldsymbol{o}_{1:t}|\lambda)$. In other words, giving an observation sequence $\boldsymbol{o}_{1:t}$ and a model λ, which sequence of states $s_{1:t'}$ best explains the given observation sequence $\boldsymbol{o}_{1:t}$.
   This problem is solved by the Viterbi algorithm specified in Appendix A.2.

3. **The Learning Problem**
   The learning problem asks for a procedure that adjusts the parameters of a given model λ to maximize the probability of observing a given observation sequence (or a set of observation sequences) $\boldsymbol{o}_{1:t}$ with that model, i.e. maximizing $p(\boldsymbol{o}_{1:t}|\lambda)$. In other words, given an observation sequence (or a set of observation sequences) $\boldsymbol{o}_{1:t}$, how to learn and optimize the model probabilities of a given model λ that would generate them.
   This problem is solved by the Baum-Welch algorithm specified in Appendix A.3.

A solution to each of the three problems stated above can be found in Appendix A. The evaluation problem can be efficiently solved by the Forward-Backward algorithm (Appendix A.1). The Viterbi algorithm (Appendix A.2) solves the decoding problem. To solve the learning problem, the Baum-Welch algorithm (Appendix A.3) is used.

2.2. Related Work

In the following some related works are discussed. An overview of the usage of Hidden Markov Models (HMMs) and Hidden semi-Markov Models (HSMMs) in Human-Robot Interaction (HRI) scenarios is provided in Section 2.2.1. Section 2.2.2 gives some insights into imitation learning and the methods it brings along. In Section 2.2.3, the idea and concepts of segmenting trajectories are discussed.

2.2.1. HMMs in HRI

HMMs in their basic form and variations of it, like the HSMMs, are very well studied. They can be used not only for HRI but in all different kinds of applications. A great overview of that is provided by Mor et. al [16] regarding HMMs and by Shun-Zheng Yu [9] regarding HSMMs. Hence the topic of this thesis is a HRI setup, we are primarily going to focus on that, which excludes many approaches that consider scenarios with only one actor like Koppula and Saxena did in [17]. Those cases are not modeling any temporal correlations, which is a crucial factor when dealing with two (or more) actors that need to interact with each other.

The goal of this thesis is to investigate the use of HSMMs in such an application. In the following, we are going to discuss the use of HMMs and HSMMs in general and in HRI scenarios with a focus on interaction learning.
The contribution of Calinon et al. [18] is pretty similar to the work in this thesis. In this paper, however, they only introduce the use of normal HMMs in HRI scenarios. An example of them using HSMMs in a HRI setup is discussed in paper [13], where a robot is taught to assist in dressing. Asfour et al. [14] use HMMs to effectively detect a set of characteristic points that can be used to represent and reconstruct the trajectory. They are as well considering two actors and the temporal dependencies between them. Nevertheless, they are again only relying on HMMs instead of HSMMs, and their results have not been tested on real robots.

Regarding the use of HSMMs, Oshikawa et al. [15] proposed an approach to interaction segmentation and learning. The difference to our approach is the use of a Gaussian Process (GP) as the underlying distribution of the HSMM. Also, they look at the interactions in a more general fashion and do not do any skill learning or reproduction from a robotic standpoint. Very similar work is provided by Nakamura et al. [19], who was also a contributor to the paper cited previously ([15]). It shows very good results regarding the unsupervised

trajectory segmentation, but again it uses a GP as the underlying distribution of the HSMM and has not been specifically applied to robots. Another paper relying on HSMMs has been proposed by Oliveira et al. [20], addressing the unsupervised segmentation of heart sounds. Although having nothing to do with robotics at all, heart sounds are continuous time series data just like trajectories from interactions. The approach in this work is quite interesting as they rely on MVNs as the emission distribution of the HSMM, just like we are doing in our work.

2.2.2. Imitation Learning

Imitation learning, also referred to as *Learning from Demonstration (LfD)* [5, 6] or *Programming by Demonstration (PbD)* [7], is a learning method mainly used in robotics that became more and more important over the last years. Hence it is the core thought on what this thesis and the proposed approach is based on, in this subsection, we want to provide an overview of its idea and the concepts behind it.
The basic idea of imitation learning arises from the natural way human beings and a lot of animal species learn tasks and skills [7]. They study movements by observing others performing the task or the skill and then trying to reproduce it as best as possible. Generally speaking, Imitation learning aims to mimic the behaviors of expert teachers performing a certain task or skill. Due to this analogy of the way human beings naturally learn skills, the most common fields of application lie within robotics which, above all, includes Human-Robot Interaction (HRI), but also all other kinds of robot interaction or assistance up to autonomous vehicle driving or flying [8].
In imitation learning, the goal of the agent (i.e. the learning machine) is to extract information from the teacher and the surroundings (e.g. interaction with objects), and learn a mapping between the situation and the demonstrated behavior [5, 6, 7, 8]. To gain the needed information many different methods exist.
Hence we want to learn the interactions from just observing the human actors, methods like teleoperation [21, 22, 23] or shadowing [24] are not relevant to us. We are rather interested in works involving motion tracking cameras and/or motion tracking suits. While using a combination of motion tracking cameras and suits is more accurate, using just cameras is more desirable since it is impractical to wear a suit, especially when it comes to the robot learning new skills on the fly at some point [5].
An approach based on RGBD videos is proposed by Prasad et al. [25], where they are learning the motions of a handshake via a Long Short Term Memory (LSTM) and a Probabilistic Movement Primitive (ProMP) distribution. Other approaches relying on RGBD videos are proposed by Shu et al. [26, 27], where they are learning multiple social

interactions between humans and robots. Vogt et al. [28] use motion capture data to learn an interaction model consisting of Interaction Meshes (IMs) to represent position pairs.

2.2.3. (Unsupervised) Trajectory Segmentation

Before explaining what it means to segment continuous time-series data (e.g. a trajectory), we want to discuss why it is even usefully to segment trajectories in the first place and also why we are doing it in an unsupervised manner.
Human beings recognize perceived continuous information, like speech or movements, by dividing it into smaller segments called primitives, like single words or unit motions. That is not only how humans learn, for example, new words or motions, but also how they speak or move in general. They concatenate those primitives to formulate sentences or to perform more complex movements, which is the reason why we believe that such kind of segmentation is a skill robots should also be capable of. In the following, we will only rely on the segmentation of movements, particularly interactions (like in [29]), since that is the topic of this thesis.
Figure 2.1 visualizes how a possible segmentation for a trajectory could look like. It also shows the feature of single primitives being reused at multiple different points. This feature, of course, does not only hold within the same trajectory but also for independent ones.

Onto the different approaches to segment such continuous time-series data like the ones we are dealing with in this thesis (e.g. [29]).
The most simple way for training a model would be to already have the segments in the training data properly labeled, like Varadarajan et al. did in [30]. This supervised approach however implies a lot of prior work to manually label all the segments. Nevertheless, for a surgical application, like the one proposed in [30], that might be worth the effort, as it is a very critical application for which the segmentation points are clearly clinically defined. In most cases though, including the one we tackle in this thesis, an unsupervised approach is desirable, which again is also more representative for the way human beings perform segmentation. It is happening more or less unconsciously, and first and foremost it is happening without the help of any particular segmentation points. Nevertheless, training a model in a way that it comes up with useful segmentation is not an easy task. There have been a few different approaches developed over time.

Two works that highly inspired the idea of this thesis in terms of segmenting and learning social interactions in HRI scenarios are contributed by Shu et al. [26, 27]. They are using

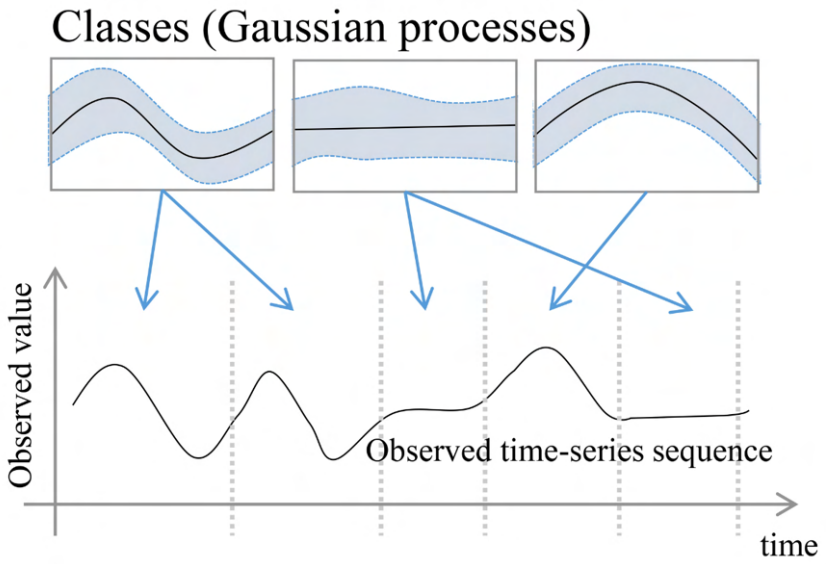


Figure 2.1.: This depiction[1] illustrates a possible segmentation on a dummy trajectory. It is to see that the whole trajectory is spitted into five different segments. Nevertheless only three different primitives are actually learned, as some of them occur multiple times within the trajectory.

RGBD videos to learn multiple social interactions between humans and robots. However, the way of representing and modeling the demonstration data differs from our approach. They use a Markov Chain Monte Carlo (MCMC) based algorithm to learn sub-events and sub-goals and their relations in the manner of a hierarchical graph.
The approach tackled in this thesis, however, is mostly based on the work by Pignat and Calinon [13] and works by first creating a Hidden Markov Model (HMM), more specifically a Hidden semi-Markov Model (HSMM), with K number of states and Multivariate Normal Distributions (MVNs) as the underlying distributions, which will represent the single segments. The model then gets initialized with the given training data by splitting each demonstration trajectory into K bins along the time, where each of the K bins corresponds to one of the K states of the HSMM. With that, each state represents one segment. Finally, the model gets optimized through an Expectation-Maximization (EM) algorithm.
Nakamura et al. [19] did something similar by using Gaussian Processes (GP) instead of MVNs as the emission distributions of the HSMM. Also, instead of an EM algorithm, they

[1] Figure taken from [19]

are using a Blocked Gibbs Sampler optimized by forward filtering-backward sampling to estimate the model parameters. However, they only consider skills performed by a single actor and do not apply their approach to social interactions between two (or more) actors. Other methods like in [31] or [32] are based on motion based heuristics like velocity peaks or zero-velocity crossing (ZVC) to segment the demonstrations. Heuristics though, are quite task dependent and therefore the quality of the results can vary a lot. For example, a model based on a heuristic that relies on contact points (like in a handshake) would not necessarily work in an interaction not involving any contact (like a hand wave).
Some approaches, however, apply heuristics to get a first abrasive segmentation that will then be fine tuned in further steps. Lioutikov et al. [33] proposed a probabilistic approach that segments demonstrations while learning a library containing probabilistic representations of the primitives. They start with an empty library and a segmentation based on heuristics that can then be further optimized.
Transition State Clustering (TSC) [34] is a two-step hierarchical segmentation algorithm. In the first step, they generate a sliding window over all demonstrations and then fit a Gaussian Mixture Model (GMM) to all the windows, assigning each state (transition state) to its most likely component. In a second step, they refine those states by clustering the transition states in terms of kinematic, sensory, and temporal features. Although achieving great results, their work only considers segmenting the trajectories and reproducing them.

3. Segmenting and Learning Interactions

This chapter provides theoretical insights into the approach proposed in this thesis. First, in Section 3.1, the problem of this work is stated. In Section 3.2, we formalize the framework used to overcome the stated problems.

3.1. Problem Statement

The goal of this work is to learn social interactions from Human-Human demonstration data. With that, a robot observing a human agent performing an interaction should be able to first, detect the intended type of interaction and second, respond to it appropriately.

More detailed, for each type of action c_k from a set $\mathcal{C} = \{c_1, c_2, ..., c_K\}$ of K possible actions, a separate Hidden semi-Markov Model (HSMM) λ_k from a set $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_K\}$ is trained in terms of the spatial and temporal dependencies between both actors. Hence, an action classifier is needed to decide which model to use for reproduction.
The robot's task is to observe a partial trajectory $\boldsymbol{o}_{1:t}$ of a human agent performing a social interaction c_k. With these observations, the robot should first classify the current interaction c_t and secondly, come up with a corresponding joint angle $\boldsymbol{q}_t(c_t)$ for response.

A graphical overview of the whole framework is depicted in Figure 3.1

3.2. Proposed Approach

The following three sections describe the overall structure of our proposed approach as well as the single components it consists of. Our framework can mainly be split into three key procedures.
The first one (Section 3.2.1) deals with the initialization and the training of the HSMMs,

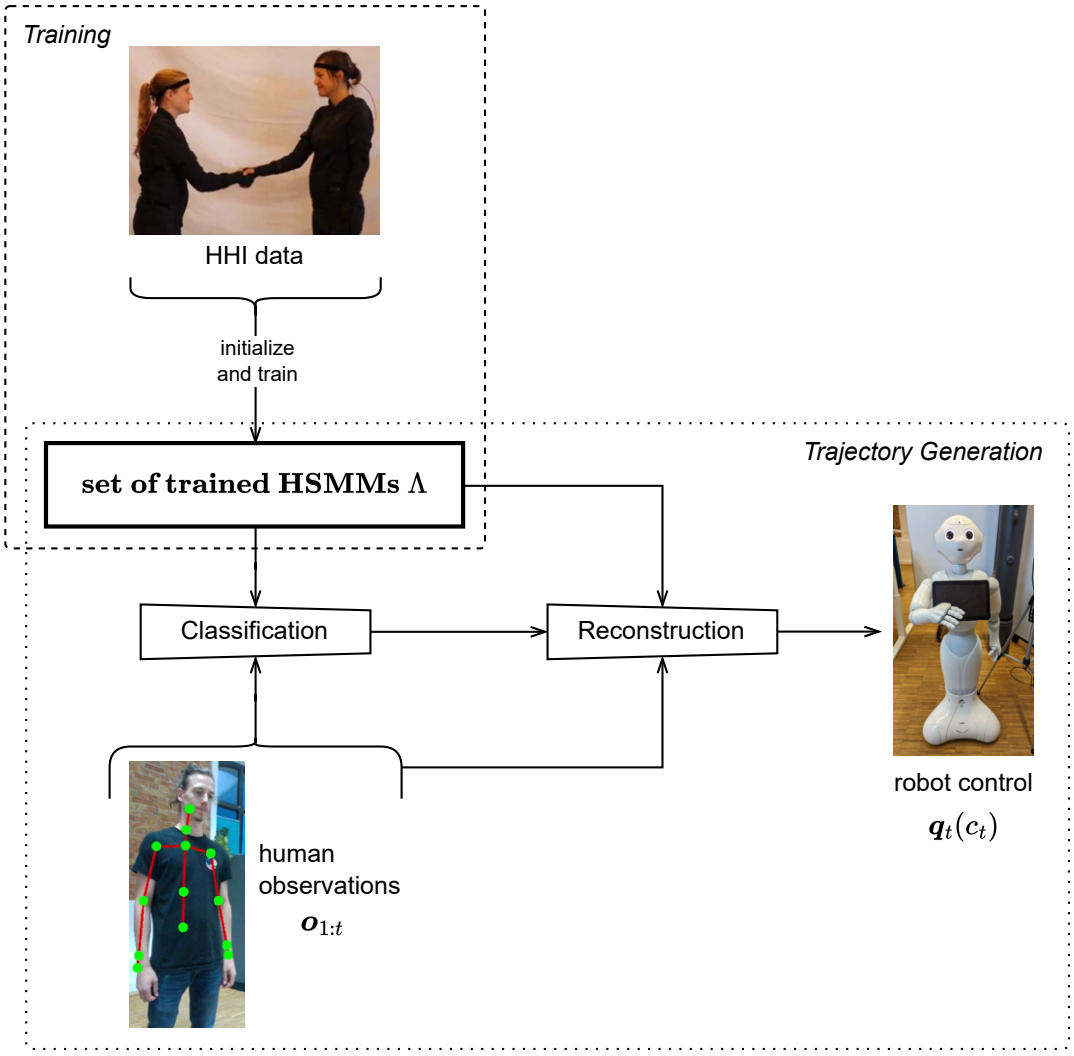


Figure 3.1.: This graphical depiction summarizes the overall framework proposed in this thesis. The content of the dashed box illustrates the training phase, where a set of models Λ is trained for each interaction from Human-Human demonstrations. The dotted box shows the reconstruction part consisting of a classifier that predicts the observed interaction, and the reconstruction procedure that eventually generates the output for the robot's response.

which also goes into the details of the trajectory segmentation discussed earlier in Section 2.2.3. The action classification is described in Section 3.2.2 and finally in Section 3.2.3, the reconstruction of trajectories is explained.

The mathematical formulations in the following three sections - regarding HSMMs - are mainly based on the notations introduced in Section 2.1. This section also explains why we are relying on HSMMs instead of HMMs, which is basically due to the fact of a more precise encoding of the durations of states.

3.2.1. Initializing and Training the Model

As discussed earlier in Section 2.2.3, segmenting a trajectory into more primitive movements can be beneficial, especially in terms of the re-usability of those segments and the better generalization to unseen scenarios while reproduction later on. A way to formulate such segmentation is through H(S)MMs (Section 2.1), where each state corresponds to a segment. Due to a better temporal encoding, we rely on HSMMs in this work.

HSMMs consist of multiple different (n) states. In the case of this thesis, the underlying distribution of each state $s_i \in \mathcal{S}$ is characterized by a Multivariate Normal Distribution (MVN) with mean $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}_i$ with $i \in \{1, 2, ..., n\}$. It represents the emission probabilities of the observations $\mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with $\boldsymbol{o}_t \in \boldsymbol{o}_{1:T}$. In essence, that can be seen as learning a Gaussian Mixture Model (GMM) over the observations and learning the

Algorithm 1 Initialize and Train Model

input:
1. training trajectories $\boldsymbol{o}_{1:T}$
2. number of states n

output:
1. trained HSMM λ

1: segment training trajectories $\boldsymbol{o}_{1:T}$ into n equally sized segments along the time t
2: initialize λ by representing the observations of each segment $i \in \{1, 2, ..., n\}$ as a MVN with $\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^1 \\ \boldsymbol{\mu}_i^2 \end{bmatrix}$; $\boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{11} & \boldsymbol{\Sigma}_i^{12} \\ \boldsymbol{\Sigma}_i^{21} & \boldsymbol{\Sigma}_i^{22} \end{bmatrix}$ (Equation (3.1))
3: train λ according to the Baum-Welch EM algorithm described in Appendix A.3
4: **return** λ

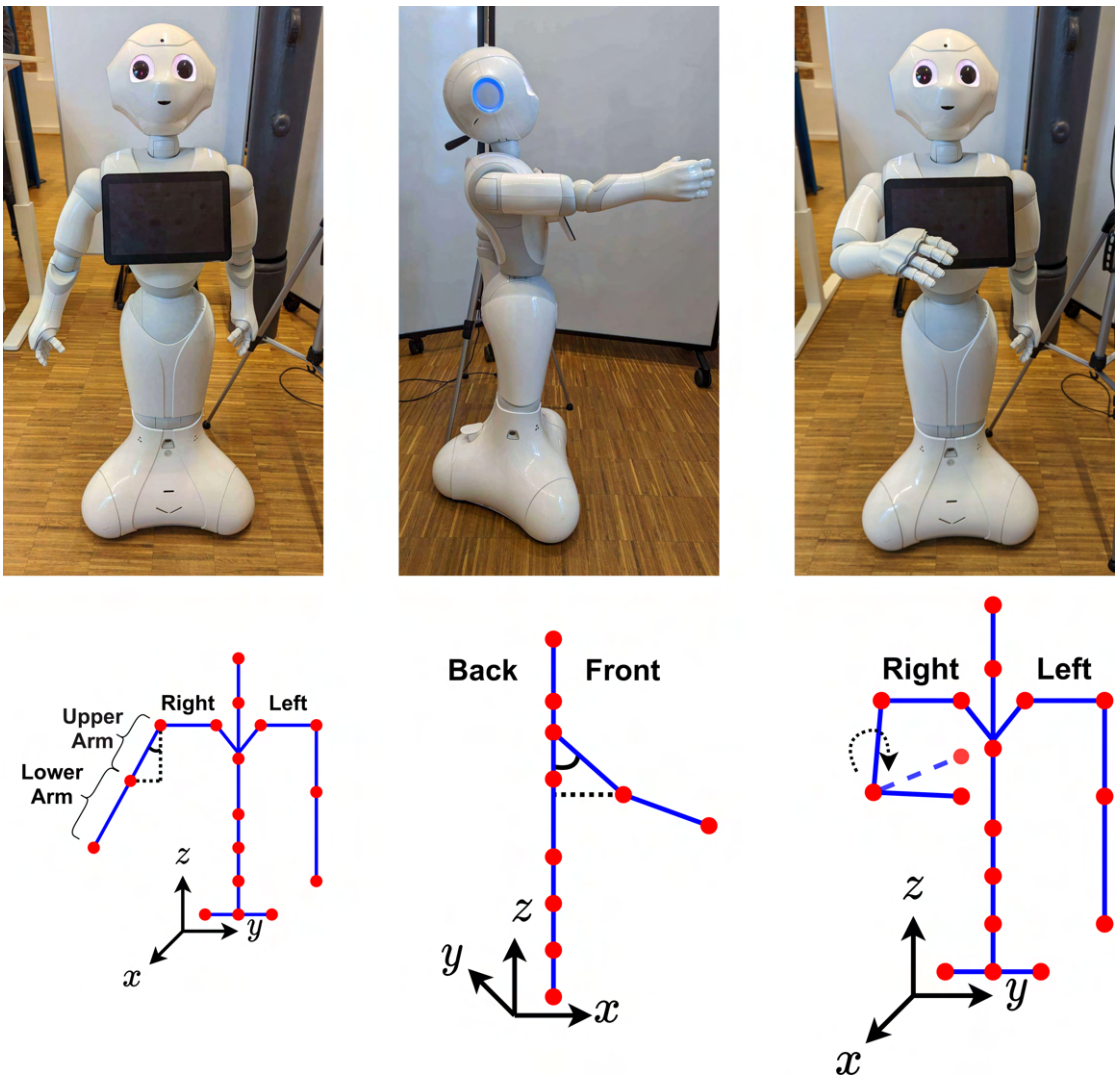


Figure 3.2.: This figure illustrates the geometric similarities between the degrees of freedom of a human's upper body and the humanoid robot Pepper. As can be seen, the shoulder roll (column 1), pitch (column 2), and the yaw and elbow angle (column 3) of a tracked human skeleton can be directly mapped to Pepper's joint angles.

temporal sequencing between the Gaussian components.
To encode the joint distribution between the interacting agents, the observations of both of them are concatenated, resulting in

$$\boldsymbol{\mu}_i = \begin{bmatrix} \boldsymbol{\mu}_i^1 \\ \boldsymbol{\mu}_i^2 \end{bmatrix} \; ; \quad \boldsymbol{\Sigma}_i = \begin{bmatrix} \boldsymbol{\Sigma}_i^{11} & \boldsymbol{\Sigma}_i^{12} \\ \boldsymbol{\Sigma}_i^{21} & \boldsymbol{\Sigma}_i^{22} \end{bmatrix} \tag{3.1}$$

This decomposition becomes helpful for predicting the trajectories of agent two later on (Section 3.2.3).

As stated before, for each interaction, an individual model is trained. The following explains the general procedure, which can universally be applied.
Before training a model, it first needs to be initialized. That happens with an arbitrary number of training demonstrations and a manually defined number of states n. The training demonstrations are the concatenated trajectories from both human actors. For actor one, those are the Cartesian positions and velocities of all relevant body joints. Although the position itself is already sufficient to track the movements, the velocity is very helpful for detecting and reconstructing repetitive movements like the "waving part" in a hand wave. It adds the feature of knowing at any particular time step, whether the arm is about to move left or right. For actor two, the joint angles and their velocities are given as input, which is because actor two is the one representing the robot in later reproduction scenarios, and since controlling a robot works by passing joint angles to it, it is useful to directly learn those.
The joint angle extraction works by utilizing the similarities between the joint spaces of a Humanoid robot and a Human [35], namely the shoulder angles (yaw, pitch, and roll) and the bending of the elbow by using the geometry of the skeleton of the demonstrations. Figure 3.2 exemplary shows that for the PEPPER robot [36] from Aldebaran Robotics[1] that is later on used for the testing in simulation (Section 4.3) and on a real robot (Section 4.4).

The initial segmentation is done by splitting the training data up into n equally sized segments along the time t. For each segment, all the observations that fall into that segment are represented by a MVN, as described above. The segments are then assigned to the states of the model with a linear sequencing of the segments along time t.

Training the model refers to the learning problem (3) stated in Section 2.1, which is solved by the Baum-Welch algorithm specified in Appendix A.3. Basically, it is a modified

[1] `https://www.aldebaran.com/en/pepper`

Expectation-Maximization (EM) algorithm that iteratively takes a model, re-estimates the parameters α, β, γ, and ξ from it, and then again uses those parameters to maximize the expectation of the model by recalculating it. This procedure continues until either a defined maximum number of iterations is reached, or until a certain threshold that describes the shift of the model is no more exceeded. In our case, the threshold is defined as the log-likelihood probability of the model given the training data.
The result is a new model $\overline{\lambda} = (\mathcal{S}, \mathcal{O}, \overline{\boldsymbol{D}}, \overline{\boldsymbol{A}}, \overline{\boldsymbol{\pi}})$ with a modified set of underlying distributions, stochastic matrix, and initial state distribution. The modification of the stochastic matrix is mainly what allows the learning of repetitive movements like the "waving part" in a hand wave.
The whole process is sketched in Algorithm 1.

3.2.2. Action Classification

Hence we are representing each action $c_k \in \mathcal{C}$ with a separate model $\lambda_k \in \Lambda$, a method to decide what model to use for reproduction is needed.
For that, we propose a classifier that predicts the intended interaction from a given (partial) demonstration trajectory. Ruffly speaking, that is done by looking for the model that best describes the observations of the given trajectory. The action that corresponds to the model that fits the best becomes the predicted action.

Formally, the classifier works by taking a (partial) observation trajectory of the leading actor (actor 1) $\boldsymbol{o}^1_{1:t}$ and a set of models $\Lambda = \lambda_1, \lambda_2, ..., \lambda_K$ as arguments. For every state

Algorithm 2 Action Classifier

input:
1. set of models $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_k\}$
2. (partial) observation trajectory of actor 1 $\boldsymbol{o}^1_{1:t}$

output:
1. predicted type of action

initialize lst_1 ▷ *type of action with highest conditional probability for every observation* $\boldsymbol{o}_{t'}$
$lst_2 \leftarrow$conditional probabilities (Algorithm 3)
for $\boldsymbol{o}^1_{t'} \in \boldsymbol{o}^1_{1:t}$ **do**
    append type of action of λ_k with highest conditional probability in lst_2 to lst_1
end for
return type of action with highest occurrence in lst_1

Algorithm 3 Conditional Probabilities

input:
1. set of models $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_k\}$
2. (partial) observation trajectory of actor 1 $\boldsymbol{o}^1_{1:t}$

output:
1. conditional probabilities of every observation $\boldsymbol{o}_{t'}$ for each model λ_k

initialize lst_1 ▷ *conditional probabilities of every observation $\boldsymbol{o}^1_{t'}$ for each model λ_k*
for $\lambda_k \in \Lambda$ **do**
  initialize lst_2 ▷ *conditional probabilities of every observation $\boldsymbol{o}^1_{t'}$ for each state s_i of λ_k*
  for $s_i \in \mathcal{S}$ **do**
    initialize lst_3 ▷ *conditional probabilities of every observation $\boldsymbol{o}^1_{t'}$ for s_i*
    for $\boldsymbol{o}^1_{t'} \in \boldsymbol{o}^1_{1:t}$ **do**
      $f = \exp\left(-\frac{1}{2}(\boldsymbol{o}^1_{t'} - \boldsymbol{\mu}^1_k)^T(\boldsymbol{\Sigma}^1_k)^{-1}(\boldsymbol{o}^1_{t'} - \boldsymbol{\mu}^1_k)\right)$ (Equation (3.2))
      $h_i(\boldsymbol{o}^1_{t'}) = \frac{\alpha_i(\boldsymbol{o}^1_{t'})}{\sum_{k=1}^{|\mathcal{S}|}\alpha_k(\boldsymbol{o}^1_{t'})}$ (Equation (3.3))
      append conditional probability $f \cdot h_i(\boldsymbol{o}^1_{t'})$ to lst_3
    end for
    append lst_3 to lst_2
  end for
  append sum over all states of lst_2 to lst_1
end for
return lst_1

$s_i \in \mathcal{S}$ of all models $\lambda_k \in \Lambda$, the algorithm iterates through the the given observations $\boldsymbol{o}^1_{1:t}$. At each iteration $\boldsymbol{o}^1_{t'} \in \boldsymbol{o}^1_{1:t}$, it considers the observation sequence $\boldsymbol{o}^1_{1:t'}$ until the time step t', and computes the element wise product of the Probability Density Function (PDF) f

$$f = \exp\left(-\frac{1}{2}(\boldsymbol{o}^1_{t'} - \boldsymbol{\mu}^1_k)^T(\boldsymbol{\Sigma}^1_k)^{-1}(\boldsymbol{o}_{t'} - \boldsymbol{\mu}^1_k)\right) \tag{3.2}$$

and the Forward Variable $h_i(\boldsymbol{o}^1_{t'})$

$$h_i(\boldsymbol{o}^1_{t'}) = \frac{\alpha_i(\boldsymbol{o}^1_{t'})}{\sum_{k=1}^{|\mathcal{S}|}\alpha_k(\boldsymbol{o}^1_{t'})} \tag{3.3}$$

where

$$\begin{aligned}\alpha_i(\boldsymbol{o}^1_{t'}) &\triangleq p(s^t_i, \boldsymbol{o}^1_{1:t}|\lambda) \\ &= \mathcal{N}(\boldsymbol{o}^1_{t'}|\boldsymbol{\mu}^1_i, \boldsymbol{\Sigma}^1_i) \sum_{k=1}^{|\mathcal{S}|} \sum_{d=1}^{|\mathcal{D}|} \alpha_k(\boldsymbol{o}^1_{t'-1}) a_{k,i} p_i(d)\end{aligned} \tag{3.4}$$

with $\alpha_i(\boldsymbol{o}^1_1 = \pi_i)$.
This conditional probability is further summed over every state of all models λ_k, resulting in one representative value for each observation $\boldsymbol{o}^1_{t'}$. Then, for each model λ_k those values are compared, and the model with the greatest value at each time step t' is noted. Finally, the model with the most occurrences of "greatest value" corresponds to the predicted action.
The whole procedure is shown in Algorithm 2 and 3. More details about the forward variable and its definition can be found in Appendix A.1.

3.2.3. Conditioning based on Human Observation

Given a (partial) trajectory of a human performing a social interaction, the overall goal is to infer a trajectory to appropriately respond to the intended interaction of the human actor.
As mentioned in Section 3.2.1, the robot will always take the place of actor two in a Human-Human Interaction (HHI) scenario. With that, the goal becomes inferring the joint angles of actor two $\boldsymbol{q}^2_{1:t}$ from the positions and velocities of the joints of actor one

Algorithm 4 Conditioning based on Human Observation

input:
1. (partial) observation trajectory $\boldsymbol{o}^1_{1:t}$
2. model λ

output:
1. predicted joint trajectory $\boldsymbol{q}^2_{1:t}$

1: condition λ using $\boldsymbol{o}^1_{1:t}$
2: use the conditioned λ to generate $\boldsymbol{q}^2_{1:t}$ according to Equation (3.5)
$\boldsymbol{q}^2_{1:t} = \sum_{k=1}^{|\mathcal{S}|} h_k\left(\boldsymbol{o}^1_{1:t}\right)\left(\boldsymbol{\mu}^2_k + \boldsymbol{\Sigma}^{21}_k\left(\boldsymbol{\Sigma}^{11}_k\right)^{-1}\left(\boldsymbol{\mu}^1_k - \boldsymbol{o}^1_{1:t}\right)\right)$
3: return $\boldsymbol{q}^2_{1:t}$

$\boldsymbol{o}^1_{1:t}$, given a trained model.
Assuming the type of interaction got correctly predicted by the action classifier (Section 3.2.2), the trajectory prediction works by conditioning the learnt joint distribution of the given action, using the observations $\boldsymbol{o}^1_{1:t}$. The conditioned HSMM is then used to reproduce the trajectories of the second agent $\boldsymbol{q}^2_{1:t}$ as

$$\boldsymbol{q}^2_{1:t} = \sum_{k=1}^{|\mathcal{S}|} h_k \left(\boldsymbol{o}^1_{1:t}\right) \left(\boldsymbol{\mu}^2_k + \boldsymbol{\Sigma}^{21}_k \left(\boldsymbol{\Sigma}^{11}_k\right)^{-1} \left(\boldsymbol{\mu}^1_k - \boldsymbol{o}^1_{1:t}\right)\right) \tag{3.5}$$

which is utilizing the aforementioned decomposition in Equation (3.1).
The procedure is illustrated in Algorithm 4.

4. Experimental Evaluation

The following chapter is discussing the overall result we achieved with the proposed approach. Section 4.1 provides some general information about the datasets we used and also the modifications we made to them to fit our needs. Section 4.2 is dealing with the setup and the evaluation of the proposed framework in general, without any robot participation. The testing on robots, in simulation and real scenarios, is finally covered in Section 4.3 and Section 4.4.

4.1. Dataset and Data Setup

This section deals with the two datasets used throughout the evaluation. The first one is proposed in Section 4.1.1 and is used for generally evaluating the framework discussed in Section 4.2, as well as for the testing in the simulation (Section 4.3). The other dataset, proposed in Section 4.1.2, is collected by ourselves and used for the testing on the real robot (Section 4.4).

4.1.1. Bütepage Interaction Dataset

For the evaluation of the proposed framework we rely on the dataset compiled by Bütepage et al. [29] [1]. It contains high frequency motion capture data of Human-Human Interactions (HHI) and Human-Robot Interactions (HRI), captured by Rokoko[2] motion capture suits. The data was recorded at a Frequency of 40Hz and consists of the 3D Cartesian joint positions of the actors.

[1] https://github.com/jbutepage/human_robot_interaction_data
[2] https://www.rokoko.com/

Action	Description
Hand Wave	Typical hand wave, where both interacting partners raise their hands at about head level, and perform an oscillatory left-and-right motion for three to four cycles. Finally, the hands go back to the neutral position.
Handshake	Typical handshake, where both interacting partners reach out, grasp each other's hand, and perform an oscillatory up-and-down motion for three to four cycles. Finally, the hands go back to the neutral position.
Rocket Fist Bump	Modified version of a fist bump, where both interacting partners reach out their fists towards each other at about waist level to bump it. Then the fists are moved upwards in a synchronized manner with the hand of the following actor being slightly below the one of the leading actors. After reaching a suitable, comfortable height, the hands go back to the neutral position.
Parachute Fist Bump	Modified version of a fist bump, where both interacting partners reach out their fists towards each other at a higher level about head to bump it. Then they bring it down in a synchronized oscillating left-and-right motion with the hand of the following actor being slightly below the one of the leading actors. After reaching about waist level, the hands go back to the neutral position.

Table 4.1.: Descriptions of the different types of interactions

(a) Fist bump collected by NuiSI

(b) Handshake collected by Bütepage at al. [29]

Figure 4.1.: Examples of two interactions from the used datasets.

The dataset includes four different social interactions: hand wave, handshake, rocket fist bump, and parachute fist bump. All actions follow a similar structure by starting in a neutral standing posture, then executing the action, and finally going back to the initial neutral posture. A more detailed description of each interaction can be found in Table 4.1. Figure 4.1a additionally provides an example image of the data collection of a handshake. Regarding the two agents, a differentiation between a “leading” and a “following” actor can be made. The leading actor determines the modality of the execution, to which the following actor needs to react and adapt. In the case of the dataset by Bütepage et al. [29], actor one always fills in the role of the leading agent, and actor two represents the following agent. For the idea of this thesis, the robot should always be the following agent, hence the robot is supposed to react to a human. With that, the data of the leading agent will also be referenced as the input data, and the data of the following agent as the output data. This differentiation is relevant since not all interactions are equal for both actors. Actions can be categorized into “symmetrical” and “asymmetrical” ones. In symmetrical movements, like the hand wave and handshake, both actors perform pretty much the same motion. For both of the fist bumps, the movements of both actors differ from each other, thus being asymmetrical.
Furthermore, the demonstrations within the same interaction are not equal in terms of the motion sequencing. That, for example, means a handshake does not always start with the arm moving to the leader’s actor right and end on the left or something similar.

More details about the dataset can be found in the paper by Bütepage et al. [29]. The

supplementary material on the publisher's website[3] additionally provides videos of all four interactions being performed in the setup of capturing them.

In the case of this thesis, we only made use of the HHI part of the dataset. On the one hand, the reason for that emerges from the general idea of this thesis, which is the learning of social interactions based on human demonstrations. On the other hand, training the following agents part from robot demonstrations is insufficient, as the data is specific only to the robot the data was captured on.
Also, we have split the data into training and test demonstrations, where the training set makes up 80%, and the test set 20%.

In the following, we are explaining further modifications we made to the dataset to better fit our needs, which however needs to be considered separately for two different evaluation stages with individual requirements.

The first stage can be seen as the process of finding optimal models by evaluating and tweaking the parameters. In that stage, no robot is involved. All the training, testing, and evaluation is purely based on error estimators and graphical depictions. The detailed process, along with its results, is described in Section 4.2.
Hence this stage involves a lot of computational expensive model training, we sampled the dataset down to constant 250 time steps per demonstration, to get rid of some computational overhead. The 250 time steps are about a quarter of the original 40Hz sampling rate and do not affect the evaluation of the models, while noticeable saving time and energy. Another sanction to encounter unnecessary computational overhead is the limitation to only the wrist joint of the actor's right arm, which also brings the advantage of the input and output data only being three-dimensional, which can still be nicely graphical depicted and evaluated. We decided on the wrist joint since it is the joint that encounters the highest spatial movement.
Nonetheless, we derived and added the joint velocities to the dataset, by computing the difference between the two consecutive position values at each time step. The idea is to get a better sequential understanding, especially for repetitive movements like the handshake or hand wave. Including the velocity makes it is easy to determine whether an arm is currently moving up or down in a handshake - or left or right in a hand wave.

The second stage deals with the testing on the robot in simulation, which is covered in Section 4.3.
In that case, we are using the dataset with its original sampling rate. For the leading actor,

[3] `https://www.frontiersin.org/articles/10.3389/frobt.2020.00047/full#supplementary-material`

we again add the velocities for a better sequential understanding, and also reduce the data to only the wrist joint to circumvent unnecessary computational overhead. Hence the wrist joint is by far the most meaningful, using only this one is totally sufficient for the input data and therefore for detecting the intended movements. For the following actor, however, that would be insufficient, as the dimension of the output data correlates with the dimensions needed for the reconstruction. Hence the movements of the proposed interactions are pretty much based on the upper body anyway, especially the right arm, we decided on curtailing to the shoulder and elbow joint of the following actor's right arm. The wrist joint can be neglected since a robot is operating in joint space and therefore needs joint angles to move. While the wrist joint is experiencing a lot of movement in the Cartesian space, it is pretty much doing nothing in joint space - at least for the interactions we are dealing with. All the movements primarily arise from the shoulder (roll and pitch angles) and elbow joints (roll and yaw angles).
Since the robot relies on joint angle data, the proposed framework needs to output such data, which we derive from the Cartesian coordinates of the following actor as stated in Section 3.2.1 and Figure 3.2. Further, we again compute and then add the velocities of the joint rotations for a better sequential understanding.

4.1.2. Nuitrack Skeleton Interaction Dataset (NuiSI)

During testing, we observed the human actor using an Intel Realsense D435[4] RGB-D camera [37]. Using RGB-D cameras for recording data is way less accurate than motion capture suits [5], like the Rokoko suit used by Bütepage et al. [29].
The observations of the human actor during our trials were therefore differing to much from the data by Bütepage et al. [29] that we initially used for training the models. Hence, we ended up training the HSMMs on a dataset recorded by the same Intel Realsense D435 RGB-D camera that we used for recording the human actor during testing.

The used NuiSI dataset was collected on our own and consists of two interactions between human partners. A usual handshake and a rocket fist bump similar to the ones by Bütepage et al. [29] that are further described in Table 4.1. In Figure 4.1b, an example of the data collection of a fist bump can be seen.
We track the upper body joints (waist, spine, neck, head, and arms) of both human partners, using the Nuitrack[5] skeleton tracking software. The skeletons are then rotated within the frame of the body such that the x-axis is going forward, the y-axis goes from

[4] https://www.intelrealsense.com/depth-camera-d435/
[5] https://nuitrack.com/

the right to the left, and the z-axis is upwards, with the origin at the shoulder. In this frame, the 3D positions of the right hand of the first partner, along with their 3D velocities represent the human Degrees of Freedom (DoFs). The robot joint angles and velocities are again extracted from the skeleton of the second partner and are used as the robot DoFs as stated in Section 3.2.1 and Figure 3.2.

4.2. Experimental Setup and Evaluation

This Section deals with the setup and the evaluation of the proposed framework in general and without any robots, which includes finding the optimal parameters based on error estimators and graphical depictions (Section 4.2.1). A comparison to three different approaches is given in Section 4.2.2 and the action classifier is evaluated in Section 4.2.3

All models were trained using the python version of the PbDlib[6] [13] and the dataset proposed in Section 4.1.1.
Due to numerical stability, we always trained all our models with only 15 demonstrations per action, which brings up the problem that the performance of a model is highly dependent on the training data batch that the model was trained on. This behavior can be seen in Figure 4.4 and 4.5, where each model has been trained 100 times on a randomly sampled set of 15 trajectories. The framework proposed in this thesis is illustrated in green. Relevant are the caps of each bar, which represent the minimum and maximum Mean Squared Error (MSE) of all 100 runs. For each run, all the test trajectories of the dataset are considered. It is to see that the maximum value is up to three times higher than the minimum value for the proposed framework, and even up to four times higher for the other approaches.
To overcome this problem while evaluating, we either did exactly what we were doing for Figure 4.4 and 4.5, where we processed the results of multiple runs with different random training batches. Or, if the model needed to be consistent - like for Figure 4.2, we used a fixed batch of demonstrations.

4.2.1. Evaluation of Human-Human Interactions

As stated before, in this section, we are first evaluating the performance of our approach only on the 3D Cartesian wrist trajectories of both human actors and not yet on a robot.

[6] `https://gitlab.idiap.ch/rli/pbdlib-python`

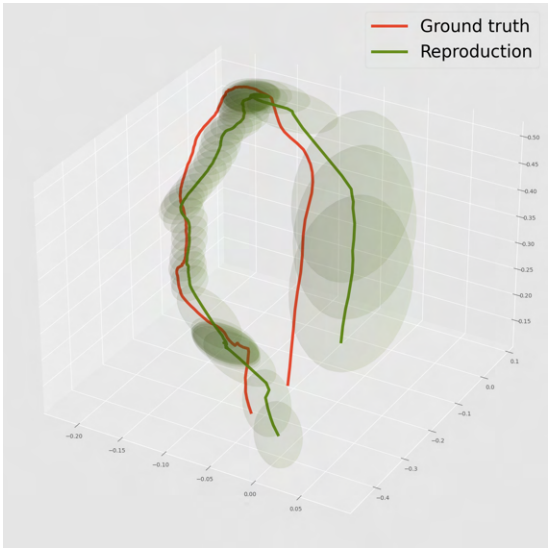
Action	Number of states	Regularization factor
Hand wave	4	1×10^{-2}
Handshake	4	1×10^{-2}
Rocket Fist Bump	8	1×10^{-2}
Parachute Fist Bump	3	1×10^{-2}

Table 4.2.: For each type of interaction, this table shows the parameters we used for training the individual models

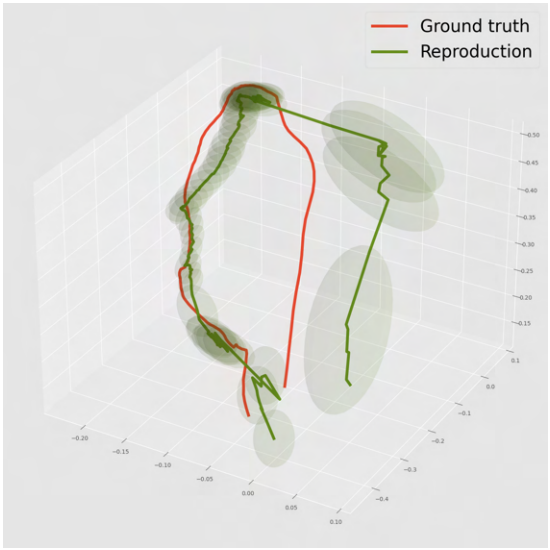
This proceeding allows for plotting the trajectories and also for easier error calculations. Moreover, the basic framework itself can be evaluated, hence it is not affected by potential misbehavior arising from the robot.

For the Expectation-Maximization (EM) algorithm that we used for training, the maximum number of iterations was set to a fixed number of 40 iterations. The change of the model within each iteration is defined by the maximum log-likelihood increase. As soon as this value is no more exceeding 1×10^{-4}, the algorithm pertains as converged and stops prematurely.

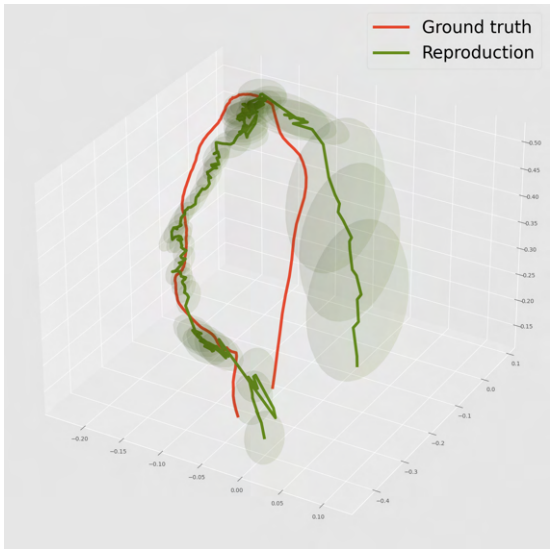
To initialize and train the models, two decisive parameters are required. The first one is simply the number of states of the HSMM. The second one is a so-called regularization factor that gets internally added to some of the intermediate variables to prevent numerical instabilities, especially when dealing with matrix calculations. Hence, it should be kept as low as possible. To find out the optimal parameters for the interactions, we used the average MSE considering all test demonstrations, as the performance criterion. For each demonstration, the MSE is calculated between the predicted and the ground truth trajectory at each time step after observing the leading agent. Further, we always verified the results by plotting both trajectories against each other. The resulting parameters, for each type of interaction, are shown in Table 4.2.
The constant regularization factor of 1×10^{-2} is because the reconstructed trajectory gets jagged for lower values. This behavior can be seen in Figure 4.2, which shows the influence of different regularization factors by the example of the wrist trajectory of a rocket fist bump interaction. Each of the graphs displays the ground truth trajectory against the one reconstructed by the HSMM trained with eight states and the respective regularization factor. Figure 4.2a is the reference with a regularization factor of 1×10^{-2}. With a slightly lower value of 1×10^{-3} (Figure 4.2b) it is to see that the trajectory already



(a) regularization = 1×10^{-2}



(b) regularization = 1×10^{-3}



(c) regularization = 1×10^{-6}

Figure 4.2.: Those three graphs show the influence of different regularization factors by the example of the wrist trajectory of a rocket fist bump interaction. Each graph displays the ground truth trajectory against the one reconstructed by the HSMM trained with eight states and the respective regularization factor.

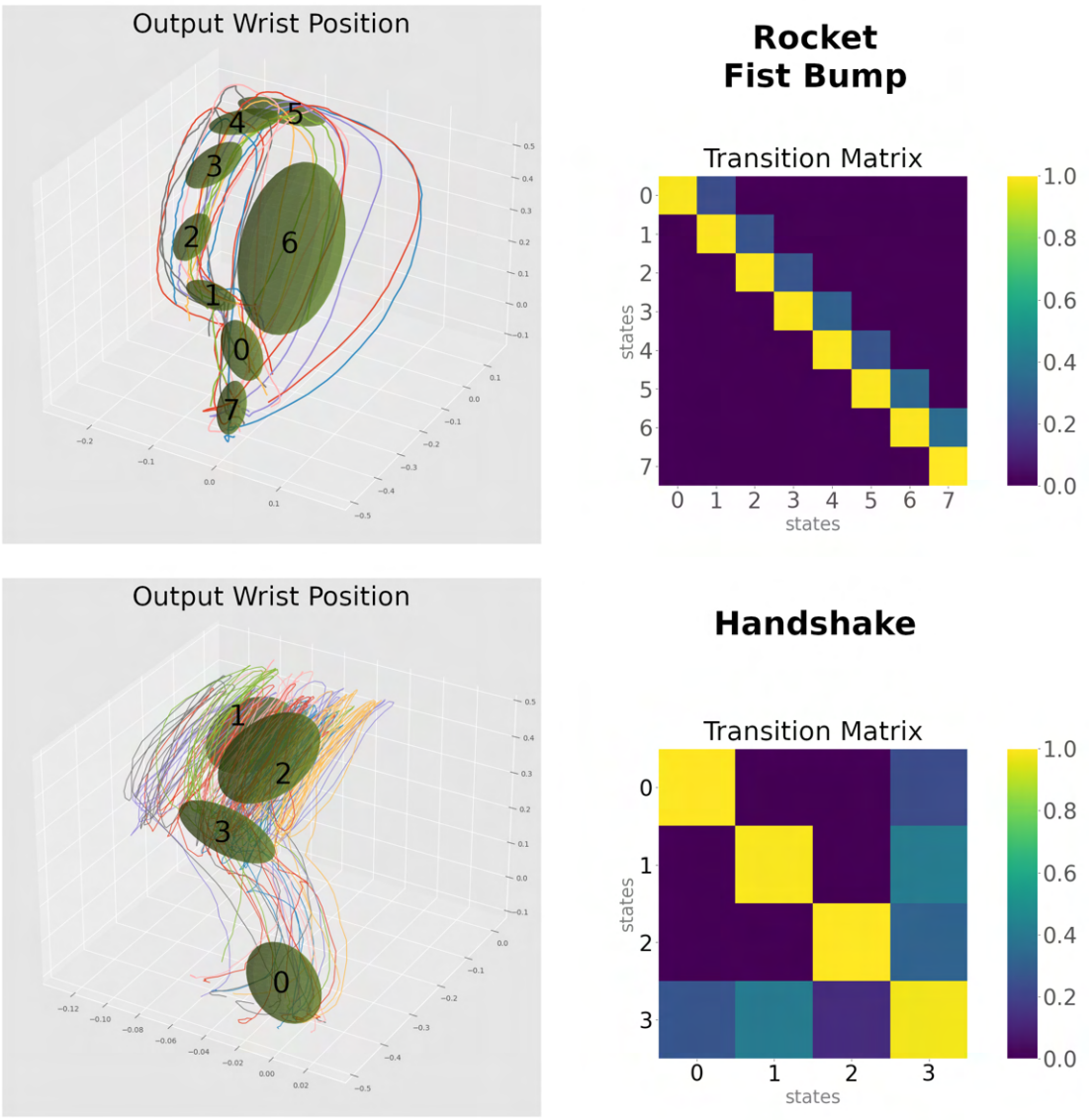


Figure 4.3.: This figure gives two examples of trained HSMMs. The upper one is for the rocket fist bump and the other one is for the handshake interaction. On the left of each subgraph, the 15 training trajectories of the following agent's right arm wrist joint, are plotted. On top of that, the trained MVNs along with their sequential order are displayed. On the right side, the transition matrix of each model is shown.

gets jagged, which just gets worse for even lower values (Figure 4.2c).

With the optimal parameters for training figured out, we can take a closer look at the trained models. For the handshake and the rocket fist bump, Figure 4.3 displays, on the left side, the position trajectories of the following agent's right arm wrist joint, of each demonstration used for training. In addition to that, the trained Multivariate Normal Distributions (MVNs) of the HSMM, along with their sequential order, are plotted. The right sight of each subgraph shows the transition matrix of the corresponding model, which illustrates the state transition probabilities. While the rocket fist bump trajectories are pretty much straightforward, the handshake is representative for interactions with a repetitive part. For better visualization and since it is the more relevant part, we only display the trajectories of the following agent.
In the left graph of the rocket fist bump, it can be seen that all demonstrations start from a certain point, conduct a quite straight motion, and end up in about the same location where they started. Along those demonstrations, the MVNs that describe the trajectory distributions can be seen. They are labeled from 0 to 7 in a timely order. This behavior is pretty much the same for the handshake, but with multiple repetitions of the conducted movement (i.e. the shaking part) and only four MVNs.
The main difference between having a straight motion versus a repetitive one also reflects to the transition matrices of the models. Those illustrate the state transition probabilities of the HSMMs. The field $(2, 3)$ for example represents the probability of getting into state 3, being in state 2. The shades go from purple (zero probability), over blue, turquoise, and green, up until yellow (100% probability). For both of the matrices, the highest probabilities are along the diagonal, i.e. the self transitions, which makes sense as each state emits multiple observations. Other than that, for the rocket fist bump, the other significant probabilities describe the transitions to the next state, which is fully in order from state 0 to state 7. In contrast to that, for the handshake, loops that describe the repetitive part of the motion can be identified. The most significant one is $(1, 3)$ and $(3, 1)$, which shows the advantage of trajectory segmentation and the use of HSMM along with it, where segments/states can be re-used multiple times.

4.2.2. Model Comparison

In this subsection, we evaluate the performance of the framework used in this thesis. As described, the proposed framework consists of a set of HSMMs of all four interactions. Each of those models is trained using the positions and the velocities.
We compare this approach against three other ones. The first one also consists of a set of

HSMMs of all four interactions. The only difference here is that the models are trained using only the positions. The second approach consists of only one single HSMM that is trained on all four interactions at the same time. Though, it considers the positions as well as the velocities. The setup of the last approach is the same as the one we are using in this thesis, but instead of HSMMs, it relies on normal HMMs.

To compare the different approaches, all of the models are trained using the same parameters as in Table 4.2. Also, all setups consider the use of only the wrist joint. To overcome the problem of the varying performance depending on the demonstrations used for training, we computed each setup 100 times, always using a random batch of 15 trajectories, which provides a broad distribution of samples for each of the approaches. For each of the 100 runs, all the test trajectories of the dataset are considered. As the performance measurement, the MSE and t-Test are used. Regarding the approaches consisting of multiple HSMMs, the MSE refers to the average of all individual models. The t-Test is a statistical hypothesis test that estimates how similar two distributions are by comparing samples of each distribution against each other. If the resulting so-called p-Value is lower than 0.05, the underlying distributions of the samples are considered significantly different. In the case of this evaluation, the t-Test helps to estimate how similar two different approaches perform.
The results of the comparison can be found in Figure 4.4 for each action individually and in Figure 4.5, the overall results are illustrated by taking the average over all four interactions.

On the x-axis of each graph, the different approaches are displayed. The green one represents the basis we are comparing against, which is the framework proposed in this thesis. The other approaches are colored blue, with - from left to right - the set of HSMMs only using the positions, the single HSMM consisting of all interactions, and the one using HMMs. The y-axis represents the MSE, which, for a better comparison, is scaled the same throughout all interactions in Figure 4.4. For each approach, all 100 samples are plotted, where the black circles illustrate the mean and the wider part of the bar the standard deviation. The caps at each end of the bars show the minimum and maximum MSE of the sample. An asterisk on top of a bar denotes the underlying distribution of the samples being significantly different from the reference (i.e. the framework used in this thesis) according to the p-Value of the t-Test. The dashed line that connects all the black circles is simply for easier comparing the different means.

Figure 4.4 considers the results for each interaction individually. According to the mean, for all the interactions apart from the handshake, the single HSMM performs the worst, followed by the one using only the positions. The best performance is provided by the

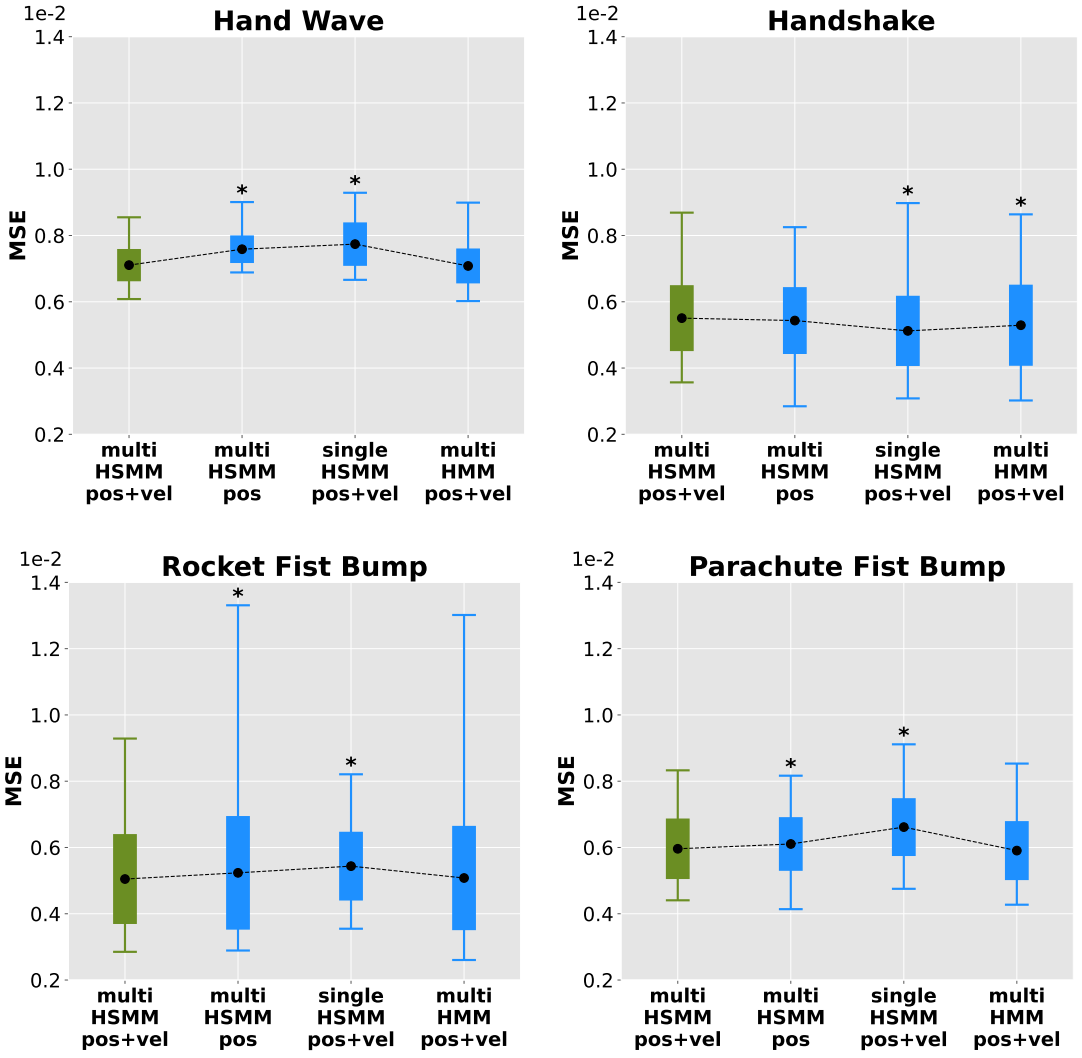


Figure 4.4.: Those plots compare, for each type of action, the proposed method against three other ones displayed on the x-axis. The y-axis shows the average MSE of 100 runs on all test trajectories in the dataset. For each run, the models are trained on a random batch of 15 demonstrations. The circles display the mean of the samples, the wider part of the bar the standard deviation, and the caps at the end of each bar the minimum and maximum MSE. An asterisk on top of a bar shows the result of the t-Test being less than 0.05, and thus the sample being significantly different from the proposed method.

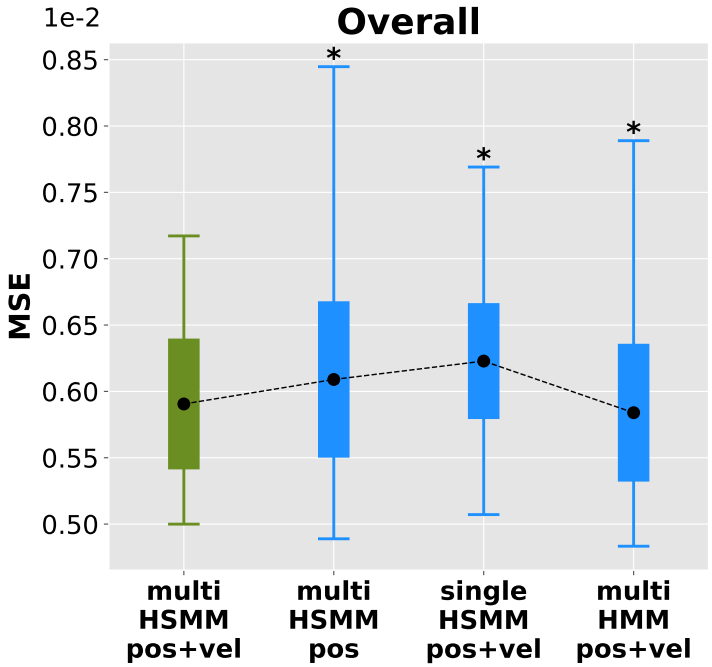


Figure 4.5.: This plot compares the method proposed in this thesis against three other ones displayed on the x-axis. The y-axis shows the average MSE of 100 runs on all test trajectories in the dataset. For each run, the models are trained on a random batch of 15 demonstrations. The circles display the mean of the samples, the wider part of the bar illustrates the standard deviation, and the caps at the end of each bar show the minimum and maximum MSE. An asterisk on top of a bar represents the result of the t-Test being less than 0.05 and therefore the sample being significantly different from the proposed method.

reference approach and the one using HMMs, which are both performing about the same. This finding is also reflected by the t-Test that considers the results of both of these approaches as similar. Nonetheless, this behavior does not hold for the handshake interaction, albeit the difference in the setups is minor.
Comparing the variance of all frameworks, they all show about the same width, which also holds when looking at the minimum and maximum outliers. In that case, however,

it is the rocket fist bump interaction that stands out. Here the HSMMs only using the positions and the HMM setup, exhibit a significantly larger variance and range of the maximum MSE.
Looking at the overall results in Table 4.5, it can be seen that the reference approach performs the best along with the one using HMMs, although they were the worst for the handshake interaction. According to the mean, the HMM approach even performs slightly better than the one used in this thesis. Compared to the other approaches though, this difference is kind of negligible. The difference in the standard deviation and maximum outliers of the HSMMs only using the positions and the HMM framework arise mainly from the results of the rocket fist bump. According to the t-Test, all of the setups differ significantly from the reference. However, the HMM approach is still quite close, as seen for the individual interactions.
An explanation of why the HSMM and the HMM approach are performing about the same can be found in Appendix B.

4.2.3. Action Classification

To evaluate the proposed action classifier, we took a set of trained models of each interaction and tested it against the whole set of test demonstrations. We did so for different numbers of observed time steps, while always noting down the number of correct predictions for each interaction. Since the performance of a model is quite dependent on the set of demonstrations it got trained on, we ran the analysis 100 times with a randomly sampled set of 15 training trajectories each run. The average results of all 100 runs are presented in Table 4.3, where for each run, all the test trajectories of the dataset are considered.

It can be seen that for 25 out of 250 observed time steps, the classifier does not perform well, especially for the hand wave and the parachute fist bump. For the hand wave that already gets significantly better, when having observed 50 time steps. At 75 observed time steps, the hand wave, handshake, and rocket fist bump get predicted pretty much perfect already, while the parachute fist bump still lacks behind. That however changes when having observed 100 time steps, which is 40% of the whole trajectory. At this point, the classifier is almost perfect for all the interactions with a total accuracy of 97.44%. From there on, the performance increase kinda stagnates. Between 125 and 150 observations, the overall score even decreases a little bit, but that can be seen as measurement tolerance.

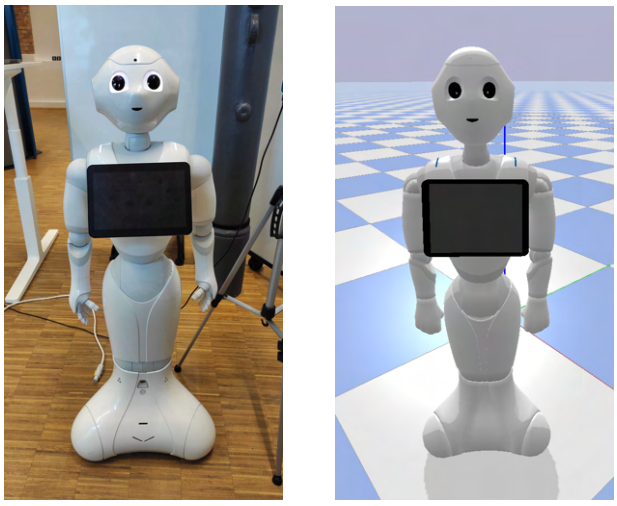
	average number of correct predictions for each action within 100 runs *(in each run, each of the models is trained on 15 randomly sampled training trajectories)*				
observed time steps *(total 250)*	hand wave *(total 7)*	handshake *(total 8)*	rocket fist bump *(total 14)*	parachute fist bump *(total 10)*	overall *(total 39)*
25 *(10%)*	0.65	5.82	11.87	0.89	19.23 *(~49.31%)*
50 *(20%)*	5.63	6.54	13.72	1.82	27.71 *(~71.05%)*
75 *(30%)*	7.0	7.65	13.78	5.8	34.23 *(~87.77%)*
100 *(40%)*	7.0	8.0	13.68	9.32	38.0 *(~97.44%)*
125 *(50%)*	7.0	8.0	13.54	9.96	38.5 *(~98.72%)*
150 *(60%)*	7.0	8.0	13.42	10.0	38.42 *(~98.51%)*

Table 4.3.: This table shows the classifier's performance on the test data for having observed a certain number of time steps. The results are listed in terms of the different actions and also the overall score regarding all actions. The "total" value corresponds to the total number of test trajectories. The values within the table show how many of those test trajectories the classifier managed to predict correctly, regarding the number of observed time steps.
The results are evaluated on 100 runs for each number of observed time steps. In each of the 100 runs, the models for each action are trained on a randomly sampled set of 15 trajectories from the set of all training trajectories. For evaluating each run, all test trajectories of the dataset are considered.

4.3. Testing in Simulation

Before testing on a real robot, we wanted to make sure that the proposed framework itself is working properly, by running it in simulation. For the real robot, the goal is to have the algorithm running on a PEPPER robot [36] from Aldebaran Robotics[7] (Figure 4.6a). For the simulation we are therefore relying on QIBULLET[8] [38], which is a simulator for the Pepper robot based on the python Bullet engine (Figure 4.6b). Pepper has six Degrees of Freedom (DoFs) in each arm (three for the shoulder, and one for the elbow, wrist, and hand closure), of which we control the shoulder and the elbow joints using qiBullet.

For the testing we are directly passing the joint angles to the robot, which are reconstructed by the HSMM conditioned on the observed trajectory of a human's right hand wrist joint 3D coordinates (Section 3.2.3). In addition to that, we also simulate the skeleton of the observed humans right arm (red boxes) in front of Pepper. This way, we can not only see the way Pepper executes the predicted trajectory, but also how responsive it is towards the human. An example of a rocket fist bump can be seen in Figure 4.7, which yields promising results.



(a) Real scenario (b) qiBullet simulation

Figure 4.6.: Pepper the Humanoid Social Robot

[7] https://www.aldebaran.com/en/pepper
[8] https://github.com/softbankrobotics-research/qibullet

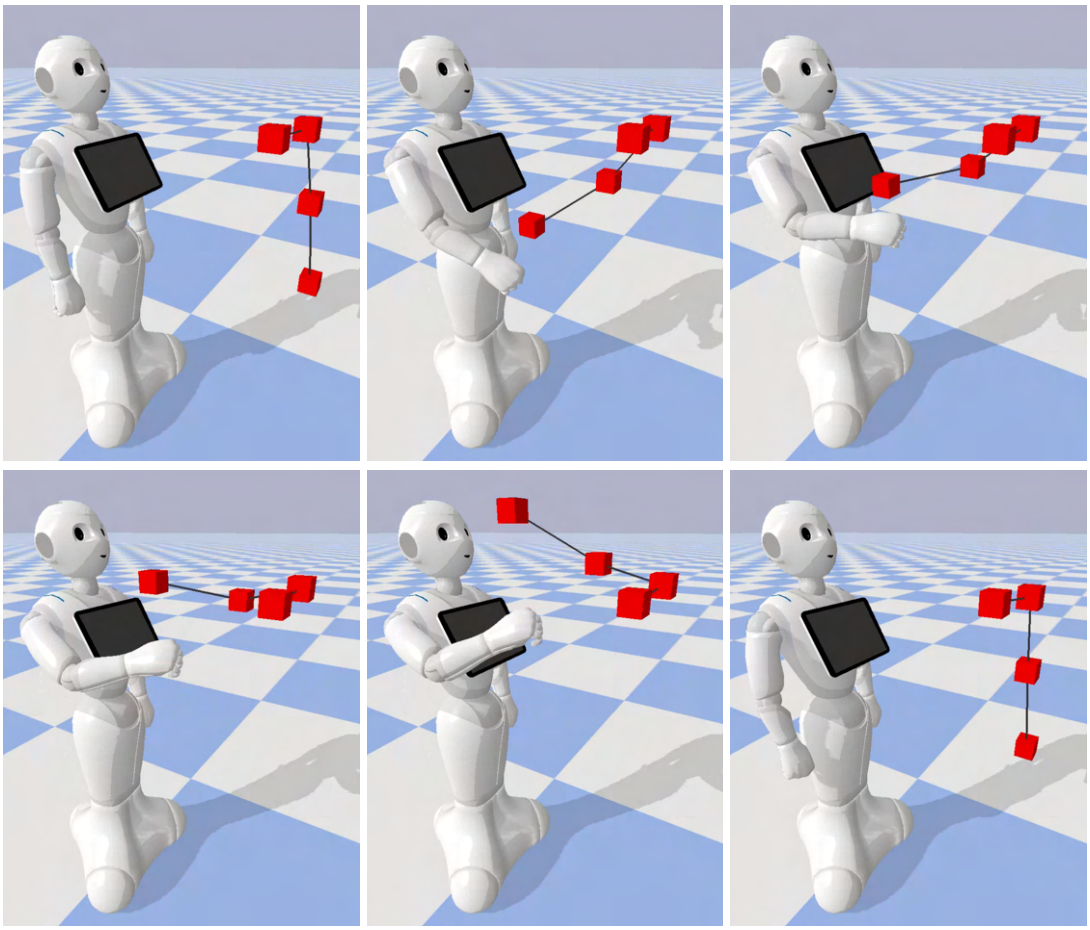


Figure 4.7.: This image sequence shows an example of a rocket fist bump in simulation using the approach proposed in this thesis. The joint trajectories of the Pepper robot are generated by conditioning the corresponding HSMMs with 3D coordinates of the right hand's wrist joint of a human. The red boxes represent the entire right arm (including the shoulder) of the human, which is placed a bit lower than the average body height of a human to compensate for the small size of Pepper.
It can be seen, that the robot is able to react and adapt to the human's motion.

4.4. Performance on a real Robot

For evaluating the performance of the proposed framework on a real robot, we were conducting a user study to see how the approach is perceived by different users, which is a key component of Human-Robot Interaction (HRI) algorithms. To have a baseline for us and the participants to compare against, we additionally introduced a rigid non-interactive and non-adaptive hard-coded algorithm.
As described in section 4.1.2, we were facing issues with the dataset proposed by Bütepage et al. [29] when executing our approach in a real scenario. Hence we decided to use a dataset collected by ourselves using the same Intel Realsense D435[9] RGB-D camera [37] that we used for observing the users. The use of a different dataset also requires different hyper-parameters to be considered optimal. For that, we decided on the ones that qualitatively worked the best, by testing the interactions themselves.

The remainder of this section is structured as follows. Section 4.4.1 defines the implementational details and the general setup of the user study. Section 4.4.2 then explains the structure of the study, as well as evaluates the results.

4.4.1. Setup

For implementation, the python version of the PbDlib[10] [13] and the dataset proposed in Section 4.1.2 is used. The HSMMs for the proposed approach are initialized and trained with six Multivariate Normal Distributions (MVNs) for each interaction, according to Section 3.2.1.
For the skeleton tracking, to track the human partner, we were using Nuitrack[11] running with an Intel Realsense D435 RGB-D camera [37]. The external calibration between the robot and the camera is done manually. The robot used is a Pepper[12] robot [36] from Aldebaran Robotics, which is a 120cm tall Humanoid robot (Figure 4.6a). It has six Degrees of Freedom (DoFs) in each arm (three for the shoulder, and one for the elbow, wrist, and hand closure), of which we control the shoulder and the elbow joints using ROS[13] [39]. Translating the human motions into robot DoFs works the same as for

[9] https://www.intelrealsense.com/depth-camera-d435/
[10] https://gitlab.idiap.ch/rli/pbdlib-python
[11] https://nuitrack.com/
[12] https://www.aldebaran.com/en/pepper
[13] https://github.com/ros-naoqi

Figure 4.8.: This picture shows the whole setup of the user story for interacting with the Pepper robot from the perspective of the participants, taken from the point of view of the participants when they first enter the setting. The camera on the right is for tracking the skeleton of the human, which is visualized on the monitor on the table behind the robot. The camera on the left is for recording the video of the interaction.

training the models described in Section 3.2.1, by adapting the work in [35] following the conventions of the Pepper robot[14].

4.4.2. User Study

For evaluation, we conduct a user study where participants interact with the robot controlled by two different algorithms, namely the approach proposed in this thesis, and a hard-coded interaction that goes to fixed joint angle goals over the duration of the interaction.

[14] http://doc.aldebaran.com/2-0/family/juliette_technical/joints_juliette.html

4.4.2.1. Methodology

The study followed a within subject design where participants interact with a Pepper robot controlled by each of the aforementioned algorithms twice in a randomized order, leading to four HRIs per participant. Each interaction was evaluated with seven different items, each rated on a 7-Point scale (ranging from 1: not at all, to 7: very much): pleasantness, naturalness/human-likeness, friendliness/comfortableness, how fun the interaction was, interactiveness, user satisfaction, and the level of connection felt with the robot. Moreover, at the end of each trial, we additionally asked the participants whether they would like to have the interaction again. Each participant had to either perform a handshake or a rocket fist bump, not both. Additionally, the focus of the study is on understanding how the different approaches are perceived by the users, therefore we do not focus on the difference between the groups or interaction types. Rather we evaluate how each participant perceived the algorithms that they interacted with.

4.4.2.2. Procedure

The study took place in a laboratory setting as shown in Figure 4.8. Participants were initially guided to a desk where they sat down to fill out and sign consent forms to take part in the study. After giving consent, they had to fill out a questionnaire that included demographic information, prior experiences with robots, their attitude towards humanoid robots, their attitude towards physical interactions, and some personality questions to gauge their extraversion [40]. After filling that out, participants were guided to a standing table where they were shown a training video of two humans performing an interaction (either a handshake or a fist bump) with one human leading the interaction and the other following the leader. Participants were then instructed that they had to perform the same interaction that they just saw with a robot four times in the role of the leader. They were then guided behind a barrier where they would see the robot for the first time. The sequence of each trial was as follows:

1. Before the start of each trial, the participants would see a video stream of the skeleton tracker so that they could see their tracked skeleton and position themselves accordingly in front of the robot, such that the skeleton tracking would not have a heavy occlusion or they would not be out of the frame of the camera.
2. Once the position was set and the tracking was stable, the experiment operator would signal them to start the trial.

3. The participant would then perform the interaction whereby the robot would respond according to the selected algorithm.
4. Apart from the hard-coded approach that always follows the same procedure, the trial would terminate when the participant goes back to a neutral position with their hands by their side.
5. After each trial, the participants were asked questions relating to their perception of the interaction on a 7-Point scale of pleasantness, naturalness/human-likeness, friendliness, interactiveness, how fun the interaction was, and how connected they felt to the robot. Further, they were asked if they would like to have the interaction again.

This process from adjusting the skeleton tracking to answering the questions was repeated four times, wherein the robot was controlled by each of the two aforementioned algorithms twice in random order. The participants were neither informed of the randomization nor the algorithm they were interacting with. Moreover, they did not even know that there were different algorithms.

4.4.2.3. Participant Sample

A total of 15 users participated in our study. We excluded the data of one participant who did not perform the interaction as was shown in the training video, which caused abnormal behaviors in the robot. Out of the 14 remaining participants, the mean age was 24.29 years with a standard deviation of 2.76, 3 were female and 11 were male. In terms of educational background, 7 participants were pursuing their Bachelor's degree, 4 were pursuing their Master's degree, 2 were pursuing their Ph.D., and 1 was pursuing a teaching degree. Participants had an overall low level of experience with robots on a scale from 1 (no experience at all) to 7 (a lot of experience) with a mean of 2.07 and a standard deviation of 2.02. Despite that, they had a relatively positive attitude towards humanoid robots on a scale from 1 (very negative) to 7 (very positive) with a mean of 5.36 and a standard deviation of 1.01. The participants had a neutral outlook towards physical interactions in general on a scale from 1 (distant) to 7 (open) with a mean of 4.86 and a standard deviation of 1.70. This statistic was also confirmed with the Big 5 extraversion scale [40] with a mean extraversion of 3.33 and standard deviation of 0.9 out of 5. 11 participants were from an engineering background and one each from pharmaceutical, medicinal, and teaching profession.

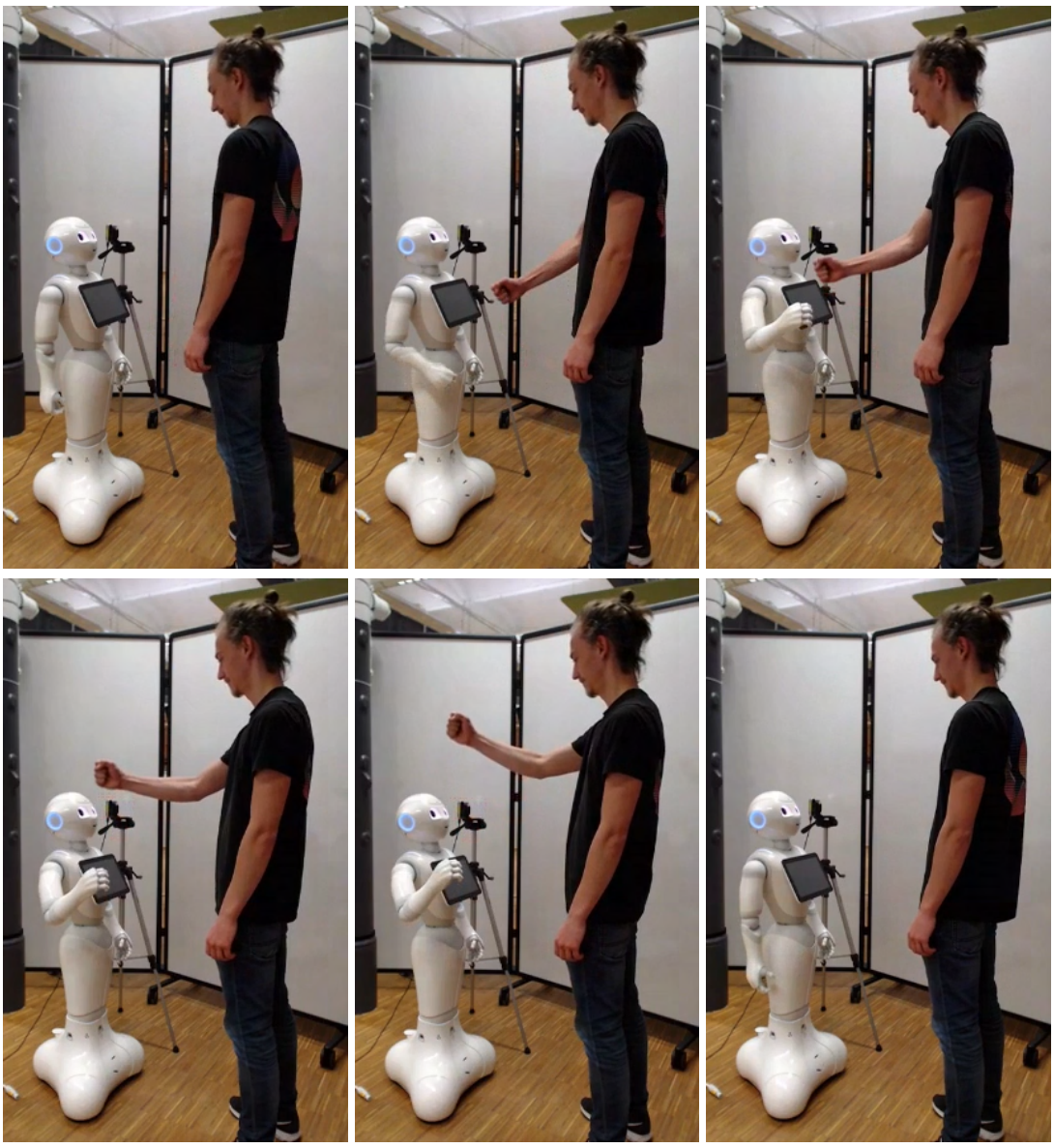


Figure 4.9.: This image sequence shows an example of a rocket fist bump in a real scenario using the approach proposed in this thesis. As it can be seen, the robot is able to react to the human's motion. However, compared to the results of the simulation shown in Figure 4.7 where the human's right arm was put lower than the one of an average sized human would be, it is further off from staying close to the human's hand.

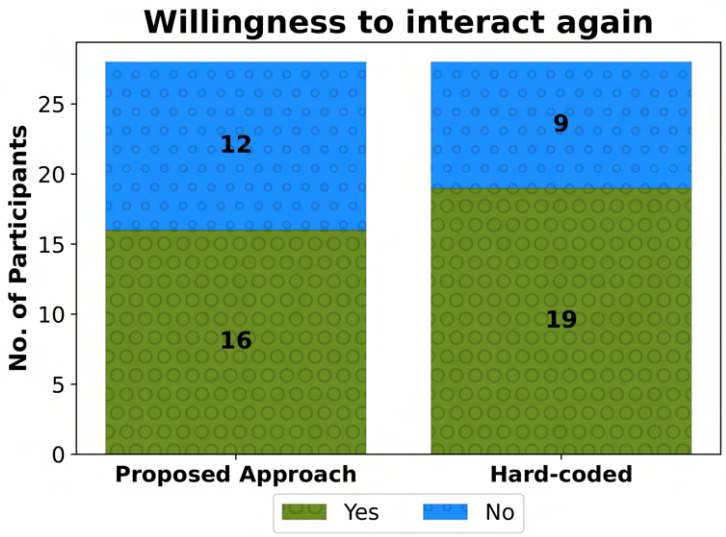


Figure 4.10.: This graphic illustrates the willingness of the participants to interact with the two different algorithms again. On the x-axis, the algorithms are plotted, and on the y-axis, the number of participants who replied yes (green) or no (blue) on the corresponding algorithm. It can be seen that both algorithms perform about the same.

4.4.2.4. Study Results

Starting with Figure 4.10, which displays the willingness of the participants to interact with each algorithm again, we can see that both algorithms are rated about the same and that more than half of the participants would like to interact with each algorithm again. However, with 16 out of 28[15] trials answered with a yes, the proposed approach is rated slightly worse than the hard-coded one.
This difference though increases when looking at the comparison items, where we take the mean of both the trials of each algorithm and compare them across both algorithms that the participants interacted with on the items in the survey, namely: pleasantness, naturalness/human-likeness, friendliness, interactiveness, how fun the interaction was and how connected they felt to the robot. We calculate the descriptive statistics (mean and standard deviation) to find the pairwise differences for each of the items, which are

[15] 14 participants $\times$ 2 trials of an algorithm

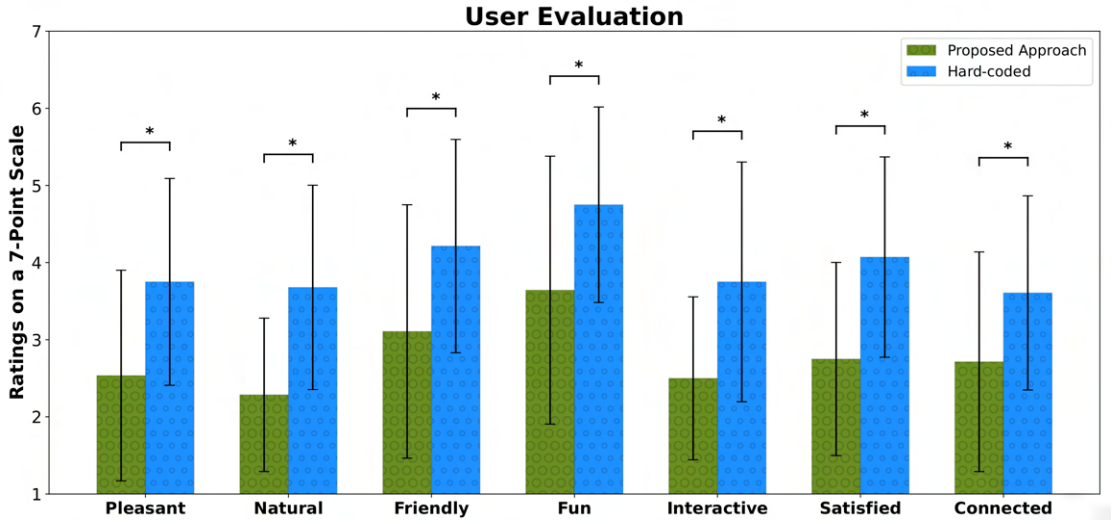


Figure 4.11.: This graphic provides a comparison of the two different approaches regarding the evaluation items, namely: pleasantness, naturalness/human-likeness, friendliness/comfortableness, how fun the interaction was, interactiveness, user satisfaction, and the level of connection felt with the robot. The y-axis displays the perception of the participants on a 7-Point scale. The colored bars illustrate the mean, and the black lines within the bar the standard deviation. An asterisk shows the two corresponding approaches differing significantly from each other according to the t-Tests p-Value (significantly different, if lower than 0.05). It can be seen that the proposed approach is throughout significantly worse than the hard-coded algorithms.

plotted in Figure 4.11. To compare the response that the different algorithms received, we ran a Repeated Measures ANOVA. It can be seen that the approach proposed in this thesis is throughout rated significantly worse than the hard-coded one. The fact of being significantly different arises from the p-Value of the t-Test being lower than 0.05, which is marked by an asterisk in the plot.
The reason for the proposed interactive framework performing worse than the hard-coded one could stem from two major facts. Firstly, it is the fact, that the executed joint motions of the robot are directly learned from the humans in the collected dataset. Since Pepper is fairly small (120cm) compared to an average human, it keeps its hands much lower than the human during an interaction. Hence, it does not enforce the contact-based nature of the interactions, resulting in the interactions not being perceived that well. In contrast,

for the hard-coded algorithm, the height of the interactions was adapted to be executed at a higher level. The small height of Pepper in general was even remarked on by some of the participants after the study. Secondly, the reason for the hard-coded algorithm performing better could stem from the fact that the trajectory is smoother than the one of the proposed approach due to the re-planning happening at every time step in that approach, resulting in the overall interaction appearing unnatural.
An example of a rocket fist bump interaction using the proposed framework is shown in Figure 4.9. It can be seen that the robot is indeed reacting to the human's motion. However, compared to the results of the simulation shown in Figure 4.7, where the human's right arm was put lower than the one of an average sized human would be, it is further off from staying close to the human's hand, due to the mentioned reasons.

5. Discussion

In this final chapter, in Section 5.1, we first summarize the idea and the results proposed in this thesis. In Section 5.2, we are finally discussing the shortcomings of this thesis and present potential future work.

5.1. Conclusion

In this thesis, we proposed a Human-Robot-Interaction (HRI) framework for segmenting and learning social interactions with the use of Hidden semi-Markov Models (HSMMs). Unlike most Learning from Demonstration (LfD) approaches that need kinesthetic teaching, our approach directly learns from Human-Human Interactions by making use of the similarity in the Degrees of Freedom (DoFs) between a human and a humanoid robot.
We discussed the importance of humanoid social robots appearing as natural and human-like as possible, which includes the ability to automatically react and adapt to the motion of the human interaction partner. HSMMs provide a great way for such kind of generalization by segmenting the demonstration trajectories and learning the joint distribution between both human actors for each segment. In combination with the learned sequencing of the segments, trajectories can be predicted by conditioning the HSMM on human observations. We show that our approach achieves great results in theory as well as in simulation. Nevertheless, a user study ($N = 14$) on a real robot reveals the importance of adapting to the user's hand position. Although the proposed framework correctly reacts to the human participant, it was rated worse than a hard-coded algorithm due to not being able to follow the actual hand position.

5.2. Outlook

The main shortcoming of the proposed framework is the lack of the robot Pepper being able to adapt to the actual hand position of human users in real scenarios, mainly due to the fact of its small size of 120cm. The most straightforward solution to that would be to just execute the framework on a robot with a taller size. Although it would be interesting to see how much of a difference that would make to the perception of the algorithm, this solution would not be the most desirable since we want the approach to be able to generalize to all kinds of humanoid robots. Hence, to overcome this issue in general, an adaption technique like Inverse Kinematics (IK) could be introduced, which aims to find an optimal joint angle configuration for the robot that reaches the desired goal position of the human's hand, while not straying too much from the robots initial joint configuration. The IK adaption would first require classifying the underlying Multivariate Normal Distributions (MVNs) of the Hidden semi-Markov Model (HSMM) into contact-based and none contact-based ones. Secondly, for the contact-based Gaussian's, the IK optimization problem would need to be solved [41]. To test this procedure, categorizing the Gaussian's could be done manually. In the long term, however, an automatic categorization would be desirable.
Another desirable feature would be to automatically determine the optimal number of Gaussian's for a HSMM. A solution to that could be the G-Means algorithms proposed by Lioutikov et al. [33]. However, applying this approach directly does not bring sufficient results, as we ascertained. Hence some adaptation would need to be done which exceeded the scope of this thesis.
Furthermore, in the user study, we selected the corresponding model by hand. Hence, it remains to evaluate the performance of having a set of models in combination with the action classifier, in real scenarios. Also, we only tested the classifier with the four different types of interactions from the dataset by Bütepage et al. [29]. It remains unknown how the classifier scales to the differentiation between more interaction types. A method to improve the classifier anyways could be by including a gesture recognition explicit for the hand, like implemented here[1]. With that, the differentiation between, for example fist bumps and handshakes or hand waves, could potentially be done much faster and easier, by determining whether the human's hand is opened or closed.
Combining the idea of automatically determining the number of Gaussian's for a HSMM and also automatically classifying them into contact-based and none contact-based ones could lead to an online learning framework, where known skills could be continuously optimized and new skills could be automatically learned.

[1] `https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe`

Bibliography

[1] D. Phutela, "The importance of non-verbal communication," *IUP Journal of Soft Skills*, vol. 9, no. 4, p. 43, 2015.

[2] M. Mori, K. F. MacDorman, and N. Kageki, "The uncanny valley [from the field]," *IEEE Robotics & automation magazine*, vol. 19, no. 2, pp. 98–100, 2012.

[3] K. M. Newell and A. B. Slifkin, "The nature of movement," *Motor behavior and human skill: A multidisciplinary approach*, p. 143, 1998.

[4] Sudar, "Non-verbal greeting of different cultures used in global society," 2018.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97, pp. 12–20, 1997.

[7] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*, pp. 1371–1394, Springer, 2008.

[8] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[9] S.-Z. Yu, "Hidden semi-markov models," *Artificial intelligence*, vol. 174, no. 2, pp. 215–243, 2010.

[10] K. P. Murphy, "Hidden semi-markov models (hsmms)," *unpublished notes*, vol. 2, 2002.

[11] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[12] C. Kohlschein, "An introduction to hidden markov models," in *Probability and Randomization in Computer Science. Seminar in winter semester*, vol. 2007, 2006.

[13] E. Pignat and S. Calinon, "Learning adaptive dressing assistance from human demonstration," *Robotics and Autonomous Systems*, vol. 93, pp. 61–75, 2017.

[14] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 02, pp. 183–202, 2008.

[15] S. Oshikawa, T. Nakamura, T. Nagai, M. Kaneko, K. Funakoshi, N. Iwahashi, and M. Nakano, "Interaction modeling based on segmenting two persons motions using coupled gp-hsmm," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 288–293, IEEE, 2018.

[16] B. Mor, S. Garhwal, and A. Kumar, "A systematic review of hidden markov models and their applications," in *Archives of Computational Methods in Engineering*, 2021.

[17] H. S. Koppula and A. Saxena, "Physically grounded spatio-temporal object affordances," in *European Conference on Computer Vision*, pp. 831–847, Springer, 2014.

[18] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[19] T. Nakamura, T. Nagai, D. Mochihashi, I. Kobayashi, H. Asoh, and M. Kaneko, "Segmenting continuous motions with hidden semi-markov models and gaussian processes," *Frontiers in neurorobotics*, vol. 11, p. 67, 2017.

[20] J. Oliveira, F. Renna, and M. Coimbra, "A subject-driven unsupervised hidden semi-markov model and gaussian mixture model for heart sound segmentation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 323–331, 2019.

[21] P. K. Pook and D. H. Ballard, "Recognizing teleoperated manipulations," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pp. 578–585, IEEE, 1993.

[22] J. D. Sweeney and R. Grupen, "A model of shared grasp affordances from demonstration," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pp. 27–35, IEEE, 2007.

[23] J. Chen and A. Zelinsky, "Programing by demonstration: Coping with suboptimal teaching actions," *The International Journal of Robotics Research*, vol. 22, no. 5, pp. 299–319, 2003.

[24] M. Ogino, H. Toichi, Y. Yoshikawa, and M. Asada, "Interaction rule learning with a human partner based on an imitation faculty with a simple visuo-motor mapping," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 414–418, 2006.

[25] V. Prasad, R. Stock-Homburg, and J. Peters, "Learning human-like hand reaching for human-robot handshaking," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3612–3618, IEEE, 2021.

[26] T. Shu, M. S. Ryoo, and S.-C. Zhu, "Learning social affordance for human-robot interaction," *arXiv preprint arXiv:1604.03692*, 2016.

[27] T. Shu, X. Gao, M. S. Ryoo, and S.-C. Zhu, "Learning social affordance grammar from videos: Transferring human interactions to human-robot interactions," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 1669–1676, IEEE, 2017.

[28] D. Vogt, S. Stepputtis, S. Grehl, B. Jung, and H. B. Amor, "A system for learning continuous human-robot interactions from human-human demonstrations," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2882–2889, IEEE, 2017.

[29] J. Bütepage, A. Ghadirzadeh, O. Öztimur Karadağ, M. Björkman, and D. Kragic, "Imitating by generating: Deep generative models for imitation of interactive tasks," *Frontiers in Robotics and AI*, vol. 7, 2020.

[30] B. Varadarajan, C. Reiley, H. Lin, S. Khudanpur, and G. Hager, "Data-derived models for segmentation with application to surgical assessment and training," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 426–434, Springer, 2009.

[31] A. Fod, M. J. Matarić, and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous robots*, vol. 12, no. 1, pp. 39–54, 2002.

[32] J. Feng-Shun Lin and D. Kulić, "Segmenting human motion for automated rehabilitation exercise analysis," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 2881–2884, 2012.

[33] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Probabilistic segmentation applied to an assembly task," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 533–540, 2015.

[34] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1595–1618, 2017.

[35] L. Fritsche, F. Unverzag, J. Peters, and R. Calandra, "First-person tele-operation of a humanoid robot," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 997–1002, IEEE, 2015.

[36] A. K. Pandey and R. Gelin, "A mass-produced sociable humanoid robot: Pepper: The first machine of its kind," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 40–48, 2018.

[37] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1–10, 2017.

[38] M. Busy and M. Caniot, "qibullet, a bullet-based simulator for the pepper and nao robots," *arXiv preprint arXiv:1909.00779*, 2019.

[39] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[40] B. Rammstedt and O. P. John, "Measuring personality in one minute or less: A 10-item short version of the big five inventory in english and german," *Journal of research in Personality*, vol. 41, no. 1, pp. 203–212, 2007.

[41] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 366–379, 2020.

[42] L. E. Baum, "Growth functions for trasformations on manifolds," *Pac. J. Math.*, vol. 27, no. 2, pp. 211–227, 1968.

Appendix

A. HMM/HSMM - Algorithms

This chapter provides a mathematical and detailed description of the techniques used to solve the three standard problems for HMMs and HSMMs defined in Section 2.1. The evaluation problem can be efficiently solved by the Forward-Backward algorithm. The Viterbi algorithm solves the decoding problem, and to solve the learning problem, the Baum-Welch algorithm is used.

For simplicity, and since it is the more general definition, the set of underlying distributions $\mathcal{D}$ is redefined by the observation matrix $\boldsymbol{B}$, which defines the probabilities of getting a certain emission in a certain state. Along with that, we are introducing two more definitions to better express the algorithms and formulas:

5. The probability $p(\boldsymbol{o}_i|s_j)$ for getting observation $\boldsymbol{o}_i$ in a state s_j is denoted as $b_{\boldsymbol{o}_i,s_j}$ with $b_{\boldsymbol{o}_i,s_j} \in \boldsymbol{B}$.
6. π_i defines the probability for state s_i being the initial state.

with definitions 1-4 being determined in Section 2.1.

A.1. Forward-Backward Algorithm

This algorithm solves the evaluation problem (Problem 1) by computing the probability of observing a (partial) observation sequence $\boldsymbol{o}_{1:t}$, given a model λ, i.e. $p(\boldsymbol{o}_{1:t}|\lambda)$. In other words, it computes how likely it is that an (partial) observation sequence $\boldsymbol{o}_{1:t}$ is produced by a given model λ, which mathematically denotes as $p(\boldsymbol{o}_{1:t}|\lambda)$. To be precise, to solve this problem, we only need the "Forward" part of the algorithm, as you will see in the following. Nevertheless, the "Backward" part will be used to solve the learning problem (Problem 3) and therefore will be introduced in this section as well.
However, since all the model parameters are known, a straightforward way of solving

the evaluation problem is by simply adding up all the probabilities of all combinations of sequences the model λ can take - denoted as $\mathcal{R}$ - in order to produce the observation sequence $\boldsymbol{o}_{1:t}$. For a HMM that can be defined as

$$p(\boldsymbol{o}_{1:t}|\lambda) = \sum^{|\mathcal{R}|} b_{\boldsymbol{o}_1,s_1} \prod_{t'=2}^{t} a_{s_{t'-1},s_{t'}} b_{\boldsymbol{o}_{t'},s_{t'}} \tag{A.1}$$

As this procedure is computationally infeasible, the actual Forward-Backward algorithm leverages the computational overhead by working recursively with the use of a so-called forward variable $\alpha_i(\boldsymbol{o}_t)$. It describes the probability of observing the (partial) observation sequence $\boldsymbol{o}_{1:t}$ and ending up in state s_i at time step t, given a model λ (Equation (A.2)).

$$\begin{aligned} \alpha_i(\boldsymbol{o}_t) &\triangleq p(s_i, \boldsymbol{o}_{1:t}|\lambda) \\ &= b_{\boldsymbol{o}_t,s_i} \sum_{k=1}^{|\mathcal{S}|} \alpha_k(\boldsymbol{o}_{t-1}) a_{k,i} \end{aligned} \tag{A.2}$$

$$\text{with } \alpha_i(\boldsymbol{o}_1) = \pi_i$$

It recursively goes through the observation sequence $\boldsymbol{o}_{1:t}$, and for each observation within the sequence $\boldsymbol{o}_{1:t}$ it adds up the forward variable for each state $s_i \in \mathcal{S}$ at time step t.
In the case of HSMMs, the distribution $p_i(d)$ that is fitted over the number of steps $d \in \{1, 2, ..., D\}$ that the model stays in a given state s_i needs to be added, which results in the following Equation (A.3)

$$\begin{aligned} \alpha_i(\boldsymbol{o}_t) &\triangleq p(s_i, \boldsymbol{o}_{1:t}|\lambda) \\ &= b_{\boldsymbol{o}_t,s_i} \sum_{k=1}^{|\mathcal{S}|} \sum_{d=1}^{|\mathcal{D}|} \alpha_k(\boldsymbol{o}_{t-1}) a_{k,i} p_i(d) \end{aligned} \tag{A.3}$$

$$\text{with } \alpha_i(\boldsymbol{o}_1) = \pi_i$$

Now, in order to gain the desired probability $p(\boldsymbol{o}_{1:t}|\lambda)$ that the evaluation problem asks for, we have to once again take sum over the forward variables $\alpha_i(\boldsymbol{o}_t)$ for all states in $\mathcal{S}$ (Equation (A.4))

$$p(\boldsymbol{o}_{1:t}|\lambda) = \sum_{k=1}^{|\mathcal{S}|} \alpha_k(\boldsymbol{o}_t) \tag{A.4}$$

As stated before, the evaluation problem is already solved by only using the “Forward” part of the Forward-Backward algorithm. Nonetheless, the “Backward” part will still be introduced in the following, as it is used to solve the learning problem (Problem 3).

The "Backward" part introduces a backward variable $\beta_i(\boldsymbol{o}_t)$ that is defined as the probability of observing an (partial) observation sequence $\boldsymbol{o}_{t:T}$, starting in a given a state s_i at time step t and given a model λ. In other words, it computes the probability of an observation sequence $\boldsymbol{o}_{t:T}$ that will occur when starting in a given state s_i at time step t, which again can be computed recursively, but instead of going forward in time, as for the forward variable, it works the other way around by going backward in time. The following Equation (A.5) defines this procedure for HMMs.

$$\begin{aligned}\beta_i(\boldsymbol{o}_t) &\triangleq p(\boldsymbol{o}_{1:T}|s_i, \lambda) \\ &= \sum_{k=1}^{|\mathcal{S}|} a_{i,k} b_{\boldsymbol{o}_{t+1}, s_k} \beta_k(\boldsymbol{o}_{t+1}) \\ &\text{with } \beta_i(\boldsymbol{o}_T) = 1\end{aligned} \tag{A.5}$$

In the case of HSMMs, again the distribution $p_i(d)$ that is fitted over the number of steps $d \in \{1, 2, ..., D\}$ that the model stays in a given state s_i needs to be added. The following Equation (A.6)

$$\begin{aligned}\beta_i(\boldsymbol{o}_t) &\triangleq p(\boldsymbol{o}_{1:T}|s_i, \lambda) \\ &= \sum_{k=1}^{|\mathcal{S}|} \sum_{d=1}^{|\mathcal{D}|} a_{i,k} b_{\boldsymbol{o}_{t+1}, s_k} \beta_k(\boldsymbol{o}_{t+1}) p_i(d) \\ &\text{with } \beta_i(\boldsymbol{o}_T) = 1\end{aligned} \tag{A.6}$$

shows the final result.

A.2. Viterbi Algorithm

This algorithm solves the decoding problem (Problem 2) by computing the most probable sequence of states $s_{1:t}$ emitting an observation sequence $\boldsymbol{o}_{1:t}$, given a model λ, i.e. $p(s_{1:t}, \boldsymbol{o}_{1:t}|\lambda)$. In other words, giving an observation sequence $\boldsymbol{o}_{1:t}$ and a model λ, it computes which sequence of states $s_{1:t}$ best "explains" the given observation sequence $\boldsymbol{o}_{1:t}$.
The solution to the Viterbi algorithm is kinda similar to the "Forward" step of the Forward-Backward algorithm (Equation (A.2) and (A.3) in Section A.1). Comparable to the forward

variable $\alpha_i(\boldsymbol{o}_t)$, it introduces a recursively defined variable $\delta_i(\boldsymbol{o}_t)$ which describes the state sequence $s_{1:t}$ with the highest probability that accounts for the first t observations and ends at a given state s_i. Simply spoken, it is doing so by replacing the summing procedure in the definition of $\alpha_i(\boldsymbol{o}_t)$ by a maximization procedure and storing the resulting state in an extra variable $\psi_i(\boldsymbol{o}_t)$. To be precise, $\psi_i(\boldsymbol{o}_t)$ describes the state s_j that lead to state s_i and the observation $\boldsymbol{o}_t$. By replacing the summing procedure with the maximization procedure, the algorithm is no longer considering all the states leading to the observation $\boldsymbol{o}_t$, but rather only considering the most likely one, which is mathematically defined in Equation (A.7) for regular HMMs.

$$\begin{aligned} \delta_i(\boldsymbol{o}_t) &\triangleq \max_{s_{1:t-1}} p(s_{1:t}, \boldsymbol{o}_{1:t}|\lambda) \\ &= b_{\boldsymbol{o}_t,s_i} \max_{n\in\mathcal{S}} \left[\delta_i(\boldsymbol{o}_{t-1})a_{n,i}\right] \\ \psi_i(\boldsymbol{o}_t) &= \arg\max_{n\in\mathcal{S}} \left[\delta_i(\boldsymbol{o}_{t-1})a_{n,i}\right] \\ &\text{with } \delta_i(\boldsymbol{o}_1) = \pi_i b_{\boldsymbol{o}_1,s_1} \text{ and } \psi_i(\boldsymbol{o}_1) = 0 \end{aligned} \tag{A.7}$$

with $\delta_i(\boldsymbol{o}_1) = \pi_i b_{\boldsymbol{o}_1,s_1}$ and $\psi_i(\boldsymbol{o}_1) = 0$. For HSMMs, again the distribution $p_i(d)$ that is fitted over the number of steps $d \in \{1, 2, ..., D\}$ that the model stays in a given state s_i needs to be added. The following Equation (A.8)

$$\begin{aligned} \delta_i(\boldsymbol{o}_t) &\triangleq \max_{s_{1:t-1}} p(s_{1:t}, \boldsymbol{o}_{1:t}|\lambda) \\ &= b_{\boldsymbol{o}_t,s_i} \max_{n\in\mathcal{S}} \left\{ \max_{d\in\mathcal{D}} \left[\delta_i(\boldsymbol{o}_{t-1})a_{n,i}p_i(d)\right] \right\} \\ \psi_i(\boldsymbol{o}_t) &= \arg\max_{n\in\mathcal{S}} \left[\delta_i(\boldsymbol{o}_{t-1})a_{n,i}\right] \\ &\text{with } \delta_i(\boldsymbol{o}_1) = \pi_i b_{\boldsymbol{o}_1,s_1} \text{ and } \psi_i(\boldsymbol{o}_1) = 0 \end{aligned} \tag{A.8}$$

shows the final results.

A.3. Baum-Welch Algorithm

The Baum-Welch algorithm is a special case of the Expectation-Maximization (EM) algorithm that solves the learning problem (Problem 3). The goal is to learn and optimize

the model parameters of a given model λ in a way that it maximizes the probability of observing a given observation sequence (or a set of observation sequences) $\boldsymbol{o}_{1:t}$, i.e. maximizing $p(\boldsymbol{o}_{1:t}|\lambda)$. It is doing so by iteratively re-estimating the parameters of the model λ and then maximizing the expectation of the model by re-calculating the parameters of the model. This procedure continues in a loop until a local optimum or a specified convergence threshold is reached. The parameters that will be changed are the stochastic matrix $\boldsymbol{A}$, the observation matrix $\boldsymbol{B}$, and the initial state distribution $\boldsymbol{\pi}$. The set of all possible states and the set of all possible observations will stay the same. The notation for the updated model will be $\overline{\lambda} = (\mathcal{S}, \mathcal{O}, \overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{\pi}})$.
In addition to the $\alpha_i(\boldsymbol{o}_t)$ and the $\beta_i(\boldsymbol{o}_t)$ variables defined for the Forward-Backward algorithm in Equations (A.2), (A.3) and (A.5), (A.6) of Section A.1, two more variables $\gamma_i(\boldsymbol{o}_t)$ and $\xi_{i,j}(\boldsymbol{o}_t)$ need to be defined in order to make solving the problem easier.
$\gamma_i(\boldsymbol{o}_t)$ can be thought of as the fusion of the $\alpha_i(\boldsymbol{o}_t)$ and the $\beta_i(\boldsymbol{o}_t)$ variables. It describes the probability of being in state s_i at time step t, given a whole observation sequence $\boldsymbol{o}_{1:T}$ and a model λ. This probability can completely be defined with the use of the $\alpha_i(\boldsymbol{o}_t)$ and the $\beta_i(\boldsymbol{o}_t)$ variables (Equation (A.9)).

$$\begin{aligned}\gamma_i(\boldsymbol{o}_t) &\triangleq p(s_i|\boldsymbol{o}_{1:T}, \lambda) \\ &= \frac{\alpha_i(\boldsymbol{o}_t)\beta_i(\boldsymbol{o}_t)}{p(\boldsymbol{o}_{1:T}|\lambda)} \\ &= \frac{\alpha_i(\boldsymbol{o}_t)\beta_i(\boldsymbol{o}_t)}{\sum_{k=1}^{|\mathcal{S}|} \alpha_k(\boldsymbol{o}_t)\beta_k(\boldsymbol{o}_t)}\end{aligned} \tag{A.9}$$

The denominator in this Equation (A.9) is used to normalize the likelihood value to obtain a probability value between 0 and 1.
$\xi_{i,j}(\boldsymbol{o}_t)$ describes the probability of being in state s_i at time step t and in state s_j at time step $t+1$, given a whole observation sequence $\boldsymbol{o}_{1:T}$ and a model λ. In other words, it asks for the probability of transitioning from s_i to s_j at time t. With that it is kind of similar to the variable $\gamma_i(\boldsymbol{o}_t)$, but considering two states and the transition probability $a_{i,j}$ between them as well as the observation probability $b_{s_j,\boldsymbol{o}_{t+1}}$, which again can be defined using the

$\alpha_i(\boldsymbol{o}_t)$ and the $\beta_i(\boldsymbol{o}_t)$ variables as shown in the following Equation (A.10)

$$\begin{aligned}\xi_{i,j}(\boldsymbol{o}_t) &\triangleq p(s_i, s_j|\boldsymbol{o}_{1:T}, \lambda) \\ &= \frac{\alpha_i(\boldsymbol{o}_t)a_{i,j}b_{s_j,\boldsymbol{o}_{t+1}}\beta_j(\boldsymbol{o}_{t+1})}{p(\boldsymbol{o}_{1:T}|\lambda)} \\ &= \frac{\alpha_i(\boldsymbol{o}_t)a_{i,j}b_{s_j,\boldsymbol{o}_{t+1}}\beta_j(\boldsymbol{o}_{t+1})}{\sum_{k=1}^{|\mathcal{S}|}\sum_{l=1}^{|\mathcal{S}|}\alpha_k(\boldsymbol{o}_t)a_{k,l}b_{s_l,\boldsymbol{o}_{t+1}}\beta_l(\boldsymbol{o}_{t+1})}\end{aligned} \tag{A.10}$$

where the denominator again is used for normalizing to a probability value between 0 and 1.
Having defined the variables $\gamma_i(\boldsymbol{o}_t)$ and $\xi_{i,j}(\boldsymbol{o}_t)$, two more useful properties can be extracted. First, if we take $\gamma_i(\boldsymbol{o}_t)$ and sum over all t, we get a quantity that can be seen as the expected number of times s_i is ever visited. Second, if we take $\xi_{i,j}(\boldsymbol{o}_t)$ and sum over all t, we get a quantity that can be seen as the expected number of times s_i ever transitions to s_j, which denotes as

$$\sum_{t=1}^{T-1}\gamma_i(\boldsymbol{o}_t) = \text{expected number of transitions from } s_i \tag{A.11}$$

$$\sum_{t=1}^{T-1}\xi_{i,j}(\boldsymbol{o}_t) = \text{expected number of transitions from } s_i \text{ to } s_j \tag{A.12}$$

Finally, having all these preliminaries defined, we can move on to describe how to use these to improve a model λ given an observation sequence $\boldsymbol{o}_{1:T}$. As three different parameters need to be optimized ($\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}$), we are going to look at each of them individually. Starting

with the most intuitive one $\boldsymbol{\pi}$.

$$\begin{aligned}\overline{\pi}_i &= \text{expected frequency in state } s_i \text{ at time } (t=1) \\ &= \gamma_i(\boldsymbol{o}_1)\end{aligned} \tag{A.13}$$

$$\begin{aligned}\overline{a}_{i,j} &= \frac{\text{expected number of transitions from } s_i \text{ to } s_j}{\text{expected number of transitions from } s_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_{i,j}(\boldsymbol{o}_t)}{\sum_{t=1}^{T-1} \gamma_i(\boldsymbol{o}_t)}\end{aligned} \tag{A.14}$$

$$\begin{aligned}\overline{b}_{s_j,\boldsymbol{o}_k} &= \frac{\text{expected number of times in state } j \text{ and observing a particular emission } v_k}{\text{expected number of times in state } j} \\ &= \frac{\sum_{t=1 \text{ s.t. } \boldsymbol{o}_t=v_k}^{T} \gamma_j(\boldsymbol{o}_t)}{\sum_{t=1}^{T} \gamma_j(\boldsymbol{o}_t)}\end{aligned} \tag{A.15}$$

With that the updated model $\overline{\lambda} = (\mathcal{S}, \mathcal{O}, \overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{\pi}})$ is defined. Now we can iteratively use this new updated model $\overline{\lambda}$ to again compute the variables $\alpha_i(\boldsymbol{o}_t), \beta_i(\boldsymbol{o}_t), \gamma_i(\boldsymbol{o}_t)$ and $\xi_{i,j}(\boldsymbol{o}_t)$, which again can be used to update the model $\overline{\lambda}$ and so on until convergence. The convergence of this procedure to a local optimum is proven by Baum et al. [42].

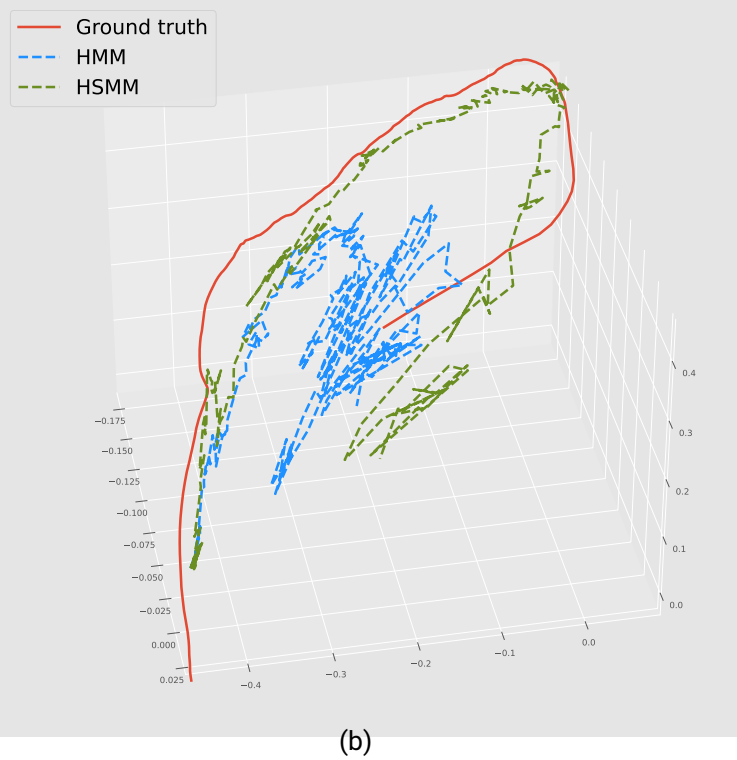
B. HSMM vs. HMM

This Appendix refers to Section 4.2.2, where the framework proposed in this thesis is compared to other methods. In particular, we are examining the reason for the Hidden semi-Markov Model (HSMM) and the Hidden Markov Model (HMM) methods performing about the same.
The reason we decided on using HSMM instead of HMM is due to their better temporal encoding, as described in Section 2.1. The obvious conclusion of that would be that HSMMs are performing better than HMMs.

In the scenarios of this thesis, we are learning Human-Human-Interactions (HHI) by representing not only the joint distributions of each agent but also the dependencies between the time series of both actors. This dependency is very strong in the sense that, while reconstruction (Section 3.2.3), the state sequence of the following actor (i.e. the robot), along with its timing, can be precisely determined by the input sequence of the leading actor. In other words, the timing of a gesture does not need to be exactly known by the robot since it is given by the leading actor. Hence, the robot only needs to synchronize, and the advantage of the HSMM becomes irrelevant.

To support this statement, we constructed a scenario where the timing should be more precisely known, which, for example, is the case when the robot should follow a complex sequence with precise timing, while the leading actor does not move. Such a behavior can be simulated when training and conditioning the models with pure random Gaussian noise as the input data for the leading actor (i.e. the human).
Figure B.1 shows the results exemplary for the rocket fist bump, where Subplot B.1a illustrates the average MSE of 100 runs on all 14 test trajectories, along with the standard deviation and the minimum and maximum values. For each run, the models are trained on a random batch of 15 demonstrations. It can be seen that the HSMM clearly outperforms the HMM.
The much better performance of the HSMM in such a scenario, which is more dependent on the encoding of precise timing, can also be seen in Subplot B.1b, where the predictions

of both models are plotted against the ground truth. The HSMM, as well as the HMM, are trained on the same data.



(b)

Figure B.1.: This figure shows the performance difference between HSMMs and HMMs trained and conditioned on Gaussian noise, based on the rocket fist bump interaction. The figure to the left (a) displays the average MSE of 100 runs on all 14 test trajectories. For each run, the models are trained on a random batch of 15 demonstrations. The black circles illustrate the mean of all 100 runs and the asterisk on top of the HMM bar denotes the sample being significantly different from the HSMM one, according to the t-Test (i.e. p-Value being lower than 0.05). The wider part of the bar displays the standard deviation and the caps at each end the minimum and the maximum value. The figure to the right (b) illustrates an exemplary prediction of the HSMM and the HMM along with the ground truth. In general, it can be seen that the HSMM outperforms the HMM in such a scenario, which is more dependent on the encoding of precise timing.