Learning Movement Primitive Libraries through Probabilistic Segmentation

Journal Title
XX(X):1–18

DOI: 10.1177/ToBeAssigned
www.sagepub.com/

Rudolf Lioutikov[1], Gerhard Neumann[2], Guilherme Maeda[1] and Jan Peters[13]

Abstract

Movement primitives are a well established approach for encoding and executing movements. While the primitives themselves have been extensively researched, the concept of movement primitive libraries has not received similar attention. Libraries of movement primitives represent the skill set of an agent. Primitives can be queried and sequenced in order to solve specific tasks. The goal of this work is to segment unlabeled demonstrations into a representative set of primitives. Our proposed method differs from current approaches by taking advantage of the often neglected, mutual dependencies between the segments contained in the demonstrations and the primitives to be encoded. By exploiting this mutual dependency, we show that we can improve both the segmentation and the movement primitive library. Based on probabilistic inference our novel approach segments the demonstrations while learning a probabilistic representation of movement primitives. We demonstrate our method on two real robot applications. First, the robot segments sequences of different letters into a library, explaining the observed trajectories. Second, the robot segments demonstrations of a chair assembly task into a movement primitive library. The library is subsequently used to assemble the chair in an order not present in the demonstrations.

Introduction

A key goal of modern robotics is to provide robots with the ability to learn new tasks. A commonly followed concept to achieve such behavior is imitation learning. The robot is provided with one or more demonstrations of a task, which the robot subsequently reproduces and improves. Often, an entire task consists of a single motion, encoded as a single movement primitive (Mülling et al. 2010),(Paraschos et al. 2013). This concept has been applied in a variety of tasks, including hitting movements in table tennis (Mülling et al. 2010) and locomotion (Nakanishi et al. 2004).

Solving more complex, non-monolithic tasks with a single movement primitive may result in a great loss of generality. Considering complex tasks as a sequence of primitives offers multiple advantages. For example, primitives can be easily generalized and optimized between the points where they connect. The same set of primitives can be reused to execute different tasks, and the movement plan can be adapted by replacing one primitive within the sequence by another one. A fundamental problem of such approaches is the autonomous acquisition of these primitives without relying on hand labeled demonstrations. In this paper, we address this problem by proposing a framework for segmenting unlabeled demonstrations into a library of movement primitives.

Essentially, such movement primitive acquisition consists of two problems, the segmentation of observed trajectories and the learning of the underlying movement primitive library. In this paper we tackle these two problems in conjunction. Each demonstrated trajectory can be considered a multidimensional time series. A common way to segment time series data is to apply heuristics. However, the quality of such heuristics and therefore, the corresponding segmentation is often task dependent. For instance, while an assembly task consisting of point to point motions might be well segmented at zero crossing velocities, the same heuristic applied on continuously written words might achieve poor, meaningless results. Furthermore,

[1]Intelligent Autonomous Systems, TU Darmstadt
[2]Computational Learning for Autonomous Systems
[3]Max Planck Institute Tuebingen

Corresponding author:
Rudolf Lioutikov
FG IAS
Hochschulstrae 10
64389 Darmstadt
Email: lioutikov@ias.tu-darmstadt.de

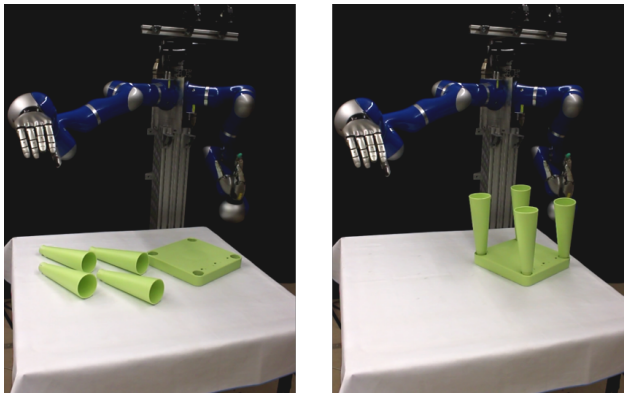


Figure 1. The robot platform used for a chair assembly experiment. We used a seven DoF KUKA lightweight arm equipped with a five finger DLR HIT Hand II as end effector. The executed movement primitives were learned by segmenting human demonstrations.

different parts of the data could be best explained by different heuristics, which raises the problem of identifying at what point to apply which heuristic.

Our approach starts from the premise that a task-specific heuristic can only segment a given trajectory sub-optimally, therefore, leading to a low-quality library. As a consequence, some movement primitives may not be meaningful while others will suitably describe the data. Our method applies probabilistic inference to reason iteratively over all possible segmentations by learning a probabilistic representation of movement primitives from a weighted set of segments. In return, the learned primitives are used to improve the set of segments by down-weighting segments that are less plausible given the current movement primitive library. We provide the mathematical formulation for the solution of this problem as an iterative Expectation-Maximization (EM) algorithm and show that our algorithm converges to a compact set of movement primitives given over-segmented demonstrations.

Another interesting aspect is the relationship between the size of the library and the complexity of the contained primitives. The learned primitives should be complex enough to represent a dedicated motion, while being simple enough to qualify as a modular unit. The size of the library is not directly proportional to the complexity of the contained primitives. The most complex primitive is the one learned from an entire demonstration. This choice would lead to a library the same size as the number of demonstrations. Simpler primitives that are shared by multiple demonstration, however, can result in a more compact library. A useful metric, that we apply in this paper, is the bit-encoding of the observed demonstrations. The encoding is described in more detail in the experiment section.

In summary, the main contribution of this article is the Probabilistic Segmentation (ProbS) algorithm that concurrently improves a given segmentation and the library of movement primitives. The method was validated on a real robot platform. We evaluated and compared our method on a letter segmentation task. The robot was taught different sequences of letters and subsequently executed the primitives learned by ProbS. Additionally, we applied ProbS to a chair assembly task. The required movement primitive library was learned by segmenting the human motion of a chair assembly. Subsequently, movement primitives were sequenced from the learned library to assemble the chair in a previously undemonstrated order. Parts of this paper consolidate previous work presented in (Lioutikov et al. 2015).

The paper is organized as follows. First the problem statement and the used notation is given. Next, related work is presented and discussed. Followed by the introduction of the Probabilistic Segmentation approach. Afterwards, we compare the proposed method to a baseline method and a state-of-the-art segmentation method, using a writing and an assembly task. Both tasks were performed on a real robot platform and show the capabilities of our method. Additionally, we use the method to segment table tennis strokes from kinesthetic teaching.

Related Work

Algorithms for automatic segmentation have been investigated extensively, not only for the purposes of generation of robot movement primitives but mainly as a general tool for movement analysis and classification. Hidden Markov Models (HMMs) have been widely adopted in this context. For example, (Brand and Kettnaker 2000) analyzed video images to train a HMM to classify if a person is walking, running or crouching. (Takano and Nakamura 2006) used automatic segmentation of motion patterns based on HMMs to group segments hierarchically, where higher level representations of symbols can then be used to orchestrate and generate low level robot movements. More recently, (Kulic et al. 2009) proposed an on-line segmentation method based on HMMs that creates a tree of primitives; the lower nodes representing detailed movements with generality increasing towards the root. HMMs have also been used in conjunction with the superposition of movement primitives for the specific case of handwriting analysis, (Williams et al. 2008). In, (Krishnan et al. 2015) a Dirichlet Process Gaussian Mixture Model is learned to identify the transition

points between the segments. The transition points are subsequently clustered spatially and temporally. In general, HMMs and methods that explicitly address temporal sequences, e.g. (d'Avella and Tresch 2001), have been generally accepted for segmentation. In this paper, however, we opt for a shape-based clustering approach on the basis that our desired library must be invariant to the possible combinations of movement primitives transitions. The encoding of trajectories that do have a sequential pattern are naturally addressed by our method as it maintains only the most probable combinations of segments.

Our work takes advantage of movement primitive representations that are time invariant, such as Dynamical Movement Primitives (DMPs), (Ijspeert et al. 2013), or Probabilistic Movement Primitives (ProMPs), (Paraschos et al. 2013). Segmentation with movement primitives is particularly suited for library construction as segments with the same profile, but with different time scales are treated as the same primitive. (Chiappa and Peters 2010), for example, had to take the expected time scales of possible segments into account with the introduction of a heuristic about the minimum and maximum duration of the movement primitives; in our approach such user-defined inputs are not necessary.

From the movement primitive perspective, our algorithm relates to the work of (Meier et al. 2011), and (Niekum et al. 2013) where DMPs have been used in different ways. In the first approach, a library of primitives is assumed given, while in our work we design our algorithm to start from an empty set. Compared to the work of (Niekum et al. 2013), the authors treat segmentation as an independent initial step which therefore, later affects the reconstruction of a task, in this case using finite state automatons. As a consequence, interactive corrections given by a human demonstrator are introduced. In contrast, our approach treats segmentation and primitive learning as an iterative optimization process where both are intrinsically connected.

Hierarchical skills have been explored by (Takano and Nakamura 2006), (Kulic et al. 2009), (Konidaris et al. 2012), and (Yamane et al. 2011), and can be very efficient for on-line applications or to represent different granularities in the task. The philosophy of our method differs in the sense that we do not enrich a model by adding branches, but instead prune unnecessary segments given by a possibly erroneous initial heuristic. We leverage on batch, off-line learning to essentially reconstruct the movement primitive library, iteratively. This leads to a single level representation which decreases the number of segments as the library is improved after each EM iteration.

A general problem in movement segmentation and library generation is the trade off between the generality of the method and its tractability, the latter usually achieved by the introduction of heuristics. For example, Zero Crossing Velocity (ZCV) has usually been used as an intuitive criterion to obtain the initial segmentation of trajectories (Fod et al. 2002), (Nakazawa et al. 2002), (Barbič et al. 2004). In the context of movement primitives, however, ZCV usually leads to over-segmentation, especially when the robot moves at low speeds. (Lemme et al. 2014) proposed segmenting demonstrations based on geometric similarities. (Wächter and Asfour 2015) introduced a two layer hierarchical approach is proposed to segment 6D motion trajectories. The top layer identifies segments based on semantic criteria, e.g. contact between objects. The bottom layer applies a heuristic to identify keyframes based on the difference of the adjacent trajectory parts. (Endres et al. 2013) apply bayesian binning in order to segment end-effector trajectories into pieces with different parameter values of the two-third power law. Other heuristics applied to segmentation include velocity profiles and minimum jerk, (Rohrer and Hogan 2006), changes of the system dynamics, (Kroemer et al. 2014). Our work differs by being insensitive to the particular choice of the heuristic. Our assumption is that a given heuristic will lead to an initial number of excessive segments, which will be then optimized by decreasing the occurrence of cuts among them when necessary.

Problem Statement and Notation

Given a set of observed trajectories $\mathcal{T} = \{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{|\mathcal{T}|}\}$, the goal of this work is to learn a set of underlying movement primitives $\mathcal{M} = \{\mathrm{m}_1, \ldots, \mathrm{m}_{|\mathcal{M}|}\}$ which explains $\mathcal{T}$, i.e., the movement primitive library which produced $\mathcal{T}$. Since we are interested in the underlying library $\mathcal{M}$ of a task domain, the trajectories in $\mathcal{T}$ can describe the same or multiple tasks, as long as they belong to the same domain, and, therefore, can be explained by the same library. The duration of each individual demonstration $\boldsymbol{\tau}$ might be different. An example of a one dimensional trajectory is illustrated in Figure 2(a). For each trajectory $\boldsymbol{\tau} \in \mathcal{T}$, a set of possible cutting points $\mathcal{C}_{\boldsymbol{\tau}}$ is defined. Each cutting point $h, i, j \in \mathcal{C}_{\boldsymbol{\tau}}$ separates two subsequent segments, e.g., $\boldsymbol{s}_{h,i}$ and $\boldsymbol{s}_{i,j}$. The indices of $\boldsymbol{s}_{i,j}$ denote that the segment starts at the i^{th} and ends at the j^{th} cutting point. We will drop the indices whenever it is irrelevant at which cutting point the segment starts $\boldsymbol{s} := \boldsymbol{s}_{i,j}$.

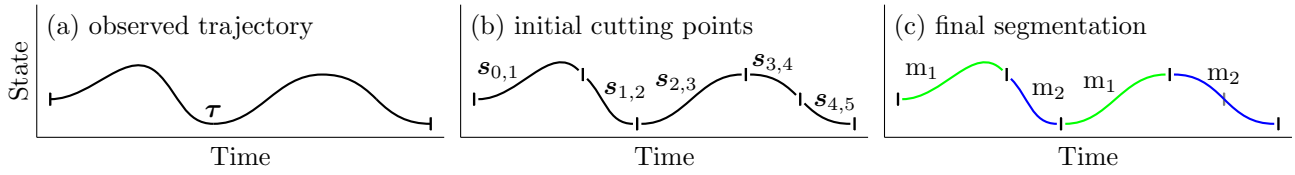


Figure 2. An illustration of a possible segmentation. (a) shows a one dimensional, continuous observation. (b) shows four initially suggested cuts, illustrated as black bars, and the five resulting segments, if all cuts were true positives. (c) shows a possible segmentation. The fourth cut was identified as a false positive cut, illustrated as a gray bar. Two primitives m_1 and m_2 were learned from the resulting four segments.

The set $\mathcal{C}_{\boldsymbol{\tau}}$ splits $\boldsymbol{\tau}$ into a set of segments $\mathcal{S}_{\boldsymbol{\tau}}$, which represents a possible segmentation of $\boldsymbol{\tau}$. We assume that $\mathcal{C}_{\boldsymbol{\tau}}$ over-segments $\boldsymbol{\tau}$, i.e., the cutting points producing the correct segmentation $\mathcal{S}^*_{\boldsymbol{\tau}}$ are a subset of the initial cutting points $\mathcal{C}^*_{\boldsymbol{\tau}} \subseteq \mathcal{C}_{\boldsymbol{\tau}}$. The cutting points included in the correct set of cuts $\mathcal{C}^*_{\boldsymbol{\tau}}$ are referred to as true positive cuts and, respectively, the cutting points not included, i.e., $\mathcal{C}_{\boldsymbol{\tau}} \setminus \mathcal{C}^*_{\boldsymbol{\tau}}$ are referred to as false positive cuts.

Unfortunately, the true positive cuts are unknown. Therefore, every possible subset $\mathcal{C}'_{\boldsymbol{\tau}} \subseteq \mathcal{C}_{\boldsymbol{\tau}}$ has to be considered. Each $\mathcal{C}'_{\boldsymbol{\tau}}$ results in a different segmentation. The set of all possible segmentations will be denoted as $\mathcal{S}_{\boldsymbol{\tau}} \in \mathcal{D}_{\boldsymbol{\tau}}$. Furthermore, some segments occur in multiple segmentations, therefore, the set of all possible segments is given as $\mathcal{A}_{\boldsymbol{\tau}} = \bigcup_{\mathcal{S}_{\boldsymbol{\tau}} \in \mathcal{D}_{\boldsymbol{\tau}}} \mathcal{S}_{\boldsymbol{\tau}}$ and the set of all segmentations containing a particular segment $\boldsymbol{s}$ is defined as $\mathcal{D}_{\boldsymbol{s}} = \{\mathcal{S} | \boldsymbol{s} \in \mathcal{S}, \forall \mathcal{S} \in \mathcal{D}\}$.

Our method tackles two challenges simultaneously: eliminating all false positive cuts $\mathcal{C}_{\boldsymbol{\tau}} \setminus \mathcal{C}^*_{\boldsymbol{\tau}}$, therefore determining the correct segmentation $\mathcal{S}^*_{\boldsymbol{\tau}} \in \mathcal{D}_{\boldsymbol{\tau}}$ and learning the underlying MP library $\mathcal{M}$ from the chosen segments $\boldsymbol{s}_{\boldsymbol{\tau}} \in \mathcal{S}^*_{\boldsymbol{\tau}}, \forall \boldsymbol{\tau} \in \mathcal{T}$. The segments $\boldsymbol{s}_{\boldsymbol{\tau}} \in \mathcal{A}_{\boldsymbol{\tau}}$, the segmentations $\mathcal{S}_{\boldsymbol{\tau}} \in \mathcal{D}_{\boldsymbol{\tau}}$ and the cuts $\mathcal{C}_{\boldsymbol{\tau}}$ are always defined with respect to a single trajectory $\boldsymbol{\tau} \in \mathcal{T}$. For simplicity, we will drop the subscript $\boldsymbol{\tau}$ from now on. Table 1 summarizes the defined entities alongside others that will be defined later in the paper.

Learning Movement Primitive Libraries using Probabilistic Segmentation

We assume that each observed trajectory $\boldsymbol{\tau} \in \mathcal{T}$ can be represented by one of the corresponding segmentations $\mathcal{S} \in \mathcal{D}$. The trajectory $\boldsymbol{\tau}$ can be explained by concatenating the segments contained in $\mathcal{S}$.

The method is initialized with a set of possible cutting points $\mathcal{C}$, which divides each trajectory into multiple segments as shown in Figure 2. We assume that $\mathcal{C}$ weakly over-segments $\boldsymbol{\tau}$ and that there is a subset of true positive cuts $\mathcal{C}^* \subseteq \mathcal{C}$ which results in the correct segmentation $\mathcal{S}^* \in \mathcal{D}$.

Our goal is to determine the correct segmentation $\mathcal{S}^*$ while simultaneously learning the underlying library $\mathcal{M}$. Since $\mathcal{S}^*$ is not known, we treat $\mathcal{S}^*$ as a latent variable. Our proposed approach assesses the quality of all possible segmentations $\mathcal{S} \in \mathcal{D}$ by applying probabilistic inference methods to learn locally optimal $\mathcal{M}$ and $\mathcal{S}$ in conjunction. The size of the library $|\mathcal{M}|$, i.e. the number of learned primitives, is determined by applying a bisecting k-means algorithm on the segments $\boldsymbol{s} \in \mathcal{S}$ defined by the current segmentation. Therefore, no prior knowledge about the number of primitives is expected. Furthermore, given that the segmentation itself is learned, the number of primitives change with the segmentation estimate. Figure 2 illustrates the segmentation process of our method. The found movement primitives are shown in Figure 2(c), the observed trajectories in Figure 2(a) and the cutting points in Figure 2(b).

Defining the Cutting Points $\mathcal{C}$ The results of the proposed method depend on the set of possible cuts $\mathcal{C}$. Given the Expectation-Maximization like nature of the method, it is unfeasible to initialize it with a cutting point for every time step of the observation. A possibility to restrict $\mathcal{C}$ to a manageable size is to initially use heuristics to determine $\mathcal{C}$. These heuristics can be chosen task specifically and different heuristics can also be combined seamlessly. However, the method only considers the cuts contained in $\mathcal{C}$, i.e., it is restricted to eliminate false positive cuts. Therefore, $\mathcal{C}$ has to provide a weak over-segmentation, i.e., $\mathcal{C}^* \subseteq \mathcal{C}$, where $\mathcal{C}^*$ denotes the set of true positive cuts.

Movement Primitive Representation

In this work we use Probabilistic Movement Primitives (ProMPs) as the primitive representation, (Paraschos et al. 2013). ProMPs project trajectories into a lower dimensional weight space using a ridge regression. Therefore, each segment $\boldsymbol{s}$ has a matching projected segment $\boldsymbol{w}$

$$\boldsymbol{w} = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \epsilon \mathbf{I}\right)^{-1} \boldsymbol{\Phi}\boldsymbol{s}, \tag{1}$$

Symbol	Description
$\boldsymbol{\tau}$	an observed multidimensional, unlabeled trajectory
$\mathcal{T}$	a set of trajectories $\mathcal{T} = \{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{\lvert\mathcal{T}\rvert}\}$
m_k	a movement primitive parameterized by $\boldsymbol{\theta}_k$
$\boldsymbol{\theta}_k$	a parameterization of primitive m_k
$\mathcal{M}$	a set of primitives $\mathcal{M} = \{\mathrm{m}_1, \ldots, \mathrm{m}_{\lvert\mathcal{M}\rvert}\}$
$\boldsymbol{\Theta}$	a set of all primitive parameters, i.e., $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{\lvert\mathcal{M}\rvert}\}$
$\mathcal{C} := \mathcal{C}_{\boldsymbol{\tau}}$	a set of possible cutting points for $\boldsymbol{\tau}$
$\boldsymbol{s} := \boldsymbol{s}_{i,j}$	a segment, i.e. the part of $\boldsymbol{\tau}$ between cutting point i and j
$\boldsymbol{w}$	a lower dimensional projection of the segment $\boldsymbol{s}$
$\mathcal{W}$	a set of all projections across all trajectories, i.e, $\mathcal{W} = \{\boldsymbol{w} \lvert \boldsymbol{w} = \boldsymbol{v}(\boldsymbol{s}), \forall \boldsymbol{s} \in \mathcal{A}_{\boldsymbol{\tau}}, \forall \boldsymbol{\tau} \in \mathcal{T}\}$
$\alpha_{\boldsymbol{s}}$, $\alpha_{\boldsymbol{w}}$	a segment weighting defining how likely $\boldsymbol{s}$ is part of the underlying segmentation
$\mathcal{S} := \mathcal{S}_{\boldsymbol{\tau}}$	a possible segmentation of $\boldsymbol{\tau}$, i.e., a set of segmentations defined by a subset of $\mathcal{C}_{\boldsymbol{\tau}}$, e.g., $\mathcal{S}_{\boldsymbol{\tau}}^{[1]} = \{\boldsymbol{s}_{0,1}, \boldsymbol{s}_{1,4}, \boldsymbol{s}_{4,5}\}$, $\mathcal{S}_{\boldsymbol{\tau}}^{[2]} = \{\boldsymbol{s}_{0,5}\}$, $\mathcal{S}_{\boldsymbol{\tau}}^{[3]} = \{\boldsymbol{s}_{0,1}, \boldsymbol{s}_{1,2}, \boldsymbol{s}_{2,3}, \boldsymbol{s}_{3,5}\}$
$\mathcal{D} := \mathcal{D}_{\boldsymbol{\tau}}$	a set of all possible segmentations for $\boldsymbol{\tau}$, i.e., $\mathcal{D}_{\boldsymbol{\tau}} = \{\mathcal{S}_{\boldsymbol{\tau}}^{[1]}, \mathcal{S}_{\boldsymbol{\tau}}^{[2]}, \mathcal{S}_{\boldsymbol{\tau}}^{[3]}, \ldots\}$
$\mathcal{D}_{\boldsymbol{s}}^{\rightarrow}$	a set containing all segmentations of the partial trajectory that starts at the beginning of $\boldsymbol{\tau}$ and ends at the beginning of $\boldsymbol{s}$
$\mathcal{D}_{\boldsymbol{s}}^{\leftarrow}$	a set containing all segmentations of the partial trajectory that starts at the end of $\boldsymbol{s}$ and ends at the end of $\boldsymbol{\tau}$
$\mathcal{A} := \mathcal{A}_{\boldsymbol{\tau}}$	a set of all possible segments across all possible segmentations for $\boldsymbol{\tau}$, i.e., $\mathcal{A}_{\boldsymbol{\tau}} = \bigcup_{\mathcal{S}_{\boldsymbol{\tau}} \in \mathcal{D}_{\boldsymbol{\tau}}} \mathcal{S}_{\boldsymbol{\tau}}$
$\mathcal{A}^{\text{start}}$	a subset $\mathcal{A}_{\boldsymbol{\tau}}^{\text{start}} \subseteq \mathcal{A}_{\boldsymbol{\tau}}$ that contains all segments that start at the beginning of $\boldsymbol{\tau}$
$\mathcal{A}^{\text{end}}$	a subset $\mathcal{A}_{\boldsymbol{\tau}}^{\text{start}} \subseteq \mathcal{A}_{\boldsymbol{\tau}}$ that contains all segments that end at the end of $\boldsymbol{\tau}$
$\mathcal{A}_{\boldsymbol{s}}^{\text{pred}}$	a subset $\mathcal{A}_{\boldsymbol{s}}^{\text{pred}} \subset \mathcal{A}_{\boldsymbol{\tau}}$ that only contains segments that end at the beginning of $\boldsymbol{s}$
$\mathcal{A}_{\boldsymbol{s}}^{\text{succ}}$	a subset $\mathcal{A}_{\boldsymbol{s}}^{\text{succ}} \subset \mathcal{A}_{\boldsymbol{\tau}}$ that only contains segments that begin at the end of $\boldsymbol{s}$
$\gamma_{\boldsymbol{s}}^{\rightarrow}$, $\delta_{\boldsymbol{s}}^{\rightarrow}$,	forward messages send from or towards $\boldsymbol{s}$ respectively
$\gamma_{\boldsymbol{s}}^{\leftarrow}$, $\delta_{\boldsymbol{s}}^{\leftarrow}$,	backward messages send from or towards $\boldsymbol{s}$ respectively

Table 1. The main entities defined across the paper. In the paper the subscript $\boldsymbol{\tau}$ and the segment indices i, j are dropped whenever irrelevant.

where $\boldsymbol{\Phi}$ denotes the feature matrices as defined in (Paraschos et al. 2013). The features $\boldsymbol{\Phi}$ are usually represented as radial basis functions and depend on the duration $|\boldsymbol{s}|$ of $\boldsymbol{s}$, and, therefore, render the projected segment $\boldsymbol{w}$ invariant to the duration of the segment itself. This invariance is in particular interesting, since it allows to compare segments of different durations purely on their shape. The duration $|\boldsymbol{s}|$ corresponds to the number of time steps in $\boldsymbol{s}$. Such projections into a lower dimensional space are not unique to ProMPs, but are quite common in movement primitives, e.g, in Dynamic Movement Primitives (Ijspeert et al. 2013). In addition, ProMPs define a Gaussian distribution over the projected trajectories.

Therefore, if the segment $\boldsymbol{s}$ is a valid segment, there exists an underlying movement primitive m_k which produced the corresponding projected segment

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{w} \,|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{2}$$

The primitive $p(\boldsymbol{s} \,|\boldsymbol{\theta}_k)$ can now be described as a distribution over the segment $\boldsymbol{s}$ parameterized by $\boldsymbol{\theta}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$.

We only consider correlations between the dimensions and not between the time steps, i.e.

$$p(s_t \,|\boldsymbol{\theta}_k) = p(s_t \,|\boldsymbol{\theta}_k, s_{t-1}),$$

where s_t describes the segment $\boldsymbol{s}$ at the time step t, i.e., s_t is the t^{th} entry of $\boldsymbol{s}$. Therefore, the probability of a segment $\boldsymbol{s}$ given a movement primitive m_k is defined as

$$p(\boldsymbol{s} \,|\boldsymbol{\theta}_k) = \prod_{t=1}^{|\boldsymbol{s}|} p(s_t \,|\boldsymbol{\theta}_k),$$

and, the probability for a single time step t given the movement primitive m_k is

$$p(s_t \,|\boldsymbol{\theta}_k) = \mathcal{N}\left(s_t \,\middle|\, \boldsymbol{\phi}_t^T \boldsymbol{\mu}_k,\ \boldsymbol{\phi}_t^T \boldsymbol{\Sigma}_k \boldsymbol{\phi}_t\right),$$

where the feature vectors $\boldsymbol{\phi}_t^T$ are the corresponding rows of the feature matrix $\boldsymbol{\Phi}$.

Probabilistic Inference on Segmentations

Each movement primitive $\mathrm{m} \in \mathcal{M}$ is represented by a parameterized, generative model

$$p(\boldsymbol{s} \,|\boldsymbol{\theta}_k), \tag{3}$$

with $\boldsymbol{\theta}_k$ denoting the parameters of the k^{th} movement primitive m_k. However, we do not consider the single primitives independently but assume that each segment was drawn from the entire library, modeled as

a mixture of primitives

$$\begin{aligned} \boldsymbol{s} &\sim p\left(\boldsymbol{s}\,|\boldsymbol{\Theta}\right), \\ p\left(\boldsymbol{s}\,|\boldsymbol{\Theta}\right) &= \sum_{k=1}^{|\mathcal{M}|} \lambda_k p\left(\boldsymbol{s}\,|\boldsymbol{\theta}_k\right), \text{ with} \\ \boldsymbol{\Theta} &= \left\{\left(\lambda_1, \boldsymbol{\theta}_1\right), \ldots, \left(\lambda_{|\mathcal{M}|}, \boldsymbol{\theta}_{|\mathcal{M}|}\right)\right\}, \end{aligned} \tag{4}$$

where λ_k denotes the mixing coefficient for movement primitive m_k.

Furthermore, we assume that every observed trajectory $\boldsymbol{\tau} \in \mathcal{T}$ was drawn from a parameterized generative model

$$\boldsymbol{\tau} \sim p\left(\boldsymbol{\tau}\,|\boldsymbol{\Theta}, \mathcal{S}^*\right) = \prod_{\boldsymbol{s} \in \mathcal{S}^*} p\left(\boldsymbol{s}\,|\boldsymbol{\Theta}\right). \tag{5}$$

Since $\mathcal{S}^*$ is unknown, we treat it as a latent variable, and integrate it out, which leads to

$$p\left(\boldsymbol{\tau}\,|\boldsymbol{\Theta}\right) = \sum_{\mathcal{S} \in \mathcal{D}} p\left(\mathcal{S}\right) \prod_{\boldsymbol{s} \in \mathcal{S}} p\left(\boldsymbol{s}\,|\boldsymbol{\Theta}\right). \tag{6}$$

The most likely model $\boldsymbol{\Theta}$ is now determined by maximizing the log-likelihood

$$\boldsymbol{\Theta}^* = \arg\max_{\boldsymbol{\Theta}} \sum_{\boldsymbol{\tau} \in \mathcal{T}} \log p\left(\boldsymbol{\tau}\,|\boldsymbol{\Theta}\right).$$

Optimizing this log-likelihood directly is unfeasible. Therefore, we resort to the EM algorithm (Bishop 2006), which finds a locally optimal model $\boldsymbol{\Theta}$ by iterating between computing the expectation over the latent variables and maximizing the model parameters.

In our approach, the EM algorithm repeatedly maximizes the auxiliary function

$$\begin{aligned} \boldsymbol{\Theta} &= \arg\max_{\boldsymbol{\Theta}} Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right), \text{ with} \\ Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) &= \sum_{\boldsymbol{\tau} \in \mathcal{T}} \sum_{\mathcal{S} \in \mathcal{D}} p\left(\mathcal{S}\,|\boldsymbol{\tau}, \boldsymbol{\Theta}'\right) \log\left(p\left(\mathcal{S}\right) p\left(\boldsymbol{\tau}\,|\boldsymbol{\Theta}, \mathcal{S}\right)\right) \end{aligned} \tag{7}$$

until convergence. In this formulation, $\boldsymbol{\Theta}'$ denotes the model parameters found in the previous iteration. The prior $p\left(\mathcal{S}\right)$ is defined as a product of priors over cutting points $p\left(c_{\boldsymbol{s}}\right)$

$$\begin{aligned} p\left(\mathcal{S}\right) &= p_c \prod_{\boldsymbol{s} \in \mathcal{S}} p\left(c_{\boldsymbol{s}}\right), \text{with} \\ p\left(c_{\boldsymbol{s}}\right) &= \left(1 - p_c\right)^{c_{\boldsymbol{s}}} p_c, \end{aligned} \tag{8}$$

where $c_{\boldsymbol{s}}$ is the number of possible cutting points the segment $\boldsymbol{s}$ spans over. The constant $0 < p_c < 1$ defines how probable it is that a cut is a true positive cut. For $p_c < 0.5$ segments which span over multiple cutting points are preferred, whereas $p_c > 0.5$ indicates that shorter segments are preferable.

Solving Equation (7) is computationally expensive, since the number of segmentations, $|\mathcal{S}| = 2^{|\mathcal{C}|}$, grows exponentially with the number of cuts $|\mathcal{C}|$.

However, we can reformulate Equation (7) such that it sums over all possible segments $\boldsymbol{s} \in \mathcal{A}$, which are only quadratic in the number of cuts $|\mathcal{A}| = 0.5(|\mathcal{C}| + 1)(|\mathcal{C}| + 2)$. Inserting Equation (5) and Equation (8) into Equation (7), moving the log inside the product and dropping the constant term yields

$$Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) = \sum_{\boldsymbol{\tau} \in \mathcal{T}} \sum_{\mathcal{S} \in \mathcal{D}} p\left(\mathcal{S}\,|\boldsymbol{\tau}, \boldsymbol{\Theta}'\right) \sum_{\boldsymbol{s} \in \mathcal{S}} \log\left(p\left(c_{\boldsymbol{s}}\right) p\left(\boldsymbol{s}\,|\boldsymbol{\Theta}\right)\right).$$

Pulling the coefficient $p\left(\mathcal{S}\,|\boldsymbol{\tau}, \boldsymbol{\Theta}'\right)$ inside the third sum and subsequently swapping the second and the third sum results in a weighted maximum-a-posteriori formulation

$$\begin{aligned} Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) &= \sum_{\boldsymbol{\tau} \in \mathcal{T}} \sum_{\boldsymbol{s} \in \mathcal{A}} \alpha_{\boldsymbol{s}} \log\left(p\left(c_{\boldsymbol{s}}\right) p\left(\boldsymbol{s}\,|\boldsymbol{\Theta}\right)\right), \\ \text{with } \alpha_{\boldsymbol{s}} &= \sum_{\mathcal{S} \in \mathcal{D}_{\boldsymbol{s}}} p\left(\mathcal{S}\,|\boldsymbol{\tau}, \boldsymbol{\Theta}'\right). \end{aligned} \tag{9}$$

After swapping the sums we first iterate over all possible segments $\boldsymbol{s} \in \mathcal{A}$ and then over the segmentations $\mathcal{S}$. It is important to make sure that we only iterate over segmentations that contain the respective segment $\mathcal{S} \in \mathcal{D}_{\boldsymbol{s}}$, since it would not be an equivalent transformation of $Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right)$ otherwise. The set of segmentations that contain $\boldsymbol{s}$ is defined as $\mathcal{D}_{\boldsymbol{s}} = \{\mathcal{S} | \boldsymbol{s} \in \mathcal{S}, \forall \mathcal{S} \in \mathcal{D}\}$.

Pulling the log into the product results in an additive constant, $\sum_{\boldsymbol{\tau} \in \mathcal{T}} \sum_{\boldsymbol{s} \in \mathcal{A}} \alpha_{\boldsymbol{s}} \log p\left(c_{\boldsymbol{s}}\right)$, which does only depend on $\boldsymbol{\Theta}'$ and not on the argument of the maximization $\boldsymbol{\Theta}$, and, hence, can be dropped yielding

$$Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) = \sum_{\boldsymbol{\tau} \in \mathcal{T}} \sum_{\boldsymbol{s} \in \mathcal{A}} \alpha_{\boldsymbol{s}} \log p\left(\boldsymbol{s}\,|\boldsymbol{\Theta}\right). \tag{10}$$

The segment weighting $\alpha_{\boldsymbol{s}}$ determines how likely the segment $\boldsymbol{s}$ belongs to the optimal segmentation $\boldsymbol{s} \in \mathcal{S}^*$, given the current model estimate $\boldsymbol{\Theta}'$. The EM algorithm iteratively computes the weighting $\alpha_{\boldsymbol{s}}$ in the E-Step, according to Equation (9), and updates the current model estimate, $\boldsymbol{\Theta}' \leftarrow \boldsymbol{\Theta}$, in the M-Step by choosing a $\boldsymbol{\Theta}$ that maximizes Equation (10).

Expectation Step: Computing the Probability of the Segments. In the E-Step, the segment weighting $\alpha_{\boldsymbol{s}}$, as described in Equation (9) is updated, and, therefore, the segments $\boldsymbol{s} \in \mathcal{A}$ in Equation (10) are re-weighted. The weighting in Equation (9) is computed by summing over all segmentations $\mathcal{S} \in \mathcal{D}_{\boldsymbol{s}}$, which

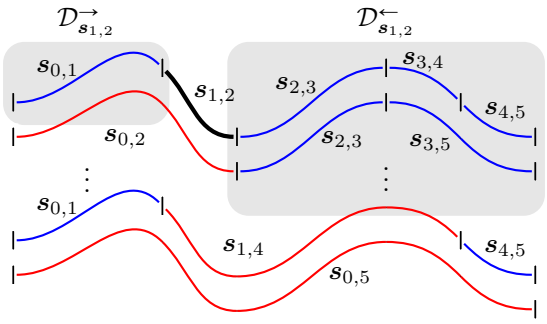


Figure 3. The figure illustrates several segmentations of a one dimensional trajectory with four potential cutting points, including the start and end of the trajectory. The indices of the segments denote at which cutting point the segment begins and at which it ends. Assuming the segment $\boldsymbol{s}_{1,2}$ is of interest, the blue segments occur alongside $\boldsymbol{s}_{1,2}$ in at least one segmentation. The red segments contain $\boldsymbol{s}_{1,2}$ and can therefore not occur in the same segmentation. The gray areas illustrate all possible preceding and succeeding segmentations of $\boldsymbol{s}_{1,2}$, denoted as $\mathcal{D}^{\rightarrow}_{\boldsymbol{s}_{1,2}}$ and $\mathcal{D}^{\leftarrow}_{\boldsymbol{s}_{1,2}}$ respectively.

contain the segment $\boldsymbol{s}$. Computing $\alpha_{\boldsymbol{s}}$ according to Equation (9) is therefore still of exponential complexity with respect to the number cuts $|\mathcal{C}|$. However, the segment weighting $\alpha_{\boldsymbol{s}}$ can be computed much more efficiently, by reformulating Equation (9) slightly. Applying Bayes Theorem on Equation (9) yields

$$\alpha_{\boldsymbol{s}} = \sum_{\mathcal{S}\in\mathcal{D}_{\boldsymbol{s}}} \frac{p\left(\boldsymbol{\tau}\,|\mathcal{S},\boldsymbol{\Theta}'\right)p\left(\mathcal{S}\right)}{p\left(\boldsymbol{\tau}\,|\boldsymbol{\Theta}'\right)}.$$

We can pull the denominator out of the sum since it is a constant with respect to the summation. Note, that the prior $p\left(\mathcal{S}\right)$ as defined in Equation (8) does not depend on the parameters $\boldsymbol{\Theta}'$.

$$\begin{aligned}\alpha_{\boldsymbol{s}} &= \frac{1}{Z}\sum_{\mathcal{S}\in\mathcal{D}_{\boldsymbol{s}}} p\left(\boldsymbol{\tau}\,|\mathcal{S},\boldsymbol{\Theta}'\right)p\left(\mathcal{S}\right)\\ Z &= p\left(\boldsymbol{\tau}\,|\boldsymbol{\Theta}'\right)\end{aligned}$$

Inserting Equation (5), Equation (6) and Equation (8) results in

$$\begin{aligned}\alpha_{\boldsymbol{s}} &= \frac{1}{Z}\sum_{\mathcal{S}\in\mathcal{D}_{\boldsymbol{s}}}\prod_{\boldsymbol{s}'\in\mathcal{S}} f\left(\boldsymbol{s}'\right), \qquad (11)\\ Z &= \sum_{\mathcal{S}\in\mathcal{D}}\prod_{\boldsymbol{s}'\in\mathcal{S}} f\left(\boldsymbol{s}'\right),\\ f\left(\boldsymbol{s}'\right) &= p\left(c_{\boldsymbol{s}'}\right)\; p\left(\boldsymbol{s}'\,|\boldsymbol{\Theta}'\right),\end{aligned}$$

where Z denotes the normalizing constant.

Such a formulation is well studied in Graphical Models and can be solved efficiently using message passing algorithms (Bishop 2006). In Equation (11) we iterate over all possible segmentations containing the segment $\boldsymbol{s}$, $\mathcal{S}\in\mathcal{D}_{\boldsymbol{s}}$. By definition, $\mathcal{D}_{\boldsymbol{s}}$ is a combination of all possible segmentations preceding $\boldsymbol{s}$ denoted as $\mathcal{D}^{\rightarrow}_{\boldsymbol{s}}$, $\boldsymbol{s}$ itself, and all possible segmentations succeeding $\boldsymbol{s}$ denoted as $\mathcal{D}^{\leftarrow}_{\boldsymbol{s}}$. The sets $\mathcal{D}^{\rightarrow}_{\boldsymbol{s}}$ and $\mathcal{D}^{\leftarrow}_{\boldsymbol{s}}$ are illustrated as gray boxes in Figure 3 with $\boldsymbol{s}=\boldsymbol{s}_{1,2}$ for a trajectory with four possible cutting points, including the start and end of the trajectory. The indices denote at which cutting points the segment starts and ends. The blue segments occur alongside $\boldsymbol{s}_{1,2}$ in at least one segmentation, and therefore, have to be considered when computing the segment weighting $\alpha_{\boldsymbol{s}_{1,2}}$. In contrast, the red segments already include $\boldsymbol{s}_{1,2}$, and therefore, must not be considered. We can rewrite Equation (11) as

$$\alpha_{\boldsymbol{s}} = \frac{1}{Z}\delta^{\rightarrow}_{\boldsymbol{s}} f\left(\boldsymbol{s}\right)\delta^{\leftarrow}_{\boldsymbol{s}}, \qquad (12)$$

$$\delta^{\rightarrow}_{\boldsymbol{s}} = \sum_{\mathcal{S}\in\mathcal{D}^{\rightarrow}_{\boldsymbol{s}}}\prod_{\boldsymbol{s}'\in\mathcal{S}} f\left(\boldsymbol{s}'\right), \qquad (13)$$

$$\delta^{\leftarrow}_{\boldsymbol{s}} = \sum_{\mathcal{S}\in\mathcal{D}^{\leftarrow}_{\boldsymbol{s}}}\prod_{\boldsymbol{s}'\in\mathcal{S}} f\left(\boldsymbol{s}'\right). \qquad (14)$$

The terms $\delta^{\rightarrow}_{\boldsymbol{s}}$ and $\delta^{\leftarrow}_{\boldsymbol{s}}$ are defined over all possible preceding and succeeding segmentations respectively, which are still exponential in the number of cutting points. We can, however, reformulate both terms as messages along a factor graph, resulting in a significantly lower computational complexity. Each segment represents a node in a factor graph, where each segment has a dedicated factor node and segments that share a possible cutting point are additionally connected via factor nodes. Figure 4 shows the factor graph corresponding to the example trajectory of Figure 3. Analogously, $\boldsymbol{s}_{1,2}$ is the segment of interest and the blue and red nodes correspond to the blue and red segments.

Instead of iterating over all succeeding segmentations $\mathcal{S}\in\mathcal{D}^{\leftarrow}_{\boldsymbol{s}}$, we can iterate over the direct successors $\boldsymbol{s}'\in\mathcal{A}^{\text{succ}}_{\boldsymbol{s}}$, and their succeeding segmentations $\mathcal{S}\in\mathcal{D}^{\leftarrow}_{\boldsymbol{s}'}$. The set $\mathcal{A}^{\text{succ}}_{\boldsymbol{s}}$ contains all segments, that begin at the cutting point where $\boldsymbol{s}$ ends. Analogously, the set $\mathcal{A}^{\text{pred}}_{\boldsymbol{s}}$ contains all segments that end immediately before $\boldsymbol{s}$. The reformulated Equation (13)

$$\delta^{\leftarrow}_{\boldsymbol{s}} = \sum_{\boldsymbol{s}'\in\mathcal{A}^{\text{succ}}_{\boldsymbol{s}}}\gamma^{\leftarrow}_{\boldsymbol{s}'}, \qquad (15)$$

$$\gamma^{\leftarrow}_{\boldsymbol{s}'} = f\left(\boldsymbol{s}'\right)\sum_{\mathcal{S}\in\mathcal{D}^{\leftarrow}_{\boldsymbol{s}'}}\prod_{\boldsymbol{s}''\in\mathcal{S}} f\left(\boldsymbol{s}''\right),$$

$$\gamma^{\leftarrow}_{\boldsymbol{s}'} = f\left(\boldsymbol{s}'\right)\delta^{\leftarrow}_{\boldsymbol{s}'}, \qquad (16)$$

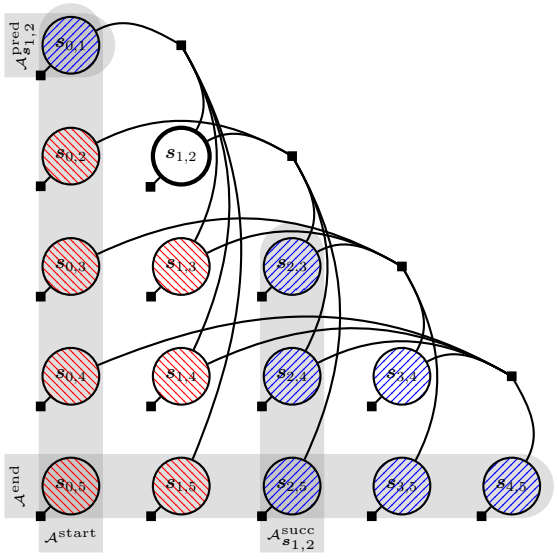


Figure 4. The factor graph corresponds to an observed trajectory shown in Figure 3. The nodes correspond to the different segments. Analogously the segment $\boldsymbol{s}_{1,2}$ is of interest and the blue and red nodes correspond to the blue and red segments. All nodes directly connected to $\boldsymbol{s}_{i,j}$ are considered its neighbors. Neighbors from above are predecessors and neighbors to the right are successors, e.g, $\mathcal{A}^{\text{pred}}_{\boldsymbol{s}_{1,2}} = \{\boldsymbol{s}_{0,1}\}$ and $\mathcal{A}^{\text{succ}}_{\boldsymbol{s}_{1,2}} = \{\boldsymbol{s}_{2,3}, \boldsymbol{s}_{2,4}, \boldsymbol{s}_{2,5}\}$. Additionally, the sets $\mathcal{A}^{\text{start}} = \{\boldsymbol{s}_{0,1}, \boldsymbol{s}_{0,2}, \boldsymbol{s}_{0,3}, \boldsymbol{s}_{0,4}, \boldsymbol{s}_{0,5}\}$ and $\mathcal{A}^{\text{end}} = \{\boldsymbol{s}_{0,5}, \boldsymbol{s}_{1,5}, \boldsymbol{s}_{2,5}, \boldsymbol{s}_{3,5}, \boldsymbol{s}_{4,5}\}$ contain all segments starting at the beginning of the trajectory or ending at the end of the trajectory respectively.

is now defined in terms of the backward messages, $\delta^{\leftarrow}_{\boldsymbol{s}}$ and $\gamma^{\leftarrow}_{\boldsymbol{s}}$ of the described factor graph. The forward messages are derived analogously,

$$\delta^{\rightarrow}_{\boldsymbol{s}} = \sum_{\boldsymbol{s}' \in \mathcal{A}^{\text{pred}}_{\boldsymbol{s}}} \gamma^{\rightarrow}_{\boldsymbol{s}'}, \tag{17}$$

$$\gamma^{\rightarrow}_{\boldsymbol{s}'} = f(\boldsymbol{s}')\, \delta^{\rightarrow}_{\boldsymbol{s}'}. \tag{18}$$

The $\gamma^{\rightarrow}_{\boldsymbol{s}}$ messages are always send from segment nodes to factor nodes and contain the product of the segment probability and all incoming $\delta^{\rightarrow}_{\boldsymbol{s}}$ messages. In our case each segment node has at most one incoming $\delta^{\rightarrow}_{\boldsymbol{s}}$ message. The $\delta^{\rightarrow}_{\boldsymbol{s}}$ messages are send from factor nodes to segment nodes and are simply the sum of all incoming $\gamma^{\rightarrow}_{\boldsymbol{s}}$ messages. The same applies equivalently to the backward messages $\gamma^{\leftarrow}_{\boldsymbol{s}}$ and $\delta^{\leftarrow}_{\boldsymbol{s}}$. Because of this representation as a sum of products this type of message passing is also referred to as sum-product algorithm.

The normalizing constant Z can also be expressed in terms of either the forward or backward messages

$$Z = \sum_{\boldsymbol{s} \in \mathcal{A}^{\text{start}}} \gamma^{\leftarrow}_{\boldsymbol{s}} = \sum_{\boldsymbol{s} \in \mathcal{A}^{\text{end}}} \gamma^{\rightarrow}_{\boldsymbol{s}},$$

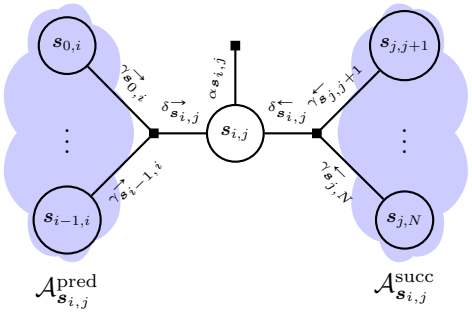


Figure 5. The factor graph illustrates the computation of the segment weighting $\alpha_{\boldsymbol{s}_{i,j}}$, formulated as massage passing. The $\delta^{\rightarrow}_{\boldsymbol{s}_{i,j}}$ and $\delta^{\leftarrow}_{\boldsymbol{s}_{i,j}}$ messages are the sums of the incoming $\gamma^{\rightarrow}_{\boldsymbol{s}_{0..i-1,i}}$ and $\gamma^{\leftarrow}_{\boldsymbol{s}_{j,j+1..N}}$ messages. The messages $\gamma^{\rightarrow}_{\boldsymbol{s}_{0..i-1,i}}$ and $\gamma^{\leftarrow}_{\boldsymbol{s}_{j,j+1..N}}$ are passed by the preceding and succeeding nodes respectively, $\mathcal{A}^{\text{pred}}_{\boldsymbol{s}_{i,j}}$ and $\mathcal{A}^{\text{succ}}_{\boldsymbol{s}_{i,j}}$. The weighting $\alpha_{\boldsymbol{s}_{i,j}}$ is the product of the incoming forward and backward messages time $f(\boldsymbol{s}_{i,j})/Z$, as described in Equation (12).

where the set of segments $\mathcal{A}^{\text{start}}$ contains all possible segments that start at the beginning of the observed trajectory. Simultaneously the set $\mathcal{A}^{\text{end}}$ contains all possible segments that end at the end of the trajectory.

The recursive nature of the messages explains the reduced computational complexity. Each forward or backward message is a combination of the incoming forward or backward messages respectively, and has to be computed exactly once. To compute the forward messages, we begin at the start nodes, $\mathcal{A}^{\text{start}}$, and pass the messages towards all of the end nodes, $\mathcal{A}^{\text{end}}$. To compute the backward messages we reverse the process, i.e., begin at the end nodes, $\mathcal{A}^{\text{end}}$, and pass the messages towards the start nodes, $\mathcal{A}^{\text{start}}$.

The segment weighting $\alpha_{\boldsymbol{s}}$ can now be interpreted as the message send from the node $\boldsymbol{s}$ to the dedicated factor node, as visualized in Figure 5, where the left subtree consists of all preceding segments and the right subtree of all succeeding segments.

Maximization Step: Learning the Movement Primitive Library. In the maximization step, the model parameters $\boldsymbol{\Theta}$ are updated by maximizing $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}')$. We assume that all observed demonstrations were generated by the same underlying model, implying that the model update considers all possible segments independently of their corresponding demonstration.

Following (Paraschos et al. 2013), we update the model based on the projected segments rather than the segments directly. Therefore, the maximization step is defined over the set of all projected segments of all observed trajectories

$$\mathcal{W} = \{\boldsymbol{w} \,|\boldsymbol{w} = \boldsymbol{v}(\boldsymbol{s}), \forall \boldsymbol{s} \in \mathcal{A}_{\boldsymbol{\tau}}, \forall \boldsymbol{\tau} \in \mathcal{T}\}, \tag{19}$$

Algorithm 1: Probabilistic Segmentation

input : The initial set of cutting points $\mathcal{C}$
output : The underlying mixture model $\boldsymbol{\Theta}^*$
The underlying segmentation $\mathcal{S}^*$

K : current number of clusters
L : current labeling of the segments
while not converged **do**
- **E-Step :** compute the weighting $\alpha_{\boldsymbol{s}}$ as described in Equation (12)
- **M-Step:** 1. compute $\mathcal{W}$ according to Equation (19)
  2. determine K and L by applying Gaussian-means on $\mathcal{W}$
  3. update $\boldsymbol{\Theta}^*$ by using a weighted EM-GMM on $\mathcal{W}$ with K clusters, initial labeling L and weights $\alpha_{\boldsymbol{w}}$

where $\boldsymbol{v}$ is the ridge regression defined in Equation (1) and the set $\mathcal{A}_{\boldsymbol{\tau}}$ contains all possible segments of the observed trajectory $\boldsymbol{\tau}$. Besides the significantly lower dimensionality of each projected segment $\boldsymbol{w}$ compared to $\boldsymbol{s}$, working in the projected space has the advantage of comparing different segments time invariantly, purely based on their shape. Therefore, similar segments, which mainly differ in their execution speed, will be assigned to the same primitive. Accordingly, we define $\alpha_{\boldsymbol{s}}$ over the projected segments

$$\alpha_{\boldsymbol{w}} = \alpha_{\boldsymbol{s}} \iff \boldsymbol{w} = \boldsymbol{v}(\boldsymbol{s}).$$

Given these definitions, the reformulated auxiliary function

$$Q_{\mathcal{W}}\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) = \sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \log p\left(\boldsymbol{w}\,|\boldsymbol{\Theta}\right), \tag{20}$$

$$p\left(\boldsymbol{w}\,|\boldsymbol{\Theta}\right) = \sum_{k=1}^{|\mathcal{M}|} \lambda_k p\left(\boldsymbol{w}\,|\boldsymbol{\theta}_k\right), \text{ with} \tag{21}$$

takes the form of a weighted log-likelihood, where $p\left(\boldsymbol{w}\,|\boldsymbol{\Theta}\right)$ is defined as standard Gaussian Mixture Model (GMM), given Equation (2). Because $\mathcal{W}$ was defined as a set which contains all the projected segments for all trajectories $\boldsymbol{\tau} \in \mathcal{T}$, the sum over $\mathcal{W}$ replaces both sums over $\boldsymbol{\tau}$ and $\boldsymbol{s}$.

It is unknown which projected segment belongs to which movement primitive, resulting in $p\left(\boldsymbol{w}\,|\boldsymbol{\Theta}\right)$ being a mixture model with latent variables. Therefore, it is unfeasible to maximize $Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right)$ directly with respect to the parameters $\boldsymbol{\Theta}$.

It is, however, possible to estimate $\boldsymbol{\Theta}$ by maximizing the weighted maximum log-likelihood. Given our model we can again apply a weighted EM algorithm for GMMs. Instead of maximizing the log-likelihood for a single projected segment $\log p\left(\boldsymbol{w}\,|\boldsymbol{\Theta}\right)$, we can maximize the auxiliary function

$$Q_{\boldsymbol{w}}\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) = \sum_{k=1}^{|\mathcal{M}|} \beta_{k,\boldsymbol{w}} \log \lambda_k p\left(\boldsymbol{w}\,|\boldsymbol{\theta}_k\right),$$

where the mixing coefficients $\sum_{k=1}^{|\mathcal{M}|} \lambda_k = 1$ sum up to one and $\beta_{k,\boldsymbol{w}} = p\left(k\,\middle|\,\boldsymbol{w}, \boldsymbol{\Theta}'\right)$ are typically referred to as responsibilities. Replacing $\log p\left(\boldsymbol{w}\,|\boldsymbol{\Theta}\right)$ for $Q_{\boldsymbol{w}}\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right)$ in Equation (22),

$$Q_{\mathcal{W}}\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) = \sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \sum_{k=1}^{|\mathcal{M}|} \beta_{k,\boldsymbol{w}} \log \lambda_k p\left(\boldsymbol{w}\,|\boldsymbol{\theta}_k\right),$$

and rearranging the sums yields the standard auxiliary function of a weighted EM for GMMs

$$Q_{\mathcal{W}}\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right) = \sum_{k=1}^{|\mathcal{M}|} \sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \beta_{k,\boldsymbol{w}} \log \lambda_k p\left(\boldsymbol{w}\,|\boldsymbol{\theta}_k\right). \tag{22}$$

Since the set $\mathcal{W}$ is defined across all observed trajectories, the model will contain primitives learned from segments of different observed trajectories. The EM algorithm for GMMs is widely applied and well known. In the expectation step, the responsibilities are updated

$$\beta_{k,\boldsymbol{w}} = \frac{\lambda'_k p\left(w\,\middle|\,\boldsymbol{\theta}'_k\right)}{\sum_{k'=1}^{|\mathcal{M}|} \lambda'_{k'} p\left(w\,\middle|\,\boldsymbol{\theta}'_{k'}\right)},$$

which emerges from applying Bayes theorem.

Maximizing the auxiliary function $Q_{\mathcal{W}}\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right)$ represents a constrained optimization problem, which can be solved by applying the method of Lagrangian multipliers. Solving the Lagrangian leads to the model updates performed in the maximization step. The updates for each parameter are given by

$$\lambda_k = \frac{\sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \beta_{k,\boldsymbol{w}}}{\sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}}},$$

$$\boldsymbol{\mu}_k = \frac{\sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \beta_{k,\boldsymbol{w}} \boldsymbol{w}}{\sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \beta_{k,\boldsymbol{w}}} \text{ and}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \beta_{k,\boldsymbol{w}} \left(\boldsymbol{w} - \boldsymbol{\mu}_k\right)\left(\boldsymbol{w} - \boldsymbol{\mu}_k\right)^T}{\sum_{\boldsymbol{w}\in\mathcal{W}} \alpha_{\boldsymbol{w}} \beta_{k,\boldsymbol{w}}}.$$

A known disadvantage of EM for mixture models is that the number of components is generally assumed to be known a priori. Additionally, the number of movement primitives $|\mathcal{M}|$ in our approach depends on the latent segmentation, and can therefore change every time we change the weighting of the segments. We circumvent this problem by using the Gaussian-means algorithm (Hamerly and Elkan 2003)

(a) While holding a pen, the robot was demonstrated writing trajectories on a white board using kinesthetic teaching.

(b) The robot executes a trajectory which was sequenced from the learned movement primitive library.

Figure 6. The experimental setup of the real robot writing task. The executed sequence is not restricted to three letter words, except by the whiteboard. The observed trajectories were demonstrated by kinesthetic teaching. Subsequently a movement primitive library was learned by segmenting the trajectories using ProbS. Finally sequences of the learned movement primitive library were executed on the real robot platform.

to determine the number of movement primitives, including an initial labeling for the EM algorithm. The Gaussian-means algorithm is a bisecting k-means algorithm which uses a test based on the Anderson-Darling statistic to determine if the data assigned to a cluster is Gaussian or not. If the data is not Gaussian, the cluster is split. We summarize our Probabilistic Segmentation (ProbS) method in Algorithm 1.

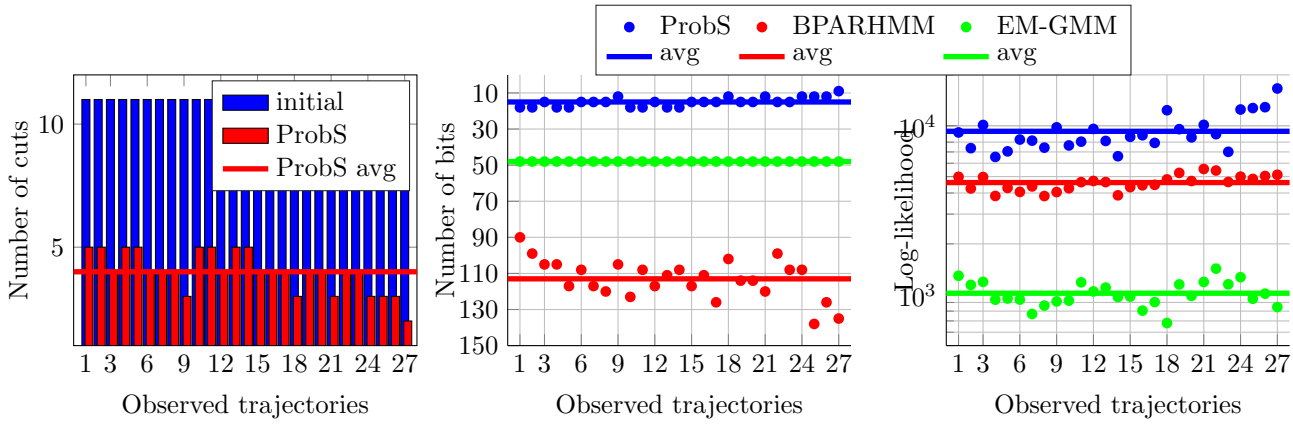
Convergence of ProbS and Complexity of the Auxiliary Function $Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right)$. The maximization step of the proposed method is itself an EM for GMMs, which is known to converge to a local maximum of the log-likelihood. Furthermore, maximizing $Q_{\mathcal{W}}\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right)$ will at least find a local maximum for $Q\left(\boldsymbol{\Theta}, \boldsymbol{\Theta}'\right)$, which guarantees the convergence of the proposed method (Wu 1983). Due to the number of segmentations $|\mathcal{D}| = 2^{|\mathcal{C}|}$, the initial formulation in Equation (7) is in $\mathcal{O}\left(2^{|\mathcal{C}|}\right)$. After the reformulations given in Equation (10) and Equation (12) the problem is defined over the segments. The number of segments is quadratic in the number of cuts, $|\mathcal{A}| = 0.5(|\mathcal{C}| + 1)(|\mathcal{C}| + 2)$, and therefore, computing Equation (10) is in $\mathcal{O}\left(|\mathcal{C}|^2\right)$. The reformulation therefore allows to consider significantly more cuts.

Experimental Evaluation of ProbS

We evaluated our method on a real robot writing task. We compared it to a baseline method, Expectation-Maximization algorithm Gaussian mixture models, and a state-of-the-art non-parametric segmentation method, Beta Process Autoregressive Hidden Markov Model,(Fox 2009),(Niekum et al. 2012). In (Lioutikov et al. 2015) we performed a real robot chair assembly task. In order to show that useful movement primitives were extracted, we replayed the movement primitives on the real robot, while conditioning the movement primitives to assemble the chair in and undemonstrated order. In both tasks the experimental platform was a seven DoF KUKA lightweight arm equipped with a five finger DLR HIT Hand II as end effector, as shown in Figure 1. Additionally, we segmented multiple demonstrations of a barrett wam robot platform returning balls in table tennis setup. In this task ProbS successfully identified primitives for forehand and backhand swings.

Setup of the Real Robot Writing Task

In the robot writing task the robot was demonstrated how to write trajectories as continuous motions via kinesthetic teaching, while holding a pen and drawing on a white board, as shown in Figure 6(a). A total of 27 words were demonstrated. Each word based on a three letter alphabet, containing the letters, “a”, “u” and “y”. The recorded data were the two dimensional trajectories on the whiteboard-plane. The initial over-segmentation was produced by a curvature-based heuristic, where the cutting points were identified by the curvature exceeding a certain threshold. The heuristic resulted in 11 cuts per observation leading to a total of 55296 possible segmentations or equivalently 2106 segments. ProbS reduced the number of active cutting points significantly, as illustrated in Figure 7(a). In average



(a) number of active cuts **(b)** bits needed for encoding (c) LOOCV

Figure 7. (a) ProbS is able to reduce the number of active cuts significantly, leading to a compact representation of the observations. (b) The quality of the found segments is quantified by the number of bits each observations requires to be encoded. Less is better. (c) The learned movement primitives are evaluated using a LOOCV. ProbS shows the advantage of including the movement primitive learning in the segmentation process.

ProbS reduced the number of cutting points to four per observation.

In order to demonstrate the advantage of optimizing both the segmentation and the movement primitive library iteratively, we chose an Expectation-Maximization algorithm over Gaussian mixture models (EM-GMM) as a baseline, where the number of clusters as well as the initial labeling is determined by the Gaussian-means algorithm(Hamerly and Elkan 2003). In addition, we compared our method to the state-of-the-art, non-parametric segmentation method Beta Process Autoregressive Hidden Markov Model (BP-AR-HMM), as applied in (Fox 2009) and (Niekum et al. 2012). Given the found number of movement primitives and the initial labeling of the Gaussian-means algorithm, the EM-GMM baseline identified a total of eight movement primitives. The BP-AR-HMM method does not rely on an initial segmentation, however, due to the sampling process, the algorithm is computationally very expensive. In our evaluation the best solutions were found with an autoregressive order of two. Furthermore, the segmentation had to be performed on the velocity of the trajectories, since the method was otherwise not able to identify the same movement primitive at different absolute positions. The final segmentation identified seven movement primitives.

ProbS was able to identify six underlying movement primitives, which are shown in Figure 8. The brighter colored background illustrates the variance of each primitive. In each demonstration, the letter "a" begins further right than "u" or "y" and is therefore proceeded by a longer tail than the other letters. ProbS separated

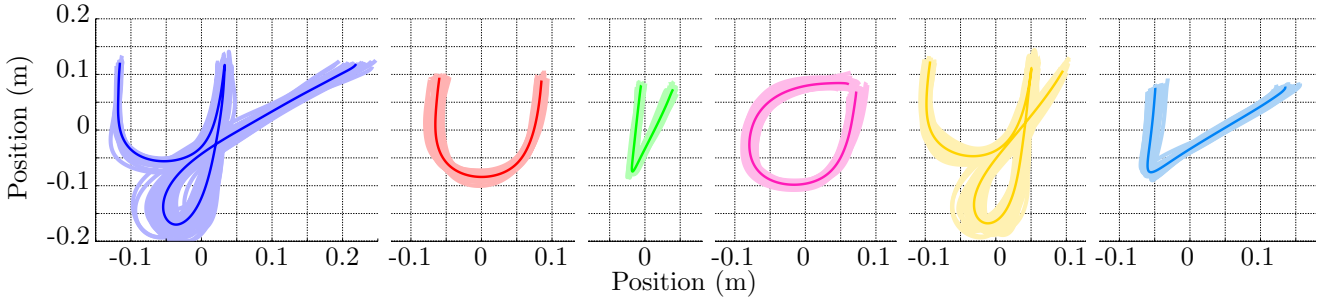


Figure 8. The six movement primitives learned by ProbS. From left to right. 1: The letter "y" when proceeding an "a". 2:The corpus of an "u". 3: The tail of the letters "a" and "u" when not followed by an "a". 4: The corpus of an "a". 5: the letter "y" when not proceeding an "a". 6: The tail of the letters "a" and "u" when followed by an "a".

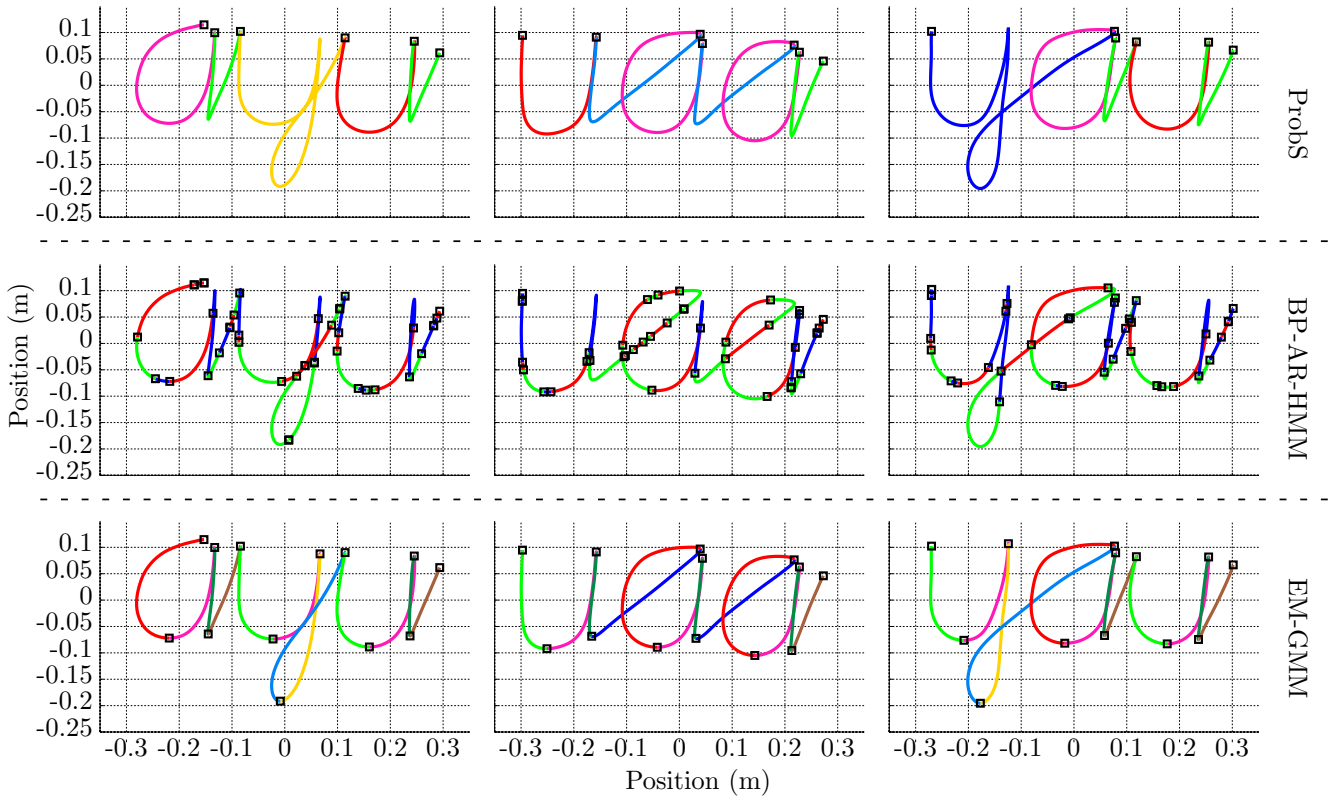
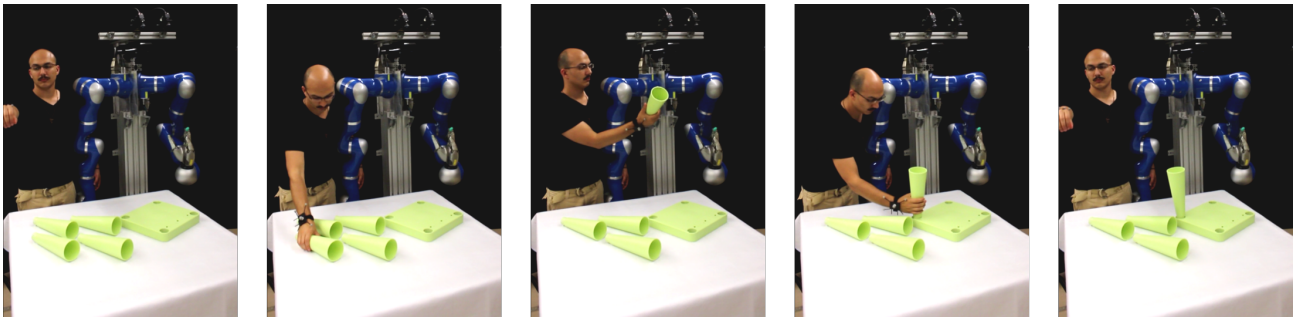


Figure 9. Comparison of ProbS, BP-AR-HMM, EM-GMM on the letter segmentation task. All methods resulted in a similar number of movement primitives (8, 6, 7), but the resulting segmentations differed significantly. Same color within one row indicates the assignment to the same movement primitive.

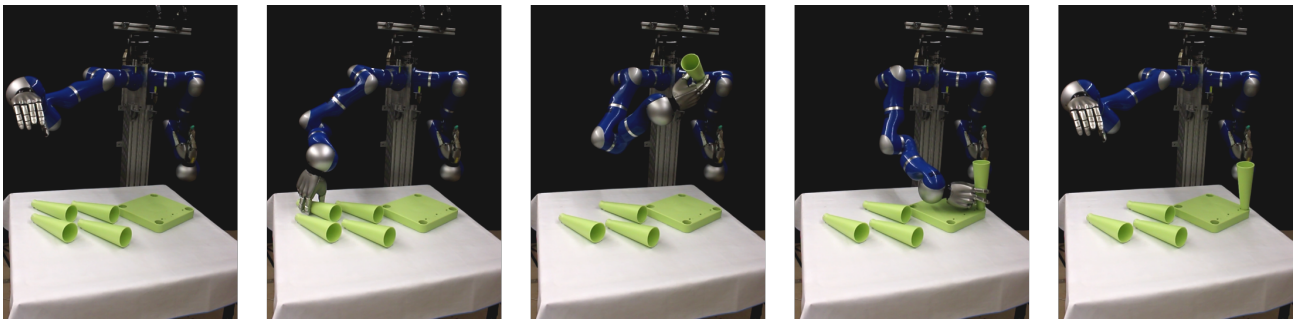
those variations for "a" and "u" into their corpus and the two different tails, leading to the movement primitives 2,3,4 and 6. Movement primitives 1 and 5 show the two variations of the letter "y". These variations were not merged into a single primitive, because the merged skill would achieve a significantly lower average likelihood across all "y"-segments than the two learned primitives achieve for their respective "y"-segments. Additional demonstrations in between those two variants would most likely yield a single "y" primitive.

Comparison on the Letter Segmentation. A comparison of the methods is shown in Figure 9. While ProbS and the EM-GMM benefit from the initial segmentation, the BP-AR-HMM found very different segments. However, many of the segments found by BP-AR-HMM occur in very different shapes and lengths, which results in movement primitives with high variance. Given the curvature based heuristic the segments found by ProbS and the EM-GMM appear semantically meaningful and have a high recall value throughout the observations.

Encoding Lengths of the Different Segmentations. As a metric to quantify the value of the segmentation we computed the number of bits necessary to encode each observed trajectory $\boldsymbol{\tau}$ given the identified segments $N_{\boldsymbol{\tau}} = |\mathcal{S}_{\boldsymbol{\tau}}| N_{\mathcal{M}}$. $\mathcal{S}_{\boldsymbol{\tau}}$ is the learned segmentation for trajectory $\boldsymbol{\tau}$ and $N_{\mathcal{M}} = \lceil \log_2 |\mathcal{M}| \rceil$ denotes the number bits necessary to encode each learned primitive uniquely. The coding length for each observation as well as the average, is shown in Figure 7(b). As the EM-GMM does not change the number of segments in any observation the encoding solely depends on the predefined number of cluster, which in this case leads to a fixed length of 48 bits per observation. By reducing the number of active cuts and learning a small set of movement primitives ProbS achieved an average encoding length of 15 bits. While BP-AR-HMM found a similar number of movement primitives, each observation was explained by significantly more segments than in either the EM-GMM or ProbS. BP-AR-HMM achieved a high average encoding length of 113 bits. This evaluation shows, that ProbS was able to find segments, which allow a compact representation of the observations.



(a) Demonstration of the chair assembly recorded via the OptiTrack motion capture system.



(b) Execution of the learned movement primitives.

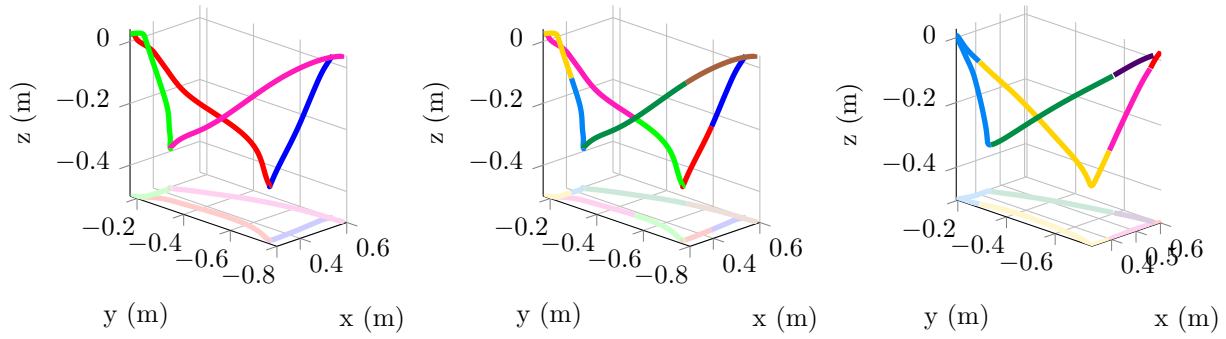
Figure 10. Demonstration and execution of the chair assembly task. The human was tracked during the chair assembly. The observed demonstrations were subsequently segmented using ProbS. The movement primitives of the learned library were subsequently sequenced and executed on a robot platform to assemble the chair.

Leave-One-Out Cross Validation of the Learned Movement Primitives. Another important aspect of ProbS is the quality of the learned movement primitives. We compared to the other two methods by applying a leave-one-out cross validation (LOOCV) on each of the learned segmentations. Thus, learning the movement primitives excluding one observation and subsequently computing the average log-likelihood of each segment in the excluded observation given the learned movement primitives. Let $\mathcal{S}_{\boldsymbol{\tau}}$ and and $\mathcal{D}_{\mathcal{T}} = \{\mathcal{S}_{\boldsymbol{\tau}} | \forall \boldsymbol{\tau} \in \mathcal{T}\}$ be the learned segmentation for trajectory $\boldsymbol{\tau}$ and respectively the set of all learned segmentations for the set of trajectories $\mathcal{T}$. The set $\mathcal{D}_{\mathcal{T}\setminus\boldsymbol{\tau}} = \mathcal{D}_{\mathcal{T}} \setminus \mathcal{S}_{\boldsymbol{\tau}}$ then denotes the set of all learned segmentations $\mathcal{D}_{\mathcal{T}}$ except $\mathcal{S}_{\boldsymbol{\tau}}$. The average log-likelihood for the LOOCV is now given as $\bar{\ell} = 1/|\mathcal{T}| \sum_{\boldsymbol{\tau}\in\mathcal{T}} \sum_{\boldsymbol{s}\in\mathcal{S}_{\boldsymbol{\tau}}} \log p\left(\boldsymbol{s} \left| \boldsymbol{\Theta}_{\mathcal{D}_{\mathcal{T}\setminus\boldsymbol{\tau}}} \right.\right)$, where $\boldsymbol{\Theta}_{\mathcal{D}_{\mathcal{T}\setminus\boldsymbol{\tau}}}$ denotes the mixture of primitives learned from $\mathcal{D}_{\mathcal{T}\setminus\boldsymbol{\tau}}$.

The results of the LOOCV are shown in Figure 7(c). The BP-AR-HMM outperformed the EM-GMM significantly, showing its superiority with an average of 4610 over the simple clustering method with an average of 1021. However, given its intrinsic optimization of the learned movement primitives, ProbS achieved a significantly higher average log-likelihood of 9272. At the same time, ProbS had the advantage of using additional knowledge given by the initial heuristic. In future work ProbS could replace the heuristic by a non-parametric method, similar to BP-AR-HMM, and hence, improve the so proposed segmentation.

Writing Using the Learned Movement Primitive Library. Finally, we show the applicability of our method in real robot scenarios, by executing a trajectory composed of the learned movement primitives. We are not restricted to a particular trajectory, but can randomly sample from our set of movement primitives, shown in Figure 8. Furthermore, we are not limited to reproducing the observations but can sequence an arbitrary number of movement primitives from the learned library. Additionally, we can take advantage of the various properties of MPs, e.g., choosing a different duration for each movement primitive and conditioning on certain positions. Figure 6 shows the execution of a sampled sequence. The sequence was randomly sampled from the mixture model. A transition table was learned from the segmented demonstrations to ensure a feasible sequence of selected components.

The initial over-segmentation was produced by a velocity based heuristic where the cutting points where



(a) ProbS **(b)** EM-GMM **(c)** BP-AR-HMM

Figure 11. The first demonstrations explained by the three different methods. Different colors within each plot illustrated different movement primitives. ProbS explains the demonstration with four movement primitives, EM-GMM identified eight movement primitives in the demonstration and BP-AR-HMM separated the demonstration into a total of 9 segments. Some of the segments have a very short duration and are not visible in the plot.

positioned at the extrema of the velocity profile for each observation.

The heuristic resulted in nine cuts per observation leading to a total of 768 possible segmentations or equivalently 216 segments. This experiment shows the feasibility of the learned movement primitives and the applicability of ProbS in real robot tasks.

Comparison on the Chair Assembly. We also compared ProbS to EM-GMM and BP-AR-HMM on the chair assembly task. Figure 11 shows the identified segments of each method for the first demonstration.

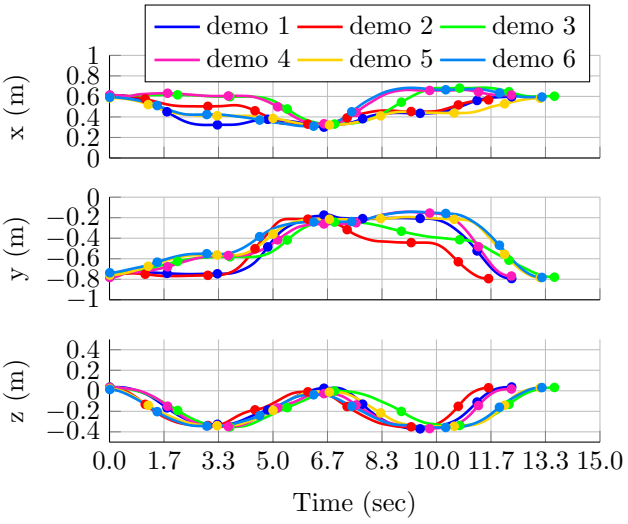


Figure 12. The first three dimensions, corresponding to the Cartesian position, of the six observations. Each observation demonstrates the insertion of a chair leg into a hole in the seat. The cuts determined by the initial heuristic are illustrated as dots.

ProbS was able to identify four underlying movement primitives. Each of the movement primitives occurs exactly once in every demonstration and corresponds to one of the four steps of each demonstration. The first three dimensions are shown in Figure 13. It is clearly visible that the movement primitives preserve a characteristic shape while the variance at certain time steps shows the adaptability of the movement primitive at those points.

Given the found number of movement primitives and the initial labeling of the Gaussian-means algorithm, the EM-GMM baseline identified a total of eight movement primitives. Analogously to ProbS each found movement primitive is present exactly once in each demonstration.

In this task BP-AR-HMM achieved the best results with an autoregressive order of one and identified four primitives. The method also explained similar segments in different demonstrations by the same primitives. However, it also identified very short segments within the demonstrations and assigned it to the same primitive as other much longer and differently shaped segments.

Setup of the Real Robot Chair Assembly Task

The chair assembly was demonstrated by a human, as seen in Figure 12. The wrist of the human was tracked using the OptiTrack motion capture system. A total of six demonstrations were performed, where each demonstration consisted of four phases, i.e., approaching and picking up a chair leg, showing the leg tip to the camera, approaching and inserting the leg into the seat and finally returning to the home position. Each data point is a seven dimensional vector

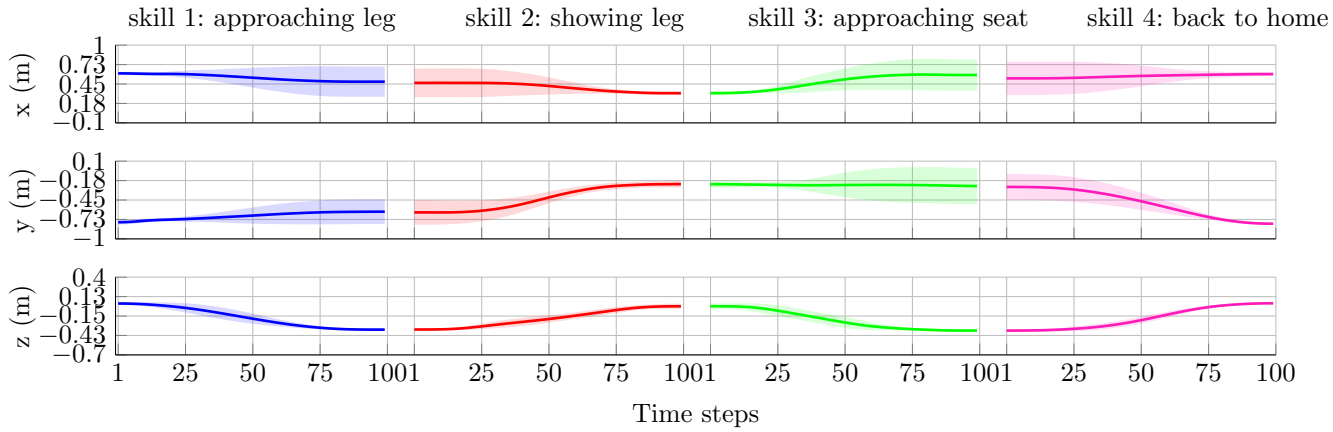


Figure 13. The first three dimensions of the four movement primitives learned by ProbS to exemplify the skills, omitting the remaining dimensions. The dark line shows the mean and the shaded area corresponds to two times the standard deviation. The movement primitives are semantically meaningful, i.e., approaching a leg, showing the leg to a camera, approaching the seat and going back to the home position.

in Cartesian space containing the three dimensional wrist position and the four dimensional orientation encoded as a quaternion. The tracked positions are shown in Figure 12.

Chair Assembly using the Learned Movement Primitives. We show the applicability of our method in real robot scenarios, by assembling an Ikea chair using the learned movement primitive library. As shown in Figure 10, the robot is able to extract the necessary movement primitives from the given demonstrations. Similarly, to the writing experiment each segment was drawn from the mixture model, while a learned transition table ensured that a sensible sequence of primitives was queried. The start and end point of each movement primitive were conditioned to the corresponding point of interest, e.g, the "inserting" movement primitive was conditioned to the hole position. Since each movement primitive was learned from only six samples, the variance at some points was too low to successfully condition the corresponding ProMP. Therefore, we scaled the covariance matrix of each movement primitive artificially. This scaling was only necessary for the execution and did not occur during the learning of the library. This experiment shows that ProbS is able to segment entire demonstrations to extract meaningful movement primitives. These movement primitives can be used and sequenced in order to solve observed as well as new tasks. For example, the chair was assembled with combinations of legs and holes which were not present in the demonstrations.

Robot Table Tennis

In this task multiple demonstrations of table tennis forehand and backhand swings were segmented into a total of four movement primitives. The swings were demonstrated on a 7 DoF barrett wam robot with a racket as an end effector. While one person threw table tennis balls in the direction of the robot another person applied kinesthetic teaching to return the balls with the robot, as shown in Figure 14. During the demonstrations the joint states were continuously recorded with a sampling rate of 100 Hz. A total of 20 forehand and 26 backhand swings were recorded randomly across 16 demonstrations in overall 189

Figure 14. Kinesthetic teaching of the barrett wam robot platform for the table tennis task. While one person throws table tennis balls towards the robot, another person moves the robot manually in order to return the balls. During the teaching all seven joints are recorded for the subsequent segmentation.

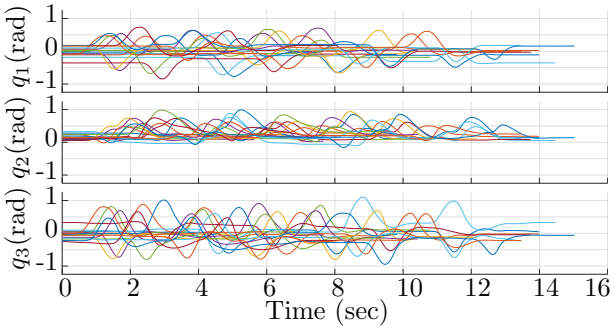


Figure 15. The first three joint states of the 16 demonstrations. The forehand and backhand swings occur randomly and are unequal in duration. Additionally, the duration between swings is not fixed and can be considered a waiting period.

seconds. The demonstrations for the first three joints are shown in Figure 15. The initial over-segmentation was given by a low velocity heuristic which resulted in a cut whenever all joints were below a certain threshold. A total of 127 cuts were initially detected with an average of 7.9 cuts per demonstration. ProbS learned a total of four primitives, a forehand swing, a backhand swing and two waiting primitives. For both waiting primitives the mean for each joint changes at most 0.03 radians, which effectively results in an almost steady racket. However, the two primitives contain the most movement in different joints, and hence, were not learned as a single primitive. The forehand and backhand primitives show characteristic joint movements, as seen in Figure 16. To illustrate the learned primitives the cartesian trajectories of the racket were computed by applying forward kinematics to the mean joint trajectories of the four primitives. The resulting movements are shown in Figure 17 alongside the original demonstrations.

Conclusion

We proposed a new algorithm for the segmentation of unlabeled trajectories. The algorithm builds a movement primitive library that is used to infer correct segmentations. The library as well as the inferred segmentations of the trajectories are iteratively optimized as there is a high dependency between both entities. We presented an efficient formulation of the algorithm, transforming multiple steps from exponential to quadratic complexity. Our algorithm takes advantage of heuristics that are used to over-segment the trajectories. In comparison to other state-of-the-art methods such as non-parametric auto-regressive HMMs, our algorithm has less hyper-parameters to fine tune and clear computational advantages. Furthermore, the returned movement primitive library looked much more compact than the ones retrieved with related approaches. The evaluation of the algorithm showed that the method can be applied in real world scenarios in both cartesian and joint space, such as the presented chair assembly task and the robot table tennis task.

In this work we did not attempt to learn libraries across various tasks. Given that both the E-Step and the M-Step treat the segments independent of the underlying task, the performance of ProbS should not change significantly. However, learning primitive libraries across multiple tasks will be part of future work. In addition, we will concentrate on reducing the reliance on the heuristics as well as learning high level control variables of each movement primitive, such as possible conditioning points.

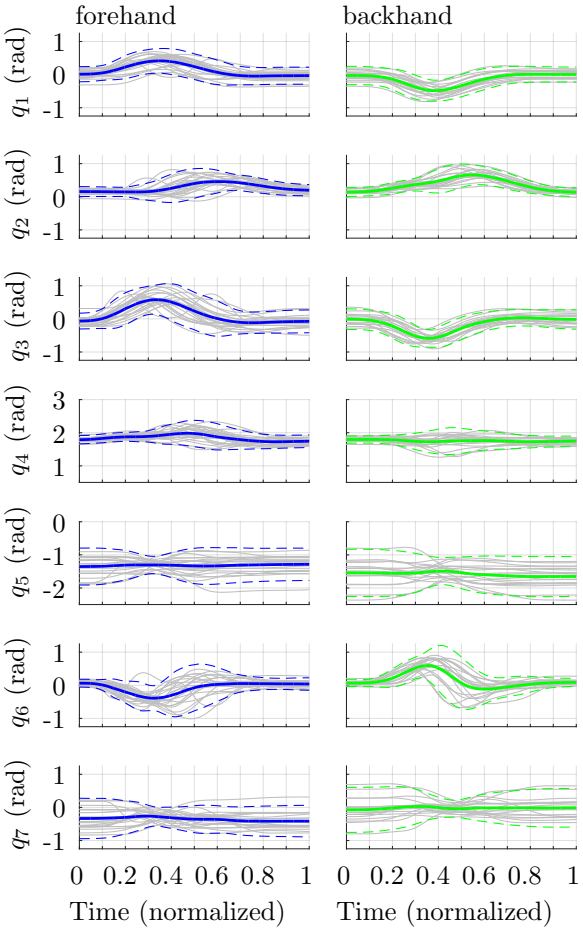


Figure 16. The mean and two times standard deviation of the forehand and backhand primitives for each joint. The grey lines indicate the corresponding segments used to learn the primitives. The segments and the primitives are normalized in time purely for illustration purposes. This normalization is not used for the actual ProbS algorithm.

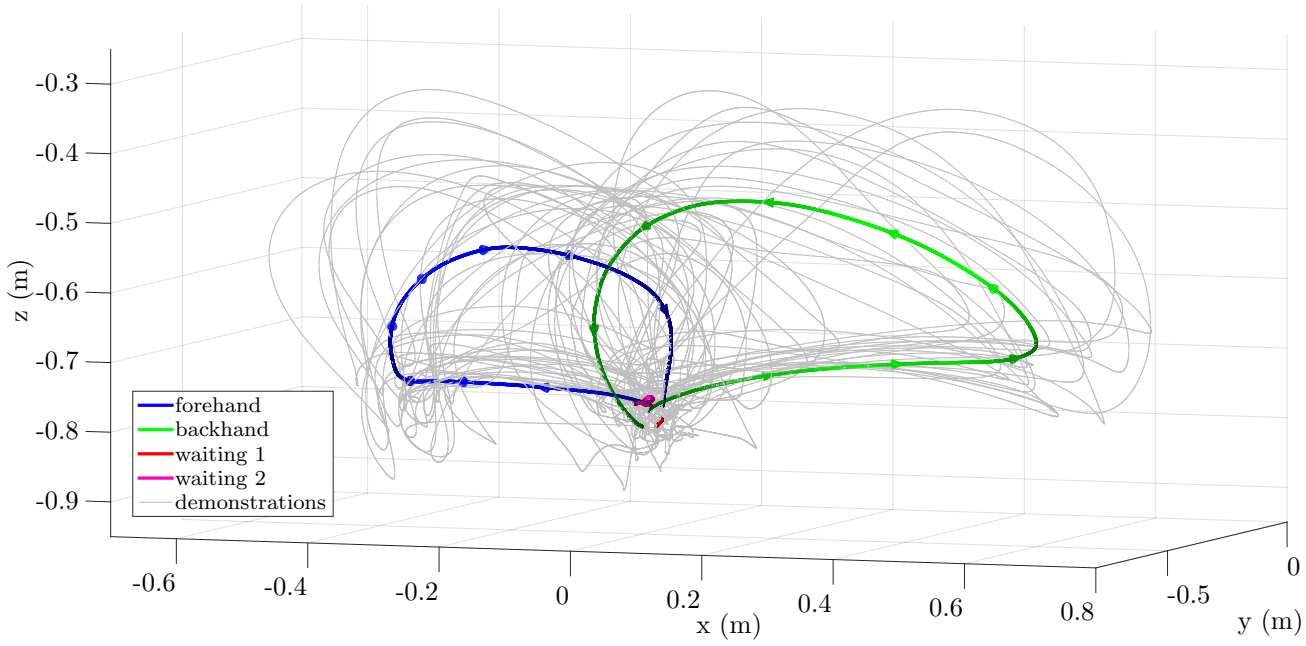


Figure 17. The four learned primitives for the robot table tennis task. The trajectories are shown in cartesian space and were computed by applying the forward kinematics to the mean of each corresponding primitive. The brightness represents the velocity of the trajectory and the arrows indicate the direction of the movement. The grey lines show the original demonstration.

Acknowledgements

We would like to thank Scott Niekum, Emily Fox and Greg Hamerly for their implementations and valuable feedback.

Funding

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-ICT-2013-10) under grant agreement 610878 (3rdHand). Furthermore, this project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 640554 (SKILLS4ROBOTS).

References

Barbič J, Safonova A, Pan JY, Faloutsos C, Hodgins JK and Pollard NS (2004) Segmenting motion capture data into distinct behaviors. In: *Proceedings of the Graphics Interface 2004 Conference, May 17-19, 2004, London, Ontario, Canada*. Canada: Canadian Human-Computer Communications Society, pp. 185–194.

Bishop CM (2006) *Pattern recognition and machine learning*, volume 1. Springer New York.

Brand M and Kettnaker V (2000) Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(8): 844–851.

Chiappa S and Peters J (2010) Movement extraction by detecting dynamics switches and repetitions. In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*. USA: Curran Associates, Inc., pp. 388–396.

d'Avella A and Tresch MC (2001) Modularity in the motor system: decomposition of muscle patterns as combinations of time-varying synergies. In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. USA: MIT Press, pp. 141–148.

Endres D, Meirovitch Y, Flash T and Giese MA (2013) Segmenting sign language into motor primitives with bayesian binning. *Frontiers in Computational Neuroscience* 7: 68.

Fod A, Matarić MJ and Jenkins OC (2002) Automated derivation of primitives for movement classification. *Autonomous Robots* 12(1): 39–54.

Fox E (2009) *Bayesian Nonparametric Learning of Complex Dynamical Phenomena*. Ph.D. thesis, MIT, Cambridge, MA.

Hamerly G and Elkan C (2003) Learning the k in k-means. In: *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*. USA: MIT Press, pp. 281–288.

Ijspeert AJ, Nakanishi J, Hoffmann H, Pastor P and Schaal S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation* 25(2): 328–373.

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

Krishnan S, Garg A, Patil S, Lea C, Hager G, Abbeel P and Goldberg K (2015) Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. In: *Proceedings of International Symposium on Robotics Research, Sep 12-15, 2015, Genova, Italy*.

Kroemer O, van Hoof H, Neumann G and Peters J (2014) Learning to predict phases of manipulation tasks as hidden states. In: *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. IEEE, pp. 4009–4014.

Kulic D, Takano W and Nakamura Y (2009) Online segmentation and clustering from continuous observation of whole body motions. *IEEE Transactions on Robotics* 25(5): 1158–1166.

Lemme A, Reinhart F and Steil JJ (2014) Self-supervised bootstrapping of a movement primitive library from complex trajectories. In: *Proceedings of the International Conference on Humanoid Robots 2014, Nov 18-20, 2014, Madrid, Spain*. IEEE, pp. 726–732.

Lioutikov R, Neumann G, Maeda G and Peters J (2015) Probabilistic segmentation applied to an assembly task. In: *15th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2015, Seoul, South Korea, November 3-5, 2015*. IEEE, pp. 533–540.

Meier F, Theodorou E, Stulp F and Schaal S (2011) Movement segmentation using a primitive library. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*. IEEE, pp. 3407–3412.

Mülling K, Kober J and Peters J (2010) Learning table tennis with a mixture of motor primitives. In: *10th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2010, Nashville, TN, USA, December 6-8, 2010*. IEEE, pp. 411–416.

Nakanishi J, Morimoto J, Endo G, Cheng G, Schaal S and Kawato M (2004) Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems* 47(2): 79–91.

Nakazawa A, Nakaoka S, Ikeuchi K and Yokoi K (2002) Imitating human dance motions through motion structure analysis. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, September 30 - October 4, 2002*, volume 3. IEEE, pp. 2539–2544.

Niekum S, Chitta S, Marthi B, Osentoski S and Barto A (2013) Incremental semantically grounded learning from demonstration. In: *Robotics: Science and Systems IX, Technische Universität Berlin, Berlin, Germany, June 24 - June 28, 2013*, volume 9.

Niekum S, Osentoski S, Konidaris G and Barto AG (2012) Learning and generalization of complex tasks from unstructured demonstrations. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*. IEEE, pp. 5239–5246.

Paraschos A, Daniel C, Peters J and Neumann G (2013) Probabilistic movement primitives. In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. USA: Curran Associates, Inc., pp. 2616–2624.

Rohrer B and Hogan N (2006) Avoiding spurious submovement decompositions ii: a scattershot algorithm. *Biological Cybernetics* 94(5): 409–414.

Takano W and Nakamura Y (2006) Humanoid robot's autonomous acquisition of proto-symbols through motion segmentation. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots, Genova, Italy, December 4-6, 2006*. IEEE, pp. 425–431.

Wächter M and Asfour T (2015) Hierarchical segmentation of manipulation actions based on object relations and motion characteristics. In: *International Conference on Advanced Robotics, ICAR 2015, Istanbul, Turkey, July 27-31, 2015*. IEEE, pp. 549–556.

Williams B, Toussaint M and Storkey AJ (2008) Modelling motion primitives and their timing in biologically executed movements. In: *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. USA: Curran Associates, Inc., pp. 1609–1616.

Wu CFJ (1983) On the convergence properties of the EM algorithm. *The Annals of Statistics* 11: 95–103.

Yamane K, Yamaguchi Y and Nakamura Y (2011) Human motion database with a binary tree and node transition graphs. *Autonomous Robots* 30(1): 87–98.