Movement Segmentation using a Primitive Library

Franziska Meier fmeier@usc.edu
Evangelos Theodorou etheodor@usc.edu
Freek Stulp stulp@clmc.usc.edu
Stefan Schaal sschaal@usc.edu
Computer Science, Neuroscience, and Biomedical Engineering, University of Southern California, Los Angeles, USA

***Abstract*— Segmenting complex movements into a sequence of primitives remains a difficult problem with many applications in the robotics and vision communities. In this work, we show how the movement segmentation problem can be reduced to a sequential movement recognition problem. To this end, we reformulate the original Dynamic Movement Primitive (DMP) formulation as a linear dynamical system with control inputs. Based on this new formulation, we develop an Expectation-Maximization algorithm to estimate the duration and goal position of a partially observed trajectory. With the help of this algorithm and the assumption that a library of movement primitives is present, we present a movement segmentation framework. We illustrate the usefulness of the new DMP formulation on the two applications of online movement recognition and movement segmentation.**

I. Introduction

Automatically observing and interpreting human movement has become an important topic in computer vision, robotics, human robot interaction, surveillance, and many other fields. One can at least distinguish two major purposes of observing movements with a computer system: i) action understanding with the help of classification algorithms, and ii) parsing observed movement into units of action that can be reproduced on a robotic system. In point (i), the representations for activity understanding only subserve the subsequent classification algorithms, and a wide variety of possibilities exist [1]. For point (ii), it is important that the representations for extracting units of actions also relate to movement generation, which is more constraining.

Our work addresses the latter issue. The larger goal of our research is to bootstrap autonomous learning robots through imitation learning, i.e., to teach robots by demonstrating motor skills, which the robot can subsequently refine by autonomous trial-and-error learning. As many others [2], we start our work with the assumption that complex movement skills are composed from smaller units of action, which we call movement primitives. A significant amount of previous work in the literature tries to segment human movement into such movement primitives, e.g., with statistical methods [3], [4]. A typical problem of such approaches is that the extracted segments are behaviorally often of little meaning, such that movement generation with such primitives is not straightforward in the context of a desired task. In contrast, we assume that useful segments have already been taught in isolation, such that a pre-existing library of movement primitives exists. Thus, in this research, our goal is to recognize these primitives in a complex movement sequence, and also indicate that a new primitive has to be learned if no element of the library provides a sufficient fit. A typical problem in this scenario is that temporally adjacent primitives have been smoothed together, such that clear demarcation points for segmentation are not easy to find.

In a fully developed stage, this research will be able to bootstrap a library of movement primitives from a continual learning process.

After giving a brief literature review in the next section, we formalize the concept of movement segmentation using a library in Section III. In Section IV we introduce the mathematical and algorithmic details of our approach. Section V and VI then present possible applications and results, respectively.

II. Related Work

The approaches to movement segmentation can be grouped into two main categories. The first category comprises of methods that perform segmentation without the use of pre-trained motion primitive models. In [5] a combination of k-means clustering and dynamic programming is presented for temporal segmentation of human motion. Most approaches can also be viewed as methods to identify movement primitives in observed data. For instance, [6] represent motion primitives as dynamical systems incorporated in a gaussian process. A sequence of dynamical models is segmented by inferring switches between different Gaussian process models. Another approach to identifying motion primitives is proposed in [7], where movement is assumed to be generated by a factorial hidden Markov model (HMM), in which each chain represents a primitive. In [8], dictionaries of human motion primitives are learned using an approach based on sparse coding. A generative probabilistic model for movement segmentation, where movement is assumed to be generated by a sequence of hidden trajectories, is introduced in [9].

The second major category of movement segmentation methods comprises of approaches that use pre-trained motion models and perform movement segmentation with simultaneous movement recognition. Many approaches in vision applications train HMMs to represent one motion primitive,

This research was supported in part by National Science Foundation grants ECS-0326095, IIS-0535282, IIS-1017134, CNS-0619937, IIS-0917318, CBET-0922784, EECS-0926052, CNS-0960061, the DARPA program on Advanced Robotic Manipulation, the Army Research Office, the Okawa Foundation, and the ATR Computational Neuroscience Laboratories. F.S. was supported by a Research Fellowship from the German Research Foundation (DFG).

for instance [10] and [11]. While in [10] stochastic context-free grammars are used to probabilistically parse likelihood outputs of HMMs to obtain the most likely sequence of segments, [11] first applies adaboost to combine several HMMs to learn a strong classifier for one primitive, and then employs a dynamic programming approach to perform simultaneous recognition and segmentation. [12] represents human motion through autoregressive models and utilizes a version of the condensation filtering algorithm that automatically switches between the models. More details on movement segmentation in the vision community is given in [1].

III. Movement Segmentation using a Library of Movement Primitives

When performing movement segmentation based on an observed trajectory $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T\}$, with $\mathbf{y}_t$ as the vector of state variables for all degrees-of-freedom (DOFs) at time t, it is, in general, not known, what and how many motion segments $\mathbf{Y} = [\mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \ldots, \mathbf{Y}^{(N)}]$ are present. Even if the sequence of N primitives within $\mathbf{Y}$ is known, it is still necessary to identify switching points between segments to carry out segmentation. Thus, the segmentation problem is divided into three subproblems:

- determining the number N of segments within $\mathbf{Y}$,
- estimating start and end time of each segment $\mathbf{Y}^{(n)}$,
- and recognizing which primitive from the library is executed in each segment $\mathbf{Y}^{(n)}$.

In this paper we aim at movement segmentation with simultaneous movement recognition.

We approach the segmentation problem by assuming that a library $\mathscr{L}$ of movement primitives exists, which contains all primitives expected to appear in any observed trajectory. Furthermore, we assume that the first observed data point is the starting point of the first motion segment $\mathbf{Y}^{(1)}$. Thus, we begin the segmentation by determining the time t_1 that corresponds to the end of segment $\mathbf{Y}^{(1)} = \mathbf{Y}_{1:t_t}$ such that $\mathbf{Y}_{1:t_1}$ optimally matches one of the movement primitives in library $\mathscr{L}$. The optimal matching is done in an online fashion by increasing the value of t_1 from a start time until the trajectory $\mathbf{Y}_{1:t_1}$ can be recognized with high statistical confidence as a primitive in library $\mathscr{L}$ – note that t_1 will, in general, not be the same as the true duration of a primitive due to the temporal overlapping of primitives when performed in sequence. Once t_1 has been found, $\mathbf{Y}^{(1)}$ is removed from the sequence $\mathbf{Y}$, and the whole process is repeated. As a result, the movement segmentation problem has been reduced to the problem of online sequential movement recognition.

In order to realize online movement recognition, we have to address two open questions. First, a representation is needed for the movement primitives in the library $\mathscr{L}$, and second we require a recognition algorithm that is suitable for recognition of *partially observed motions*, as we have to assume that temporally adjacent primitives have been smoothed together, a phenomenon addressed as co-articulation in the behavioral literature [13]. We have chosen to represent the movements as Dynamic Movement Primitives (DMP) [14]. DMPs have a compact representation and can be formulated such that they are parameterized by the movement duration τ and goal position g of the encoded motion, i.e., the segmentation point if no co-articulation were present. Having the duration and goal position as open parameters enables us to represent many different versions of the same movement primitives, as DMPs have especially designed temporal and spatial generalization properties. We show how the original DMP formulation is expressed as a linear dynamical system (LDS) with control inputs. Additionally, we demonstrate how the new formulation together with the library of primitives is used to solve both movement recognition and movement duration in a well-define statistial maximum-likelihood estimation problem, which is solved with an Expectation-Maximization algorithm.

IV. Kalman Filter Formulation of Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) encode a desired movement trajectory in terms of the attractor dynamics of nonlinear differential equations [14]. For a 1 DOF system, the equations are:

$$\begin{aligned} \frac{1}{\tau}\dot{z} &= \alpha_z(\beta_z(g-p)-z)+sf \\ \frac{1}{\tau}\dot{p} &= z \end{aligned} \tag{1}$$

such that $p, \dot{p}, \ddot{p} = \dot{z}$ are position, velocity, and acceleration of the movement trajectory, where

$$f(x) = \frac{\sum_{i=1}^{N} \psi_i w_i x}{\sum_{i=1}^{N} \psi_i}, \text{ with } \psi_i = \exp\left(-h_i(x-c_i)^2\right)$$

with

$$\frac{1}{\tau}\dot{x} = -\alpha_x x$$

and

$$s = \frac{g-p_0}{g_{fit}-p_{0,fit}} = \frac{g-p_0}{\Delta g}$$

In general, it is assumed that the duration τ and goal position g are known. Thus, given τ and g the DMP is parametrized through weights $\mathbf{w} = (w_1, ..., w_N)^T$ which are learned to represent the shape of any smooth movement. During this fitting process, the scaling variable s is set to one, and the value of Δg is stored as a constant for the DMP.

In our problem setting the parameter roles are reversed. We are given a library of dynamic movement primitives

$$\mathscr{L} = \{\Theta^{(1)}, \ldots, \Theta^{(m)}, \ldots, \Theta^{(M)}\},$$

where M is the number of movement primitive in the library and $\Theta^{(m)} = \{\mathbf{w}^{(m)}, \Delta g^{(m)}\}$. Assuming we know which primitive has generated a partially observed motion $\mathbf{Y}$, we can plug the corresponding $\mathbf{w}$ and Δg in (1), but are left wondering about which value to use for τ and g. We would like to determine these values such that the distance between the trajectory $\mathbf{P}$ produced by (1) and the observed trajectory $\mathbf{Y}$ is minimal.

To accomplish this goal, we reformulate the original DMP to take the form of a linear dynamical system, with τ and g as the key system parameters. As a result, the estimation of τ, g becomes a system identification problem, and the similarity measure between $\mathbf{Y}$ and $\mathbf{P}$ is given through the likelihood $p(\mathbf{Y}|\tau, g)$.

A. Reformulation

The DMP equations are discretized using Euler discretization with time step Δt, resulting in

$$\begin{aligned} x_t &= -\alpha_x\, x_{t-1}\tau\, \Delta t + x_{t-1} \\ z_t &= (\alpha_z(\beta_z(g - p_{t-1}) - z_{t-1}) + sf(x_{t-1}))\tau\, \Delta t + z_{t-1} \\ p_t &= z_{t-1}\tau\, \Delta t + p_{t-1} \end{aligned}$$

Next, we formulate the discrete time DMP as a linear dynamical system with inputs and Gaussian noise. Let $\mathbf{s}_t = \begin{pmatrix} z_t & p_t \end{pmatrix}^T$ be the (hidden) state of the primitive and y_t the observed trajectory point at time step t. We can write the stochastic DMP as

$$\begin{aligned} \mathbf{s}_t &= \mathbf{A}_1\mathbf{s}_{t-1} + \mathbf{A}_2\mathbf{s}_{t-1}\tau + \mathbf{B} * \tau * u_{t-1} + \varepsilon \\ y_t &= \mathbf{C}\,\mathbf{s}_t + v \end{aligned}$$

where $\varepsilon \sim N(0, \mathbf{Q})$ and $v \sim N(0, \mathbf{R})$. The state transition matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ are defined as

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \mathbf{A}_2 = \begin{pmatrix} -\alpha_z\, \Delta t & -\alpha_z\beta_z\, \Delta t \\ \Delta t & 0 \end{pmatrix}.$$

The control input matrix $\mathbf{B}$ and the observation matrix $\mathbf{C}$ are set to

$$\mathbf{B} = \begin{pmatrix} \Delta t \\ 0 \end{pmatrix},\ \mathbf{C} = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

and the control input u_t is computed as

$$u_t = \alpha_z\beta_z g + sf(x_t).$$

Note that the phase variable x is not part of the state s_t and only influences the input u_t.

B. Parameter Estimation

The parameters of the stochastic DMP formulation are given by

$$\theta_{all} = \{\mathbf{w}, \Delta g, \tau, g, \mathbf{A}_1, \mathbf{A}_2, \mathbf{B}, \mathbf{C}, \mathbf{Q}, \mathbf{R}\}.$$

However, the state transition, control input and observation matrices are fixed and do not depend on the problem setting. Furthermore, we assume that the weights $\mathbf{w}$ and the value of Δg are known for primitives in the library. Thus the open parameters that can influence the likelihood of observing a trajectory $\mathbf{Y}$ are

$$\theta = \{\tau, g, \mathbf{Q}, \mathbf{R}\}$$

The goal is to estimate these parameters given an observed trajectory $\mathbf{Y} = \{y_t\}_{t=1}^T$ using maximum likelihood estimation. Because the model has hidden variables $\mathbf{S} = \{\mathbf{s}_t\}_{t=1}^T$ we use an Expectation Maximization algorithm to estimate these variables.

TABLE I: Duration and Goal Estimation

- **Given**
  - Partially observed movement $\mathbf{Y}_{1:T}$
  - weights $\mathbf{w}$ and Δg of movement
  - initial estimate of goal g and duration τ
  - state transition matrix $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2\tau$ and control matrix $\mathbf{B}$
  - observation matrix $\mathbf{C}$ and noise covariance matrices $\mathbf{Q}, \mathbf{R}$
- **for** $i = 1 : iter_{max}$
  % perform forward kalman filter
  - init $\hat{\mathbf{s}}_{f,0}^+ = \mathbb{E}[\mathbf{s}_0]$ and $\mathbf{V}_{f,0} = \mathbb{E}[(\mathbf{s}_0 - \hat{\mathbf{s}}_{f,0}^+)(\mathbf{s}_0 - \hat{\mathbf{s}}_{f,0}^+)^T]$
  - **for** $t = 1 : T$
    - $\mathbf{P}_{f,t} = \mathbf{A}\,\mathbf{V}_{f,t-1}\mathbf{A} + \mathbf{Q}$
    - $\mathbf{K} = \mathbf{P}_{f,t}\mathbf{C}^T(\mathbf{C}\mathbf{P}_{f,t}\mathbf{C}^T + \mathbf{R})^{-1}$
    - $\hat{\mathbf{s}}_{f,t}^- = \mathbf{A}\,\hat{\mathbf{s}}_{f,t-1}^+ + \mathbf{B}\,u_{t-1}$
    - $\hat{\mathbf{s}}_{f,t}^+ = \hat{\mathbf{s}}_{f,t}^- + \mathbf{K}\,(y_t - \mathbf{C}\,\hat{\mathbf{s}}_{f,t}^-)$
    - $\mathbf{V}_{f,t} = (\mathbf{I} - \mathbf{K}\,\mathbf{C})\mathbf{P}_{f,t}$
    - $c_t = \mathcal{N}(y_t|\ \mathbf{C}\,\hat{\mathbf{s}}_{f,t}^-,\ \mathbf{C}\,\mathbf{P}_{f,t}\mathbf{C}^T + \mathbf{R})$

    end
    % perform smoothing and compute needed expectations
  - init $\hat{\mathbf{s}}_T = \hat{\mathbf{s}}_{f,T}^+$ and $\mathbf{V}_T = \mathbf{V}_{f,T}$
  - **for** $t = (T-1) : 1$
    - $\mathbf{J}_t = \mathbf{V}_{f,t}\mathbf{A}(\mathbf{P}_{f,t+1})^{-1}$
    - $\mathbf{V}_t = \mathbf{V}_{f,t} - \mathbf{J}_t(\mathbf{P}_{f,t+1} - \mathbf{V}_{t+1})\mathbf{J}_t^T$
    - $\hat{\mathbf{s}}_t = \hat{\mathbf{s}}_{f,t}^+ + \mathbf{J}_t(\hat{\mathbf{s}}_{t+1} - \hat{\mathbf{s}}_{f,t+1}^-)$
    - $\mathbb{E}[\mathbf{s}_t] = \hat{\mathbf{s}}_t$
    - $\mathbb{E}[\mathbf{s}_t\mathbf{s}_t^T] = \mathbf{V}_t + \hat{\mathbf{s}}_t\hat{\mathbf{s}}_t^T$
    - $\mathbb{E}[\mathbf{s}_t\mathbf{s}_{t-1}^T] = \mathbf{J}_{t-1}\mathbf{V}_t + \hat{\mathbf{s}}_t\hat{\mathbf{s}}_{t-1}$

    end
    % compute log likelihood
  - $l = \sum_{t=1}^T \ln(c_t)$
    % collect statistics and update parameters g and τ
  - $\Sigma_1 = 0$; $\Sigma_2 = 0$; $\Sigma_3 = 0$; $\Sigma_4 = 0$
  - **for** $t = 2 : T$
    - $\Sigma_1 = \Sigma_1 + Tr(\mathbf{A}_2^T\mathbb{E}[\mathbf{s}_t\mathbf{s}_{t-1}^T]) + \mathbb{E}[\mathbf{s}_t]^T\mathbf{B}u_{t-1}$
      $\Sigma_1 = \Sigma_1 - Tr(\mathbf{A}_2\mathbb{E}[\mathbf{s}_{t-1}\mathbf{s}_{t-1}^T]\mathbf{A}_1^T) - \mathbb{E}[\mathbf{s}_{t-1}]^T\mathbf{A}_1^T\mathbf{B}u_{t-1}$
    - $\Sigma_2 = \Sigma_2 + Tr(\mathbf{A}_2\mathbb{E}[\mathbf{s}_{t-1}\mathbf{s}_{t-1}^T]\mathbf{A}_2^T) + 2\mathbb{E}[\mathbf{s}_{t-1}]^T\mathbf{A}_2^T\mathbf{B}u_{t-1}$
      $\Sigma_2 = \Sigma_2 + u_{t-1}^T\mathbf{B}^T\mathbf{B}u_{t-1}$
    - $\hat{u} = f(x_{t-1})/(\Delta g) + \alpha_z\beta_z$
    - $\Sigma_3 = \Sigma_3 + \hat{u}\,\mathbf{B}^T(\mathbb{E}[\mathbf{s}_t] - \mathbf{A}_1\mathbb{E}[\mathbf{s}_{t-1}] - \mathbf{A}_2\mathbb{E}[\mathbf{s}_{t-1}]\tau)$
    - $\Sigma_4 = \Sigma_4 + \hat{u}\,\mathbf{B}^T\mathbf{B}\,\hat{u}$;

  - **end**
  - $\tau = \Sigma_1/\Sigma_2$;
  - $g = 1/\tau * (\Sigma_4)^{-1}\Sigma_3$;

end
return log likelihood l **and updated parameters** τ, g

The complete data log likelihood is given by:

$$\begin{aligned} \ln p(\mathbf{Y}, \mathbf{S}|\tau, g) &= \ln p(\mathbf{s}_1) \\ &\quad + \sum_{t=2}^T \ln p(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{A}_1, \mathbf{A}_2, \mathbf{B}, u_{t-1}, \mathbf{Q}, \tau, g) \\ &\quad + \sum_{t=1}^T \ln p(y_t|\mathbf{s}_t, \mathbf{C}, \mathbf{R}) \end{aligned}$$

Taking the expectation of the complete-data log likelihood with respect to the posterior $p(\mathbf{S}|\mathbf{Y}, \theta^{old})$ defines the function

$$Q(\theta, \theta^{old}) = \mathbb{E}_{\mathbf{S}|\theta^{old}}\left[\ln p(\mathbf{Y}, \mathbf{S}|\theta)\right],$$

which we want to maximize with respect to θ. The maximum likelihood solution of all parameters in θ are found analytically. Taking the derivative of $Q(\theta, \theta^{old})$ with respect

to τ[1] and solving for τ, yields

$$\tau = \frac{\Sigma_1}{\Sigma_2}$$

where Σ_1 and Σ_2 are given through

$$\begin{aligned}\Sigma_1 &= \sum_{t=2}^{T} \{Tr(\mathbf{A}_2^T \mathbb{E}[\mathbf{s}_t \mathbf{s}_{t-1}^T]) + \mathbb{E}[\mathbf{s}_t^T]\mathbf{B}u_{t-1} \\ &\quad -Tr(\mathbb{E}[\mathbf{s}_{t-1}\mathbf{s}_{t-1}^T]\mathbf{A}_1^T\mathbf{A}_2) - \mathbb{E}[\mathbf{s}_{t-1}]^T\mathbf{A}_1^T\mathbf{B}u_{t-1}\}\end{aligned}$$

$$\begin{aligned}\Sigma_2 &= \sum_{t=2}^{T} \{Tr(\mathbb{E}[\mathbf{s}_{t-1}\mathbf{s}_{t-1}^T]\mathbf{A}_2^T\mathbf{A}_2) \\ &\quad +2\mathbb{E}[\mathbf{s}_{t-1}^T]\mathbf{A}_2^T\mathbf{B}u_{t-1} + u_{t-1}^T\mathbf{B}^T\mathbf{B}u_{t-1}\}.\end{aligned}$$

Optimizing for g results in the update

$$g = \frac{1}{\tau}(\Sigma_4)^{-1}\Sigma_3,$$

with

$$\begin{aligned}\Sigma_3 &= \sum_{t=2}^{T} \hat{u}_{t-1}^T \mathbf{B}^T \mathbf{B}\hat{u}_{t-1} \\ \Sigma_4 &= \sum_{t=2}^{T} \hat{u}_{t-1}^T \mathbf{B}^T(\mathbf{s}_t - \mathbf{A}_1\mathbf{s}_{t-1} - \mathbf{A}_2\mathbf{s}_{t-1}\tau)\end{aligned}$$

and

$$\hat{u}_t = \frac{f(x_t)}{\Delta g} + \alpha_z\beta_z$$

To compute the maximum likelihood estimation of θ, the following expectations are needed:

$$\begin{aligned}\mathbb{E}[\mathbf{s}_{t-1}] &= \hat{\mathbf{s}}_{t-1} \\ \mathbb{E}[\mathbf{s}_{t-1}\mathbf{s}_{t-1}^T] &= cov(\mathbf{s}_{t-1}, \mathbf{s}_{t-1}) + \hat{\mathbf{s}}_{t-1}\hat{\mathbf{s}}_{t-1}^T \\ \mathbb{E}[\mathbf{s}_t\mathbf{s}_{t-1}^T] &= cov(\mathbf{s}_t, \mathbf{s}_{t-1}) + \hat{\mathbf{s}}_t\hat{\mathbf{s}}_{t-1}^T.\end{aligned}$$

The estimates of the state and covariance matrices are calculated through Kalman smoothing. In Table I the complete algorithm for updating the goal and duration parameter of a $1D$ trajectory is given.

So far we have presented a probabilistic version of Dynamic Movement Primitives. With the probabilistic formulation it is possible to estimate the duration τ and the goal position g of a partially observed trajectory. In the next section we describe how this new formulation is used to perform movement recognition and prediction of the goal position of a partially observed motion, and how to perform movement segmentation.

V. Applications

In the previous section we have seen that given the weights $\mathbf{w}$ of the DMP we can estimate the optimal τ and g for an observed trajectory $\mathbf{Y}$. However, in reality we do not know the $\mathbf{w}$ that generated the observations. Thus we build a library of movement primitives containing all primitives expected to be observed. Each primitive is parametrized through

$$\Theta^{(m)} = \{\mathbf{w}^{(m)}, \Delta g^{(m)}, \theta^{(m)} = \{\tau^{(m)}, g^{(m)}, \mathbf{Q}^{(m)}, \mathbf{R}^{(m)}\}\}.$$

Given the library, we can optimize the parameters $\theta^{(m)}$ for each primitive by plugging $\mathbf{w}^{(m)}$ and $\Delta g^{(m)}$ into (1) and executing the EM algorithm as described in Table I. This is illustrated in Fig. 1, where the library consists of the three letters 'a', 'o' and 'u'. The blue line represents the partially observed trajectory that is used to fit the duration and goal parameters for all three primitives. For letters 'a' and 'o' we see that the resulting goal position makes sense, and that both letters are possible. Freek: How can we see that? I don't understand where this is visualized.

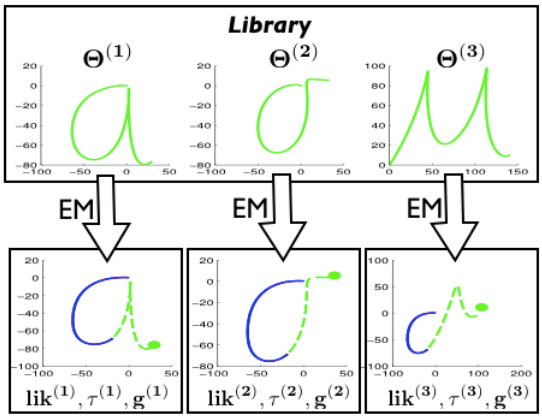


Fig. 1: Illustration of EM fitting of a partially observed trajectory. (Top) 3 primitives in the library. (Bottom) Observed trajectory in blue, and the estimated goal, duration and likelihood estimates for each primitive obtained through the EM algorithm. The green dashed line shows the predicted rest trajectory, given the estimates of $g^{(m)}, \tau^{(m)}$.

A. Creating the Library

During the creation of a primitive library we want to get an average estimate of all parameters in $\Theta^{(m)}$. Given N examples of each primitive m, we first need to learn the corresponding averaged weights $\mathbf{w}^{(m)}$. Since for each training instance, we know the duration and goal position, this is easily done by weighted linear regression. Note that in this phase, the value of s in Equation (1) is set to one. However, for recognition purposes we will need an average estimate of $\Delta g^{(m)}$, thus the mean of the goal offset across all training instances for primitive m is computed. The same is done for duration τ and goal position g, as these averaged values serve as plausible initial values for the EM algorithm.

Given the weights $\mathbf{w}^{(m)}$ the duration $\tau^{(m)}$ and the goal distance $\Delta g^{(m)}$ for each primitive, we can learn the noise matrices $\mathbf{Q}^{(m)}$ and $\mathbf{R}^{(m)}$ as well. This is done by using the expectation maximization algorithm as described in Table I, with the difference that τ and g values are known, and thus fixed, and values for $\mathbf{Q}$ and $\mathbf{R}$ need to be updated. Update equations for the covariance matrices are found following the

[1]Note that the phase variable x_t is not part of state s_t although x_t depends on τ. However, one can show that the inclusion of x_t in s_t with the assumption of no noise on x_t, leads to the same update equation for τ.

steps in [15] in the section on how to learn linear dynamical systems.

B. Movement Recognition and Prediction

Given the primitive library, it is now possible to perform online movement recognition. While observing a motion, we hypothesize which primitive m in $\mathscr{L}$ is responsible for the observed trajectory, and where and when it is going to finish. This is achieved by executing the EM algorithm in Table II on $\mathbf{Y}_{1:t}$ for all primitives m at each time step t until all data points y_t have been observed, or one of the primitives' likelihood crosses a threshold.

First, the state transition matrices A_1,A_2, control input matrix B and the observation matrix C are initialized. These matrices are the same across all primitives in $\mathscr{L}$ and do not change their value during the recognition process. Next, the weights $\mathbf{w}^{(m)}$ that determine the shape of each primitive are set to the average estimates from the library. Then for each primitive m, the noise covariances, $Q^{(m)}$ and $R^{(m)}$, and the duration $\tau^{(m)}$ and goal $g^{(m)}$ are initialized with the values from the library. The noise covariances will not be updated once initialized. Thus the only parameters that change as more data points y_t become available are the duration and goal parameter for each primitive.

At each time step t we optimize each primitive for $\tau^{(m)}$ and $g^{(m)}$ as to best fit the partially observed trajectory $Y_{1:t}$. This is done by running the EM algorithm Freek: missing word? updating the current estimates of $\tau^{(m)}$ and $g^{(m)}$. Some attention is required when updating the parameter τ. When trying to optimize the duration of a primitive that is incompatible with $\mathbf{Y}_{1:t}$, it can happen that τ approaches extremely small or large values that are unreasonable. Thus we constrain τ to be in the interval $[\tau_{min}, \tau_{max}]$. If at time step t the update step produces a τ outside that interval, the EM step is aborted for this time step, and the recognition procedure will retry for that primitive at the next time step.

Furthermore, instead of running the EM algorithm until convergence for each time step, we choose to perform 10 iterations at each time step. This saves computational effort and avoids overfitting of τ and g in the very beginning when only few data points have been observed. Through the expectation step, we obtain the likelihood of all primitives in the library to have generated $\mathbf{Y}_{1:t}$. These likelihood values are used to classify the partially observed trajectory as the letter with the currently highest likelihood.

C. Movement Segmentation

We now show how the online movement recognition process is used to perform segmentation on a trajectory that is a sequence of primitives. Let us assume that the complete trajectory $\mathbf{Y}_{1:T}$ consists of two concatenated primitives, for instance a word with two letters. Furthermore, let t_1 be the time step of the last data point of the first letter. It is assumed that the first observed data point Y_1 represents the start position of the first primitive. Then, we try to find the end of the first segment by performing online movement recognition, starting with two data points and keep adding

TABLE II: Movement Segmentation

- **Given**
  - first data point Y_1 of new segment
  - Library of motion primitives $\Theta = \{\Theta^{(m)}\}_{m=1}^{M}$
- **for each new data point** y_t
  - add new data point y_t to trajectory $\mathbf{Y}_{t-1}$ to obtain $\mathbf{Y}_{1:t}$
  - **for each primitive in library**
    - execute 10 iterations of EM on $\mathbf{Y}_{1:t}$ to update the parameters $\tau^{(m)}, g^{(m)}$ and compute the new log likelihood $l_t^{(m)} = \ln p(\mathbf{Y}_{1:t}|\theta^{(m)})$
  - find the currently most likely primitive: $m_{ml,t} = \arg\max_m l_t^{(m)}$
  - if we have surpassed the expected duration of the best primitive, i.e. $t > \tau^{(m_{ml,t})}$, set $t^{(*)} = t$, then **break**
- find segmentation point $t_s = \arg\max_{t \in [t^*-k,\, t^*]} l_t^{(m_{ml})}$

data points. At each time step we monitor the likelihood of all primitives. Let $m_{ml,t}$ be the primitive with the highest likelihood at time step t, and $\tau^{(m_{ml,t})}$ its predicted duration.

Once we reach time step t^*, such that $t^* > \tau^{(m_{ml,t})}$, we assume that we have found or already passed the end point of the first segment. Thus, we stop the procedure and choose the segmentation point t_1 as the point with the highest likelihood in the time interval $[t^*-k,\, t^*]$, $t_1 = \arg\max_{t \in [t^*-k,\, t^*]} (l_t^{(m_{ml})})$ where k is a small value. At the same time the discovered segment is labeled as primitive $m_{ml,t}$. Our procedure to finding the end point of a segment is given in Table II. The next segment is found by cutting off the previous segment and restarting the online movement recognition process.

VI. Experiments

For evaluation purposes we recorded a dataset of $2D$ trajectories of letters and words with a digitizing tablet. All letters that are easily written with one stroke have been recorded, which is a total of 22, and for each letter we created 20 samples. The primitive library was generated using 15 samples of each letter. The 5 remaining instances were used as test instances to evaluate the movement recognition performance.

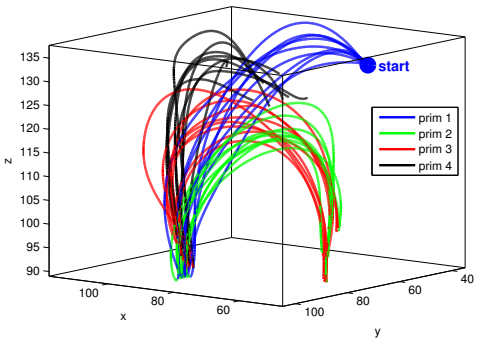


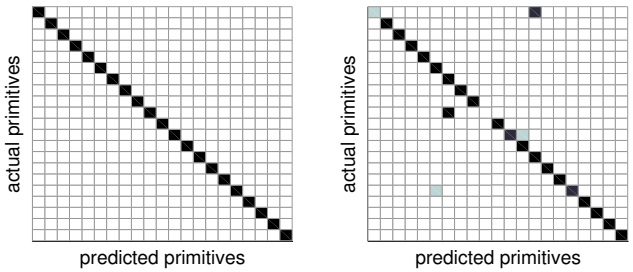
Fig. 2: Training data obtained from robot.

Additionally, we recorded $3D$ trajectories through kinesthetic teaching using the endeffector of a robotic manip-

ulator (Barrett WAM). The recorded movement trajectory was a demonstration of how to reach for a bottle, move the endeffector to pour the content of water into a cup, to place the bottle to its original start position, and move the endeffector back to its starting pose. Each motion primitive was performed 10 times, of which 9 were used for training the library. The training data for this experiment is shown in Fig. 2. For segmentation evaluation purposes, the complete sequence of all four segments was performed and recorded as a continuous motion.

In the following, first the movement recognition is evaluated, then we present some results showing how the online movement recognition is used for movement segmentation.

A. Movement Recognition and τ, g prediction



(a) Full Trajectory (b) Half Trajectory

Fig. 3: Confusion matrices of recognition results. Rows represent the actual primitives 'a' to 'z', whereas columns represent predicted primitives. The darker a square at row_i, col_j the more often primitive i was classified as primitive j.

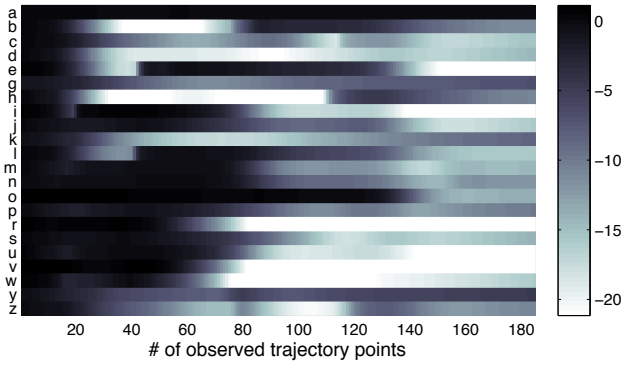


Fig. 4: Log Likelihood plots: (Top) online recognition of letter 'a'. The plot illustrates how the likelihood values change for each primitive in the library as more data points become available.

First we evaluate the recognition rate on the test instances of the letter data set. In total there are 110 test instances. When given only half of the trajectory of the test instances 11 of the 110 were miss classified, resulting in a recognition rate of 90%. Given the full trajectory the recognition rate was perfect. The confusion plots of the recognition results are shown in Fig. 3. The typical reason behind a miss classification is that the two confused letters, for instance letters 'a' and 'o', share a similar first trajectory half.

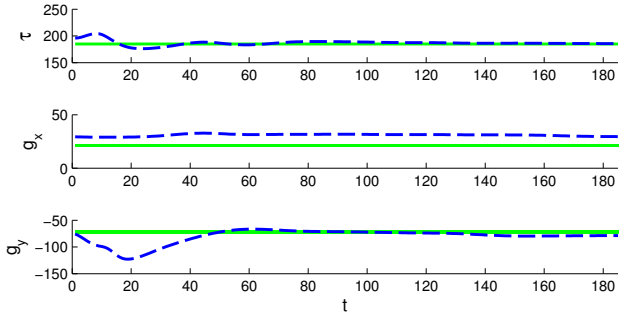


Fig. 5: Parameter evolution plot. For each parameter, the ground truth is shown as the green line.

A typical result of the online recognition is shown in Fig. 4. The task was to recognize a test instance of letter 'a'. The log likelihood at each time step for each movement primitive is shown, where red encodes high likelihood and blue low likelihood. In the beginning all primitives have a high likelihood of having generated the observed partial trajectory. However, as more data points become available, likelihoods of most letters that are not 'a' are dropping. Towards the end, the primitive causing the observations becomes obvious. Similar results are observed when carrying out the same experiment on the robot data, Fig. 4 (bottom). Here an instance of the second primitive is sequentially classified. While the first and fourth primitive have visibly lower likelihoods in the end, the third primitive is almost as likely as the second. This makes sense when looking at the training data. The second (green) and third (red) primitive have almost the same shape.

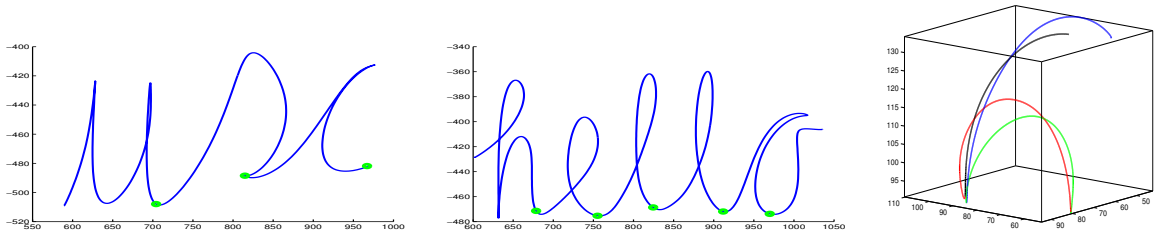
For the same instance of 'a', we also show how the parameters τ and g evolve over time, see Fig. 5. After using half of the trajectory points the estimate of τ is already close to the ground truth.

B. Movement Segmentation

Some typical segmentation results using the algorithm in Table II are shown in Fig. 6. In examples 6a and 6c all segments have been correctly found and classified. Instance 6b illustrates a case where the current segmentation procedure fails. While the first four segmentation points have been found correctly, the last segmentation point was selected too greedily. The fifth segment was cut off too early and recognized as letter 'c' instead of 'o'. The left over trajectory can now not be classified anymore.

VII. Conclusion

In this paper, we present a movement segmentation algorithm that assumes the presence of a library and reduces the segmentation problem to online movement recognition. We show how the original DMP formulation is reformulated so that given a partially observed trajectory its duration τ and goal position g are estimated using an expectation maximization algorithm. Furthermore, we demonstrate how the recognition of partially observed trajectories and the prediction of τ and g is used in a segmentation framework.



(a) (b) (c)

Fig. 6: Segmentation results: For (a) and (b) the green dots indicate the selected segmentation points. For (c) the colors represent the found primitives.

Future work will augment this framework so that it is possible to recover from wrongly chosen segmentation points and deal with cases in which no primitive in $\mathscr{L}$ can be found to fit parts of a trajectory.

REFERENCES

[1] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, 2011.

[2] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999, clmc.

[3] T. Inamura, T. Iwaki, H. Tanie, and Y. Nakamura, "Embodied symbol emergence based on mimesis theory," *International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 363–377, 2004.

[4] O. Jenkins and M. Mataric, "Performance-derived behavior vocabularies: Data-driven acqusition of skills from motion," *International Journal of Humanoid Robotics*, vol. 1, no. 2, pp. 237–288, Jun 2004.

[5] F. Zhou, F. Torre, and J. Hodgins, "Aligned cluster analysis for temporal segmentation of human motion," in *Automatic Face & Gesture Recognition*, 2008, pp. 1–7.

[6] M. Alvarez, J. Peters, B. Schölkopf, and N. Lawrence, "Switched latent force models for movement segmentation," *NIPS*, 2010.

[7] B. Williams, M. Toussaint, and A. Storkey, "Extracting motion primitives from natural handwriting data," *ICANN*, pp. 634–643, 2006.

[8] T. Kim, T. Chicago, G. Shakhnarovich, and R. Urtasun, "Sparse coding for learning interpretable spatio-temporal primitives," *NIPS*, 2010.

[9] S. Chiappa and J. Peters, "Movement extraction by detecting dynamics switches and repetitions," *NIPS*, 2010.

[10] A. Bobick and Y. Ivanov, "Action recognition using probabilistic parsing," in *CVPR*, 1998, pp. 196 –202.

[11] F. Lv and R. Nevatia, "Recognition and segmentation of 3-d human action using hmm and multi-class adaboost," in *ECCV*, 2006, pp. 359–372.

[12] J. Rittscher and A. Blake, "Classification of human body motion," in *ICCV*, vol. 1, 1999, pp. 634 –639 vol.1.

[13] R. Sosnik, T. Flash, B. Hauptmann, and A. Karni, "The acquisition and implementation of the smoothness maximization motion strategy is dependent on spatial accuracy demands," *EXPERIMENTAL BRAIN RESEARCH*, vol. 176, no. 2, pp. 311–331, 2007.

[14] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, pp. 1547–1554, 2003.

[15] C. Bishop, *Pattern recognition and machine learning*. Springer New York, 2006, vol. 4.