Modeling Human-Robot Interaction with Probabilistic Movement Representations

Modellierung von Mensch-Roboter Interaktion mit Probabilistischen Bewegungsrepräsentationen

Master-Thesis von Marco Antônio Sousa Ewerton aus São Luís - MA, Brasilien

Dezember 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Elektrotechnik und Informationstechnik
Schwerpunkt Datentechnik

Modeling Human-Robot Interaction with Probabilistic Movement Representations
Modellierung von Mensch-Roboter Interaktion mit Probabilistischen Bewegungsrepräsentationen

Vorgelegte Master-Thesis von Marco Antônio Sousa Ewerton aus São Luís - MA, Brasilien

Direkte Betreuer: Prof. Dr. Gerhard Neumann, Ph.D. Guilherme Jorge Maeda
1. Gutachten: Prof. Dr. Jan Peters
2. Gutachten: Prof. Dr. Klaus Hofmann

Tag der Einreichung:

Erklärung zur Master-Thesis

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

Darmstadt, den 18ten Dezember, 2014

(M. Ewerton)

Abstract

Robots that can interact with humans represent a potential benefit to society. They may, among many other applications, help increasing the quality of life of motor impaired people. However, in order to assist humans in their daily lives, robots must be able to deal with dynamic environments and uncertainties in how they perceive and interact with the world. Those are still challenging and open problems in robotics. Humans can in general deal with changing environments and uncertainty much better than robots. For this reason, it is useful if robots can learn from human demonstrations. Probability theory offers effective tools to design and operate models that capture the uncertainty in the environment, including the interaction with a human partner. Therefore it is reasonable to model human-robot interactions with probabilistic movement representations.

A recently introduced method to learn probabilistic interaction models from demonstrations is the framework of Interaction Primitives. Primitives are representations of elementary movements, usually in the form of trajectories. Primitives are essential for robot movement generation as they provide the building blocks from which more complex skills can be achieved. Interaction Primitives are elementary representations of the movements that two or more agents execute to achieve a task in a collaborative manner. So far the framework of Interaction Primitives has been limited to representing a single interaction pattern, i.e., an interaction that represents a single collaborative task. As it will be discussed in this work, a single interaction pattern can be modeled by a single multivariate Gaussian distribution and assumes linear correlation between interacting agents.

In practice, however, interactions between a human and a robot can consist of many different patterns, allowing for large variability in shape and duration of trajectories. Multiple interaction patterns also mean nonlinear correlation between interacting agents. To overcome the limitations of the conventional Interaction Primitives, this work proposes a Mixture of Interaction Primitives to learn multiple interaction patterns from unlabeled demonstrations. Specifically the proposed method uses Gaussian Mixture Models of Interaction Primitives to model nonlinear correlations between the movements of the different agents. The proposed algorithm is validated with two experiments involving interactive tasks between a human and a lightweight robotic arm. In the first, the proposed method is compared with conventional Interaction Primitives in a toy problem scenario where the robot and the human are not linearly correlated. In the second, a proof-of-concept experiment is presented, where the robot assists a human in assembling a box.

One of the aspects of the uncertainty in the interaction with humans is related to the fact that the execution of a human movement can have different velocity profiles. These changes occur not only from movement to movement, for example, by reaching a tool on a table multiple times at different speeds, but also within the same movement, for example, by moving the arm fast while far but decelerating when carefully grasping a glass full of water. Those changes and uncertainties in the velocity profiles must be addressed as they are essential to recognize the phase in which a human is executing the task. An algorithm to estimate online the phase, i.e., the stage of execution of a trajectory is also proposed. Online phase estimation allows the prediction of trajectories with different durations, as it is experimentally demonstrated in this work. It is reasonable to assume that this algorithm may play a important role in a future work on Human-Robot Interaction, since the necessary operations in the framework of Interaction Primitives for prediction of trajectories and inference of reactions are the same.

As a final contribution, this work proposes a method to encode human trajectories, even when the different demonstrations are only partially provided. This is a problem that occurs frequently due to occlusions, noise or lack of sensor coverage during recordings of the demonstrations. Finally, future work where the proposed algorithm can be potentially extended to deal with complex relations between time and phase is discussed.

Acknowledgements

I would like to thank Jan Peters not only for his valuable research advices but also for advising me on how to improve and acquire the skills of writing papers, making better presentations and becoming a better researcher. I also would like to thank Klaus Hofmann for kindly accepting me as his Master's student, evaluating my work and being such an accessible professor.

I have had two amazing advisors: Guilherme Jorge Maeda and Gerhard Neumann. Guilherme has always been available, advising me throughout my masters and during the writing of papers and this thesis. Together, we spent many hours working on the setup for our experiments with the real robot. He has provided me with numerous useful advices and insights. Geri was the source of many of the fundamental ideas behind this work, he was always available to discuss whenever I had problems with my implementations and derivations.

Rudolf Lioutikov also deserves a special thanks. His work was fundamental in order to perform experiments with the real robot.

I am indebted to Heni Ben Amor, who hired me almost three years ago as a research assistant, giving me the opportunity to start working in robotics. He also took part in discussions along this work, giving valuable ideas and feedback.

I am thankful to my parents and to my brothers for their love, support and encouragement.

Finally, I sincerely thank my fiancée, Marie, for being so kind to me and for even taking some of my responsibilities during the most time-consuming phases along this work.

Contents

1 Introduction

Robots that can assist humans in the industry, in the household, in hospitals, etc. can be of great benefit to society. They could help people with movement impairment in their daily lives; help moving heavy objects around; help executing repetitive or physically demanding tasks in the same work environment of the human, while avoiding any dangerous situations, etc. Also intelligent prosthetics can be seen as robots that assist humans.

The variety of tasks in which a human may need assistance is, however, practically unlimited. Thus, it is very hard (if not impossible) to program a robot in the traditional way, devising a fixed set of rules, to assist humans in scenarios that have not been exactly prespecified.

In order to assist different people in different situations, the robot should be able to adapt. Humans can in general adapt much better to new circumstances than current robots can. Therefore, in the face of situations to which the robot still cannot react appropriately, it would be useful for the robot to be able to learn from human demonstrations.

Learning from human demonstrations does not mean only repeating the movements that have been shown to the robot. The robot must infer from a reasonably small set of demonstrations how to react to situations that are not quite the same as the trained ones. For example, if it was demonstrated to the robot how to hand over an object at two different positions, the robot should be able to hand over an object also at a position between the demonstrated ones. Probability theory offers effective tools to make this sort of inferences, from a limited set of observations and under uncertainty.

Based on the ideas of learning from demonstration and of using probability theory, Interaction Primitive (IP) is a framework that has been recently proposed to alleviate the problem of programming a robot for physical collaboration and assistive tasks [1, 16]. At the core, IPs are primitives that capture the correlation between the movements of two agents—usually a human and a robot. Then, by observing one of the agents, say the human, it is possible to infer the controls for the robot such that collaboration can be achieved.

A main limitation of IPs is the assumption that the movements of the human and the movements of the robot assistant are linearly correlated. This assumption is reflected in the underlying Gaussian distribution that is used to model the demonstrations. While this assumption holds for tasks that cover a small region of the workspace (a high-five task in [1] or handover of objects in [16]), it limits the use of IPs in two aspects. First, as illustrated in Fig. 1, a task such as the assembly of a toolbox consists of several interaction patterns, i.e., interactions that differ significantly from each other in shape and duration of trajectories, and therefore can not be captured by a single Gaussian.

Moreover, even within a relatively small region of the workspace, the correlation between the two agents may not be linear, for example, if the movements of the human are measured in the Cartesian space, while the movements of the robot are measured in joint space (joint angles in the case of revolute joints).

Manually labeling each subtask (e.g. “plate handover”, “screw handover”, “holding screw driver”) is a way to model interactions with multiple subtasks. Ideally, however, robots should be able to identify different subtasks by themselves. Moreover, it may not be clear to a human how to separate a number of demonstrated interactions in different, linearly correlated groups. Thus, a method to learn multiple interaction patterns from unlabeled demonstrations is necessary. The first main contribution of this work is the development of such a method. In particular, this work uses Gaussian Mixture Models (GMMs) to create a Mixture of Interaction Probabilistic Movement Primitives (Mixture of Interaction ProMPs) [16].

In order to interact successfully with humans, a robot must also be able to react to actions executed in different speeds. The robot should infer from a partial observation of the action of the human partner if this action requires a fast or a slow reaction for example. The second main contribution of this work is the design of a method to estimate the phase of trajectories online. In other words, this method estimates, at each time step of a trajectory under observation, at which stage of the whole trajectory the current observation is. This estimation allows for identifying actions with different durations and

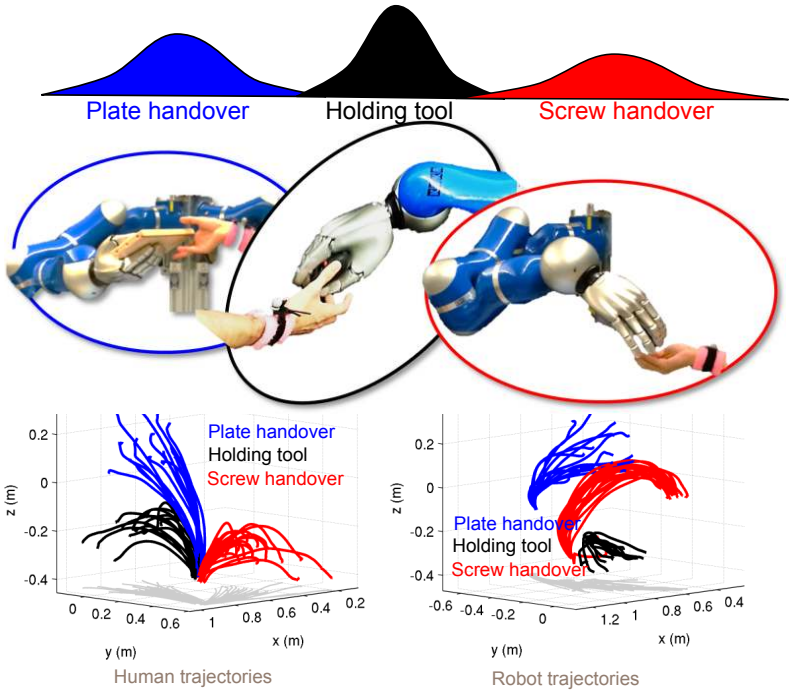


Figure 1: Illustration of a task consisting of multiple interaction patterns, where each can be represented as an Interaction Primitive. In this work, one of the main objectives is to learn multiple interaction patterns from an unlabeled data set of interaction trajectories.

answering questions such as whether the human has completed his/her action or not, when the action is expected to be completed, etc. Phase estimation may also play a role in switching from one interaction to another during execution.

A third main contribution of this work involves learning from demonstrations with missing data points. Occlusions and situations where the sensors capturing the positions along a trajectory cannot observe the whole trajectory occur quite frequently. To deal with those problems, this work also proposes an Expectation-Maximization (EM) approach to learn probabilistic distributions over trajectories from demonstrations with missing data points. Furthermore a possible extension of this algorithm to deal with complex relations between time and phase is discussed. As it will be discussed, this work uses a very simple formulation for the phase variable. Different formulations could lead to more accurate phase estimation in a future work.

In summary, the main contributions of this work are: the design of an algorithm that models interactions with a Mixture of Interaction Primitives, which can learn multiple interaction patterns from unlabeled demonstrations and can model nonlinear correlations between interacting agents; the design of an algorithm to estimate the phase of trajectories online, allowing for prediction of trajectories with different durations; the derivation and implementation of an Expectation-Maximization (EM) algorithm that allows for learning from training trajectories with missing data points.

The remainder of this thesis is organized as follows: Chapter 2 presents related work on Human-Robot Interaction and phase estimation. Chapter 3 explains the proposed approach to Human-Robot Interaction without yet treating the problem of phase estimation. This chapter corresponds to a work submitted to *IEEE International Conference on Robotics and Automation (ICRA)*, 2015. Section 3.1 briefly introduces Probabilistic Movement Primitives (ProMPs) and Section 3.2 provides an overview of Interaction ProMPs. Section 3.3 presents the first main contribution of this work: a Mixture of Interaction ProMPs based on Gaussian Mixture Models (GMMs). Afterwards, Section 3.4 evaluates the proposed method, first on a toy problem that is useful to clarify the characteristics of the method, and then on a practical application of a collaborative toolbox assembly. Chapter 4 deals with the problem of phase estimation, in principle left

aside in the previous chapter. More specifically, an approach to estimate online the phase of trajectories is explained in Section 4.1, followed by the explanation in Section 4.2 of an EM algorithm that can deal with incomplete training trajectories and will be extended in a future work to deal with more complex relations between time and phase than the one used in this work. Experiments validating those algorithms are described in Section 4.3. Chapter 5 summarizes the thesis and presents ideas for future work.

2 Related Work

This chapter discusses related work on Human-Robot Interaction and on estimating the time or the phase along a trajectory, which is also pertinent to interaction. A brief overview of the literature on those topics is provided and it is discussed how the work described in this thesis builds upon or differs from the cited works.

2.1 Human-Robot Interaction

Physical human-robot interaction poses the problem of both action recognition and movement control. Interaction dynamics need to be specified in a way that allows for robust reproduction of the collaborative task under different external disturbances, and a common approach is based on direct force sensing or emulation. Rozo et al. [20] propose a framework for haptic collaboration between a human and a robot manipulator. Given a set of kinesthetic demonstrations, their method learns a mapping between measured forces and the impedance parameters used for actuating the robot, e.g., the stiffness of virtual springs governing the collaborative task. In another force-based approach, Lawitzky et al. [13] propose learning physical assistance in a collaborative transportation task. In the early learning phase, the robot uses the measured force values to follow the human guidance during the task. Recorded force and motion patterns are then used to learn a Hidden Markov Model (HMM) that can predict the human's next action, and over time the robot learns to take over a more active role in the interaction. Kulvicius et al. [12] also address a transportation task where the two agents are modeled as two point particles coupled by a spring. The forces applied by the other agent tell the robot how to adapt its own trajectory.

The method presented in this work differs significantly from the previously cited works in the sense that it does not use nor emulate force signals, but instead learns the correlation between the trajectories of two agents. Correlating trajectories not only simplifies the problem in terms of hardware and planning/control but also allows for correlating multi-agent movements that do not generate force during the interaction, for example, the simple gesture of asking and receiving an object.

Graphical models have also been used to describe interaction dynamics. In the computer vision community, HMMs have been widely adopted to model interaction dynamics from input video streams [4, 5, 17]. As a result, graphical models have also gained considerable attention in the field of Human-Robot Interaction. The work described in [25] uses HMMs for human intention estimation in order to perform realistic haptic human-robot interaction. In [9], Hawkins and colleagues use a Bayes network to improve the fluency in a joint assembly task. The Bayes network learns to infer the current state of the interaction, as well as task constraints and the anticipated timing of human actions. Tanaka et al. [22] use a Markov model to predict the positions of a worker in an assembly line. Wang et al. [26] propose the Intention-Driven Dynamics Model (IDDM) as a probabilistic graphical model with observations, latent states and intentions where the transitions between latent states and the mapping from latent states to observations are modeled as Gaussian Processes. Koppula et al. [11] use a conditional random field with sub-activities, human poses, object affordances and object locations over time. Inference on the graphical model allows a robot to anticipate human activity and choose a corresponding, preprogrammed robot response. Lee et al. [14] learn a hierarchical HMM that triggers action primitives in response to observed behaviors of a human partner.

While very successful for classifying actions, graphical models, however, may not be the best option when it comes to generating motions. In [2], for example, the use of a HMM with discrete states, although very successful in action classification, introduces artifacts into the motion generation part that hinders motion generalization. Therefore, a clear problem in physical human-robot interaction is that while graphical models may be suitable in the action recognition domain, motion generation at the continuous level must also be taken into account. Llorens et al. [15] present a hybrid design for a robot to be used on the shoulder. In their work, Petri Nets accounts for discrete control transitions while at the motion level, Partial Least Squares Regression has been used to find the best action of the robot at future time steps.

Perhaps the principal distinction of the method proposed in this work is the use of Interaction Primitives (IPs), proposed by Ben Amor et al. [1], which were initially based on Dynamical Movement Primitives [10] and later have been extended to Probabilistic Movement Primitives [18] with action recognition in the work of Maeda et al. [16]. As shown in [16], Interaction Primitives can be used not only to recognize the action of an agent, but also to coordinate the actions of a collaborator at the movement level; thus overcoming in a single framework both layers of discrete action recognition and continuous movement control. Differently from [16], where different interaction patterns must be hand-labeled, one of the main contributions of this work is the unsupervised learning of a Mixture of Interaction Primitives.

2.2 Phase Estimation

A Dynamical Movement Primitive (DMP) [10] is a compact, time-dependent representation of a trajectory, comprising a proportional-derivative (PD) controller and a nonlinear forcing function. This forcing function corresponds to a weighted sum of time-dependent basis functions. The weights for the basis functions are learned for example through Linear Regression to approximate any smooth trajectory.

Probabilistic Movement Primitives (ProMPs) [18], which will be addressed in more detail in Section 3.1, are movement representations based on a probabilistic distribution over trajectories. In ProMPs, not a forcing function, but the trajectories themselves are approximated by a weighted sum of time-dependent basis functions.

In both DMPs and ProMPs, the time-dependency of the basis functions can be given through a phase variable. Defining the basis functions through a phase variable allows for time-rescaling trajectories and synchronizing the movements of different limbs.

The Interaction Primitives (IPs), proposed by Ben Amor at al. [1], are based on DMPs. The Interaction Probabilistic Movement Primitives, proposed by Maeda et al. [16], and the Mixture of Interaction Primitives, proposed in this work, are based on ProMPs. Therefore, all those interaction models are time-dependent. In their current form, those models require the time-alignment of the demonstrations. In [1], Dynamic Time Warping [21] is used to time-align the demonstrations. In [16], a local optimization method is proposed to this end. However, as it will be discussed in Section 4.1, those procedures do not allow for reacting to actions executed in different speeds.

Time-independent methods to learn trajectories by imitation have been proposed. For example, Calinon et al. [6] propose an approach based on Hidden Markov Models (HMMs) and Gaussian Mixture Regression (GMR) to learn and reproduce gestures by imitation. Each hidden state corresponds to a Gaussian over positions and velocities, locally encoding variation and correlation. ProMPs, however, offer very useful properties for approaching the problem of Human-Robot Interaction, such as the parameterization of trajectories with a relatively small set of weights and the global encoding of variation and correlation. Therefore, ProMPs have been chosen in this work and a method is proposed to estimate the phase of the ProMPs online to be able to react to actions executed in different speeds.

A number of other methods deal with the problem of phase estimation or time-alignment. Englert et al. [8] present a method to adapt the shape and the phase of a trajectory to changes in the environment, as changes in the position of the goal or in the position of obstacles. Vuga et al. [24] present a modified form of dynamical movement primitives where the rate of phase change is related to the speed of movement. They use Reinforcement Learning and Iterative Learning Control (ILC) to speed up the execution of a movement as much as possible without violating some arbitrary constraints. For example, they speed up the movements of a robot carrying a glass full of liquid without spilling the liquid. Coates et al. [7] learn how to follow a desired trajectory from multiple sub-optimal expert's demonstrations. They apply their algorithm to the problem of autonomous helicopter flight. They use Dynamic Time Warping in order to find the relation between the time steps of the demonstrations and the time steps of the desired trajectory. Similarly, van den Berg et al. [23] use Dynamic Time Warping to learn the time mapping between demonstrated trajectories and a reference trajectory. With their approach, robots are able to perform surgical tasks with superhuman performance.

Differently from the cited works, the work on phase estimation presented in this thesis aims at inferring the phase from a partial observation of a trajectory. The position of the goal is not known a priori and there is no reference trajectory with a fixed number of time steps.

3 Human-Robot Interaction

In this chapter, the Interaction Primitive framework based on Probabilistic Movement Primitives [16, 18] will be briefly discussed, followed by the presentation of the proposed method, based on Gaussian Mixture Models. After that, two experiments are described. In the first experiment, it is demonstrated how the proposed method can deal with nonlinear correlations between the interacting agents. The second experiment is a proof of concept for a practical application, building a toolbox with the assistance of a robot.

3.1 Probabilistic Movement Primitives

A Probabilistic Movement Primitive (ProMP) [18] is a representation of movement based on a distribution over trajectories. The probabilistic formulation of a movement primitive allows operations from probability theory to seamlessly combine primitives, specify via points, and correlate joints via conditioning. Given a number of demonstrations, ProMPs are designed to capture the variance of the positions q and velocities $\dot{q}$ as well as the covariance between different joints.

For simplicity, at first, only the positions q for one degree of freedom (DOF) are considered. The position q_t at time step t can be approximated by a linear combination of basis functions,

$$q_t = \boldsymbol{\psi}_t^T \boldsymbol{w} + \epsilon, \tag{1}$$

where ϵ is Gaussian noise. The vector $\boldsymbol{\psi}_t$ contains the N basis functions ψ_i, $i \in \{1,\ 2,\ 3, ...,\ N\}$, evaluated at time step t. Standard normalized Gaussian basis functions will be used.

Twenty Gaussian basis functions evaluated between the values 0 and 100 of a phase variable are depicted by Fig. 2. In this chapter, the phase variable is treated as being simply the time. In Chapter 4 a different treatment will be given to it. It may be helpful to normalize the Gaussian basis functions, especially if those Gaussians are not so close to each other. This practice helps avoiding numerical errors due to weights with a very high magnitude. Fig. 3 depicts twenty normalized Gaussian basis functions. The sum of all those basis functions evaluated at the same time is equal to one. Fig. 4 shows weighted normalized Gaussian basis functions and Fig. 5 shows the sum of those weighted functions. By choosing different sets of weights, it is possible to approximate any smooth trajectory.

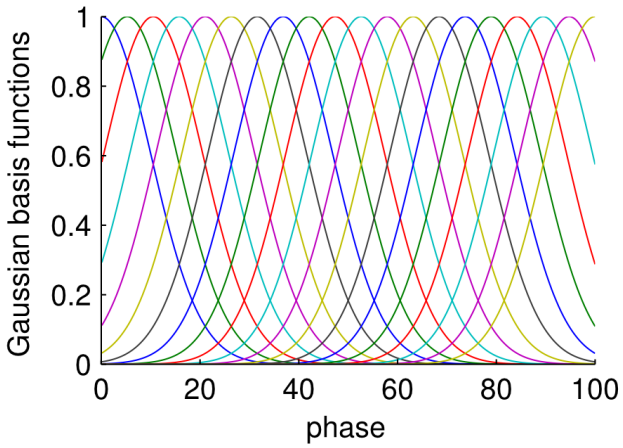


Figure 2: Gaussian basis functions.

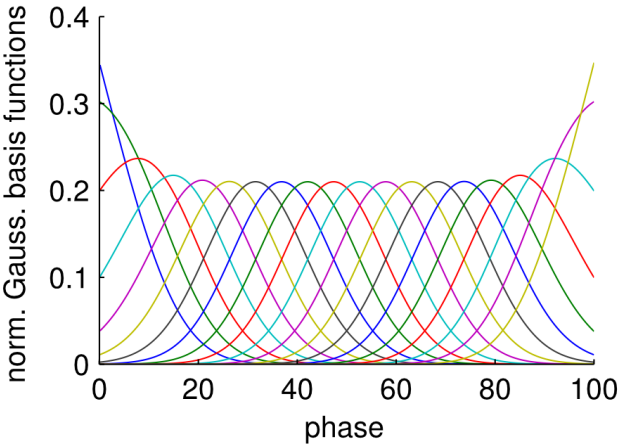


Figure 3: Normalized Gaussian basis functions.

The weight vector $\boldsymbol{w}$ can be seen as a compact representation of a trajectory. Having recorded a number of trajectories of q, the probability distribution over the weights $\boldsymbol{w}$ can be inferred. Typically, a single Gaussian distribution is used to represent $p(\boldsymbol{w})$. While a single $\boldsymbol{w}$ represents a single trajectory, a distribution $p(q_{1:T})$ over trajectories $q_{1:T}$ can be obtained by integrating $\boldsymbol{w}$ out,

$$p(q_{1:T}) = \int p(q_{1:T}|\boldsymbol{w})\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}. \tag{2}$$

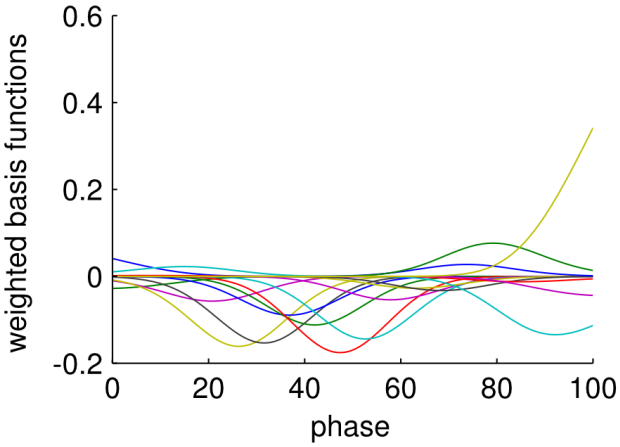


Figure 4: Weighted normalized Gaussian basis functions.

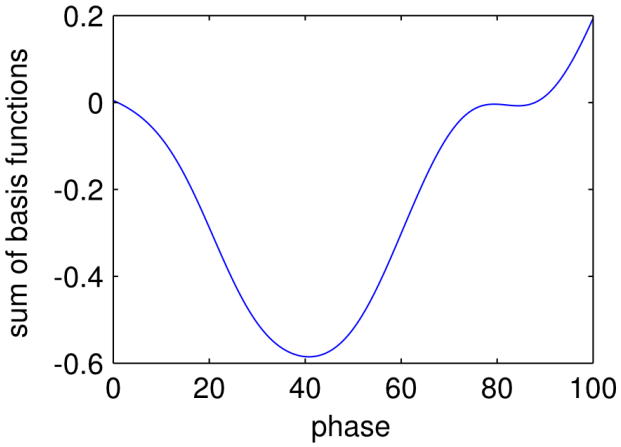


Figure 5: Sum of weighted normalized Gaussian basis functions.

If $p(\boldsymbol{w})$ is a Gaussian, $p(q_{1:T})$ is also Gaussian. The distribution $p(q_{1:T})$ is called a Probabilistic Movement Primitive (ProMP).

3.2 Interaction ProMP

An Interaction ProMP builds upon the ProMP formulation, with the fundamental difference that a distribution over the trajectories of all agents involved in the interaction is used instead. Hence, $\boldsymbol{q}$ is multidimensional and contains the positions in joint angles or Cartesian coordinates of all agents. This work focuses on the interaction between two agents, here defined as the observed agent (human) and the controlled agent (robot). Thus, the vector $\boldsymbol{q}$ is now given as $\boldsymbol{q} = \left[\left(\boldsymbol{q}^o\right)^T, \left(\boldsymbol{q}^c\right)^T\right]^T$, where $\boldsymbol{q}^o$ and $\boldsymbol{q}^c$ contain the positions of the observed and controlled agent, respectively.

Suppose a sequence of positions $\boldsymbol{q}_t^o$ of an interaction was observed, comprising m specific time steps t, $m \leq T$, where T is the duration of the whole interaction. This sequence will be denoted by D. Given those observations, the objective is to infer the most likely remaining trajectory of both the human and the robot.

Defining $\bar{\boldsymbol{w}} = \left[\boldsymbol{w}_o^T, \boldsymbol{w}_c^T\right]^T$ as an augmented vector that contains the weights of the human and of the robot for one demonstration, the conditional probability over trajectories $\boldsymbol{q}_{1:T}$ given the observations D of the human is written as

$$p\left(\boldsymbol{q}_{1:T}|D\right) = \int p\left(\boldsymbol{q}_{1:T}|\bar{\boldsymbol{w}}\right) p\left(\bar{\boldsymbol{w}}|D\right) \mathrm{d}\bar{\boldsymbol{w}}. \tag{3}$$

A normal distribution of n demonstrations is computed by stacking several weight vectors $\left[\bar{\boldsymbol{w}}_1, \ldots, \bar{\boldsymbol{w}}_n\right]^T$, one for each demonstration, such that $\bar{\boldsymbol{w}} \sim \mathcal{N}\left(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right)$. A posterior distribution can be obtained after observing D with

$$\begin{aligned} \boldsymbol{\mu}_w^+ &= \boldsymbol{\mu}_w^- + \boldsymbol{K}\left(D - \boldsymbol{H}_t^T \boldsymbol{\mu}_w^-\right) \\ \boldsymbol{\Sigma}_w^+ &= \boldsymbol{\Sigma}_w^- - \boldsymbol{K}\left(\boldsymbol{H}_t^T \boldsymbol{\Sigma}_w^-\right) \end{aligned} \tag{4}$$

where $\boldsymbol{K} = \boldsymbol{\Sigma}_w^- \boldsymbol{H}_t^T \left(\boldsymbol{\Sigma}_D + \boldsymbol{H}_t^T \boldsymbol{\Sigma}_w^+ \boldsymbol{H}_t\right)^{-1}$, $\boldsymbol{\Sigma}_D$ is the observation noise, and

$$\boldsymbol{H}_t = \left[\begin{array}{cc:cc} \left(\boldsymbol{\psi}_t^o\right)_{(1,1)} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \left(\boldsymbol{\psi}_t^o\right)_{(P,P)} & \boldsymbol{0} & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}^c_{(1,1)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}^c_{(Q,Q)} \end{array}\right] \tag{5}$$

is the observation matrix where the unobserved states of the robot are filled with zero bases. Here, the human and the robot are assumed to have P and Q DOFs, respectively.

Now, by combining (1), (3) and (4), the probability distribution over the trajectories $\boldsymbol{q}_{1:T}$ given the observation D can be computed. For a detailed implementation the interested reader is referred to [16].

3.3 Mixture of Interaction ProMPs

The goal of the proposed method is to learn several interaction patterns given the weight vectors that parameterize the unlabeled training trajectories. For this purpose, a GMM in the weight space is learned, using the Expectation-Maximization algorithm (EM) [3].

Assume a training set with n vectors $\bar{\boldsymbol{w}}$ representing the concatenated vectors of human-robot weights as defined in Section 3.2. In order to implement EM for a GMM with a number K of Gaussian mixture components, the Expectation and the Maximization step need to be implemented and the algorithm must iterate over those steps until convergence of the probability distribution over the weights, $p\left(\bar{\boldsymbol{w}}; \alpha_{1:K}, \boldsymbol{\mu}_{1:K}, \boldsymbol{\Sigma}_{1:K}\right)$, where $\alpha_{1:K} = \left\{\alpha_1, \alpha_2, \cdots, \alpha_K\right\}$, $\boldsymbol{\mu}_{1:K} = \left\{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K\right\}$ and $\boldsymbol{\Sigma}_{1:K} = \left\{\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \cdots, \boldsymbol{\Sigma}_K\right\}$. Here, $\alpha_k = p(k)$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the prior probability, the mean and the covariance matrix of mixture component k, respectively. The parameters $\alpha_{1:K}$, $\boldsymbol{\mu}_{1:K}$ and $\boldsymbol{\Sigma}_{1:K}$ are initialized using k-means clustering [3] before starting the Expectation-Maximization loop. The number K of Gaussian mixture components is found by leave-one-out cross-validation.

The mixture model can be formalized as

$$p(\bar{\boldsymbol{w}}) = \sum_{k=1}^{K} p(k)\, p(\bar{\boldsymbol{w}}|k) = \sum_{k=1}^{K} \alpha_k \mathcal{N}\left(\bar{\boldsymbol{w}}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right). \tag{6}$$

Expectation step: Compute the responsibilities r_{ik}, where r_{ik} is the probability of cluster k given weight vector $\bar{\boldsymbol{w}}_i$.

$$r_{ik} = p\left(k|\bar{\boldsymbol{w}}_i\right) = \frac{\mathcal{N}\left(\bar{\boldsymbol{w}}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right) \alpha_k}{\sum_{l=1}^{K} \alpha_l \mathcal{N}\left(\bar{\boldsymbol{w}}_i; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\right)} \tag{7}$$

Maximization step: Update the parameters α_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ of each cluster k, using

$$n_k = \sum_{i=1}^{n} r_{ik}, \;\; \alpha_k = \frac{n_k}{n}, \tag{8}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} r_{ik} \bar{\boldsymbol{w}}_i}{n_k}, \tag{9}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{n_k} \left(\sum_{i=1}^{n} r_{ik} \left(\bar{\boldsymbol{w}}_i - \boldsymbol{\mu}_k\right) \left(\bar{\boldsymbol{w}}_i - \boldsymbol{\mu}_k\right)^T \right). \tag{10}$$

Finally, this model can be used to infer the trajectories of the controlled agent, given observations from the observed agents. The posterior probability distribution over trajectories $\mathbf{q}_{1:T}$ given the observations D must be found, as in Section 3.2.

In order to compute this posterior using the GMM prior, first the algorithm finds the most probable cluster k^* given the observation D, using the Bayes' theorem [3]. The posterior over the clusters k given the observation D is given by

$$p(k|D) \propto p(D|k)\, p(k), \tag{11}$$

where

$$p(D|k) = \int p(D|\bar{\boldsymbol{w}})\, p(\bar{\boldsymbol{w}}|k)\, \mathrm{d}\bar{\boldsymbol{w}}$$

and

$$p\left(\bar{\boldsymbol{w}}|k\right)=\mathcal{N}\left(\bar{\boldsymbol{w}};\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right).$$

Thus the most probable cluster k^* given the observation D is

$$k^*=\arg\max_k p\left(k|D\right). \tag{12}$$

The output of the proposed algorithm is the posterior probability distribution over trajectories $\mathbf{q}_{1:T}$, conditioning cluster k^* to the observation D

$$p\left(\mathbf{q}_{1:T}|D\right)=\int p\left(\mathbf{q}_{1:T}|\bar{\boldsymbol{w}}\right)p\left(\bar{\boldsymbol{w}}|k^*,D\right)\mathrm{d}\bar{\boldsymbol{w}}. \tag{13}$$

Algorithms 1 and 2 provide a compact description of the proposed methods for training and inference, respectively.

Algorithm 1 Training

1) Parameterize demonstrated trajectories:
Find vector of weights $\bar{\boldsymbol{w}}$ for each trajectory, such that $\mathbf{q}_t\approx\boldsymbol{\psi}_t^T\bar{\boldsymbol{w}}$.
2) Find GMM in parameter space, using EM:
Initialize GMM parameters $\alpha_{1:K}$, $\boldsymbol{\mu}_{1:K}$ and $\boldsymbol{\Sigma}_{1:K}$ with k-means clustering.
repeat
E step

$$r_{ik}=p\left(k|\bar{\boldsymbol{w}}_i\right)=\frac{\mathcal{N}\left(\bar{\boldsymbol{w}}_i;\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k\right)\alpha_k}{\sum_{l=1}^{K}\alpha_l\mathcal{N}\left(\bar{\boldsymbol{w}}_i;\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l\right)}$$

M step

$$n_k=\sum_{i=1}^{n}r_{ik},\ \alpha_k=\frac{n_k}{n}$$

$$\boldsymbol{\mu}_k=\frac{\sum_{i=1}^{n}r_{ik}\bar{\boldsymbol{w}}_i}{n_k}$$

$$\boldsymbol{\Sigma}_k=\frac{1}{n_k}\left(\sum_{i=1}^{n}r_{ik}\left(\bar{\boldsymbol{w}}_i-\boldsymbol{\mu}_k\right)\left(\bar{\boldsymbol{w}}_i-\boldsymbol{\mu}_k\right)^T\right)$$

until $p\left(\bar{\boldsymbol{w}};\alpha_{1:K},\boldsymbol{\mu}_{1:K},\boldsymbol{\Sigma}_{1:K}\right)$ converges

Algorithm 2 Inference

1) Find most probable cluster given observation:

$$p\left(k|D\right) \propto p\left(D|k\right) p\left(k\right)$$

$$k^* = \arg\max_k p\left(k|D\right)$$

2) Condition on observation, using cluster k^*:

$$p\left(\mathbf{q}_{1:T}|D\right) = \int p\left(\mathbf{q}_{1:T}|\bar{\boldsymbol{w}}\right) p\left(\bar{\boldsymbol{w}}|k^*, D\right) \mathrm{d}\bar{\boldsymbol{w}}$$

3.4 Experiments

This section presents experimental results in two different scenarios using a 7-DOF KUKA lightweight arm with a 5-finger hand[1].

The goal of the first scenario is to expose the issue of the original Interaction Primitives [1, 16] when dealing with trajectories that have a clear multimodal distribution. In the second scenario a real application of the method designed in this work is proposed, where the robot assistant acts as a third hand of a worker assembling a toolbox.

3.4.1 Nonlinear Correlations between the Human and the Robot on a Single Task

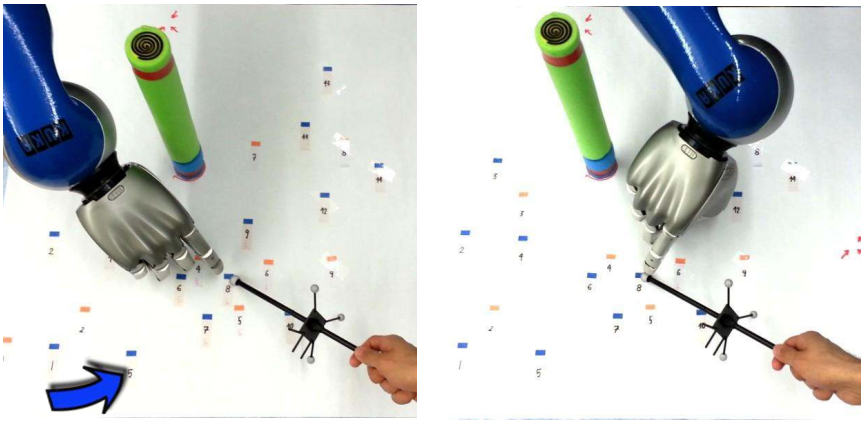
To expose the capability of the proposed method for dealing with multimodal distributions, a toy problem is proposed, where a human specifies a position on a table and the robot must point at the same position. The robot is not provided any form of exteroceptive sensors; the only way it is capable to generate the appropriate pointing trajectory is by correlating its movement with the trajectories of the human. As shown in Fig. 6, however, a pole was placed in front of the robot such that the robot can only achieve the position specified by the human by moving either to the right or to the left of the pole. This scenario forces the robot to assume quite different configurations, depending on which side of the pole its arm is moving around.

During demonstrations the robot was moved by kinesthetic teach-in to point at the same positions indicated by the human (tracked by motion capture) without touching the pole. For certain positions, as the one indicated by the arrow in Fig. 6(a), only one demonstration was possible. For other positions, both right and left demonstrations could be provided as shown in Fig. 6(a) and 6(b). The demonstrations, totaling 28 pairs of human-robot trajectories, resulted in a multimodal distribution of right and left trajectory patterns moving around the pole.

In this scenario, modeling the whole distribution over the parameters of the trajectories with one single Gaussian (as in the original Interaction Primitive formulation) is not capable of generalizing the movements of the robot to other positions in a way that resembles the training, as the original framework is limited by assuming a single pattern. This limitation is clearly shown in Fig. 7(a) where several trajectories generated by a single cluster GMM (as in the original Interaction Primitive) cross over the middle of the demonstrated trajectories, which, in fact, represents the mean of the single Gaussian distribution.

Fig. 7(b) shows the predictions using the proposed method with a mixture of Gaussians. By modeling the distribution over the parameters of the trajectories using GMMs as described in Section 3.3, a much

[1] Regarding the control of the robot, the design of a stochastic controller capable of reproducing the distribution of trajectories is also part of ProMPs and the interested reader is referred to [18] for details. Here a compliant, human-safe standard inverse-dynamics based feedback controller is used.



(a) (b)

Figure 6: Experimental setup of a toy problem used to illustrate the properties of the Mixture of Interaction Primitives. The robot is driven by kinesthetic teach-in to point at the positions specified by the human (pointed with the wand). Certain pointed positions can be achieved by either moving the arm to the right (a) or to left (b) of the pole placed on the table. Other positions, such as the one indicated by the arrow, can only be achieved by one interaction pattern.

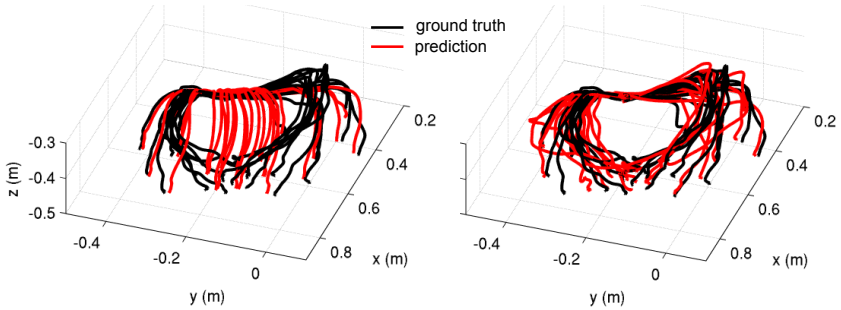


Figure 7: Results of the predictions of the robot trajectories in Cartesian space. Both subplots show the same ground truth trajectories generated by driving the robot in kinesthetic teach-in. The predictions are generated by leave-one-out cross-validation on the whole data set comprised of 28 demonstrations. (a) Prediction using the conventional Interaction ProMPs with a single Gaussian. (b) Prediction using the proposed method with a mixture of Gaussians.

better performance could be achieved. The GMM assumption that the parameters are only locally linear correlated seemed to represent the data much more accurately. As shown in Fig. 8, this improvement is quantified in terms of the Root Mean Square (RMS) error of the prediction of the trajectory in relation to the ground truth using leave-one-out cross-validation over the whole data set. The same figure also shows that there is a sharp decrease in the RMS error up to six clusters, especially when taking into account the variance among the 28 tests. Beyond seven clusters it is observed that the prediction error fluctuates around 4 cm. The experiments previously shown in Fig. 7(b) were done with eight clusters.

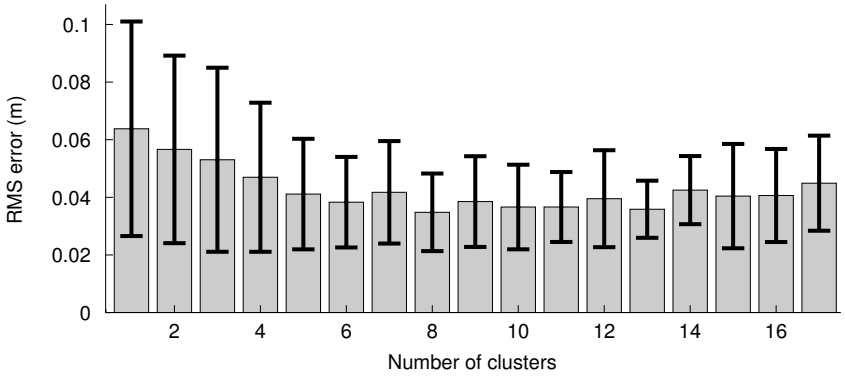


Figure 8: Root Mean Square error with models using up to 17 Gaussians. The error and its variance drop significantly until six clusters. Using six or more clusters, the error fluctuates around 4cm.

3.4.2 Assembling a Box with a Robot Assistant

In this experiment, a number of demonstrations of different interaction patterns between a human and the robot cooperating to assemble a box were recorded. The same robot described in the previous experiment was used. During demonstrations the human wore a bracelet with markers whose trajectories in Cartesian coordinates were recorded by motion capture. Similarly to the first scenario, the robot was moved in gravity compensation mode by another human during the training phase and the trajectories of the robot in joint space were recorded.

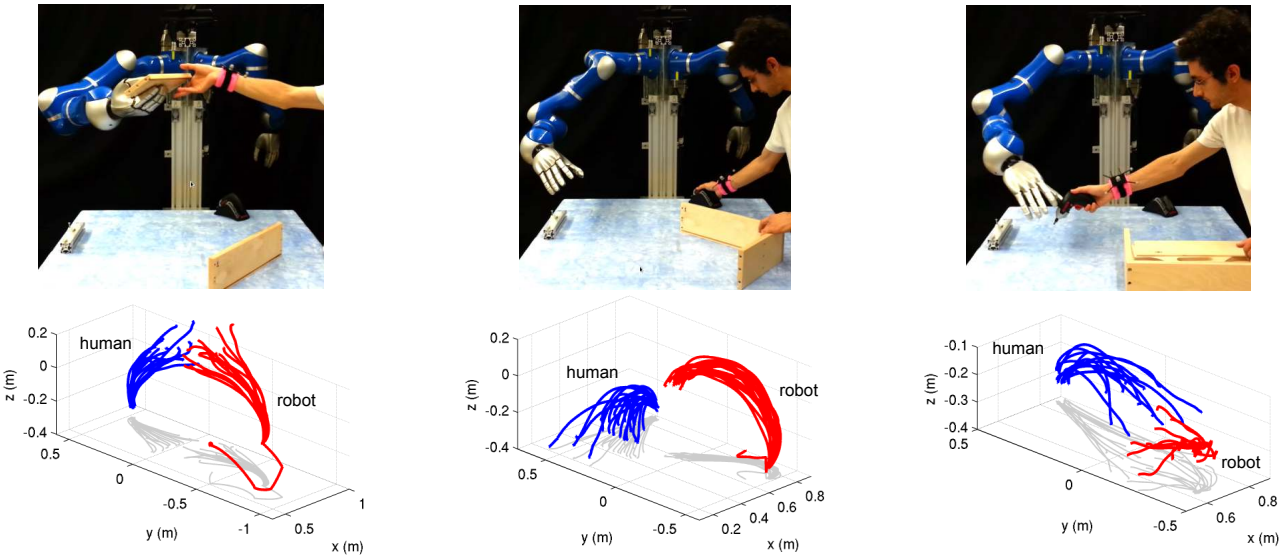
There are three interaction patterns. Each interaction pattern was demonstrated several times to reveal the variance of the movements. In one of them, the human extends his/her hand to receive a plate. The robot fetches a plate from a stand and gives it to the human. In a second interaction the human fetches the screwdriver and the robot grasps and gives a screw to the human as a pre-emptive collaborator would do. The third type of interaction consists of giving/receiving a screwdriver. The interactions of plate handover, screw handover and holding the screwdriver were demonstrated 15, 20, and 13 times, respectively. The pairs of trajectories of each interaction are shown in Fig. 9[2].

As described in Section 3.2, all training data are fed to the algorithm resulting in 48 human-robot pairs of unlabeled demonstrations as shown in the upper row of Fig. 11. The presented method parameterizes the trajectories and performs clustering in the parameter space in order to encode the mixture of primitives. In the figure, the human is represented by the (x, y, z) Cartesian coordinates while the robot is represented by the seven joints of the arm. The figure only shows up to the first four joints (starting from the base).

Figure 10 shows the RMS prediction error averaged over all tests as the number of mixture components increases. The prediction is obtained by leave-one-out cross-validation over the whole set of 48 demonstrations. As one would expect, since the unlabeled data set contains three distinct interaction patterns, the improvement is clearly visible up to three mixture components. No significant improvement is obtained afterwards, thus the GMM with three mixture components was selected for experiments.

In the inference/execution phase, the algorithm first computes the most probable Interaction Primitive mixture component based on the observation of the position of the wrist of the human with (12). Using the same observation, the algorithm conditions the most probable Interaction Primitive, which allows computing a posterior distribution over trajectories for all seven joints of the robot arm as in (13).

[2] Due to the experimental setup, for the sub-tasks of plate and screw handover an initial hand-coded trajectory that runs before the kinesthetic teach-in effectively starts was added. Those trajectories are used to make the robot grasp and remove the plate or screw from their respective stands. This is reflected in the figure as the deterministic part at the beginning of the trajectory of the robot. This initial trajectory, however, has no effect on the proposed method itself.



(a) Handing over a plate

(b) Handing over a screw

(c) Holding the screw driver

Figure 9: Demonstrations of the three different interactions and their respective trajectories. For the case of plate and screw handover the beginning of the robot trajectory shows a deterministic part that accounts for the fact that the robot has to remove objects from their respective stands, which is not part of the kinesthetic teach-in and which does not affect the algorithm in any sense.

Finally, the mean of each joint posterior distribution is fed to a standard inverse dynamics feedback tracking controller.

The lower row of Fig. 11 depicts the posterior distribution for one test example where a three-cluster GMM was trained with the other 47 trajectories. The GMM prior is shown in gray where the patches of different clusters overlap. The observation consists only of the final position of the wrist, shown as asterisks in the figure. The black lines are the ground truth trajectories of each degree of freedom. The posterior, in red, is represented by its mean and by the region inside $\pm$ two standard deviations. The mean of this posterior is the most probable trajectory for each degree of freedom given the observed end position of the wrist of the human.

The toolbox, consisting of seven parts and 12 screws, was assembled two times. The experiments demanded more than 40 executions of the Interaction Primitives. The selection of the right mixture component was 100% correct.

The precision of the interactions was evaluated by computing the final position of the hand of the robot with forward kinematics. The forward kinematics was fed with the conditioned robot trajectories predicted by leave-one-out cross validation. The interactions of plate handover and holding screwdriver resulted in mean error with two standard deviations (mean error $\pm 2\sigma$) of 3.2 ± 2.6 cm and 2.1 ± 2.3 cm, respectively. The precision of the handover of the screw has not been evaluated, as the position at which the robot hands the screw is not correlated with the human.

As an example, Fig. 12 shows the robot executing the plate handover at three different positions based on the location of the wrist marker. Note that the postures of the arm are very different, although they are all captured by the same Interaction Primitive.

A video showing this experiment can be found at `http://youtu.be/9XwqW_V0bDw`.

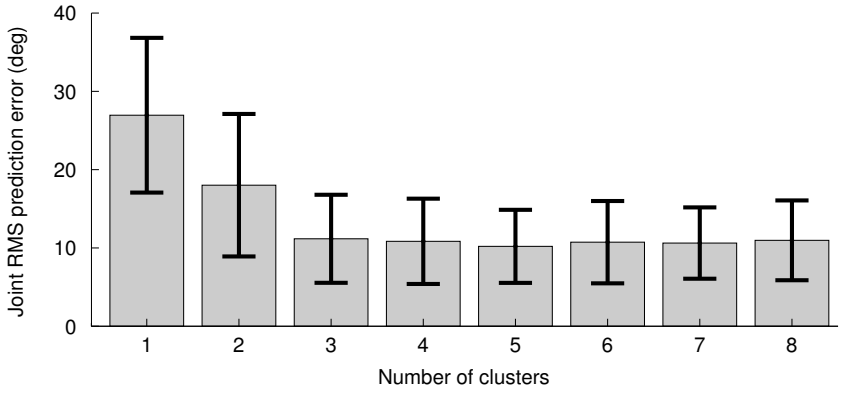


Figure 10: Root Mean Square error of the joint trajectories (averaged over all tests) using a leave-one-out cross-validation as a function of the number of clusters (mixture components). The plateau after three clusters seems to be consistent with the training data set, since it consists of three distinct interaction patterns.

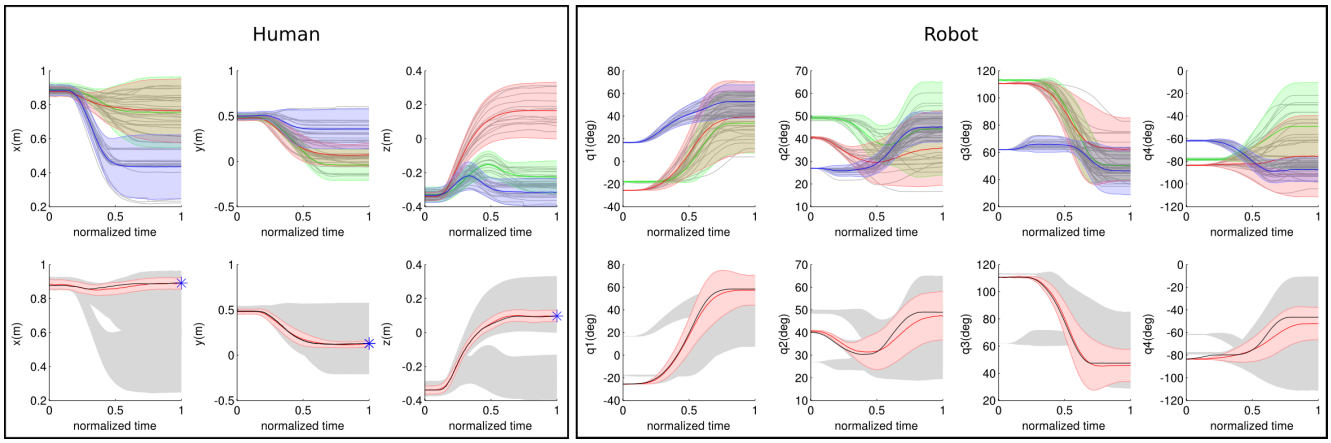


Figure 11: Upper row: Mixture components represented by their mean trajectories and the region inside two standard deviations ($\mu \pm 2\sigma$). Obs.: The plots show only the part of the trajectories generated by kinesthetic teach-in. **Lower row**: Posterior probability distribution given observation depicted by the blue asterisks.

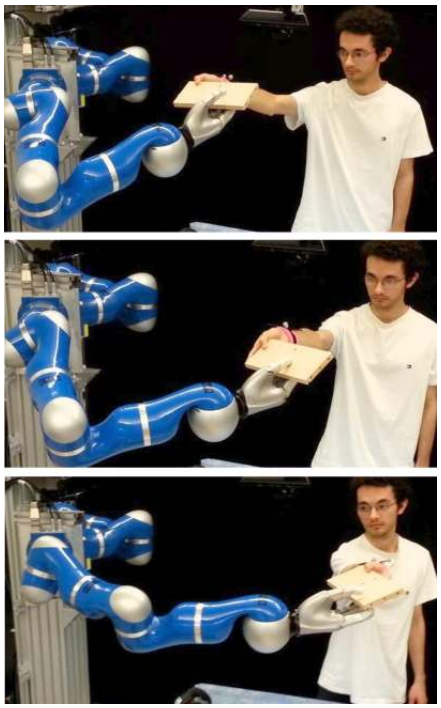


Figure 12: Handover of a plate. Conditioning on three different positions of the wrist (using motion capture) of a human coworker.

4 Phase Estimation

This chapter addresses the problem of estimating the phase of a trajectory. This estimation allows for predicting trajectories with different durations from partial observations. Phase estimation may also play a role in reacting to actions with different durations and in switching from one interaction to another during execution. A human interacting with the robot may for example move sometimes faster and sometimes slower. He/she may also change his/her mind in the middle of a movement and decide to do something else.

In Section 4.1, a method to estimate the phase online, i.e., during execution of the trajectory in test phase, is proposed. Section 4.2 presents a method to determine the weight vectors for each training trajectory and the probability distribution over those weights using Expectation-Maximization (EM). The EM-based method is compared to the traditional one, which uses Linear Regression. The importance of the EM-based method to the phase estimation problem is pointed out.

4.1 Online Phase Estimation

For the experiments presented in Section 3.4, the trajectories in the training data set were time-aligned using a local optimization method proposed by Maeda et al. in [16]. After time-alignment, all trajectories have the same number T of time steps. This time-alignment simplifies the problem of finding the weight vectors for each trajectory. Equation (1) expresses the relation between positions q_t and weight vectors $\boldsymbol{w}$. The relation between a whole trajectory $q_{1:T}$ and a weight vector can thus be expressed as

$$\begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_T \end{pmatrix} = \begin{pmatrix} \psi_{1,1} & \psi_{2,1} & \cdots & \psi_{N,1} \\ \psi_{1,2} & \psi_{2,2} & \cdots & \psi_{N,2} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{1,T} & \psi_{2,T} & \cdots & \psi_{N,T} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}, \tag{14}$$

where $\psi_{i,j}$ stands for the basis function i evaluated at time step j. If all trajectories have the same number T of time steps, the matrix with elements $\psi_{i,j}$ needs to be computed only once.

The drawback of this simplification is that the unobserved part of test trajectories can not be predicted unless it is assumed that the test trajectories have also T time steps. For interaction, the correlation between a partial observation of the movement of the human and the reaction of the robot must be found. For this purpose, it is necessary to assign a time index or a phase to each observed position. Previous Interaction Primitive formulations do not provide this assignment of positions to time index or phase, unless it is assumed that the interactions have the same duration T. In practice however, it may be necessary that the robot react to actions with different durations. Due to the fact that a solution to this problem was missing, the experiments presented in Section 3.4 assumed that the only observation provided by the human was at the end of the trajectory, such that phase estimation becomes obvious. Conversely, the instant of observations given halfway the execution of a trajectory could not be associated with the Interaction ProMP as the phase estimation component was not available.

In order to deal with interactions with different durations, the basis functions are defined this time as functions of a phase variable $z(t)$,

$$q(t) = \boldsymbol{\psi}(z(t))^T \boldsymbol{w} + \epsilon, \tag{15}$$

where $z(t) = \alpha t$ and $\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$. The phase variable $z(t)$ assumes real values from 0 to 100. A trajectory with an arbitrary number of time steps T has therefore α equal to $\frac{100}{T}$.

In the face of a partial observation D of an action, the duration of the whole action is still unknown. It is possible though to infer the number of time steps of the whole trajectory by computing the posterior probability distribution over the parameters α, given observation D. This posterior is denoted by $p(\alpha|D)$.

Using Bayes' theorem,

$$p(\alpha|D) \propto p(D|\alpha)\, p(\alpha). \tag{16}$$

Assuming $p(\alpha)$ a Gaussian probability distribution, it can be determined by computing the individual parameters α_i for each trajectory in the training data set and by computing the mean and the variance of this set of values. This assumption is reasonably close to reality as long as the durations of the trajectories range around a single value.

The term $p(D|\alpha)$ can be computed by

$$p(D|\alpha) = \int p(D|\boldsymbol{w}, \alpha)\, p(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}. \tag{17}$$

Here it has been assumed that the weight vector $\boldsymbol{w}$ is independent of the parameter α. This assumption can be verified by the analysis of plots of weights versus α. Fig. 13 shows plots of weights for different basis functions versus the parameter α. The sum of weighted basis functions in this case is approximating trajectories of hand-drawn letters “a” (see Fig. 28). There are twenty different trajectories. Each asterisk corresponds to a different trajectory. Those plots point to no strong correlation between weights and α. This means that weights basically determine the shape of the trajectory, but not how fast or slow it is executed.

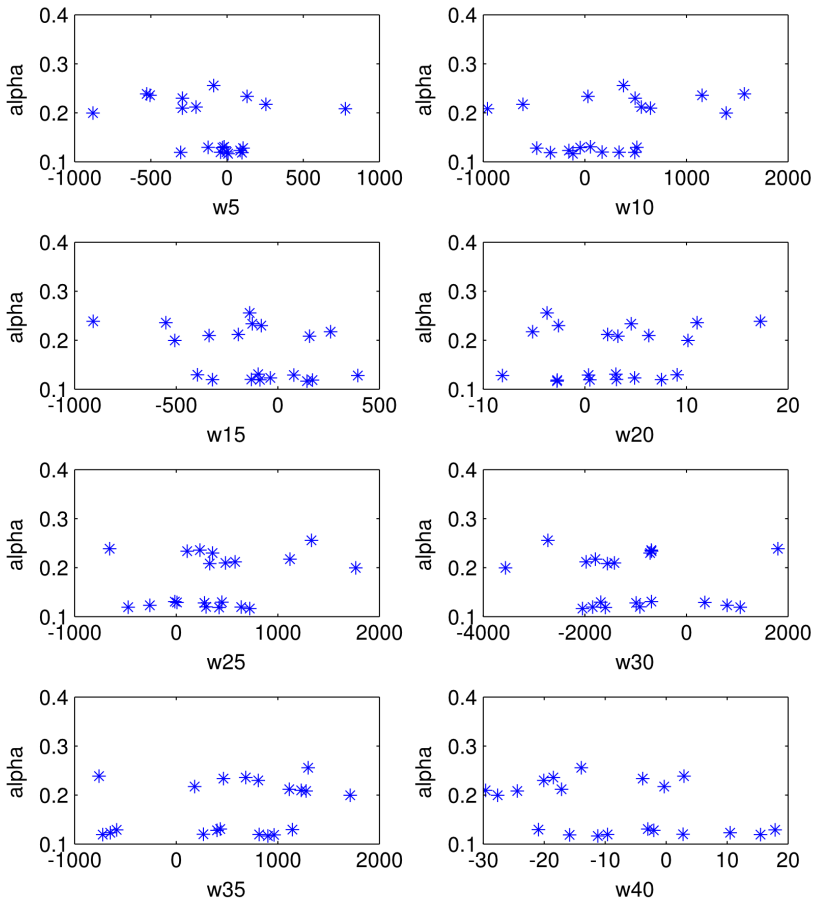


Figure 13: Weights w vs. phase parameters α. There is no strong correlation between w and α. Therefore they can be treated as independent variables.

For simplicity, it is also assumed that $p(\boldsymbol{w})$ is a single multivariate Gaussian. One way of computing $p(\boldsymbol{w})$ is by estimating the weights vector $\boldsymbol{w}_i$ for each training trajectory through Linear Regression, given (15). Subsequently, the mean $\boldsymbol{\mu}_w$ and the covariance $\boldsymbol{\Sigma}_w$ of this set of weight vectors are computed.

Equation (17) can be solved in closed form, yielding $p(D|\alpha) \sim \mathcal{N}\left(\boldsymbol{\mu}_D, \boldsymbol{\Sigma}_D\right)$ with

$$\begin{aligned} \boldsymbol{\mu}_D &= \boldsymbol{A}\boldsymbol{\mu}_w \\ \boldsymbol{\Sigma}_D &= \sigma^2 \mathrm{I} + \boldsymbol{A}\boldsymbol{\Sigma}_w \boldsymbol{A}^T. \end{aligned} \tag{18}$$

The parameter α determines the expected duration T_α of the whole trajectory. For an observation D, comprising m time steps ($m \leq T_\alpha$), $\boldsymbol{A}$ assumes the form

$$\boldsymbol{A} = \begin{pmatrix} \psi_1(z(1)) & \psi_2(z(1)) & \cdots & \psi_N(z(1)) \\ \psi_1(z(2)) & \psi_2(z(2)) & \cdots & \psi_N(z(2)) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(z(m)) & \psi_2(z(m)) & \cdots & \psi_N(z(m)) \end{pmatrix}, \tag{19}$$

with $z(t) = \alpha t$.

In order to estimate α online, the algorithm first samples a number of values for α from the prior probability distribution $p(\alpha)$. After each new observed position, $p(\alpha|D)$ is computed according to (16) for each of the sampled values of α. The estimated value of α is the one that maximizes $p(\alpha|D)$,

$$\alpha^* = \arg\max_{\alpha} p(\alpha|D). \tag{20}$$

Having estimated the phase parameter α, it is then possible to compute the phase z at each time step until the expected duration T_α of the whole trajectory. Afterwards, a matrix of basis functions is defined, as in (19) with the difference that this new matrix has T_α, instead of only m time steps. The mean and covariance of the whole trajectory can be computed as in (18), using now the new matrix of basis functions defined over the whole expected duration of the trajectory.

4.2 Expectation-Maximization for ProMPs

This section explains an Expectation-Maximization approach to determine the vectors of weights for the training trajectories of a ProMP and the probability distribution over those weights. This approach is especially important in the face of missing data. An extension of this approach to deal with more complex relations between time t and phase variable $z(t)$ is motivated. The implementation of this extension will be part of a future work.

4.2.1 Dealing with Missing Data

When the training data set is comprised of only complete trajectories, using Linear Regression for each trajectory separately is enough to estimate weights for the basis functions that approximate each training trajectory with good accuracy, as it was quantitatively evaluated by experiments that will be described in Section 4.3.1. However, occlusions, noise and lack of sensor coverage when recording training trajectories are frequent occurrences. For example, when using a motion capture system, situations where some markers are not detected are quite common. Other systems have similar problems. Therefore, a good training algorithm should also be robust to missing data.

The procedure of using Linear Regression on each trajectory separately to estimate the weights vector for each training trajectory and afterwards computing the mean and the covariance of all those weight vectors to define a multivariate Gaussian distribution over the weights is not suitable to deal with missing data. The reason is that this procedure only observes each trajectory separately. If one trajectory has been completely observed until 50% of its execution and a second trajectory has been completely observed

only after 50% of its execution, for example, parameterizing those partial observations separately is not enough to learn how the whole trajectory should look like. A method that learns at the same time from all the training trajectories that are available is necessary.

Learning from all the training trajectories at the same time is possible by using an Expectation-Maximization approach to estimate the weights. In this approach, even when estimating the vectors of weights for the individual observations, all the training trajectories are taken into consideration.

This EM approach does not require the time-alignment of the training trajectories as a more naive method such as finding the mean of all points available would. Besides, finding a mean trajectory from the partial observations that are available could potentially result in discontinuities. Moreover, this EM approach will be extended in a future work to deal with more complex relations between phase and time as the one that has been used in this work. This extension may allow for learning the parameters that determine the relationship between phase and time. A time-alignment of the training trajectories already would require some assumption about the phase of the observed parts.

Given a number n of training trajectories, the objective is to find the mean and the covariance of $p(\boldsymbol{w})$, assuming $p(\boldsymbol{w})$ a single multivariate Gaussian. The training trajectories may have missing values. The observed parts of each of those trajectories is represented by D_i. This problem can be formulated as an Expectation-Maximization [3] problem, where $\boldsymbol{w}$ is a hidden vector-valued variable.

The joint probability of the observations,

$$\prod_i p_\theta\left(D_i\right) = \prod_i \int p\left(D_i|\boldsymbol{w},\alpha_i\right) p\left(\boldsymbol{w}\right) \mathrm{d}\boldsymbol{w}, \tag{21}$$

must be maximized with respect to θ, where $p\left(D_i|\boldsymbol{w},\alpha_i\right) = \mathcal{N}\left(D_i; \boldsymbol{A}_i\boldsymbol{w}, \sigma^2\boldsymbol{I}\right)$. The parameter θ in $p_\theta\left(D_i\right)$ stands for the set $\left\{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right\}$, the mean and the covariance of the Gaussian distribution over the weight vectors $\boldsymbol{w}$.

The matrix $\boldsymbol{A}_i$ has the values of each basis function evaluated at the phase value correspondent to the observed time steps. Assuming one of the observations D_i extends from time step 1 to time step m, where $m \leq T_i$, T_i being the total number of time steps of the trajectory i,

$$D_i = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{pmatrix} = \boldsymbol{A}_i\boldsymbol{w} = \begin{pmatrix} \psi_1\left(z_i(1)\right) & \psi_2\left(z_i(1)\right) & \cdots & \psi_N\left(z_i(1)\right) \\ \psi_1\left(z_i(2)\right) & \psi_2\left(z_i(2)\right) & \cdots & \psi_N\left(z_i(2)\right) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1\left(z_i(m)\right) & \psi_2\left(z_i(m)\right) & \cdots & \psi_N\left(z_i(m)\right) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}, \tag{22}$$

where $z_i(t) = \alpha_i t$.

The observations D_i do not need to be comprised of positions at successive time steps. There may be gaps between the observed parts as well.

The parameters α_i are determined by

$$\alpha_i = \frac{100}{T_i}, \tag{23}$$

assuming the duration T_i of each training trajectory is known.

The estimation of θ goes as follows. First, θ is initialized with arbitrary values,

$$\theta_0 = \left\{\boldsymbol{\mu}_{w0}, \boldsymbol{\Sigma}_{w0}\right\}. \tag{24}$$

Next, the algorithm performs the Expectation Step (E-Step), by computing the function $Q\left(\theta, \theta_0\right)$,

$$Q\left(\theta, \theta_0\right) = \sum_i \mathbb{E}_{\theta_0}\left(\log p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right) | D = D_i\right), \tag{25}$$

and performs the Maximization Step (M-Step),

$$\theta \in \arg\max_{\theta} Q\left(\theta, \theta_0\right). \tag{26}$$

The algorithm keeps iterating over the E-Step and the M-Step until $\prod_i p_\theta\left(D_i\right)$ converges. A detailed derivation of the following equations can be found in Appendix A.

E step:

$$\boldsymbol{\Sigma}_{wi} = \left(\boldsymbol{A}_i^T \left(\sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{A}_i + \boldsymbol{\Sigma}_{w0}^{-1}\right)^{-1}, \tag{27}$$

$$\boldsymbol{\mu}_{wi} = \boldsymbol{\Sigma}_{wi}\left(\boldsymbol{A}_i^T \left(\sigma^2 \boldsymbol{I}\right)^{-1} D_i + \boldsymbol{\Sigma}_{w0}^{-1} \boldsymbol{\mu}_{w0}\right). \tag{28}$$

M step:

$$\boldsymbol{\mu}_w^* = \frac{\sum_i \boldsymbol{\mu}_{wi}}{n}, \tag{29}$$

$$\boldsymbol{\Sigma}_w^* = \frac{\left(\boldsymbol{E}^T \boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{w_i}\right)}{n}, \tag{30}$$

with

$$\boldsymbol{\mu}_{wi} = \begin{pmatrix} \mu_{w_{i1}} \\ \mu_{w_{i2}} \\ \vdots \\ \mu_{w_{iN}} \end{pmatrix}, \qquad \boldsymbol{\mu}_w = \begin{pmatrix} \mu_{w_1} \\ \mu_{w_2} \\ \vdots \\ \mu_{w_N} \end{pmatrix}, \tag{31}$$

$$\boldsymbol{e}_i = \boldsymbol{\mu}_{wi} - \boldsymbol{\mu}_w = \begin{pmatrix} \mu_{w_{i1}} - \mu_{w1} \\ \mu_{w_{i2}} - \mu_{w2} \\ \vdots \\ \mu_{w_{iN}} - \mu_{wN} \end{pmatrix}, \tag{32}$$

$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \vdots \\ \boldsymbol{e}_n^T \end{pmatrix}. \tag{33}$$

Note that this approach incorporates all available observations when estimating the mean $\boldsymbol{\mu}_{wi}$ and the covariance $\boldsymbol{\Sigma}_{wi}$ that define the probability distribution over the weights for each training trajectory i. Equations (27) and (28) involve the terms $\boldsymbol{\mu}_{w0}$ and $\boldsymbol{\Sigma}_{w0}$, which represent the mean and the covariance from the previous EM iteration of the Gaussian over the weights of all the training trajectories.

4.2.2 Discussion on Dealing with Complex Phase Variables

In this work, the phase variable has been defined as a linear function of time, $z(t) = \alpha t$. By this definition, the higher the value of α, the faster the trajectory is executed and the lower the number of time steps of the whole trajectory. For example, a letter "a" that is written in 391 time steps has $\alpha = \frac{100}{391} \approx 0.2558$. A letter "a" that is written in 858 time steps has $\alpha = \frac{100}{858} \approx 0.1166$. By changing the

α, the whole trajectory is executed faster or slower. According to this definition of the phase variable, there is no way to make some parts of the trajectory faster and other parts slower.

In practice, however, this sort of variability occurs. For example, consider the movement of grasping a glass of water followed by the movement of handing it over to another person and treat the concatenation of those two movements as a single trajectory. Someone can move slowly towards the glass of water and hand it over quickly to another person; or move quickly towards the glass and hand it over slowly. The previous formulation of the phase variable accounts only for the possibility of executing both movements faster or slower.

In order to account for this sort of variability, which makes some parts of a trajectory slower and other parts faster, one possibility is to define the phase variable as

$$z(t) = \boldsymbol{\phi}(t)^T \boldsymbol{\beta}, \tag{34}$$

where $\boldsymbol{\phi}(t)$ is a vector of basis functions defined on time t and $\boldsymbol{\beta}$ is a vector of weights for each basis function. Since the movement is always advancing forward along the trajectory, the phase must be a monotonically increasing function of time. Therefore, positive sigmoid basis functions are a sensible choice.

With this new definition of phase, according to (34), it is not possible to find the correspondent $\boldsymbol{\beta}$ to each training trajectory just by knowing the duration of each trajectory and the maximum value of the phase, as it was possible before for the parameters α. To estimate both the parameters $\boldsymbol{\beta}$ and the vectors of weights $\boldsymbol{w}$ for each training trajectory, an EM approach may be used that tries to maximize the probability of the observations with respect to the mean of the Gaussian distribution over the weights, $\boldsymbol{\mu}_w$, the covariance of this distribution, $\boldsymbol{\Sigma}_w$, the mean of the Gaussian distribution over the parameters $\boldsymbol{\beta}$, $\boldsymbol{\mu}_\beta$, and the covariance of this distribution, $\boldsymbol{\Sigma}_\beta$. In this approach, both $\boldsymbol{w}$ and $\boldsymbol{\beta}$ are treated as hidden variables.

The derivation, implementation and test of the method proposed in this section are planned as part of a future work.

4.3 Experiments

This section presents experiments on online phase estimation, using the method proposed in Section 4.1, when all training trajectories were completely observed. After that, experiments dealing with incomplete training data are presented that demonstrate some of the advantages of using the proposed EM approach for ProMPs explained in Section 4.2.1 in comparison to using the traditional Linear Regression method.

4.3.1 Experiments on Online Phase Estimation

In these experiments, there are twenty training trajectories and twenty test trajectories, different from the ones in the training data set. The trajectories correspond to the letter "a" with different shapes and durations. Given a partial observation of a test trajectory, the objective is to predict the unobserved part. The prediction should be as close to the ground truth as possible. In the experiments presented in this section, the training data set was comprised only of whole trajectories.

First, both training and test data were constructed by sampling weights and parameters α from Gaussian distributions. Forty trajectories were generated with the sampled values, using (22). Twenty of them were selected at random as training and the other twenty as test data. This data set fulfills perfectly the assumptions that both the weights of the basis functions and the parameters α are normal distributed, therefore it was chosen as a start point to test the proposed algorithm for estimating the phase online. Fig. 14 depicts the training data and Fig. 15, the test data.

In test phase, fifty values for the α parameter are sampled from the prior probability distribution $p(\alpha)$. Afterwards each test trajectory is observed, one position at a time. After each new observed position, the most probable value for α from the set of sampled values is computed with (20). Given the most probable α, the matrix with the values of each basis function at each time step is computed with (19).

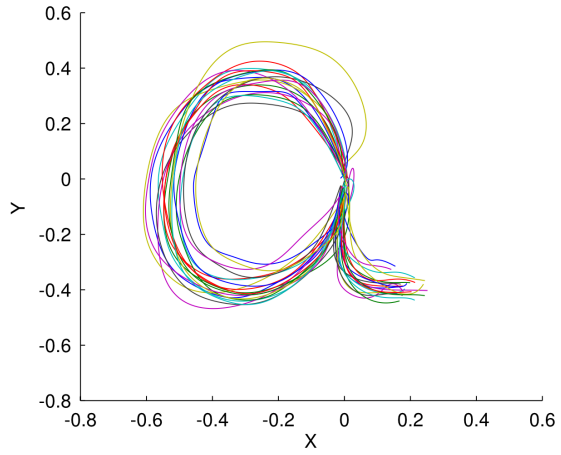


Figure 14: Training data with normal distributed weights and α.

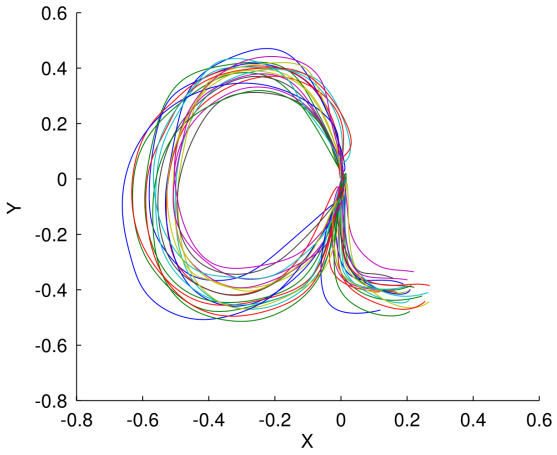


Figure 15: Test data with normal distributed weights and α.

The unobserved part is predict with (3). Figs. 16 to 20 show the phase estimation and the completion after observing 5%, 10%, 20%, 40% and 60% of a test trajectory, respectively. The vertical red line on the right plot denotes the index of the most probable α from the set of fifty sampled values. In general, the larger the observed part, the better the estimation of α and the better the prediction. In this example, after observing 40% of the trajectory, the best α was correctly estimated with approximately 85% of confidence.

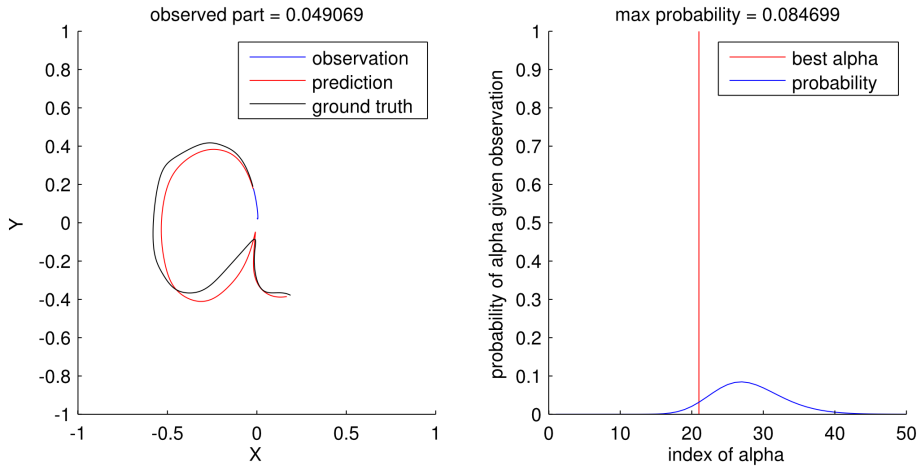


Figure 16: Completion after observing 5 percent of a test trajectory.

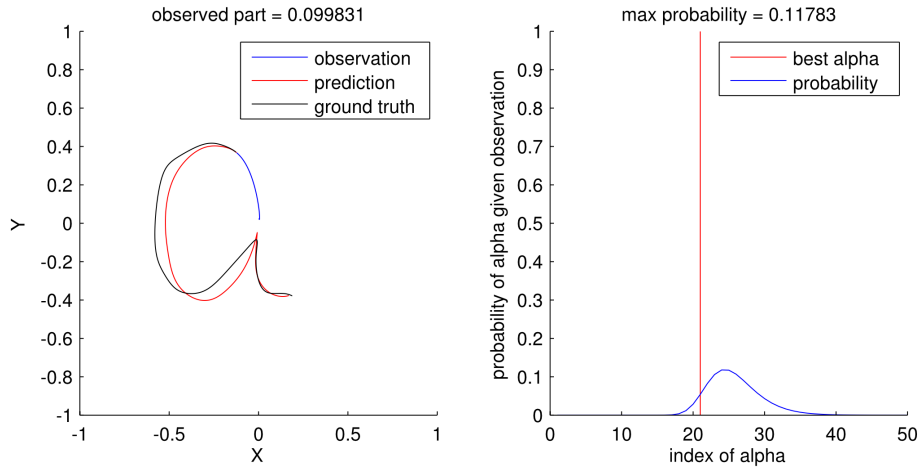


Figure 17: Completion after observing 10 percent of a test trajectory.

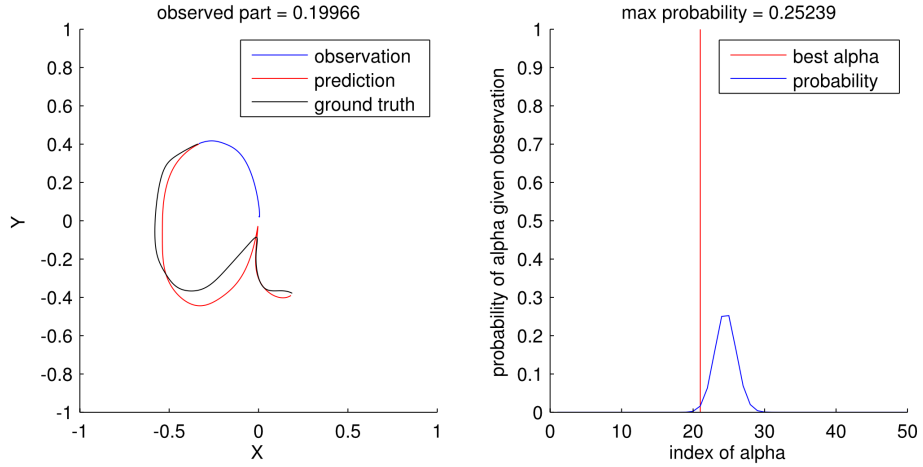


Figure 18: Completion after observing 20 percent of a test trajectory.

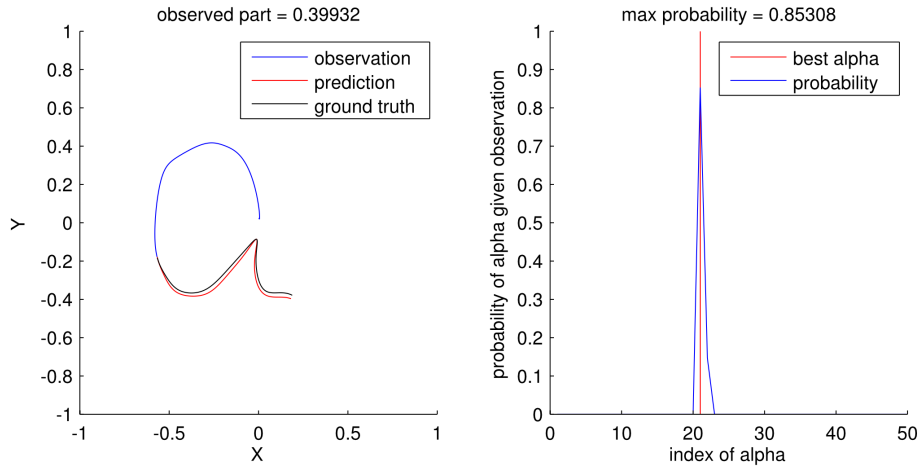


Figure 19: Completion after observing 40 percent of a test trajectory.

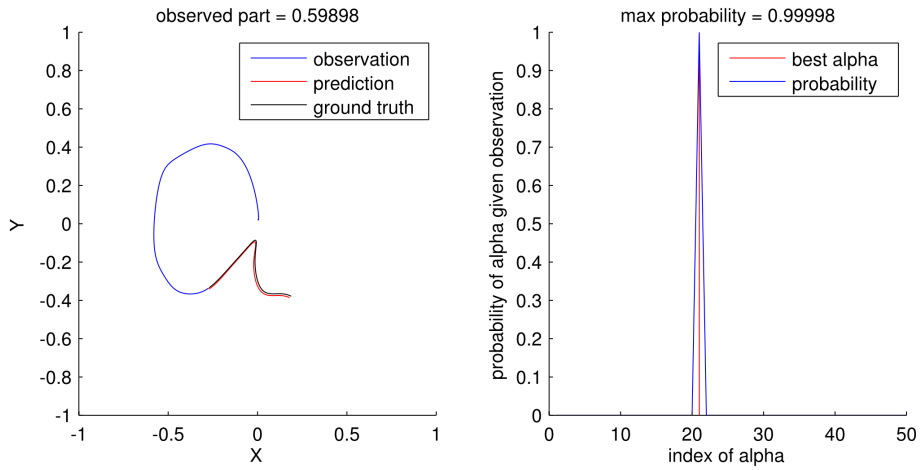


Figure 20: Completion after observing 60 percent of a test trajectory.

Figs. 21 to 25 correspond to Figs. 16 to 20, respectively. In Figs. 21 to 25, the x trajectory and the y trajectory as functions of time are plotted separately. The mean and the region inside two standard deviations ($\mu \pm 2\sigma$) of the prediction is depicted in red. The ground truth is depicted in blue and the observed positions are represented by the blue asterisks. Observe that the prediction after observing 40% has approximately the same number of time steps as the ground truth and achieves high accuracy.

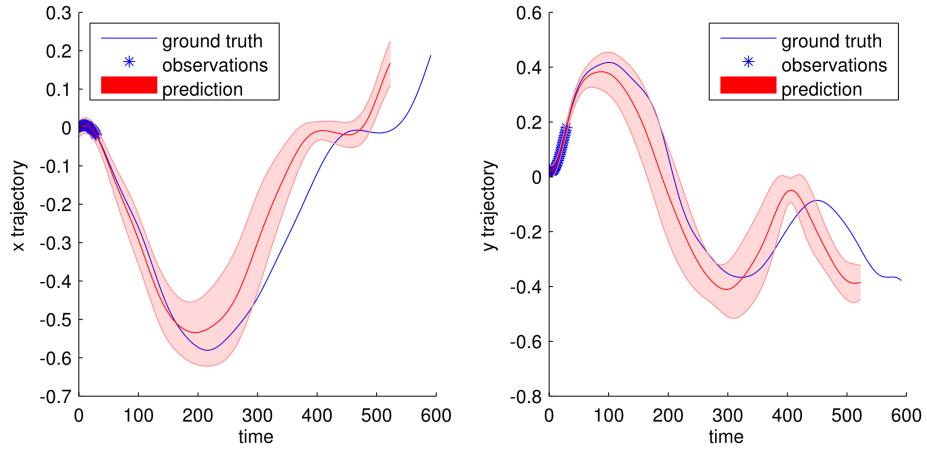


Figure 21: Completion after observing 5 percent of a test trajectory (X and Y trajectories).

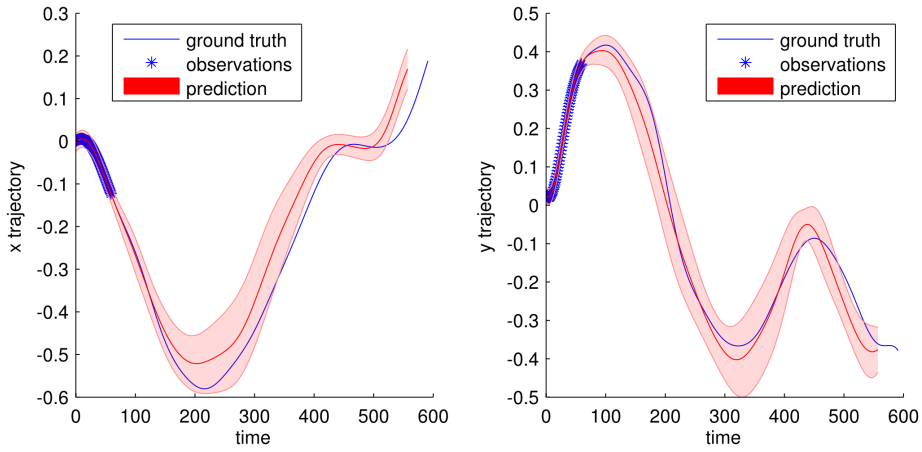


Figure 22: Completion after observing 10 percent of a test trajectory (X and Y trajectories).

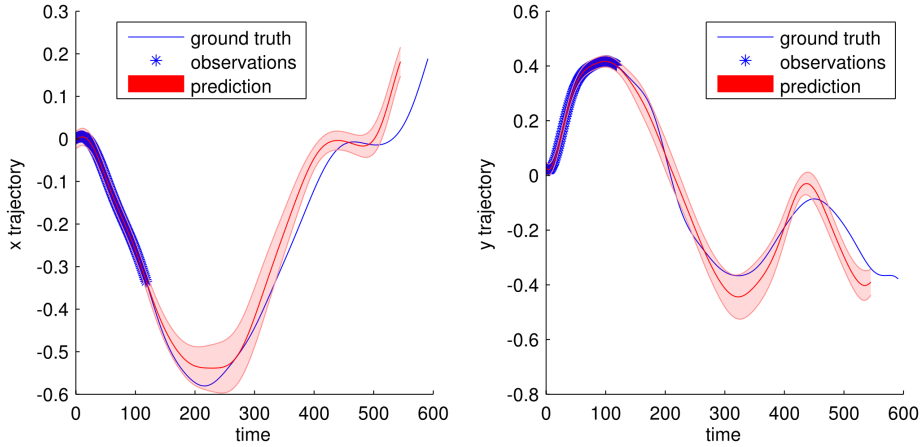


Figure 23: Completion after observing 20 percent of a test trajectory (X and Y trajectories).

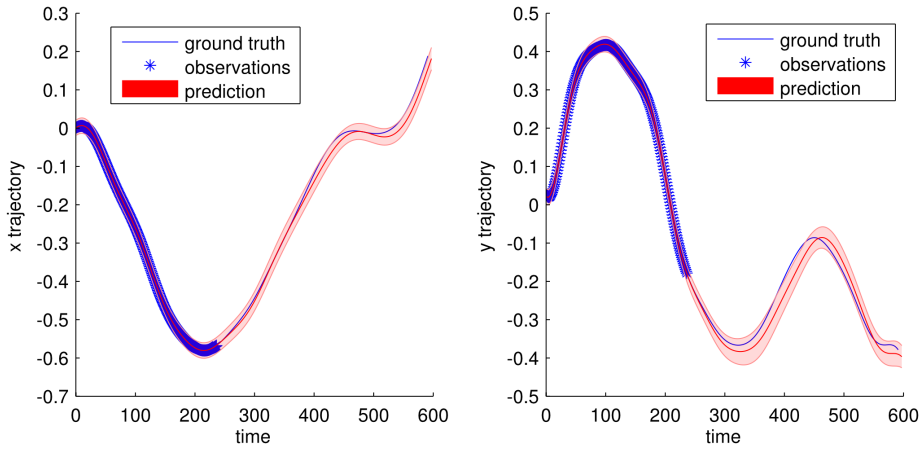


Figure 24: Completion after observing 40 percent of a test trajectory (X and Y trajectories).

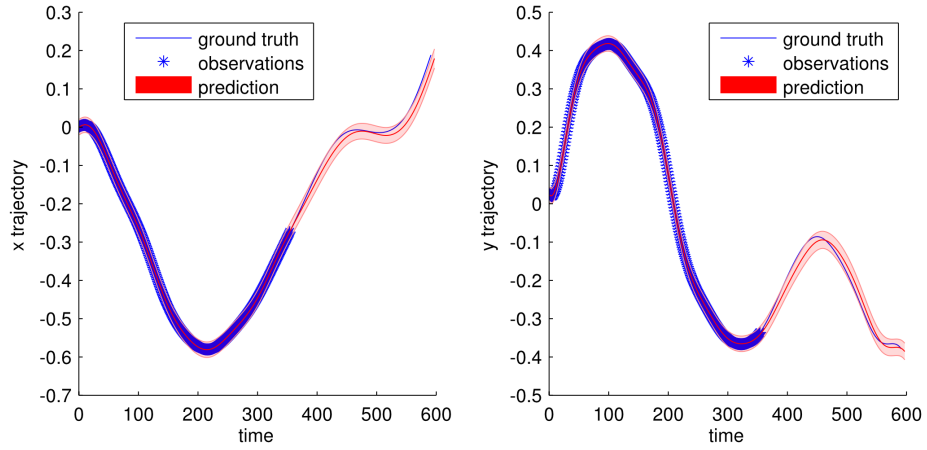


Figure 25: Completion after observing 60 percent of a test trajectory (X and Y trajectories).

The accuracy of the predictions was evaluated by computing the Root Mean Square (RMS) error between prediction and ground truth. The RMS error was computed for each of the twenty test trajectories every time the number of time steps of the observed part increased 1% in relation to the total number of time steps of the trajectory. Fig. 26 shows how the RMS error changes as the observed part increases for each of the twenty test trajectories. In general, the RMS error gets lower as the observed part increases, but it is also possible to observe the RMS error getting sometimes larger with a increase in the observed part and getting lower again afterwards. This has not been thoroughly investigated in this work, but may be due to the assumption that the phase variable is a linear function of time, $z(t) = \alpha t$. This assumption is enough to predict the duration of a trajectory, but is just a rough approximation to reality, as discussed in Section 4.2.2. The Gaussians from which weights and α were sampled were built based on real letters "a" that were drawn by hand. Those trajectories were not drawn with constant speeds, therefore test trajectories may have parts that point towards a higher value of α followed by parts that point towards a lower value, for example. Trajectories that have actually different durations may look at certain intervals of time like they should have the same duration. Fig. 27 shows the mean and the variance of the RMS error as the observed part increases.

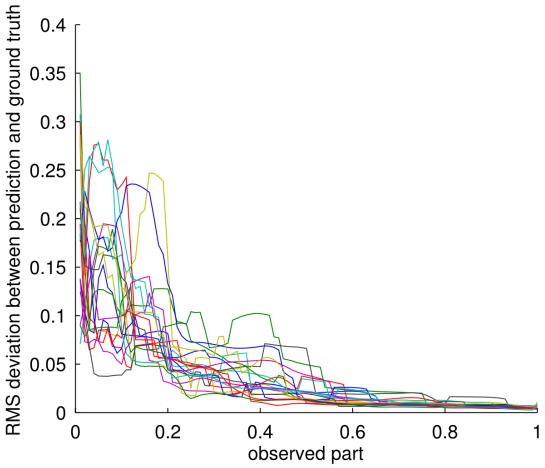


Figure 26: Observed part versus RMS error for each test trajectory with normal distributed weights and α.

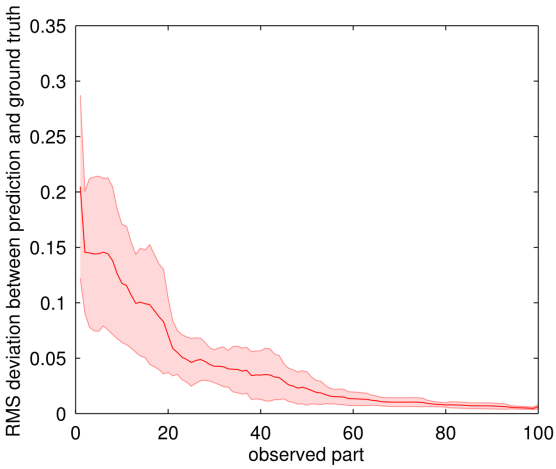


Figure 27: Observed part versus mean and standard deviation of the RMS error.

In a second experiment on online phase estimation, the training and the test trajectories were actually drawn by hand, instead of being constructed by sampling normal distributed parameters as in the previous experiment. The prior probability distribution over weights, $p(\boldsymbol{w})$, was built by first computing the weight vectors for each training trajectory by Linear Regression and then computing the mean and the standard deviation of this set of weight vectors, as explained in Section 4.1. Fig. 28 shows the twenty training trajectories and Fig. 29, the twenty test trajectories for this experiment.

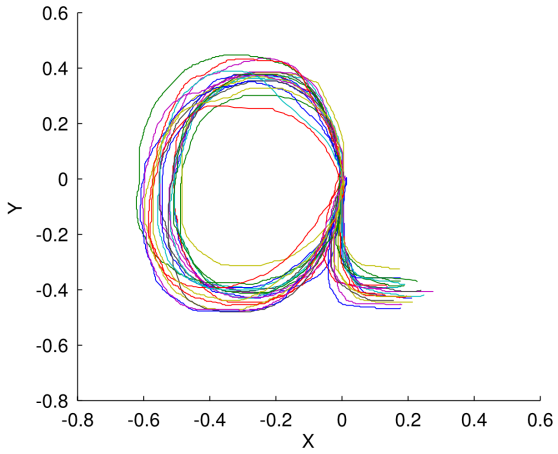


Figure 28: Training data with hand-drawn trajectories.

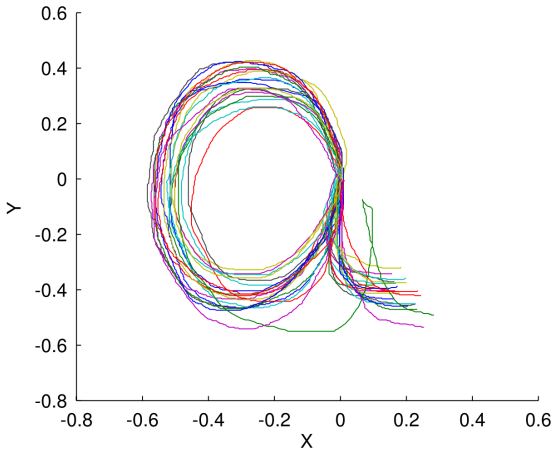


Figure 29: Test data with hand-drawn trajectories.

Fig. 30 shows how the RMS error of the prediction changes as the observed part increases for each of the twenty test trajectories. Clearly there are three trajectories for which the RMS error remained higher than for the other trajectories. Those were trajectories that did not fit very well the Gaussian distribution learned in the training phase. Alternatives to improve the quality of the predictions would be to increase the number and the diversity of the training trajectories and to learn a Gaussian Mixture Model over the weights as opposed to using one single Gaussian. Fig. 31 shows the mean and the standard deviation of the RMS error as the observed part increases.

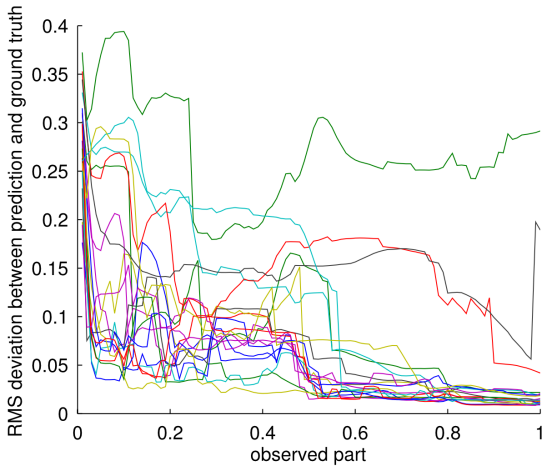


Figure 30: Observed part versus RMS error for each hand-drawn test trajectory after training with Linear Regression.

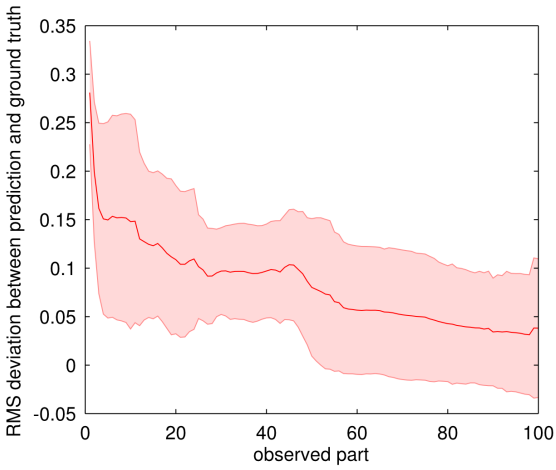


Figure 31: Observed part versus mean and standard deviation of the RMS error after training with Linear Regression.

In a further experiment with the hand-drawn trajectories, instead of using Linear Regression to learn the vectors of weights for the training trajectories, the Expectation-Maximization method presented in Section 4.2 was applied. The correspondent plots for the RMS error of the predictions are shown in Figs. 32 and 33. By superimposing Figs. 31 and 33, as depicted by Fig. 34, no big difference can be noticed between the two methods, except for the fact that the method where the EM approach was used during the training phase resulted in less variance for situations where more than 90% of the test trajectory was already observed.

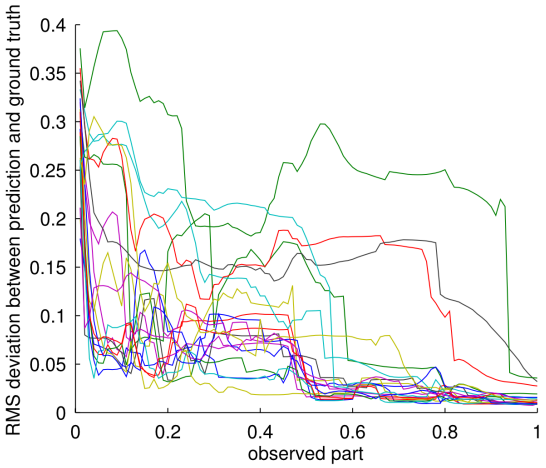


Figure 32: Observed part versus RMS error for each hand-drawn test trajectory after training with EM.

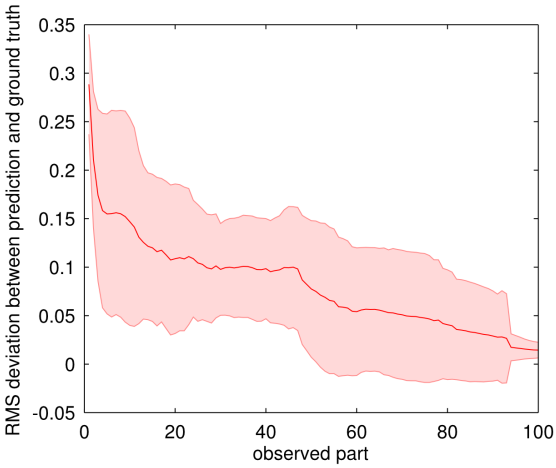


Figure 33: Observed part versus mean and standard deviation of the RMS error after training with EM.

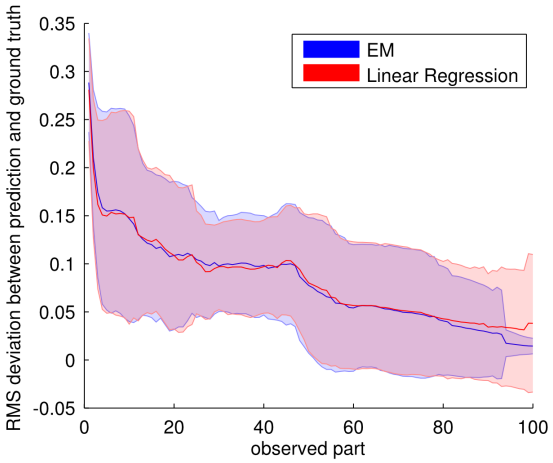


Figure 34: EM versus Linear Regression with fully observed training trajectories. In this case, both training methods resulted in similar prediction performances. The EM method resulted in less variance than the Linear Regression method basically only when more than 90% of the test trajectory was already observed.

4.3.2 Experiments on Dealing with Missing Data

Often during the recording of movements with a motion capture system, for example, there are occlusions or situations where not all the markers can be detected by the cameras around the scene, resulting in recorded training trajectories with some missing data points. Problems such as occlusion also hold for video cameras and other sensors. Especially if robots are to be used in dynamic environments, such as at home, in hospitals or in a disaster scenario, possibly interacting physically with people, being able to learn from partial observed trajectories becomes a key ability to successfully solve many different tasks.

This section presents experiments that show the applicability of the EM approach proposed in Section 4.2.1 to the learning of ProMPs in the face of training trajectories with missing data points. Those ProMPs, which are probabilistic distributions over trajectories (see Section 3.1), are used in these experiments to predict the unobserved part of trajectories in test phase, by conditioning the models learned in training phase to the observations made in test phase with (3).

The training and test trajectories are, as in the experiments previously described, letters “a”. The trajectories were in this case artificially generated by sampling weight vectors for the Gaussian basis functions and α parameters from Gaussian distributions. Afterwards, random sequences of 180 time steps were removed from the training trajectories. Since the shortest training trajectory comprises 355 time steps and the longest, 1129, the missing parts correspond approximately to between 16% and 51% of the entire trajectories. Fig. 35 shows three of those training trajectories.

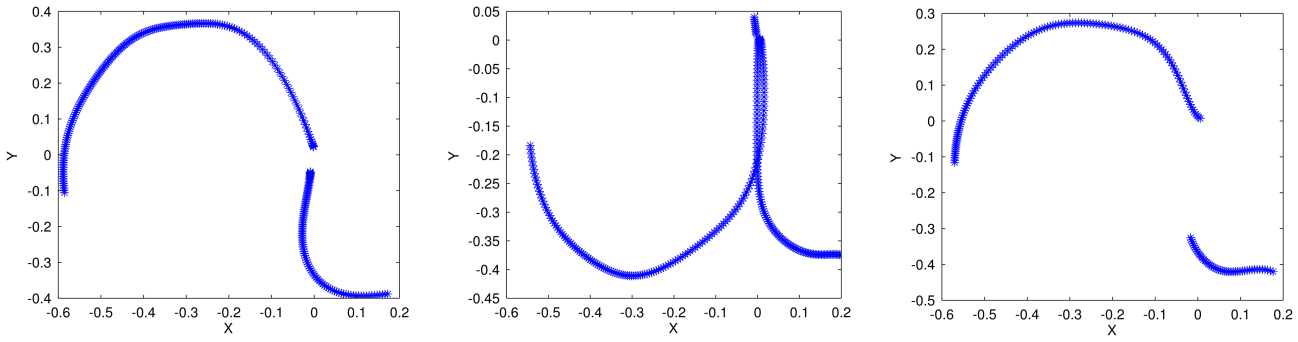


Figure 35: Training trajectories with missing data points. Those trajectories miss random sequences of 180 time steps.

In a first experiment, Linear Regression was used to learn the weight vectors for each of those training trajectories and a Gaussian distribution was fitted to the resulting set of weight vectors. Fig. 36 depicts the approximation generated by Linear Regression to one of the training trajectories.

In a second experiment, the EM approach proposed in Section 4.2.1 was used instead of Linear Regression to learn the weights. Fig. 37 depicts the approximation generated by the EM approach to the same training trajectory as in Fig. 36.

The approximation generated by the EM approach resembles a letter “a”, while the one generated by the Linear Regression method does not. The reason is that, while learning the weights for one training trajectory, the Linear Regression method only takes into consideration the observed positions of this trajectory, not using any information from other training trajectories, which may have observed positions complementing the observations currently under consideration. The EM method, on the other hand, takes into consideration all training trajectories while computing the mean and the covariance that define a probability distribution over the weights for each trajectory i with (27) and (28). Note that those equations involve the terms $\boldsymbol{\mu}_{w0}$ and $\boldsymbol{\Sigma}_{w0}$, which represent the mean and the covariance from the previous EM iteration of the Gaussian over the weights of all training trajectories.

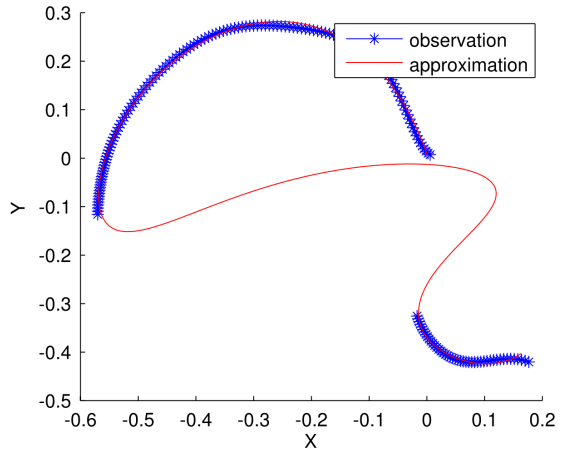


Figure 36: Approximation to a training trajectory with Linear Regression.

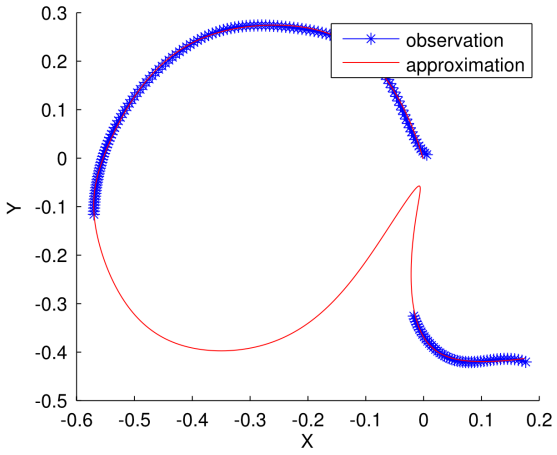


Figure 37: Approximation to a training trajectory with EM.

After training a ProMP with the Linear Regression method and another ProMP with the EM method, both models were used to estimate the phase and predict the unobserved part of test trajectories. Figs. 38 and 39 show the results obtained after training with the Linear Regression approach and observing 50% of a test trajectory. Figs. 40 and 41 show the results obtained after training with the EM approach and observing 50% of the same test trajectory. The EM algorithm could estimate the phase and predict the unobserved part much better than the algorithm that simply used Linear Regression for each training trajectory separately.

Fig. 42 shows the mean and the standard deviation of the Root Mean Square (RMS) error as the observed part of the test trajectories increases. As the observed part gets larger, the RMS gets in general lower for both methods, because the prior probability distribution over the weights gets less and less relevant in the face of new observations. However, the EM approach reaches lower values for the RMS sooner than the Linear Regression method.

Finally, Fig. 43 shows the number of EM iterations versus the log-likelihood of the training data, illustrating the learning progress of EM in the training phase.

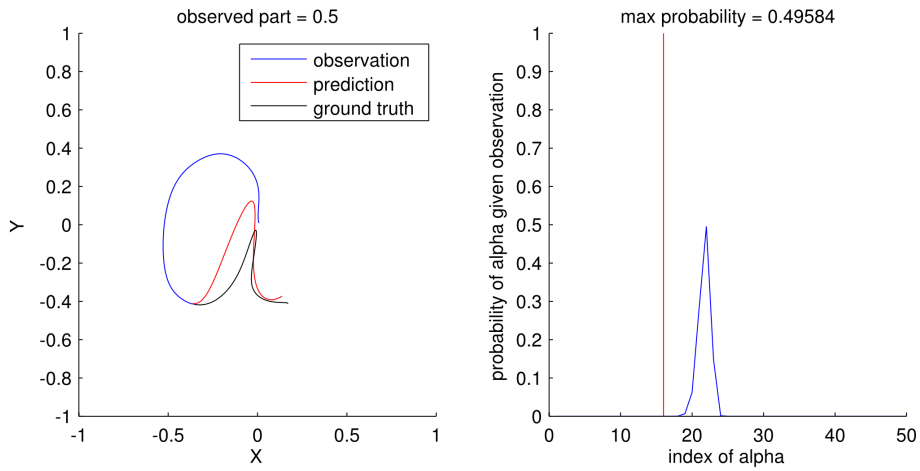


Figure 38: Completion after training with Linear Regression.

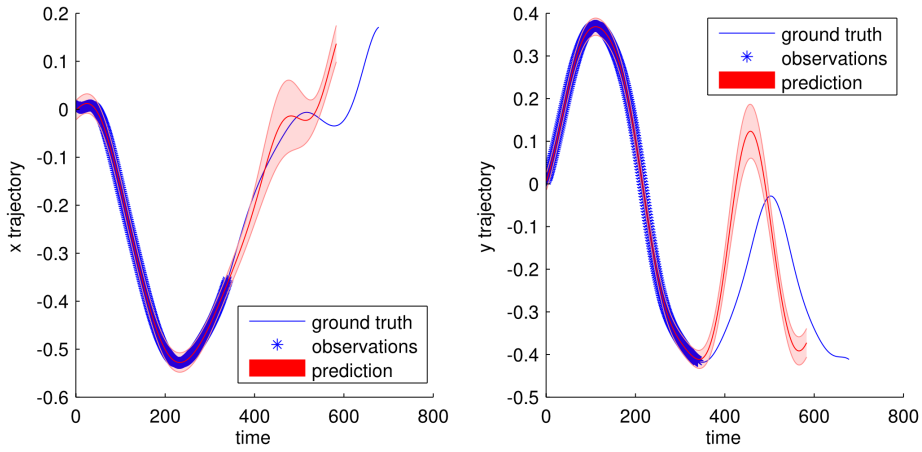


Figure 39: Completion after training with Linear Regression (X and Y trajectories).

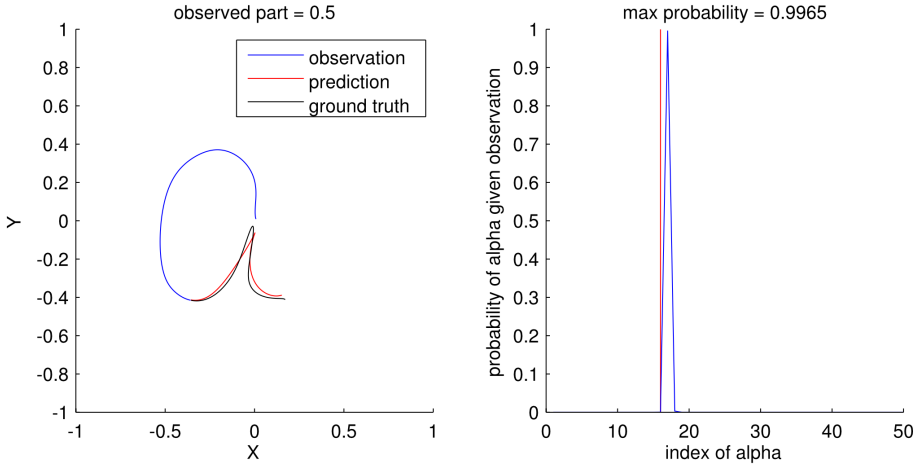


Figure 40: Completion after training with EM.

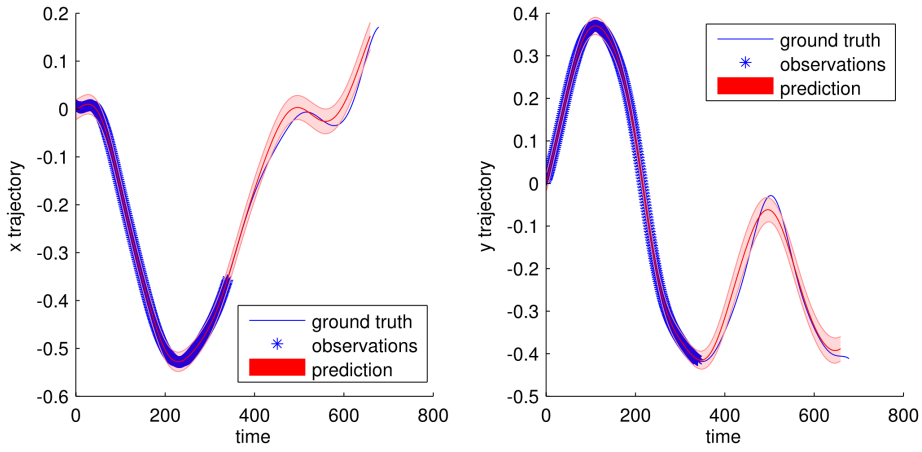


Figure 41: Completion after training with EM (X and Y trajectories).

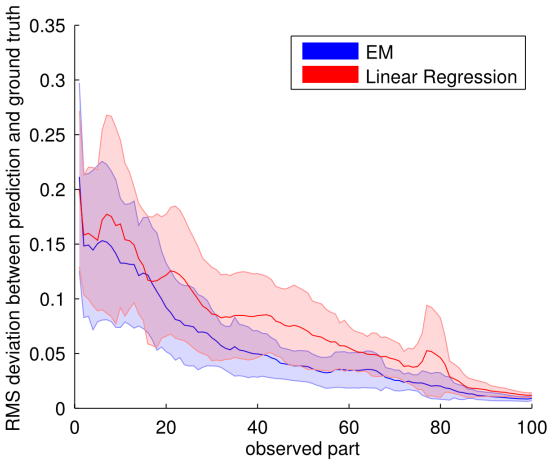


Figure 42: EM versus Linear Regression with partially observed training trajectories. The prediction performance after training with the EM method is visibly better than the prediction performance after training with the Linear Regression method.

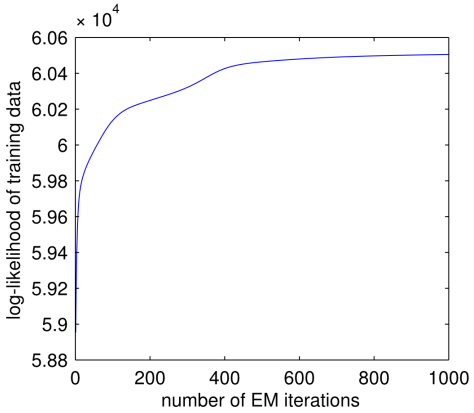


Figure 43: Number of EM iterations versus log-likelihood of the training data. The log-likelihood of the training data increases monotonically with the number of EM iterations.

5 Conclusion and Future Work

This work presented how probabilistic movement representations can be used to model human-robot interactions. Specifically this work proposed the concept of a Mixture of Interaction Primitives where Gaussian Mixture Models are used to learn multiple interaction patterns from unlabeled data. The multimodal prior probability distribution is obtained over parameterized demonstration trajectories of two agents working in collaboration. During execution, the algorithm selects the mixture component with the highest probability given the observation of the human, which is then conditioned to infer the appropriate robot reaction. The proposed method is able to learn and recognize multiple human-robot collaboration tasks from an arbitrary number of demonstrations consisting of unlabeled interaction patterns, what was not possible with previous Interaction Primitive frameworks [1, 16].

One of the issues faced when using Interaction Primitives is the fact that humans may change the speed of the execution of a movement. This change in speed requires adaptation in the phase, what was not part of previous Interaction Primitives formulations. This work addressed this limitation by presenting an algorithm for online phase estimation of trajectories that can be used in the context of Human-Robot Interaction to predict the trajectory of human movements with different durations given the beginning of the movements and to estimate the best reaction of the robot.

Furthermore, an Expectation-Maximization (EM) algorithm was presented that generates Probabilistic Movement Primitives (ProMPs) that fit well the training trajectories even in the face of missing data points. The ProMPs learned by the EM approach are able to estimate the phase and predict the unobserved part of test trajectories better than the ProMPs learned by using a Linear Regression approach conventionally applied to train Interaction Primitives so far [1, 16].

As future work, the Mixture of Interaction Primitives and the algorithm for online phase estimation will be brought together to allow for reacting to actions with different durations. The estimation of the phase of execution of the primitive for switching tasks in real time will also be addressed.

Extensions of this work are being considered, where the ProMPs are conditioned on other variables of interest instead of on human movements. For example, the same framework can be used to correlate joint and end-effector trajectories of the same robot to learn nonlinear forward/inverse kinematics models. Similarly the Mixture of Interaction Primitives can be used to correlate the interaction between motor commands and joint trajectories to learn inverse dynamics models.

As already pointed out in Section 4.2.2, an extension of the EM algorithm proposed in Section 4.2.1 will be designed to deal with more complex phase variables, which may result in a better performance on phase estimation, accounting for local variability in speed of execution.

6 Papers Related to this Thesis

The work presented in this thesis has been partially presented in or extended from the following papers:

G.J. Maeda, **M. Ewerton**, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann. Learning interaction for collaborative tasks with probabilistic movement primitives. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2014.

M. Ewerton, G. Neumann, R. Lioutikov, H. Ben Amor, J. Peters, and Guilherme Maeda. Learning Multiple Collaborative Tasks with a Mixture of Interaction Primitives. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

M. Ewerton, G. Maeda, J. Peters, G. Neumann. Learning Motor Skills from Partially Observed Movements Executed at Different Speeds. Submitted to *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

References

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

[2] Heni Ben Amor, David Vogt, Marco Ewerton, Erik Berger, Bernhard Jung, and Jan Peters. Learning responsive robot behavior by imitation. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3257–3264, 2013.

[3] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[4] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 994–, Washington, DC, USA, 1997. IEEE Computer Society.

[5] Matthew Brand. Coupled hidden markov models for modeling interacting processes. Technical report, 1997.

[6] Sylvain Calinon, Eric L Sauser, Aude G Billard, and Darwin G Caldwell. Evaluation of a probabilistic approach to learn and reproduce gestures by imitation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2671–2676. IEEE, 2010.

[7] Adam Coates, Pieter Abbeel, and Andrew Y Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151. ACM, 2008.

[8] Peter Englert and Marc Toussaint. Reactive phase and task space adaptation for robust motion execution. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 109–116. IEEE, 2014.

[9] K.P. Hawkins, N. Vo, S. Bansal, and Aaron F. Bobic. Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2013.

[10] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[11] Hema Swetha Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. In *Robotics: Science and Systems*, 2013.

[12] Tomas Kulvicius, Martin Biehl, Mohamad Javad Aein, Minija Tamosiunaite, and Florentin Wörgötter. Interaction learning for dynamic movement primitives used in cooperative robotic tasks. *Robotics and Autonomous Systems*, 61(12):1450–1459, 2013.

[13] M. Lawitzky, J.R. Medina, Dongheui Lee, and S. Hirche. Feedback motion planning and learning from demonstration in physical robotic assistance: differences and synergies. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3646–3652, Oct 2012.

[14] Dongheui Lee, Christian Ott, Yoshihiko Nakamura, and Gerd Hirzinger. Physical human robot interaction in imitation learning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3439–3440. IEEE, 2011.

[15] Baldin Llorens-Bonilla and H Harry Asada. A robot on the shoulder: Coordinated human-wearable robot control using coloured petri nets and partial least squares predictions. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*, 2014.

[16] G.J. Maeda, M. Ewerton, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann. Learning interaction for collaborative tasks with probabilistic movement primitives. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, 2014.

[17] N.M. Oliver, B. Rosario, and AP. Pentland. A bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):831–843, Aug 2000.

[18] A. Paraschos, C. Daniel, J. Peters, and G Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2616–2624, 2013.

[19] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook. *Technical University of Denmark*, pages 7–15, 2008.

[20] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras. Learning collaborative impedance-based robot behaviors. In *AAAI Conference on Artificial Intelligence*, Bellevue, Washington, USA, 2013.

[21] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.

[22] Yasufumi Tanaka, Jun Kinugawa, Yusuke Sugahara, and Kazuhiro Kosuge. Motion planning with worker's trajectory prediction for assembly task partner robot. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1525–1532. IEEE, 2012.

[23] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.Y. Fu, K. Goldberg, and P. Abbeel. Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pages 2074–2081, 2010.

[24] Rok Vuga, Bojan Nemec, and Ales Ude. Speed profile optimization through directed explorative learning. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 547–553. IEEE, 2014.

[25] Zheng Wang, Angelika Peer, and Martin Buss. An hmm approach to realistic haptic human-robot interaction. In *EuroHaptics conference, 2009 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics 2009. Third Joint*, pages 374–379. IEEE, 2009.

[26] Zhikun Wang, Katharina Mülling, Marc Peter Deisenroth, Heni Ben Amor, David Vogt, Bernhard Schölkopf, and Jan Peters. Probabilistic movement modeling for intention inference in human–robot interaction. *The International Journal of Robotics Research*, 32(7):841–858, 2013.

Appendix

A Derivation of EM for ProMPs

Statement of the problem: Assume a number n of training trajectories. Each training trajectory can be approximated by a weighted sum of normalized Gaussian basis functions. Therefore, each training trajectory can be compactly represented by a weight vector $\boldsymbol{w}$. Those weight vectors, however, are unobserved. Find the mean and the covariance of the probability distribution over weight vectors $p(\boldsymbol{w})$, assuming $p(\boldsymbol{w})$ a single multivariate Gaussian. The training trajectories may have missing values.

This can be formulated as an Expectation-Maximization problem, where $\boldsymbol{w}$ is a hidden vector-valued variable.

The objective is to maximize the joint probability of the observations D_i of each training trajectory with index i with respect to θ,

$$\prod_i p_\theta\left(D_i\right) = \prod_i \int p\left(D_i|\boldsymbol{w},\alpha_i\right) p(\boldsymbol{w})\,\mathrm{d}\boldsymbol{w}, \tag{35}$$

where $p\left(D_i|\boldsymbol{w},\alpha_i\right) = \mathcal{N}\left(D_i;\boldsymbol{A}_i\boldsymbol{w},\sigma^2\boldsymbol{I}\right)$.

The matrix $\boldsymbol{A}_i$ has the values of each basis function evaluated at the observed time steps. Assuming one of the observations D_i extends from time step 1 to time step m, where $m \leq T_i$, T_i being the total number of time steps of the trajectory i,

$$D_i = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{pmatrix} = \boldsymbol{A}_i\boldsymbol{w} = \begin{pmatrix} \psi_{1,1} & \psi_{2,1} & \cdots & \psi_{N,1} \\ \psi_{1,2} & \psi_{2,2} & \cdots & \psi_{N,2} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{1,m} & \psi_{2,m} & \cdots & \psi_{N,m} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}. \tag{36}$$

The observations D_i do not need to be comprised of positions at successive time steps. There may be gaps between the observed parts as well.

The parameter θ in $p_\theta\left(D_i\right)$ stands for the set $\{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$, the mean and the covariance of the Gaussian distribution over the weight vectors $\boldsymbol{w}$.

The estimation of θ goes as follows. First, θ is initialized with arbitrary values,

$$\theta_0 = \{\boldsymbol{\mu}_{w0}, \boldsymbol{\Sigma}_{w0}\}. \tag{37}$$

Next, the algorithm performs the Expectation Step (E-Step), by determining the function $Q\left(\theta,\theta_0\right)$,

$$Q\left(\theta,\theta_0\right) = \sum_i \mathbb{E}_{\theta_0}\left(\log p_\theta\left(D_i,\boldsymbol{w},\alpha_i\right)|D = D_i\right) \tag{38}$$

and perform the Maximization Step (M-Step),

$$\theta \in \arg\max_\theta Q\left(\theta,\theta_0\right). \tag{39}$$

The algorithm keeps iterating over the E-Step and the M-Step until $\prod_i p_\theta\left(D_i\right)$ converges.

A.1 E-Step

In order to determine the function $Q(\theta, \theta_0)$, we start by defining the term $p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right)$,

$$p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right) = p\left(D_i|\boldsymbol{w}, \alpha_i\right) p\left(\boldsymbol{w}\right) \tag{40}$$

$$= \mathcal{N}\left(D_i; \boldsymbol{A}_i\boldsymbol{w}, \sigma^2\boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{w}; \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right). \tag{41}$$

Then, we take the logarithm of $p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right)$,

$$\log p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right) = \log \mathcal{N}\left(D_i; \boldsymbol{A}_i\boldsymbol{w}, \sigma^2\boldsymbol{I}\right) + \log \mathcal{N}\left(\boldsymbol{w}; \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right). \tag{42}$$

Writing explicitly the Gaussians and taking their logarithm, we get

$$\mathcal{N}\left(D_i; \boldsymbol{A}_i\boldsymbol{w}, \sigma^2\boldsymbol{I}\right) = \frac{1}{\sqrt{|2\pi\sigma^2\boldsymbol{I}|}} \exp\left(-\frac{1}{2}\left(D_i - \boldsymbol{A}_i\boldsymbol{w}\right)^T \left(\sigma^2\boldsymbol{I}\right)^{-1} \left(D_i - \boldsymbol{A}_i\boldsymbol{w}\right)\right), \tag{43}$$

$$\mathcal{N}\left(\boldsymbol{w}; \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{\mu}_w\right)^T \boldsymbol{\Sigma}_w^{-1} \left(\boldsymbol{w} - \boldsymbol{\mu}_w\right)\right), \tag{44}$$

$$\begin{aligned}\log p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right) = &\log\left(\frac{1}{\sqrt{|2\pi\sigma^2\boldsymbol{I}|}}\right) - \frac{1}{2}\left(D_i - \boldsymbol{A}_i\boldsymbol{w}\right)^T \left(\sigma^2\boldsymbol{I}\right)^{-1} \left(D_i - \boldsymbol{A}_i\boldsymbol{w}\right) \\ &+ \log\left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}}\right) - \frac{1}{2}\left(\boldsymbol{w} - \boldsymbol{\mu}_w\right)^T \boldsymbol{\Sigma}_w^{-1} \left(\boldsymbol{w} - \boldsymbol{\mu}_w\right).\end{aligned} \tag{45}$$

The expectation term of $Q\left(\theta, \theta_0\right)$ can be written as

$$\mathbb{E}_{\theta_0}\left(\log p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right) | D = D_i\right) = \int \log p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right) p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right) \mathrm{d}\boldsymbol{w}. \tag{46}$$

In order to calculate this expectation term, we will need to define $p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right)$,

$$p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right) \propto p_{\theta_0}\left(D_i|\boldsymbol{w}\right) p_{\theta_0}\left(\boldsymbol{w}\right), \tag{47}$$

$$p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right) \propto \mathcal{N}\left(D_i; \boldsymbol{A}_i\boldsymbol{w}, \sigma^2\boldsymbol{I}\right) \mathcal{N}\left(\boldsymbol{w}; \boldsymbol{\mu}_{w0}, \boldsymbol{\Sigma}_{w0}\right). \tag{48}$$

Using (45) and (46), we get

$$\begin{aligned}\mathbb{E}_{\theta_0}\left(\log p_\theta\left(D_i, \boldsymbol{w}, \alpha_i\right)|D = D_i\right) = &\int \log\left(\frac{1}{\sqrt{|2\pi\sigma^2\boldsymbol{I}|}}\right) p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right) \mathrm{d}\boldsymbol{w} \\ &- \frac{1}{2}\int \left(D_i - \boldsymbol{A}_i\boldsymbol{w}\right)^T \left(\sigma^2\boldsymbol{I}\right)^{-1} \left(D_i - \boldsymbol{A}_i\boldsymbol{w}\right) p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right) \mathrm{d}\boldsymbol{w} \\ &+ \int \log\left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}}\right) p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right) \mathrm{d}\boldsymbol{w} \\ &- \frac{1}{2}\int \left(\boldsymbol{w} - \boldsymbol{\mu}_w\right)^T \boldsymbol{\Sigma}_w^{-1} \left(\boldsymbol{w} - \boldsymbol{\mu}_w\right) p_{\theta_0}\left(\boldsymbol{w}|D = D_i\right) \mathrm{d}\boldsymbol{w}.\end{aligned} \tag{49}$$

Now, let us expand the last term in (49):

$$\begin{aligned}
&\left(\boldsymbol{w}-\boldsymbol{\mu}_w\right)^T \boldsymbol{\Sigma}_w^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_w\right), \\
&\left(\boldsymbol{w}^T-\boldsymbol{\mu}_w^T\right) \boldsymbol{\Sigma}_w^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_w\right), \\
&\left(\boldsymbol{w}^T-\boldsymbol{\mu}_w^T\right)\left(\boldsymbol{\Sigma}_w^{-1} \boldsymbol{w}-\boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w\right), \\
&\boldsymbol{w}^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{w}-\boldsymbol{w}^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w-\boldsymbol{\mu}_w^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{w}+\boldsymbol{\mu}_w^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w, \\
&\boldsymbol{w}^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{w}-2 \boldsymbol{w}^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w+\boldsymbol{\mu}_w^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w,
\end{aligned} \tag{50}$$

$$\begin{aligned}
-\frac{1}{2} \int &\left(\boldsymbol{w}-\boldsymbol{\mu}_w\right)^T \boldsymbol{\Sigma}_w^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_w\right) p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right) \mathrm{d} \boldsymbol{w}= \\
&-\frac{1}{2} \int \boldsymbol{w}^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{w} p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right) \mathrm{d} \boldsymbol{w}-\frac{1}{2} \int-2 \boldsymbol{w}^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right) \mathrm{d} \boldsymbol{w} \\
&-\frac{1}{2} \int \boldsymbol{\mu}_w^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_w p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right) \mathrm{d} \boldsymbol{w}.
\end{aligned} \tag{51}$$

Expanding (48),

$$\begin{aligned}
p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right) \propto & \frac{1}{\sqrt{\left|2 \pi \sigma^2 \boldsymbol{I}\right|}} \exp\left(-\frac{1}{2}\left(D_i-\boldsymbol{A}_i \boldsymbol{w}\right)^T\left(\sigma^2 \boldsymbol{I}\right)^{-1}\left(D_i-\boldsymbol{A}_i \boldsymbol{w}\right)\right) \\
& \times \frac{1}{\sqrt{\left|2 \pi \boldsymbol{\Sigma}_{w 0}\right|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{w}-\boldsymbol{\mu}_{w 0}\right)^T \boldsymbol{\Sigma}_{w 0}^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_{w 0}\right)\right),
\end{aligned} \tag{52}$$

$$\begin{aligned}
&p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right) \propto \\
&\quad \exp\left(-\frac{1}{2}\left(D_i-\boldsymbol{A}_i \boldsymbol{w}\right)^T\left(\sigma^2 \boldsymbol{I}\right)^{-1}\left(D_i-\boldsymbol{A}_i \boldsymbol{w}\right)-\frac{1}{2}\left(\boldsymbol{w}-\boldsymbol{\mu}_{w 0}\right)^T \boldsymbol{\Sigma}_{w 0}^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_{w 0}\right)\right).
\end{aligned} \tag{53}$$

At the same time, we know that $p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right)$ is a Gaussian, since $p_{\theta_0}(\boldsymbol{w})$ is a Gaussian. Therefore, we can write $p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right)$ as

$$\begin{aligned}
p_{\theta_0}\left(\boldsymbol{w} \mid D=D_i\right) &= \mathcal{N}\left(\boldsymbol{w} ; \boldsymbol{\mu}_{w i}, \boldsymbol{\Sigma}_{w i}\right) \\
&\propto \exp\left(-\frac{1}{2}\left(\boldsymbol{w}-\boldsymbol{\mu}_{w i}\right)^T \boldsymbol{\Sigma}_{w i}^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_{w i}\right)\right).
\end{aligned} \tag{54}$$

The argument of the exponential function in (53) can be further expanded, ignoring the coefficients $-\frac{1}{2}$ for now,

$$\begin{aligned}
&\left(D_i^T-\boldsymbol{w}^T \boldsymbol{A}_i^T\right)\left(\left(\sigma^2 \boldsymbol{I}\right)^{-1} D_i-\left(\sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{A}_i \boldsymbol{w}\right)+\left(\boldsymbol{w}^T-\boldsymbol{\mu}_{w 0}^T\right)\left(\boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{w}-\boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{\mu}_{w 0}\right)= \\
&\quad D_i^T\left(\sigma^2 \boldsymbol{I}\right)^{-1} D_i-D_i^T\left(\sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{A}_i \boldsymbol{w}-\boldsymbol{w}^T \boldsymbol{A}_i^T\left(\sigma^2 \boldsymbol{I}\right)^{-1} D_i+\boldsymbol{w}^T \boldsymbol{A}_i^T\left(\sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{A}_i \boldsymbol{w} \\
&\qquad +\boldsymbol{w}^t \boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{w}-\boldsymbol{w}^T \boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{\mu}_{w 0}-\boldsymbol{\mu}_{w 0}^T \boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{w}+\boldsymbol{\mu}_{w 0}^T \boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{\mu}_{w 0}= \\
&\quad D_i^T\left(\sigma^2 \boldsymbol{I}\right)^{-1} D_i-2 D_i^T\left(\sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{A}_i \boldsymbol{w}+\boldsymbol{w}^T \boldsymbol{A}_i^T\left(\sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{A}_i \boldsymbol{w} \\
&\qquad +\boldsymbol{w}^T \boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{w}-2 \boldsymbol{w}^T \boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{\mu}_{w 0}+\boldsymbol{\mu}_{w 0}^T \boldsymbol{\Sigma}_{w 0}^{-1} \boldsymbol{\mu}_{w 0}
\end{aligned} \tag{55}$$

The argument of the exponential function in (54) can be similarly expanded,

$$\begin{aligned}\left(\boldsymbol{w}^T-\boldsymbol{\mu}_w^T\right)\left(\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{w}-\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}\right)=\\ \boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{w}-\boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}-\boldsymbol{\mu}_{wi}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{w}+\boldsymbol{\mu}_{wi}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}=\\ \boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{w}-2\boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}+\boldsymbol{\mu}_{wi}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}.\end{aligned}\tag{56}$$

Comparing (55) with (56), we get

$$\begin{aligned}\boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{w}&=\boldsymbol{w}^T\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i\boldsymbol{w}+\boldsymbol{w}^T\boldsymbol{\Sigma}_{w0}^{-1}\boldsymbol{w},\\ \boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{w}&=\boldsymbol{w}^T\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i\boldsymbol{w}+\boldsymbol{w}^T\boldsymbol{\Sigma}_{w0}^{-1}\boldsymbol{w},\\ \boldsymbol{\Sigma}_{wi}^{-1}&=\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i+\boldsymbol{\Sigma}_{w0}^{-1},\\ \boldsymbol{\Sigma}_{wi}&=\left(\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i+\boldsymbol{\Sigma}_{w0}^{-1}\right)^{-1}.\end{aligned}\tag{57}$$

$$\begin{aligned}-2\boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}&=-2D_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i\boldsymbol{w}-2\boldsymbol{w}^T\boldsymbol{\Sigma}_{w0}^{-1}\boldsymbol{\mu}_{w0},\\ \boldsymbol{w}^T\boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}&=\boldsymbol{w}^T\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}D_i+\boldsymbol{w}^T\boldsymbol{\Sigma}_{w0}^{-1}\boldsymbol{\mu}_{w0},\\ \boldsymbol{\Sigma}_{wi}^{-1}\boldsymbol{\mu}_{wi}&=\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}D_i+\boldsymbol{\Sigma}_{w0}^{-1}\boldsymbol{\mu}_{w0},\\ \boldsymbol{\mu}_{wi}&=\boldsymbol{\Sigma}_{wi}\left(\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}D_i+\boldsymbol{\Sigma}_{w0}^{-1}\boldsymbol{\mu}_{w0}\right).\end{aligned}\tag{58}$$

Now let us go back to (46),

$$\int\log p_\theta\left(D_i,\boldsymbol{w},\alpha_i\right)p_{\theta_0}\left(\boldsymbol{w}|D=D_i\right)\mathrm{d}\boldsymbol{w}=\int\log p_\theta\left(D_i,\boldsymbol{w},\alpha_i\right)\mathcal{N}\left(\boldsymbol{w};\boldsymbol{\mu}_{wi},\boldsymbol{\Sigma}_{wi}\right)\mathrm{d}\boldsymbol{w}.\tag{59}$$

This expression can be expanded, using (49),

$$\begin{aligned}&\int\log\left(\frac{1}{\sqrt{|2\pi\sigma^2\boldsymbol{I}|}}\right)\mathcal{N}\left(\boldsymbol{w};\boldsymbol{\mu}_{wi},\boldsymbol{\Sigma}_{wi}\right)\mathrm{d}\boldsymbol{w}\\ &\quad-\frac{1}{2}\int\left(D_i-\boldsymbol{A}_i\boldsymbol{w}\right)^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\left(D_i-\boldsymbol{A}_i\boldsymbol{w}\right)\mathcal{N}\left(\boldsymbol{w};\boldsymbol{\mu}_{wi},\boldsymbol{\Sigma}_{wi}\right)\mathrm{d}\boldsymbol{w}\\ &\quad+\int\log\left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}}\right)\mathcal{N}\left(\boldsymbol{w};\boldsymbol{\mu}_{wi},\boldsymbol{\Sigma}_{wi}\right)\mathrm{d}\boldsymbol{w}\\ &\quad-\frac{1}{2}\int\left(\boldsymbol{w}-\boldsymbol{\mu}_w\right)^T\boldsymbol{\Sigma}_w^{-1}\left(\boldsymbol{w}-\boldsymbol{\mu}_w\right)\mathcal{N}\left(\boldsymbol{w};\boldsymbol{\mu}_{wi},\boldsymbol{\Sigma}_{wi}\right)\mathrm{d}\boldsymbol{w}.\end{aligned}\tag{60}$$

Using expectation identities provided by [19], we get finally this expression for Q,

$$\begin{aligned}Q=&\sum_i\log\left(\frac{1}{\sqrt{|2\pi\sigma^2\boldsymbol{I}|}}\right)\\ &-\frac{1}{2}\sum_i\left(D_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}D_i-2D_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i\boldsymbol{\mu}_{wi}+\boldsymbol{\mu}_{w_i}^T\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i\boldsymbol{\mu}_{w_i}+\mathrm{Tr}\left(\boldsymbol{A}_i^T\left(\sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{A}_i\boldsymbol{\Sigma}_{w_i}\right)\right)\\ &+\sum_i\log\left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}}\right)\\ &-\frac{1}{2}\sum_i\left(\left(\boldsymbol{\mu}_{wi}-\boldsymbol{\mu}_w\right)^T\boldsymbol{\Sigma}_w^{-1}\left(\boldsymbol{\mu}_{wi}-\boldsymbol{\mu}_w\right)+\mathrm{Tr}\left(\boldsymbol{\Sigma}_w^{-1}\boldsymbol{\Sigma}_{w_i}\right)\right).\end{aligned}\tag{61}$$

A.2 M-Step

In the Maximization Step, we want to find the set of parameters θ that maximize $Q\left(\theta, \theta_0\right)$. The set of parameters θ comprises in this case $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$.

In order to find the $\boldsymbol{\mu}_w$ that maximizes Q, we take the partial derivative of Q with respect to $\boldsymbol{\mu}_w$ and set it equal to 0,

$$\begin{aligned}
\frac{\partial Q}{\partial \boldsymbol{\mu}_w} = \sum_i \left(\boldsymbol{\mu}_{wi}^T \boldsymbol{\Sigma}_w^{-1} - \frac{1}{2} 2 \boldsymbol{\mu}_w^T \boldsymbol{\Sigma}_w^{-1} \right) = 0, \\
\sum_i \boldsymbol{\mu}_{wi}^T \boldsymbol{\Sigma}_w^{-1} = \sum_i \boldsymbol{\mu}_w^T \boldsymbol{\Sigma}_w^{-1}, \\
\sum_i \boldsymbol{\mu}_{wi}^T = \boldsymbol{\mu}_w^T n, \\
\boldsymbol{\mu}_w^* = \frac{\sum_i \boldsymbol{\mu}_{wi}}{n}.
\end{aligned} \tag{62}$$

Now, let us apply the same procedure to find the $\boldsymbol{\Sigma}_w$ that maximizes Q,

$$\begin{aligned}
\frac{\partial Q}{\partial \boldsymbol{\Sigma}_w} = \sum_i \left(\frac{\partial \log\left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}} \right)}{\partial \boldsymbol{\Sigma}_w} \right) \\
- \frac{1}{2} \sum_i \frac{\partial \left(\left(\boldsymbol{\mu}_{wi} - \boldsymbol{\mu}_w \right)^T \boldsymbol{\Sigma}_w^{-1} \left(\boldsymbol{\mu}_{wi} - \boldsymbol{\mu}_w \right) + \mathrm{Tr}\left(\boldsymbol{\Sigma}_w^{-1} \boldsymbol{\Sigma}_{wi} \right) \right)}{\partial \boldsymbol{\Sigma}_w} = 0.
\end{aligned} \tag{63}$$

Let us first reformulate the logarithm term in the argument of the first sum,

$$\begin{aligned}
\log\left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}} \right) &= \log |2\pi\boldsymbol{\Sigma}_w|^{-\frac{1}{2}}, \\
&= -\frac{1}{2} \log |2\pi\boldsymbol{\Sigma}_w|, \\
&= -\frac{1}{2} \log\left((2\pi)^N |\boldsymbol{\Sigma}_w| \right), \\
&= -\frac{1}{2} \log (2\pi)^N - \frac{1}{2} \log |\boldsymbol{\Sigma}_w|.
\end{aligned} \tag{64}$$

Now we can compute the derivative in the argument of the first sum,

$$\begin{aligned}
\frac{\partial \log\left(\frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}_w|}} \right)}{\partial \boldsymbol{\Sigma}_w} &= \frac{\partial \left(-\frac{1}{2} \log |\boldsymbol{\Sigma}_w| \right)}{\partial \boldsymbol{\Sigma}_w}, \\
&= -\frac{1}{2} \frac{\partial \log |\boldsymbol{\Sigma}_w|}{\partial \boldsymbol{\Sigma}_w}, \\
&= -\frac{1}{2} \left(\boldsymbol{\Sigma}_w^{-1} \right)^T.
\end{aligned} \tag{65}$$

Equation (65) uses an identity on derivatives of determinants provided by [19].

Now let us tackle the second sum in (63),

$$\boldsymbol{\mu}_{wi} = \begin{pmatrix} \mu_{w_{i1}} \\ \mu_{w_{i2}} \\ \vdots \\ \mu_{w_{iN}} \end{pmatrix}, \qquad \boldsymbol{\mu}_w = \begin{pmatrix} \mu_{w_1} \\ \mu_{w_2} \\ \vdots \\ \mu_{w_N} \end{pmatrix}, \tag{66}$$

$$\boldsymbol{e}_i = \boldsymbol{\mu}_{wi} - \boldsymbol{\mu}_w = \begin{pmatrix} \mu_{w_{i1}} - \mu_{w1} \\ \mu_{w_{i2}} - \mu_{w2} \\ \vdots \\ \mu_{w_{iN}} - \mu_{wN} \end{pmatrix}, \tag{67}$$

$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \vdots \\ \boldsymbol{e}_n^T \end{pmatrix}, \tag{68}$$

$$\sum_i \boldsymbol{e}_i^T \boldsymbol{\Sigma}_w^{-1} \boldsymbol{e}_i = \mathrm{Tr}\left(\boldsymbol{E}\boldsymbol{\Sigma}_w^{-1}\boldsymbol{E}^T\right), \tag{69}$$

$$-\frac{1}{2}\frac{\partial \mathrm{Tr}\left(\boldsymbol{E}\boldsymbol{\Sigma}_w^{-1}\boldsymbol{E}^T\right)}{\partial \boldsymbol{\Sigma}_w} = \frac{1}{2}\left(\boldsymbol{\Sigma}_w^{-1}\boldsymbol{E}^T\boldsymbol{E}\boldsymbol{\Sigma}_w^{-1}\right)^T, \tag{70}$$

$$-\frac{1}{2}\sum_i \frac{\partial \mathrm{Tr}\left(\boldsymbol{\Sigma}_w^{-1}\boldsymbol{\Sigma}_{w_i}\right)}{\partial \boldsymbol{\Sigma}_w} = \frac{1}{2}\sum_i \left(\boldsymbol{\Sigma}_w^{-1}\boldsymbol{\Sigma}_{w_i}\boldsymbol{\Sigma}_w^{-1}\right)^T, \tag{71}$$

using the equation

$$\frac{\partial \mathrm{Tr}\left(\boldsymbol{I}\boldsymbol{\Sigma}_w^{-1}\boldsymbol{\Sigma}_{wi}\right)}{\partial \boldsymbol{\Sigma}_w} = -\left(\boldsymbol{\Sigma}_w^{-1}\boldsymbol{\Sigma}_{wi}\boldsymbol{\Sigma}_w^{-1}\right)^T. \tag{72}$$

Equations (70) and (72) use identities on derivatives of traces provided by [19].
Finally, using (63), (65), (70) and (71), we can find the $\boldsymbol{\Sigma}_w$ that maximizes Q,

$$\begin{gathered}
-\frac{1}{2}n\left(\boldsymbol{\Sigma}_w^{-1}\right)^T + \frac{1}{2}\left(\boldsymbol{\Sigma}_w^{-1}\boldsymbol{E}^T\boldsymbol{E}\boldsymbol{\Sigma}_w^{-1}\right)^T + \frac{1}{2}\sum_i \left(\boldsymbol{\Sigma}_w^{-1}\boldsymbol{\Sigma}_{w_i}\boldsymbol{\Sigma}_w^{-1}\right)^T = 0, \\
-n\left(\boldsymbol{\Sigma}_w^{-1}\right)^T + \left(\boldsymbol{\Sigma}_w^{-1}\right)^T \boldsymbol{E}^T\boldsymbol{E}\left(\boldsymbol{\Sigma}_w^{-1}\right)^T + \sum_i \left(\left(\boldsymbol{\Sigma}_w^{-1}\right)^T \boldsymbol{\Sigma}_{w_i}\left(\boldsymbol{\Sigma}_w^{-1}\right)^T\right) = 0, \\
-n + \left(\boldsymbol{\Sigma}_w^{-1}\right)^T \boldsymbol{E}^T\boldsymbol{E} + \left(\boldsymbol{\Sigma}_w^{-1}\right)^T \sum_i \boldsymbol{\Sigma}_{w_i} = 0, \\
\left(\boldsymbol{\Sigma}_w^{-1}\right)^T \left(\boldsymbol{E}^T\boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{w_i}\right) = n, \\
\left(\boldsymbol{E}^T\boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{wi}\right)^T \boldsymbol{\Sigma}_w^{-1} = n, \\
\left(\boldsymbol{E}^T\boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{w_i}\right) \boldsymbol{\Sigma}_w^{-1} = n, \\
\boldsymbol{\Sigma}_w^{-1} = \left(\boldsymbol{E}^T\boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{w_i}\right)^{-1} n, \\
\boldsymbol{\Sigma}_w^* = \frac{\left(\boldsymbol{E}^T\boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{w_i}\right)}{n}.
\end{gathered} \tag{73}$$