2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Learning Motor Skills from Partially Observed Movements Executed at Different Speeds

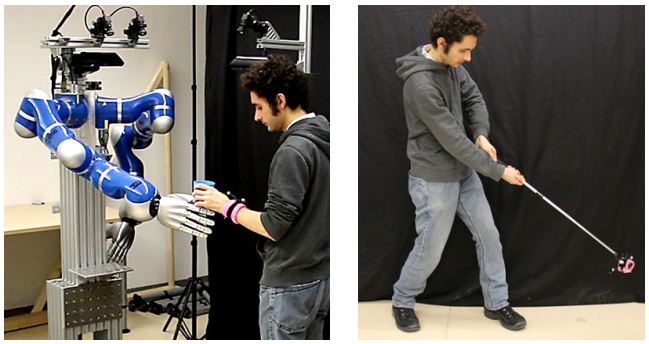
Marco Ewerton[1], Guilherme Maeda[1], Jan Peters[1,2] and Gerhard Neumann[3]

***Abstract*— Learning motor skills from multiple demonstrations presents a number of challenges. One of those challenges is the occurrence of occlusions and lack of sensor coverage, which may corrupt part of the recorded data. Another issue is the variability in speed of execution of the demonstrations, which may require a way of finding the correspondence between the time steps of the different demonstrations. In this paper, an approach to learn motor skills is proposed that accounts both for spatial and temporal variability of movements. This approach, based on an *Expectation-Maximization* algorithm to learn *Probabilistic Movement Primitives*, also allows for learning motor skills from partially observed demonstrations, which may result from occlusion or lack of sensor coverage. An application of the algorithm proposed in this work lies in the field of Human-Robot Interaction when the robot has to react to human movements executed at different speeds. Experiments in which a robotic arm receives a cup handed over by a human illustrate this application. The capabilities of the algorithm in learning and predicting movements are also evaluated in experiments using a data set of letters and a data set of golf putting movements.**

I. Introduction

Researchers investigating how a robot can learn motor skills from multiple human demonstrations face numerous challenges. The recorded data may, for example, be corrupted by occlusions or lack of sensor coverage and the demonstrations may vary considerably in speed of execution. It is necessary to identify the corrupted data or to be able to extract useful information from the corrupted recordings. A common approach to represent the learned movement is to use time-dependent models. In this case, the variability in speed of execution requires a way of finding the correspondence between the time steps of the different demonstrations. This paper presents an approach to deal with both challenges. This approach is based on an Expectation-Maximization (EM) algorithm to learn Probabilistic Movement Primitives (ProMPs).

ProMPs [1] are movement representations based on a probability distribution over trajectories. By representing movements with ProMPs, it is possible to condition the distributions over trajectories on observed positions. One of the applications of this framework lies in the field of Human-Robot Interaction, where it is possible to use probabilistic operations to condition the reactions of the robot on the actions of the human [2], [3].



(a) (b)

Fig. 1. Experimental setups to evaluate our EM algorithm to learn ProMPs. (a) The robot receives a cup from a human who executes the handover at different speeds. (b) A golf putting data set is used to evaluate different formulations of our phase function.

This work builds upon the concept of ProMPs and proposes learning from multiple demonstrations probability distributions not only over trajectories but also over speed profiles or the phase of the movement. The phase of the movement is a function of time that can be associated with a movement. By changing the phase of a movement, it is possible to change its speed of execution. Learning a distribution over phase profiles is important to learn motor skills from demonstrations executed at different speeds and to allow a robot to react to human actions executed at different speeds.

The remainder of this paper is organized as follows: Section II presents related work. Section III presents the main contribution of this work, an EM algorithm to learn ProMPs, capturing variability in the trajectories and in the phase profiles. Section IV describes a number of experiments that evaluate the proposed algorithm. Our experiments evaluate the capabilities of the presented algorithm in learning movements from demonstrations with missing data. Furthermore, we evaluate two different formulations of the phase function: a simple formulation with one single phase parameter and a more general formulation with multiple phase parameters. Section V summarizes the paper and presents ideas for future work.

II. Related Work

Time-dependent movement representations are heavily used due to their small number of parameters. The most prominent time-dependent movement representation is the

[1]Intelligent Autonomous Systems group, department of Computer Science, Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany {ewerton, maeda, peters}@ias.tu-darmstadt.de

[2]Max Planck Institute for Intelligent Systems, Spemannstr. 38, 72076 Tuebingen, Germany jan.peters@tuebingen.mpg.de

[3]Computational Learning for Autonomous Systems group, department of Computer Science, Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany geri@robot-learning.de

Dynamical Movement Primitive (DMP) [4], which comprises a proportional-derivative (PD) controller and a non-linear 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.

In the Probabilistic Movement Primitive (ProMP), which is also a time-dependent movement representation, not a forcing function, but the trajectories themselves are approximated by a weighted sum of time-dependent basis functions.

In both approaches, the time-dependency of the basis functions can be given through a phase variable, which is in turn a function of time. Defining the basis functions through a phase variable allows for time-rescaling trajectories and synchronizing the movements of different limbs.

Those formulations of movement primitives have found a large number of applications. They are used, for instance, to model Human-Robot Interaction. The Interaction Primitives (IPs), proposed by Ben Amor at al. [5], are based on DMPs. The Interaction Probabilistic Movement Primitives, proposed by Maeda et al. [2], and the Mixture of Interaction Primitives, proposed by Ewerton et al. [3], are based on ProMPs. All those interaction models are time-dependent. In their current form, those models require the time-alignment of the demonstrations. In [5], Dynamic Time Warping [6] is used to time-align the demonstrations. In [2], a local optimization method is proposed to this end. This same method is also used in [3]. However, as it will be discussed in Section IV-A.2, those procedures do not allow for reacting to actions executed at different speeds.

Time-independent methods to learn trajectories by imitation have also been proposed. For example, Calinon et al. [7] propose an approach based on Hidden Markov Models (HMM) 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 learning motor skills and 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 our work.

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 trajectory and the phase of a movement to changes in the environment, such as changes in the position of the goal or in the position of obstacles. Vuga et al. [9] present a modified form of DMPs 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 given constraints. For example, they speed up the movements of a robot carrying a glass full of liquid without spilling the liquid. Coates et al. [10] 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. [11] 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 online phase estimation presented in this paper 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.

Meier et al. [12] propose a technique to perform movement recognition, prediction and segmentation. This technique assumes the existence of a library of DMPs with specific weights and uses a reformulation of DMPs as linear dynamical systems. Given a partial observation of a trajectory, an EM algorithm is able to estimate the most probable primitive corresponding to this trajectory, the duration and the goal of this primitive. Their work is related to ours, especially since they also use an EM algorithm that can also estimate the completion of trajectories. In our work, an EM algorithm is used to infer a different set of parameters of movement primitives, specifically, the weights and the phase parameters of ProMPs. While their technique is especially suitable for movement segmentation, ours is able to model the correlation between positions at different time steps and the correlation between different joints, what is suitable for Human-Robot Interaction applications, for instance.

III. EM Algorithm to Learn ProMPs

This section explains an EM algorithm to determine the vectors of weights that compactly represent the training movements of a ProMP and a probability distribution over those weights as well as the phase parameters for each training movement and a probability distribution over those phase parameters. First, the algorithm is introduced for the case in which the phase function can be defined by one single parameter. Afterwards, an extension of this method for the case in which the phase function depends on multiple parameters is presented.

A. Algorithm with a Single Phase Parameter

Assume there are a number K of training movements in the form of trajectories. Each training movement can be compactly represented by a vector $\boldsymbol{w}$ of weights for basis functions[1] and, for now, by a single non-negative phase parameter α. The phase function $z(t)$ is defined as $z(t) = \alpha t$ and assumes values between 0 and Z. The higher the value of α, the faster the phase goes from 0 to Z and the faster the movement gets executed.

The phase parameter α_i of a trajectory indexed by i is given by

$$\alpha_i = \frac{Z}{T_i}, \tag{1}$$

[1]We use normalized Gaussian basis functions.

where T_i is the duration of the training trajectory i.

Assuming $p_\theta(\boldsymbol{w})$ is a multivariate Gaussian defined by a set of parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$, i.e. its mean and covariance, one of the objectives of the algorithm is to determine $\boldsymbol{\theta}$. The training trajectories may have missing values, as it would be the case for example due to occlusion or lack of sensor coverage. The observed part of trajectory i is represented by D_i. Finding the parameters $\boldsymbol{\theta}$ of $p_\theta(\boldsymbol{w})$ can be formulated as an Expectation-Maximization [13] problem, where $\boldsymbol{w}$ is a hidden, vector-valued variable.

The joint probability of the observations[2],

$$\prod_i p_{\boldsymbol{\theta}}(D_i|\alpha_i) = \prod_i \int p(D_i|\boldsymbol{w}, \alpha_i)\, p_{\boldsymbol{\theta}}(\boldsymbol{w})\, \mathrm{d}\boldsymbol{w}, \quad (2)$$

with $p(D_i|\boldsymbol{w}, \alpha_i) = \mathcal{N}\left(D_i; \boldsymbol{\Psi}_i \boldsymbol{w}, \sigma^2 \boldsymbol{I}\right)$, must be maximized with respect to $\boldsymbol{\theta}$. The term $\sigma^2 \boldsymbol{I}$ is a covariance matrix modeling a uniform observation noise.

Assuming that an observation D_i comprises the positions q_t from time step $t = 1$ to time step $t = m$, where $m \leq T_i$, this observation can be represented by

$$D_i = [q_1, q_2, \cdots, q_m]^T \approx \boldsymbol{\Psi}_i \boldsymbol{w}_i, \quad (3)$$

where $\boldsymbol{w}_i$ is the unobserved vector of weights that compactly represents the training trajectory i. We assume a number N of basis functions, i.e. the weight vectors are given by $\boldsymbol{w}_i = [w_{i1}, w_{i2}, \cdots, w_{iN}]^T$.

The matrix $\boldsymbol{\Psi}_i$ has each basis function ψ_n evaluated at the phase value correspondent to the observed time steps t. This matrix can be written as

$$\boldsymbol{\Psi}_i = \begin{bmatrix} \psi_1(z_i(1)) & \psi_2(z_i(1)) & \cdots & \psi_N(z_i(1)) \\ \psi_1(z_i(2)) & \psi_2(z_i(2)) & \cdots & \psi_N(z_i(2)) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(z_i(m)) & \psi_2(z_i(m)) & \cdots & \psi_N(z_i(m)) \end{bmatrix}, \quad (4)$$

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

Note that 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.

As the parameter vector $\boldsymbol{w}_i$ for each demonstration D_i is unobserved, an EM algorithm is used to maximize the likelihood of the demonstrations. First, $\boldsymbol{\theta}$ is initialized with a rough estimate $\boldsymbol{\theta}_0 = \{\boldsymbol{\mu}_{w0}, \boldsymbol{\Sigma}_{w0}\}$. Next, the algorithm performs the Expectation step (E step), by estimating for each observation D_i the posterior $p_{\theta_0}(\boldsymbol{w}|D_i, \alpha_i) \propto p(D_i|\boldsymbol{w}, \alpha_i) p_{\theta_0}(\boldsymbol{w})$, which is a Gaussian with mean $\boldsymbol{\mu}_{\boldsymbol{wi}}$ and covariance $\boldsymbol{\Sigma}_{wi}$. The posterior for each observation D_i is necessary to define the complete data log-likelihood $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}})$,

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}) = \sum_i \mathbb{E}_{\boldsymbol{\theta}_{\text{old}}} \left[\log p_{\boldsymbol{\theta}}(D_i, \boldsymbol{w}, \alpha_i) \,|D = D_i\right], \quad (5)$$

which is maximized with respect to $\boldsymbol{\theta}$ in the Maximization step (M step), i.e.

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_{\text{old}}). \quad (6)$$

[2]Assuming $\boldsymbol{w}$ and α independent variables.

The algorithm keeps iterating over the E step and the M step until $\prod_i p_{\boldsymbol{\theta}}(D_i|\alpha_i)$ converges. The parameters $\boldsymbol{\mu}_{\boldsymbol{wi}}$, $\boldsymbol{\Sigma}_{wi}$ and Equation (6) have closed-formed solutions as follows:

E step:

$$\boldsymbol{\Sigma}_{wi} = \left(\sigma^{-2} \boldsymbol{\Psi}_i^T \boldsymbol{\Psi}_i + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1}\right)^{-1}, \quad (7)$$

$$\boldsymbol{\mu}_{wi} = \boldsymbol{\Sigma}_{wi} \left(\sigma^{-2} \boldsymbol{\Psi}_i^T D_i + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1} \boldsymbol{\mu}_{w_{\text{old}}}\right), \quad (8)$$

M step:

$$\boldsymbol{\mu}_w^* = \frac{\sum_i \boldsymbol{\mu}_{wi}}{K}, \quad (9)$$

$$\boldsymbol{\Sigma}_w^* = \frac{\left(\boldsymbol{E}^T \boldsymbol{E} + \sum_i \boldsymbol{\Sigma}_{w_i}\right)}{K}, \quad (10)$$

with

$$\boldsymbol{\mu}_{wi} = [\mu_{w_{i1}}, \mu_{w_{i2}}, \cdots, \mu_{w_{iN}}]^T,$$
$$\boldsymbol{\mu}_w^* = \left[\mu_{w_1}^*, \mu_{w_2}^*, \cdots, \mu_{w_N}^*\right]^T,$$
$$\boldsymbol{e}_i = \boldsymbol{\mu}_{wi} - \boldsymbol{\mu}_w^*, \quad \boldsymbol{E} = [\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_K]^T.$$

Note that this approach to learn ProMPs incorporates all available observations when estimating the mean $\boldsymbol{\mu}_{wi}$ and the covariance $\boldsymbol{\Sigma}_{wi}$, which define the probability distribution over the weights for each training trajectory i. Equations (7) and (8) involve the terms $\boldsymbol{\mu}_{w_{\text{old}}}$ and $\boldsymbol{\Sigma}_{w_{\text{old}}}$, which represent the mean and the covariance from the previous EM iteration of the Gaussian over the weights of all the training trajectories.

B. Algorithm with Multiple Phase Parameters

A set of movement demonstrations may present local variabilities in speed of execution. For example, consider a data set in which the beginning of the movements has approximately the same speed of execution, but the end differs considerably in speed. A phase function with one single parameter cannot account for this kind of variability, because changing its phase parameter would result in accelerating or decelerating the whole movement.

In order to learn movement primitives that model local variabilities in speed of execution, the rate of change $\dot{z}(t)$ of the phase function with respect to time t can be defined as a weighted sum of Gaussian basis functions,

$$\dot{z}(t) = \boldsymbol{\phi}(t)^T \boldsymbol{\alpha}, \quad (11)$$

where $\boldsymbol{\phi}(t)$ is a vector of normalized Gaussian basis functions evaluated at time step t and $\boldsymbol{\alpha}$ is a vector of phase parameters, which are the weights for the basis functions.

The phase function can be obtained by first defining its value at time step $t = 0$ and then using Euler Integration until a time limit t_{max} is achieved,

$$z(0) = 0, \quad z(t+1) = z(t) + \Delta t \dot{z}(t).$$

However, once $z(t) = Z$, it no longer increases, remaining with the value Z until $t = t_{max}$. Movements may have any duration that does not surpass the time limit t_{max}.

When learning ProMPs with multiple phase parameters, the estimation of the phase parameters is incorporated into the EM algorithm. This time, $\boldsymbol{w}$ and $\boldsymbol{\alpha}$ are both hidden,

vector-valued variables. The set of parameters $\boldsymbol{\theta}$ is initialized with a rough estimate $\boldsymbol{\theta}_0 = \{\boldsymbol{\mu}_{w0}, \boldsymbol{\Sigma}_{w0}, \boldsymbol{\mu}_{\alpha 0}, \boldsymbol{\Sigma}_{\alpha 0}\}$. In the beginning of each iteration of the algorithm, a number S of vectors $\boldsymbol{\alpha}_j$ are sampled from $\mathcal{N}(\boldsymbol{\alpha}; \boldsymbol{\mu}_{\alpha_{\text{old}}}, \boldsymbol{\Sigma}_{\alpha_{\text{old}}})$, where $\boldsymbol{\mu}_{\alpha_{\text{old}}}$ and $\boldsymbol{\Sigma}_{\alpha_{\text{old}}}$ are parameters determined by the previous iteration. The algorithm optimizes the parameter vector $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha\}$ to maximize $\prod_{i,j} p_{\boldsymbol{\theta}}(D_i|\boldsymbol{\alpha}_j)$. This operation is performed by executing iteratively an EM to determine $\{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$ and an EM to determine $\{\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha\}$:

E step for $\boldsymbol{w}$:

$$\boldsymbol{\Sigma}_{wij} = \left(\sigma^{-2}\boldsymbol{\Psi}_{ij}^T\boldsymbol{\Psi}_{ij} + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1}\right)^{-1}, \tag{12}$$

$$\boldsymbol{\mu}_{wij} = \boldsymbol{\Sigma}_{wij}\left(\sigma^{-2}\boldsymbol{\Psi}_{ij}^T D_{ij} + \boldsymbol{\Sigma}_{w_{\text{old}}}^{-1}\boldsymbol{\mu}_{w_{\text{old}}}\right), \tag{13}$$

M step for $\boldsymbol{w}$:

$$\boldsymbol{\mu}_w^* = \frac{\sum_{i,j} p(\boldsymbol{\alpha}_j|D_i)\,\boldsymbol{\mu}_{wij}}{\sum_{i,j} p(\boldsymbol{\alpha}_j|D_i)}, \tag{14}$$

$$\boldsymbol{\Sigma}_w^* = \frac{\left(\boldsymbol{E}_d^T\boldsymbol{E} + \sum_{i,j} p(\boldsymbol{\alpha}_j|D_i)\,\boldsymbol{\Sigma}_{w_{ij}}\right)}{\sum_{i,j} p(\boldsymbol{\alpha}_j|D_i)}, \tag{15}$$

with

$$\begin{aligned}
&\boldsymbol{\mu}_{wij} = \left[\mu_{w_{ij1}}, \mu_{w_{ij2}}, \cdots, \mu_{w_{ijN}}\right]^T,\\
&\boldsymbol{\mu}_w^* = \left[\mu_{w_1}^*, \mu_{w_2}^*, \cdots, \mu_{w_N}^*\right]^T,\\
&\boldsymbol{e}_{ij} = \boldsymbol{\mu}_{wij} - \boldsymbol{\mu}_w^*, \quad \boldsymbol{E} = \left[\boldsymbol{e}_{11}, \boldsymbol{e}_{12}, \cdots, \boldsymbol{e}_{KS}\right]^T,\\
&\boldsymbol{e}_{dij} = p(\boldsymbol{\alpha_j}|\boldsymbol{D_i})\left(\boldsymbol{\mu}_{wij} - \boldsymbol{\mu}_w^*\right),\\
&\boldsymbol{E}_d = \left[\boldsymbol{e}_{d11}, \boldsymbol{e}_{d12}, \cdots, \boldsymbol{e}_{dKS}\right]^T.
\end{aligned}$$

Having executed the EM for $\boldsymbol{w}$, the algorithm recomputes the terms $p(\boldsymbol{\alpha}_j|D_i)$ according to $\{\boldsymbol{\mu}_w^*, \boldsymbol{\Sigma}_w^*\}$ and executes the EM for $\boldsymbol{\alpha}$:

E step for $\boldsymbol{\alpha}$:

$$\mathbb{E}[\boldsymbol{\alpha}|D_i] = \frac{\sum_j p(\boldsymbol{\alpha}_j|D_i)\,\boldsymbol{\alpha}_j}{\sum_j p(\boldsymbol{\alpha}_j|D_i)}, \tag{16}$$

M step for $\boldsymbol{\alpha}$:

$$\boldsymbol{\mu}_\alpha^* = \frac{\sum_i \mathbb{E}[\boldsymbol{\alpha}|D_i]}{K}, \tag{17}$$

$$\boldsymbol{\Sigma}_\alpha^* = \frac{\sum_{i,j} p(\boldsymbol{\alpha}_j|D_i)\left(\boldsymbol{\alpha}_j - \boldsymbol{\mu}_\alpha^*\right)\left(\boldsymbol{\alpha}_j - \boldsymbol{\mu}_\alpha^*\right)^T}{\sum_{i,j} p(\boldsymbol{\alpha}_j|D_i)}. \tag{18}$$

The mean $\boldsymbol{\mu}_\alpha^*$ and the covariance $\boldsymbol{\Sigma}_\alpha^*$ become the mean $\boldsymbol{\mu}_{\alpha_{\text{old}}}$ and the covariance $\boldsymbol{\Sigma}_{\alpha_{\text{old}}}$ of the Gaussian probability distribution from which the vectors $\boldsymbol{\alpha}_j$ are sampled in the beginning of the next iteration of the algorithm. The iterations continue until $\prod_{i,j} p_{\boldsymbol{\theta}}(D_i|\boldsymbol{\alpha}_j)$ converges.

C. Online Phase Estimation and Movement Prediction

Given a set D_i of observed positions at specific time steps of a movement i being observed[3], the phase associated with this movement can be online estimated and the unobserved part of this movement can subsequently be predicted, as long as its trajectory and phase fit into the probability distributions learned in the training phase.

In order to perform this prediction, a number of vectors $\boldsymbol{\alpha}_j$ are sampled from $\mathcal{N}(\boldsymbol{\alpha}; \boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha)$ resulting from the training phase performed with the algorithm explained in Section III-B. The probability of $\boldsymbol{\alpha}_j$ given observation D_i is given by[4]

$$p(\boldsymbol{\alpha}_j|D_i) \propto p(D_i|\boldsymbol{\alpha}_j), \tag{19}$$

where[5]

$$p(D_i|\boldsymbol{\alpha}_j) = \int p(D_i|\boldsymbol{w}, \boldsymbol{\alpha}_j)\,p(\boldsymbol{w})\,\mathrm{d}\boldsymbol{w}. \tag{20}$$

Equation 20 can be solved in closed form, yielding $p(D_i|\boldsymbol{\alpha}_j) = \mathcal{N}(D_i; \boldsymbol{\mu}_{Dij}, \boldsymbol{\Sigma}_{Dij})$ with

$$\boldsymbol{\mu}_{Dij} = \boldsymbol{\Psi}_{ij}\boldsymbol{\mu}_{wij}, \tag{21}$$

$$\boldsymbol{\Sigma}_{Dij} = \sigma^2\mathbf{I} + \boldsymbol{\Psi}_{ij}\boldsymbol{\Sigma}_{wij}\boldsymbol{\Psi}_{ij}^T. \tag{22}$$

The terms $\boldsymbol{\mu}_{wij}$ and $\boldsymbol{\Sigma}_{wij}$ can be computed using (13) and (12), respectively.

Finally, the mean $\boldsymbol{\mu}_\tau$ and covariance $\boldsymbol{\Sigma}_\tau$ of the predicted trajectory τ for the whole duration T_i of the movement i can be computed with

$$\boldsymbol{\mu}_\tau = \frac{\sum_j p(\boldsymbol{\alpha}_j|D_i)\,\boldsymbol{\mu}_{\tau j}}{\sum_j p(\boldsymbol{\alpha}_j|D_i)}, \tag{23}$$

$$\boldsymbol{\Sigma}_\tau = \frac{\sum_j p(\boldsymbol{\alpha}_j|D_i)\left(\boldsymbol{\Sigma}_{\tau j} + \boldsymbol{\mu}_{\tau j}\boldsymbol{\mu}_{\tau j}^T - \boldsymbol{\mu}_\tau\boldsymbol{\mu}_\tau^T\right)}{\sum_j p(\boldsymbol{\alpha}_j|D_i)}, \tag{24}$$

where

$$\boldsymbol{\mu}_{\tau j} = \boldsymbol{\Psi}_j\boldsymbol{\mu}_{wij}, \tag{25}$$

$$\boldsymbol{\Sigma}_{\tau j} = \sigma^2\mathbf{I} + \boldsymbol{\Psi}_j\boldsymbol{\Sigma}_{wij}\boldsymbol{\Psi}_j^T. \tag{26}$$

While $\boldsymbol{\Psi}_{ij}$ is defined only at the time steps of observed positions, $\boldsymbol{\Psi}_j$ is defined over the whole duration T_i of the movement according to $\boldsymbol{\alpha}_j$.

This prediction is thus a weighted average of the predictions with all sampled vectors $\boldsymbol{\alpha}_j$. Equations (23) and (24) can be derived by minimizing the Kullback-Leibler divergence [13] between a mixture of Gaussians with parameters $\{p(\boldsymbol{\alpha}_j|D_i), \boldsymbol{\mu}_{\tau j}, \boldsymbol{\Sigma}_{\tau j}\}$ and a single Gaussian with parameters $\{\boldsymbol{\mu}_\tau, \boldsymbol{\Sigma}_\tau\}$.

IV. Experiments

This section presents a number of experiments that evaluate some of the applications of our algorithm described in Section III. Those applications are online phase estimation with subsequent movement prediction, learning from incomplete data and inferring distributions over trajectories associated with phase functions defined by multiple parameters.

[3]This movement does not need to be part of the training data. In fact, for any practical application, this movement most likely does not match exactly any of the demonstrations.

[4]Note that $p(\boldsymbol{\alpha}_j)$ is not part of the expression, since the vectors $\boldsymbol{\alpha}_j$ are being sampled.

[5]Assuming $\boldsymbol{w}$ and $\boldsymbol{\alpha}$ independent variables.

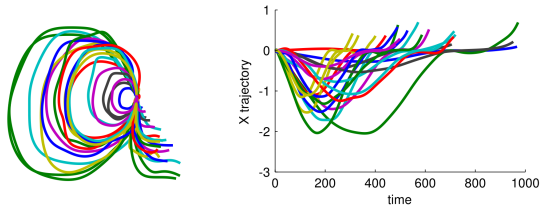


Fig. 2. Data set consisting of different trajectories of the letter "a". The trajectories differ in shape, scale and duration.

A. Experiments on Online Phase Estimation and Movement Prediction

In this first set of experiments, we evaluate online phase estimation and movement prediction assuming all movements can be associated with a phase function defined by one single phase parameter. Each movement has, however, its own phase parameter α. This evaluation is performed in two scenarios: with artificially generated trajectories of the letter "a" and in a Human-Robot Interaction scenario, in which a robot receives a cup from a human.

1) Using Artificially Generated "a" Trajectories: Consider a data set comprising several (x, y) trajectories of the letter "a", differing in shape, scale and duration. Fig. 2 exemplifies such a data set, showing different letters "a" and their correspondent x trajectories. Given a partial observation of a test trajectory, our objective is to predict the unobserved part. The prediction should be as close as possible to the ground truth. In the experiments presented in this section, the training data set was comprised only of whole trajectories, without any gaps.

The training and test data were constructed by sampling weights $\boldsymbol{w}$ for a set of 20 Gaussian basis functions and phase parameters α from Gaussian distributions. Forty trajectories were generated with the sampled values, using (3). 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 $\boldsymbol{w}$ of the basis functions and the phase parameters α are normal distributed. For this reason, it was chosen as a starting point to test the method proposed in Section III-C to estimate the phase online and predict the unobserved part of a movement.

In training phase, the weights $\boldsymbol{w}$ for each training trajectory were computed with linear regression and the phase variables α were computed with Equation (1). Then, Gaussian probability distributions $p(\boldsymbol{w})$ and $p(\alpha)$ over the computed values were defined.

In test phase, 50 values for the α parameter were sampled from the prior probability distribution $p(\alpha)$. For testing, we provided for each test trajectory observations with an increasing number of time steps. After each new observed position, the remaining test trajectory was predicted using the method described in Section III-C. Fig. 3 shows the probability density function $p(\alpha|D)$ over the sampled values for α after observing 5%, 10%, 20%, 40% and 60% of a test trajectory. The vertical red line corresponds to the ground truth phase parameter. In general, the larger the observed part, the better the estimation of α and the better the prediction. Fig. 4 shows the prediction after observing 5% and 40% of a test trajectory in comparison to the ground truth.

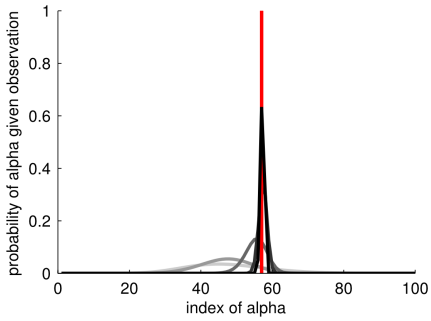


Fig. 3. Phase estimation after observing 5, 10, 20, 40 and 60 percent of a test trajectory. The darker the curve representing the probability density function, the larger the observed part. The vertical red line corresponds to the ground truth phase parameter.

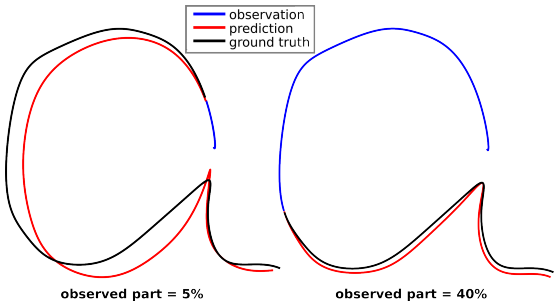


Fig. 4. Predictions after observing 5 and 40 percent of a test trajectory.

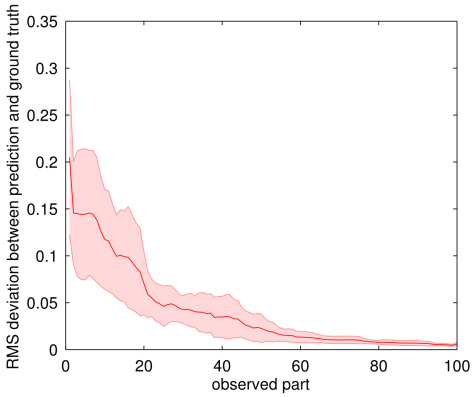


Fig. 5. Observed part versus RMS error with mean and standard deviation.

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. 5 shows the mean and the variance of the RMS error as the observed part increases.

2) Human-Robot Interaction Experiment: We evaluated the same algorithm on a human-robot collaborative task of an object handover (see Fig. 6 and accompanying video).

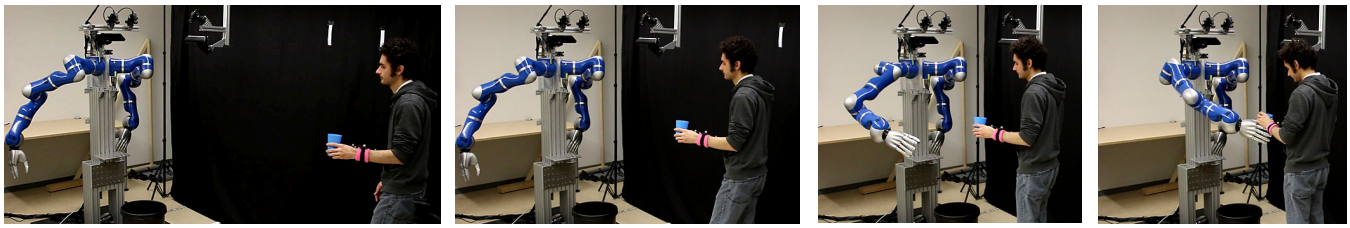


Fig. 6. A sequence of snapshots of a cup handover interaction between a human and a robot. The robot infers online the phase of the human movement and computes the expected reaction to this movement, according to the training.

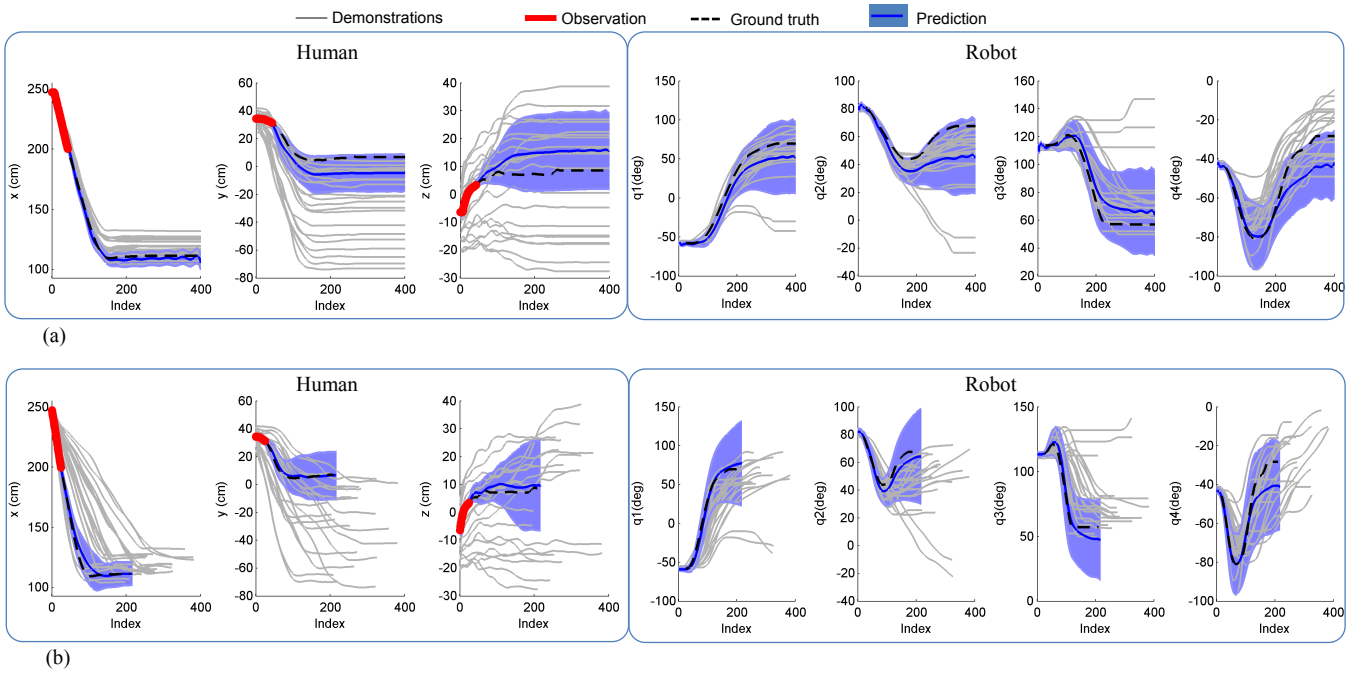


Fig. 7. One of the test cases where the robot reaction is computed based on a partial observation of the human movement. The (x, y, z) coordinates of the human wrist and four joint angles of the 7 DOF robotic arm are shown. (a) Conditioning after time-alignment with the method proposed in [2], which does not allow for online conditioning. (b) Conditioning online, without time-alignment, using the method proposed in this paper.

During training, the human moved with different speeds in the direction of the robot but also varying the position at which he handed over the cup, while the robot was moved by kinesthetic teaching to receive the cup at the correct position. The trajectories of the human were about 150 cm long. Each of those trajectories was a sequence of (x, y, z) coordinates of the human's left wrist, which had markers attached to it detectable by a motion capture system.

During test, the human was observed only during the first 50 cm of his trajectory. Then, 15 values for the phase parameter α were sampled from the $p(\alpha)$ learned in the training phase. Subsequently, the rest of the movement of the human and the expected reaction of the robot[6] were computed online using the method presented in Section III-C.

Fig. 7 shows one of the test cases from a leave-one-out cross-validation (LOOCV) procedure run over a data set of recorded trajectories (22 in total). Fig. 7(a) shows the result of conditioning the learned probability distribution over trajectories on the observed sequence of positions of the human with the method proposed in [2]. The method proposed in that work performs first the time-alignment of the recorded trajectories using a local optimization algorithm and then performs the conditioning. That method cannot condition online on the beginning of the human's movement, since it is not possible to time-align his trajectory before it has been completed. Fig. 7(b) shows the result of our method, in which the training movements, in gray, preserve their original speed profile. Using the same observations as in the previous case, the robot's joint trajectories could still be predicted with similar accuracy, but now also with variable phases, inferred from the human movement.

[6]For more details on how to apply the ProMP framework to interaction scenarios, the interested reader is referred to [2].

B. Experiments on Dealing with Missing Data

This section presents experiments that show the applicability of the EM algorithm proposed in Section III to learn ProMPs in the face of training trajectories with missing data points.

The training and test trajectories were artificially generated letters "a" as in Section IV-A.1. This time, however, random

sequences of 180 time steps were removed from the training trajectories. Since the shortest training trajectory comprised 355 time steps and the longest, 1129, the missing parts corresponded approximately to between 16% and 51% of the entire trajectories. Fig. 8 shows three of those training trajectories.

In a first experiment, linear regression was used to learn the weight vectors $\boldsymbol{w}$ for each of those training trajectories and a Gaussian distribution was fitted to the resulting set of weight vectors. Fig. 9(a) shows the approximation generated by linear regression to one of the training trajectories. In a second experiment, the EM algorithm proposed in Section III-A was used instead of linear regression to learn the weights $\boldsymbol{w}$. Fig. 9(b) shows the approximation generated by the EM algorithm to the same training trajectory as in Fig. 9(a).

The approximation generated by the EM algorithm 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 algorithm, 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 $\boldsymbol{w}$ for each trajectory i with (7) and (8). Note that those equations involve the terms $\boldsymbol{\mu}_{w_{\text{old}}}$ and $\boldsymbol{\Sigma}_{w_{\text{old}}}$, which represent the mean and the covariance from the previous EM iteration of the Gaussian over the weights of all the training trajectories.

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. Fig. 10 shows the mean and the standard deviation of the 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 $\boldsymbol{w}$ 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.

C. Experiments on Using Multiple Phase Parameters

In this section, we evaluate differences between models with only one phase parameter and models with multiple phase parameters. In the experiments here presented, whole trajectories, with no missing data, were used for training. Nevertheless, the data sets used here have speed profiles that make predicting unobserved parts more challenging than with the previously used data sets.

In a first set of experiments, training and test data comprised artificially generated "a" trajectories. This time, however, these trajectories were associated to phase functions with two phase parameters, i.e. these trajectories could for instance start slowly and end fast or start fast and end slowly.

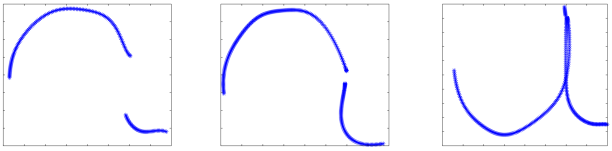


Fig. 8. Training trajectories with missing data points.

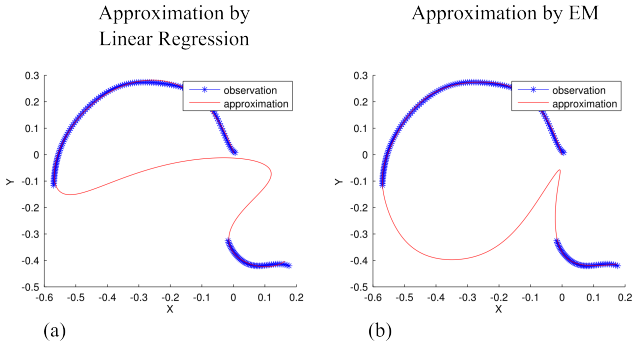


Fig. 9. Approximation to a training trajectory by linear regression versus approximation by EM.

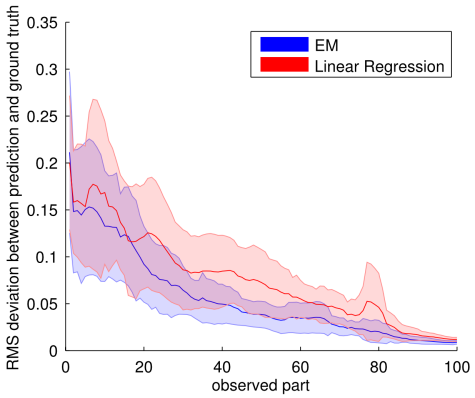


Fig. 10. Comparison between error of the predictions after training with EM and error of the predictions after training with linear regression, for the case in which the training trajectories have missing data points.

After training and having observed a part of a test trajectory, our objective was to predict the unobserved part. The prediction should be as close as possible to the ground truth. Fig. 11 shows the prediction produced by the model with two phase parameters, in red, and the prediction produced by the model with only one phase parameter, in green, after observing 30% of a test trajectory. The model with two phase parameters produced a much better prediction, because it accounts for local changes in speed of execution.

We performed the same experiment also using a data set of golf putting movements executed by a human. Fig. 12 shows the prediction produced by the model with two phase parameters, in red, and the prediction produced by the model with only one phase parameter, in green, after observing a sequence of positions in the beginning of a test trajectory. This sequence of observations comprised positions before the x degree of freedom (DOF) had reached its minimum value. The x DOF is the one with the largest range of values,

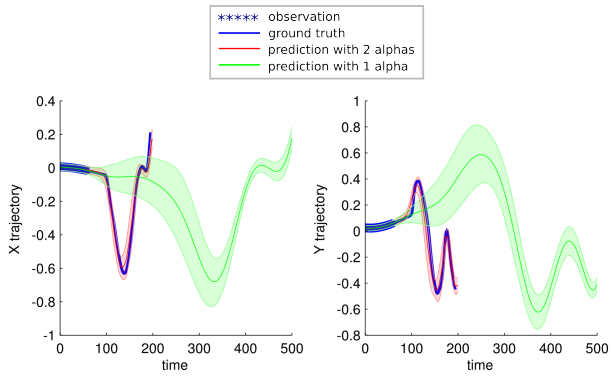


Fig. 11. Comparison between prediction ($\mu \pm 2\sigma$) of the model with two phase parameters (red) and prediction of the model with only one phase parameter (green), having observed 30% of a test trajectory. The training and test trajectories were in this case artificially generated letters "a".

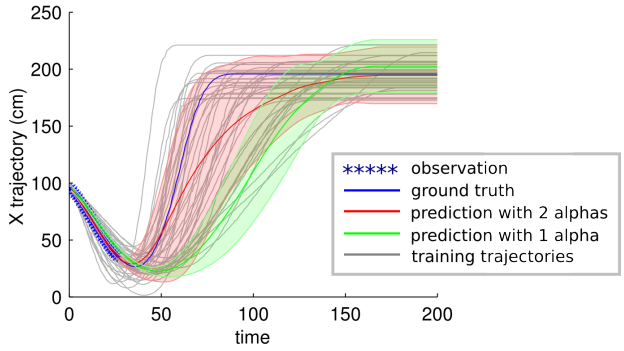


Fig. 12. Comparison between prediction ($\mu \pm 2\sigma$) of the model with two phase parameters (red) and prediction of the model with only one phase parameter (green), having observed part of a test trajectory before the minimum value of x had been reached. The training and test trajectories were in this case golf putting trajectories demonstrated by a human.

from left to right in Fig. 1(b). While both predictions are considerably far away from the ground truth, the prediction from the model with two phase parameters has a covariance that better represents the training trajectories, what can be observed in Fig. 12 by the fact that the red shade covers more of the training trajectories than the green shade. Moreover, the prediction of the model with two phase parameters had an average RMS error of approximately 18.93 cm with a standard deviation of 8.23 cm, while the model with only one phase parameter had an average RMS error of approximately 24.69 cm with a standard deviation of 13.52 cm.

V. Conclusions

This paper presented an *Expectation-Maximization* algorithm to learn motor skills in the form of *Probabilistic Movement Primitives*. The presented algorithm allows for learning from trajectories with missing data and accounts for the spatial-temporal variability of the demonstrations. Experiments demonstrated the applicability of this algorithm to movement prediction and to Human-Robot Interaction scenarios in which the robot must react to human movements executed at different speeds.

Possible extensions of this work involve using Gaussian Mixture Models to account for nonlinear correlations between joints or interacting agents and to learn multiple tasks as in [3]. Applications to real data of models with multiple phase parameters deserve further investigation, as well as different phase function formulations with different numbers of parameters.

VI. Acknowledgments

The research leading to these results has received funding from the project BIMROB of the "Forum für interdisziplinäre Forschung" (FiF) of the TU Darmstadt, from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand) and from the European Community's Seventh Framework Programme (FP7-ICT-2009-6) under grant agreement 270327 (CompLACS).

References

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

[2] G. 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.

[3] M. Ewerton, G. Neumann, R. Lioutikov, H. Ben Amor, J. Peters, and G. Maeda, "Learning multiple collaborative tasks with a mixture of interaction primitives," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2015.

[4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[5] 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.

[6] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, 1978.

[7] S. Calinon, E. L. Sauser, A. G. Billard, and D. G. Caldwell, "Evaluation of a probabilistic approach to learn and reproduce gestures by imitation," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2671–2676.

[8] P. Englert and M. Toussaint, "Reactive phase and task space adaptation for robust motion execution," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 109–116.

[9] R. Vuga, B. Nemec, and A. Ude, "Speed profile optimization through directed explorative learning," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 547–553.

[10] A. Coates, P. Abbeel, and A. Y. Ng, "Learning for control from multiple demonstrations," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 144–151.

[11] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X. 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*, 2010, pp. 2074–2081.

[12] F. Meier, E. Theodorou, F. Stulp, and S. Schaal, "Movement segmentation using a primitive library," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 3407–3412.

[13] C. M. Bishop *et al.*, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1.