Mixture of Attractors: A novel Movement Primitive Representation for Learning Motor Skills from Demonstrations

Simon Manschitz[1,2], Michael Gienger[2], Jens Kober[3], and Jan Peters[1,4]

***Abstract*—In this paper, we introduce Mixture of Attractors, a novel movement primitive representation which allows for learning complex object-relative movements. The movement primitive representation inherently supports multiple coordinate frames, enabling the system to generalize a skill to unseen object positions and orientations. In contrast to most other approaches, a skill is learned by solving a convex optimization problem Therefore, the quality of the skill does not depend on a good initial estimate of parameters. The resulting movements are automatically smooth and can be of arbitrary shape. The approach is evaluated and compared to other movement primitive representations on data from the Omniglot handwriting data set and on real demonstrations of a handwriting task. The evaluations show that the presented approach outperforms other state-of-the-art concepts in terms of generalization capabilities and accuracy.**

***Index Terms*—Learning from Demonstration, Learning and Adaptive Systems, Motion Control**

I. Introduction

DESPITE impressive results in the recent years, some of the main challenges in the domain of Learning from Demonstration (LFD) remain unsolved. For instance, learning complex tasks usually requires more demonstrations than a user would be willing to provide. One reason for the large number of required demonstrations is that learning a skill requires finding a mapping of a potentially large input space (e.g., camera input), to a potentially large output space (e.g., desired joint positions). Depending on the task and robot, such a mapping can become highly non-linear and almost arbitrarily complex. If only a few demonstrations of a task are performed, then only a small fraction of the input space is covered with data. Learning an input to output mapping from this sparse data often results in skills that are able to perform a task if the environmental conditions are about the same as in the demonstrations. Yet, it is often not clear how the system will generalize to unseen situations.

Manuscript received: September, 8th, 2017; Revised December, 1st, 2017; Accepted December, 29th, 2017.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments.

[1]S. Manschitz and J. Peters are with the Institute for Intelligent Autonomous Systems, Technische Universität Darmstadt, 64289 Darmstadt, Germany, manschitz@ias.tu-darmstadt.de, mail@jan-peters.net

[2]S. Manschitz and M. Gienger are with the Honda Research Institute Europe, 63073 Offenbach, Germany, michael.gienger@honda-ri.de

[3]J. Kober is with the Cognitive Robotics Department, Delft University of Technology, 2628 CD, Delft, The Netherlands, j.kober@tudelft.nl

[4]J. Peters is with the Max Planck Institute for Intelligent Systems, 72076 Tübingen, Germany

Digital Object Identifier (DOI): see top of this page.

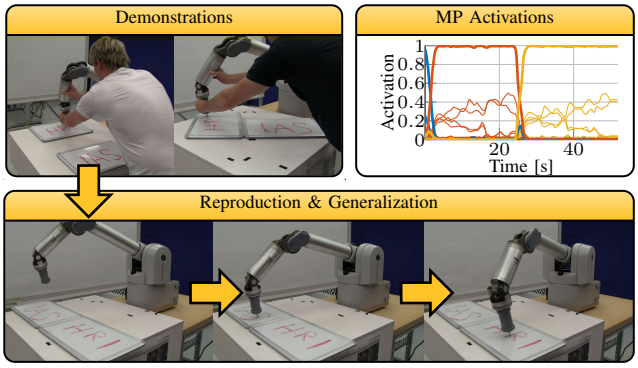


Fig. 1: The system learns a handwriting skill from kinesthetic demonstrations by learning a set of attractors and their continuous activations over time. The attractors can be defined in different coordinate frames, enabling the system to generalize the learned skill to unseen whiteboard positions. The plot shows the learned attractor activations for the handwriting task (thin lines). Blue lines correspond to attractors represented in the world frame, red in the IAS frame and yellow the HRI frame. The thick lines show the sum of the attractors defined in the individual coordinate frames.

Our method aims at learning skills for tasks that require handling multiple objects. We assume a task is demonstrated a few times with varying object positions and orientations, as depicted in Figure 1. From these demonstrations, the system is supposed to learn a skill which generalizes to unseen positions and orientations of the involved objects.

The main contribution of the paper is a novel movement primitive (MP) representation, which we call Mixture of Attractors (MOA). MOA represents movements in multiple coordinate frames. When learning a skill, a weighting of the coordinate frames is learned that explains the demonstrations well. For instance, in a task phase where the robot is supposed to manipulate an object, the weights of the coordinate frame attached to this object will be large, allowing the robot to manipulate the object at arbitrary positions. Moreover, a continuous representation of the weights is learned, allowing the robot to smoothly blend between successive movements. The proposed learning algorithm for MOA is formalized as convex optimization problem. Therefore, it does not rely on a good initialization of the parameters.

A. Related Work

MPs are a tool for increasing the data efficiency of skill learning algorithms. They are basic reusable building blocks that are able to generate complex movements. By using MPs,

the movement generation can be parametrized which leads to a complexity reduction of the learning problem. Many different MP representations have been proposed. Among the most prominent ones are Dynamic Movement Primitives (DMPs, [1]), Gaussian Mixture Models (GMMs, [2]), Stable Estimator of Dynamical Systems (SEDS, [3]), Interaction Primitives [4] or Probabilistic Movement Primitives (PROMPs, [5]). In the following section, we discuss some of the approaches utilizing MPs for learning skills from demonstrations. With our approach, we aim for contributing to three important aspects in the learning from demonstration domain: Learning from a few demonstrations, Coordinate frame selection and Co-articulated movements.

a) Learning from few Demonstrations: One of the main challenges when learning skills from demonstrations is to extract as much information as possible from the demonstrations. A human teacher usually is not willing to provide tens or hundreds of demonstrations. Ideally, only a handful demonstrations should be sufficient to learn a skill. Lee et al. [6] use Principal Component Analysis for reducing the dimensionality of the input data before segmenting the demonstrations into a set of MPs. Extracting a set of MPs from the demonstrations is a frequently used technique for decomposing the overall learning problem into smaller parts which might be easier to learn (e.g., [7], [8], [9], [10]). Kappler et al. [11] learn to activate MPs based on sensory input which may be potentially of a high dimension. Yet, they assume the MPs are learned at a previous stage. Kim et al. [12] learn a skill from few demonstrations which may even be inaccurate. The movements generated by many of the aforementioned MP representations are usually modulated temporally (e.g., DMPs or PROMPs). In that case, the input dimension is inherently reduced to one, which simplifies the learning process. In this paper, we also modulate the movements temporally. The reason why our approach is able to learn complex tasks from a few demonstrations is the way coordinate frames are integrated into the learning process.

b) Coordinate Frame Selection: When controlling a robot in task-space, the generalization capabilities of a system can be greatly improved if the MPs operate in task-spaces which are defined relative to objects. If each object is associated with a coordinate frame and a task-space which controls the robot in this coordinate frame, the system is inherently able to generalize the movements to setups which have not been seen in the demonstrations. As the teacher usually does not want to specify which MP performs a movement relative to which object, this parameter has also to be learned from the demonstrations. Niekum et al. [13] present an approach that extracts a set of MPs and selects an appropriate task-space for each MP. The task-spaces are selected using a heuristic. Heuristics are also used by [14], [15], [16] to select a task-space for each MP. The Task-Parametrized Gaussian Mixture Model (TP-GMM, [17]) generalizes the GMM to support multiple coordinate frames and is for instance used in [18]. It does not select the coordinate frame explicitly, but instead learns a probability distribution over the coordinate frame weights. A newer version of this approach is the Task-Parametrized Hidden Semi-Markov Model, where also the temporal correlation of the individual mixture models is learned [19], [20]. Our approach also does not select the coordinate frames explicitly. Instead, it learns to activate a set of attractors represented in different coordinate frames over time in a way which is most consistent with the demonstrations. Estimating the activations is formulated as convex optimization problem which does not rely on a good initialization of the parameters, which for instance is the case for the TP-GMM.

c) Co-Articulated Motions: Humans often tend to co-articulate between successive movements [21], which renders the problem of detecting start and end of individual movements more difficult. Two MP representations which explicitly support co-articulated movements are GMMs and SEDS. Calinon et al. [2], encode a movement as joint probability distribution over the positions and velocities by using GMMs. A movement is generated by conditioning the velocity on the current position. SEDS also makes use of GMMs, but additionally guarantees convergence to a desired target at the cost of a larger computational effort. GMMs allow for modeling an entire demonstration with a single probability distribution. As the density function of a Gaussian is smooth, the movements generated by a GMM are smooth as well and therefore, the model can be utilized for modeling co-articulated movements. PROMPs can also blend between two successive MPs. Yet, it is not clear how to learn a blending factor from demonstrations. Our approach learns to activate a set of attractors so that the generated movement follows the demonstrated behavior. If the human teacher transitioned smoothly between two successive movements, the resulting MP activations will change slowly, leading to the same behavior.

In summary, our framework learns to continuously activate a set of attractors over time by solving a convex optimization problem. The attractors can be represented in different coordinate frames. Additionally, co-articulated movements are supported explicitly. Altogether, the framework is able to learn complex skills from a few demonstrations and is able to generalize a skill to novel setups. The remainder of the paper is organized as follows. In Section II, the MOA MP representation is introduced formally. Next, Section III shows how the representation can be used for robot control. The approach is then evaluated in Section IV before concluding and giving a short outlook on future work in Section V.

II. Mixture of Attractors

The basic idea behind MOA is to represent a movement as composition of very simple Dynamical Systems (DS). We refer to an attractor as a spring-mass-damper system of the form

$$\ddot{\boldsymbol{x}}(t) = \alpha(\beta(\boldsymbol{g} - \boldsymbol{x}(t)) - \dot{\boldsymbol{x}}(t)), \tag{1}$$

where α and β are controller parameters. The parameters can be set to guarantee the stability of the DS. In that case, the DS converges to its attractor goal $\boldsymbol{g}$ for $t \to \infty$. In this paper, $\boldsymbol{x}$ corresponds to the Cartesian position of the robot's end-effector. Instead of having a single attractor, we assume that complex movements are generated by a linear combination of K attractors

$$\ddot{\boldsymbol{x}}(t) = \sum_{k=1}^{K} a_k(t)\alpha(\beta(\boldsymbol{g}_k - \boldsymbol{x}(t)) - \dot{\boldsymbol{x}}(t)). \tag{2}$$

Hence, MOA defines a movement by a set of K attractor goals $\boldsymbol{g}_k$ and their activations $a_k(t)$. Note that the activations explicitly depend on the time which allows for shaping the trajectory and generate complex movements. We assume the activations of each time-step sum up to one and therefore, the equation can also be written as

$$\ddot{\boldsymbol{x}}(t) = \alpha(\beta(\boldsymbol{G}\boldsymbol{a}(t) - \boldsymbol{x}(t)) - \dot{\boldsymbol{x}}(t)), \tag{3}$$

where the attractor goals are summarized in the matrix $\boldsymbol{G} = [\boldsymbol{g}_1, \dots, \boldsymbol{g}_K]$ and the activations of one time step are summarized in the vector $\boldsymbol{a}(t)$. In the following sections, we show how the attractor goals and activations can be learned from demonstrations and discuss some of the properties of the MP representation.

A. Trajectory Tracking

We introduce the learning method by assuming we want to follow a desired trajectory. Later, we focus on learning skills from demonstrations which go beyond pure replaying of a demonstration and extend the approach to support multiple coordinate frames. The first step is to time-discretize (3) using step-size h

$$\begin{aligned}\boldsymbol{x}_t &= \frac{2+\alpha h}{1+\alpha h+\alpha\beta h^2}\boldsymbol{x}_{t-1} - \frac{1}{1+\alpha h+\alpha\beta h^2}\boldsymbol{x}_{t-2} \\ &\quad + \frac{\alpha\beta h^2}{1+\alpha h+\alpha\beta h^2}\boldsymbol{G}\boldsymbol{a}_t \\ &= c_1\boldsymbol{x}_{t-1} + c_2\boldsymbol{x}_{t-2} + c_3\boldsymbol{G}\boldsymbol{a}_t,\end{aligned} \tag{4}$$

where c_1, c_2 and c_3 are constants to keep the equation compact. We want to learn the parameters $\boldsymbol{G}$ and $\boldsymbol{a}_t$, so that we track a demonstrated trajectory $\tau = \{\boldsymbol{y}_0, \boldsymbol{y}_1, \dots, \boldsymbol{y}_{N-1}\}$ as closely as possible. Here, N is the length of the demonstration. In this section, we assume to know the goals $\boldsymbol{G}$ and want to estimate the movement primitive activations $\boldsymbol{a}_t$. In order to track the trajectory, we minimize the mean squared error (MSE) over time between the demonstrated trajectory and the trajectory generated by MOA

$$J = \sum_{t=0}^{N-1} (\boldsymbol{x}_t - \boldsymbol{y}_t)^T (\boldsymbol{x}_t - \boldsymbol{y}_t). \tag{5}$$

The aim of this section is to minimize the cost function with respect to the activations $\boldsymbol{a}_t$. Note that our system is fully determined by the attractor goals, their activations over time and the start points $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$. Therefore, as a first step $\boldsymbol{x}_t$ is expressed in terms of these variables

$$\boldsymbol{x}_t = \lambda_t\boldsymbol{x}_1 + \mu_t\boldsymbol{x}_0 + \sum_{i=0}^{N-1}\gamma_{t,i}\boldsymbol{G}\boldsymbol{a}_i. \tag{6}$$

The scalar constants λ_t, μ_t and $\gamma_{t,i}$ can be specified recursively

$$\begin{aligned}\lambda_t &= c_1\lambda_{t-1} + c_2\lambda_{t-2}, & \lambda_0 &= 0, & \lambda_1 &= 1,\\ \mu_t &= c_1\mu_{t-1} + c_2\mu_{t-2}, & \mu_0 &= 1, & \mu_1 &= 0,\\ \gamma_{t,i} &= c_1\gamma_{t-1,i} + c_2\gamma_{t-2,i} + c_3\delta_{t,k}, & \gamma_{0,i} &= 0, & \gamma_{1,i} &= 0,\end{aligned}$$

where δ is the Kronecker delta with $\delta_{i,j} = 1$ for $i = j$ and $\delta_{i,j} = 0$ otherwise. Given a trajectory τ, we can compute the constants, set $\boldsymbol{x}_0 = \boldsymbol{y}_0$, $\boldsymbol{x}_1 = \boldsymbol{y}_1$ substitute $\hat{\boldsymbol{y}}_t = \boldsymbol{y}_t - \lambda_t\boldsymbol{x}_1 - \mu_t\boldsymbol{x}_0$ and plug this term into the cost function

$$J = \sum_{t=0}^{N-1}\left(\sum_{i=0}^{N-1}\gamma_{t,i}\boldsymbol{G}\boldsymbol{a}_i - \hat{\boldsymbol{y}}_t\right)^T\left(\sum_{i=0}^{N-1}\gamma_{t,i}\boldsymbol{G}\boldsymbol{a}_i - \hat{\boldsymbol{y}}_t\right). \tag{7}$$

If we concatenate all movement primitive activations in a single vector $\boldsymbol{a} = \left[\boldsymbol{a}_0^T, \dots, \boldsymbol{a}_{N-1}^T\right]^T$ and concatenate $\hat{\boldsymbol{G}}_t = [\gamma_{t,0}\boldsymbol{G}, \dots, \gamma_{t,N-1}\boldsymbol{G}]$, then (7) can be rearranged to

$$\begin{aligned}J &= \sum_{t=0}^{N-1}\left(\hat{\boldsymbol{G}}_t\boldsymbol{a} - \hat{\boldsymbol{y}}_t\right)^T\left(\hat{\boldsymbol{G}}_t\boldsymbol{a} - \hat{\boldsymbol{y}}_t\right),\\ &= \boldsymbol{a}^T\underbrace{\left(\sum_{t=0}^{N-1}\hat{\boldsymbol{G}}_t^T\hat{\boldsymbol{G}}_t\right)}_{0.5\boldsymbol{H}}\boldsymbol{a} - 2\underbrace{\left(\sum_{t=0}^{N-1}\hat{\boldsymbol{y}}_t^T\hat{\boldsymbol{G}}_t\right)}_{-\boldsymbol{f}^T}\boldsymbol{a} + \underbrace{\sum_{t=0}^{N-1}\hat{\boldsymbol{y}}_t^T\hat{\boldsymbol{y}}_t}_{const},\\ &= \frac{1}{2}\boldsymbol{a}^T\boldsymbol{H}\boldsymbol{a} + \boldsymbol{f}^T\boldsymbol{a} + const,\end{aligned} \tag{8}$$

which can be minimized via Quadratic Programming (QP). In addition to the cost function, constraints have to be added to ensure the individual activations are in the range $[0, 1]$ and the activations for each time step t sum up to one. Therefore, the overall minimization problem is

$$\min_{\boldsymbol{a}}\frac{1}{2}\boldsymbol{a}^T\boldsymbol{H}\boldsymbol{a} + \boldsymbol{f}^T\boldsymbol{a}, \text{ such that } \begin{cases}\|\boldsymbol{a}_t\|_1 = 1,\\ \boldsymbol{0} \le \boldsymbol{a} \le \boldsymbol{1}.\end{cases} \tag{9}$$

The optimization problem (9) can now be solved using an out of the box standard QP solver. Please note that the matrix $\boldsymbol{H}$ is a sum of the form $\sum\hat{\boldsymbol{G}}_t^T\hat{\boldsymbol{G}}_t$. Therefore, the matrix is positive semi-definite and a standard QP solver is guaranteed to find a global minimum.

B. Parametrizing the Activations

So far, the optimizer could freely choose the activations for each point in time. This freedom may lead to jumps in the activations, potentially resulting in jerky movements. In order to generate more natural, smooth movements, the activations can be parametrized. In that case, the optimizer is only allowed to change the activations at fixed points in time (e.g., every 50ms). We call these points support points. In between support points, the activations are interpolated. If we summarize all activations of the support points in a matrix $\boldsymbol{S} \in \mathbb{R}^{K\times N_S}$, where N_S is the number of support points and k the number of attractors, the activations at time-step i are interpolated according to

$$\boldsymbol{a}_i = \boldsymbol{S}\boldsymbol{w}_i. \tag{10}$$

Here, $\boldsymbol{w}_i$ is a weight vector for time-step i. It determines how the activations will be interpolated. In order to generate smooth movements, we use Radial Basis Functions (RBFs)

$$w_{i,s} = \frac{e^{-\gamma^2(t_i-t_s)^2}}{\sum_{u=0}^{N_S}e^{-\gamma^2(t_i-t_u)^2}}, \tag{11}$$

where $w_{i,s}$ is the sth value of vector $\boldsymbol{w}_i$, t_i is the time at time step i and t_s is the (temporal) center of the RBF. The bandwidth γ of the RBF is a hyperparameter that determines

how smoothly the weights change over time. If we plug (10) into (9) and rewrite the matrix $\boldsymbol{S}$ as a vector $\boldsymbol{s}$ by concatenating the columns of the matrix, the Quadratic Program can be reformulated as

$$J = \frac{1}{2}\boldsymbol{s}^T \hat{\boldsymbol{H}} \boldsymbol{s} - \hat{\boldsymbol{f}}^T \boldsymbol{s}, \tag{12}$$

where the size of the matrix $\hat{\boldsymbol{H}}$ is $KN_S \times KN_S$. Note that the matrix $\boldsymbol{H}$ from the original formulation in (9) has a size of $KN \times KN$, where N is the number of time steps. As $N_S < N$, parametrizing the activations does not only lead to smoother movements but also reduces the computational costs and memory requirements of the optimization.

C. Support for Multiple Coordinate Frames

In the previous sections, we concatenated $\hat{\boldsymbol{G}}_t = [\gamma_{t,0}\boldsymbol{G}, \ldots, \gamma_{t,N-1}\boldsymbol{G}]$. Here, the goal matrix $\boldsymbol{G}$ is constant over time. In order to support multiple coordinate frames, we can relax this assumption. An attractor can be defined in a coordinate frame which is not the world frame. In that case, it's attractor goal can be transformed into the world frame for each time-step. Hence, the goal matrix at time-step t becomes $\boldsymbol{G}_t$, where the columns of the matrix correspond to the attractor goals transformed into the global world frame. Changing the fixed goal matrix to a matrix which varies over time does not change the fact that the quadratic matrix $\hat{\boldsymbol{H}}$ is positive semi-definite. Therefore, the QP can still be solved in a globally optimal manner. We would like to point out that the QP has to be solved only once. In a real-world task, if a coordinate frame is associated with an object, the attractor goals defined in this frame automatically move together with this object. Therefore, a movement which is defined relative to these attractors is automatically adapted to changing object positions and orientations without solving the QP again.

III. Using Mixture of Attractors for Robot Control

So far, we introduced the MOA framework and showed how the activations can be learned to track a demonstrated trajectory. Next, we show how MOA can be used for learning a skill from demonstrations of a task. To do so, we discuss two aspects that were not covered in the paper so far. First, we show how the number of attractors and their goals can be learned from the demonstrations. Second, we discuss which coordinate frame to choose for each attractor. At the end of the section, we present the final skill learning algorithm.

First of all, it is notable that a skill is learned using multiple demonstrations of a task. So far, the attractor activations were learned by minimizing the MSE between a trajectory generated by MOA and a single demonstration. For learning a skill, the MSE of M demonstrations add up to

$$J = \sum_{m=0}^{M-1} \sum_{t=0}^{N-1} \left(\boldsymbol{x}_t^{(m)} - \boldsymbol{y}_t^{(m)}\right)^T \left(\boldsymbol{x}_t^{(m)} - \boldsymbol{y}_t^{(m)}\right). \tag{13}$$

As the MSE of the different demonstrations simply add up, the form of the QP does not change when minimizing J. Therefore, for M demonstrations, we can compute M independent QPs and then compute the sum of all $\hat{\boldsymbol{H}}$'s and $\hat{\boldsymbol{f}}$'s to form a single QP which has the same form as (12).

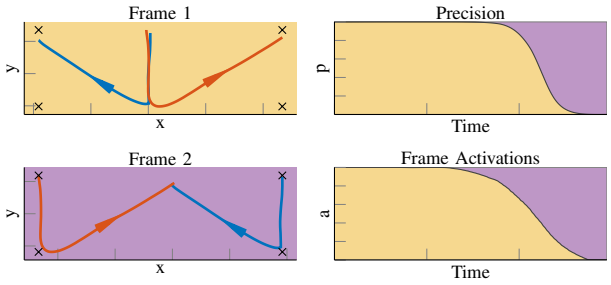


Fig. 2: Toy example to illustrate the MOA optimization. On the left, two demonstrations (blue and red) are shown in two different coordinate frames. The demonstrations start at the same position in the first frame, approach a target in this frame and then approach a target in the second frame. The attractor goals are marked with an x. On the top right, the ratio of the precision (reciprocal of the variance) in the two coordinate frames is shown over time. The frame activations resulting from the MOA optimization (bottom right) are akin to the precision.

A. Choosing the Number of Attractors and their Goals

No matter how many attractors are chosen, the points $\boldsymbol{Ga}$ generated by MOA will always lie within the convex hull of their attractor goals, as the activations sum up to one. For two linearly independent attractors, the system would generate points on a line. For three linearly independent attractors points within a triangle. A generic solution to estimate the number of attractors in a D dimensional space is to choose it so that any trajectory within this space can be generated by the system. Therefore, we propose to use 2^D attractors, where D is the dimension of the space the MPs operate on[1]. We propose to choose the attractor goals by computing the minimum and maximum values for each dimension of the demonstrations. The attractor goals then build the corner points of the bounding box of the demonstrations. If multiple coordinate frames are used, 2^D attractors are chosen for each coordinate frame independently. The attractor goals are illustrated in Figure 2 for a simple $2D$ toy example.

B. Determining the most relevant Coordinate Frame

When learning a skill, we associate the world and each object in the scene with one coordinate frame. The attractor activations should be learned in a way that yields the best generalization performance. For instance, if a task requires approaching an object, the activations of the attractors which control the robot in the coordinate frame of this object should be large during this phase of the task. A common approach for choosing coordinate frames is to compare the variance of the data over multiple demonstrations in the individual coordinate frames (e.g., [14], [15]). For a certain task phase, the movement will be represented in the frame that has the lowest variance. Our approach leads to similar results without using a heuristic to decide which coordinate frame to choose for which phase of a task. Instead, the optimization converges

[1]In general, a set $D+1$ attractors can be found which covers the demonstrations. We use a 2^D bounding box as it is more intuitive and we represent movements only in $2D$ and $3D$ space in this paper.

to a solution where the activations of the attractors represented in a coordinate frame will be large if the variance is low.

Our main assumption is that the demonstrations are aligned in time and the movement will be modulated temporally. For aligning the demonstrations, we use Dynamic Time Warping (DTW). As the demonstrations are represented in different coordinate frames, it is not straightforward to align them temporally. As cost function to measure the distance between two points $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$ from different demonstrations we measure the Euclidean distance in each frame k and use the overall minimum as cost term

$$\mathrm{DTW}(i,j) = \min_k \left\| \boldsymbol{x}_i^{(k)} - \hat{\boldsymbol{x}}_j^{(k)} \right\|_2, \tag{14}$$

where $\boldsymbol{x}_i^{(k)}$ is the ith data point of a demonstration represented in the kth coordinate frame. After aligning the demonstrations in time, the attractor goals are computed for each frame separately according to the previous section and subsequently transformed into the world frame for all non-world frames. The 2^D attractor goals of frame k at time step t will be noted as $\boldsymbol{G}_t^{(k)}$. The attractor goals of the frames are stacked together, resulting in a single goal matrix

$$\boldsymbol{G}_t = \left[\boldsymbol{G}_t^{(1)}, \ldots, \boldsymbol{G}_t^{(K)} \right] \tag{15}$$

for each time step. Now, the attractor activations can be computed according to (12). The process of estimating the activations is illustrated in Figure 2 by using a simple toy example. The optimizer can freely choose the attractor activations over time, but has to explain all demonstrations with the same sequence of activations. As it is not possible to do so in a single coordinate frame, the optimizer converges to a solution where the frame activations (sum of attractor activations represented in the same frame) vary over time.

C. Choosing the Hyperparameters

When using the suggested bounding box method for choosing the positions and numbers of the attractors goals, the only hyperparameters a user has to choose are the number of support points and the bandwidth γ of the corresponding RBF. In all experiments presented in this paper, the bandwidth of each RBF was set so that the function evaluates to a value $\alpha = 0.1$ at the center's of the neighboring RBFs. The numbers of support points were chosen by increasing the number until we were satisfied with the results. In future work, we plan to develop a method for optimizing the hyperparameters in a principled manner.

D. Final Algorithm

The steps MOA performed for learning a skill from demonstrations are summarized in Algorithm 1. First, the demonstrations are aligned in time using DTW. Next, the centers and bandwidth of the support points are computed. The attractor goals are computed for each frame separately. Then, the goals of the frames are transformed into the world frame for each time-step. Finally, the QP is generated and solved. The learned skill is composed of the resulting support point activation matrix $\boldsymbol{S}$, the attractor goals $\boldsymbol{G}$ and the parameters of the support points.

Algorithm 1 MoA Learning Algorithm

Require: Trajectories $\boldsymbol{X}^{(1)}, \ldots, \boldsymbol{X}^{(M)}$, Coordinate Frames $k = 1, \ldots, K$
1: Align trajectories using DTW (14)
2: Compute support point centers $\boldsymbol{t}_s$ and bandwidth γ according to (11)
3: Compute support point weights $\boldsymbol{W}$
4: **for each** Frame k **do**
5: Compute attractor goals $\boldsymbol{G}^{(k)}$ (Section III-A)
6: $\hat{\boldsymbol{H}} = \boldsymbol{0}, \hat{\boldsymbol{f}} = \boldsymbol{0}$
7: **for each** Trajectory m **do**
8: **for each** Frame k **do**
9: Convert goals to world frame $\boldsymbol{G}_t^{(k)}$
10: Concatenate goal matrices to single matrix for each time-step $\boldsymbol{G}_t$ (15)
11: $[\hat{\boldsymbol{H}}, \hat{\boldsymbol{f}}]$ += generateQP($\boldsymbol{X}^{(m)}, \boldsymbol{W}, \boldsymbol{G}_t$) (12)
12: Solve QP($\hat{\boldsymbol{H}}, \hat{\boldsymbol{f}}$) to find support point activations $\boldsymbol{S}$
13: **return** Goals $\boldsymbol{G}$, Support point activations $\boldsymbol{S}$, centers $\boldsymbol{t}_s$ and bandwidth γ

IV. Evaluation of the Approach

For evaluating MOA, we performed two experiments. The aim of the first experiment was to compare some of its properties to DMPs and GMMs. These MP representations have similar properties to MOA, which allows for a fair comparison. The comparison is carried out on letters from the Omniglot handwriting data set. In a second experiment, we evaluated the generalization capabilities of the system on a real robot handwriting task. The task is demonstrated kinesthetically and later reproduced on a real seven degrees of freedom (DOF) Barrett WAM robot. Additionally, we compare the generalization capabilities with two state-of-the-art approaches.

A. Handwriting Evaluation

The Omniglot data set was introduced by Lake et al. [22] as a $2D$ data set for one-shot learning. It contains over 1500 different handwritten characters from 50 different alphabets. Each character was drawn online using Amazon's Mechanical Turk by 20 different people. The images come with stroke data as sequences of $2D$ coordinates associated with time information t. We use the data set to compare MOA with DMPs and GMMs. All MP representations are trained on characters from the Latin alphabet. Before training, we removed all characters that were not drawn continuously (e.g., more than one stroke was used as the participant lifted the pen). The remaining characters were preprocessed as follows. First, we shifted each character so that the mean of the image is at $[0, 0]$. For each character, we computed the average standard deviation from the mean and subsequently scaled each image to have the same standard deviation. As a last preprocessing step, we aligned the trajectories in time using DTW. The preprocessing of the data was the same for all methods.

For a fair comparison, the movements generated by all methods were modulated temporally. For the GMM approach, a joint probability $p(t, \boldsymbol{v})$ was learned. The movements were then generated by conditioning the current velocity on the time $p(\boldsymbol{v}|t)$. For MOA and DMPs, the basis functions were activated temporally. We did our best to tune the hyperparameters of all methods. For the GMM approach, we increased the number of Gaussians to 15 until the Mean Squared Error (MSE) between the generated and demonstrated movements did not increase anymore. For MOA and DMPs we used the method suggested in Section III-C, which resulted in 20 support points. In contrast to MOA, the movements generated by DMPs are conditioned on a desired start and end point. To add this

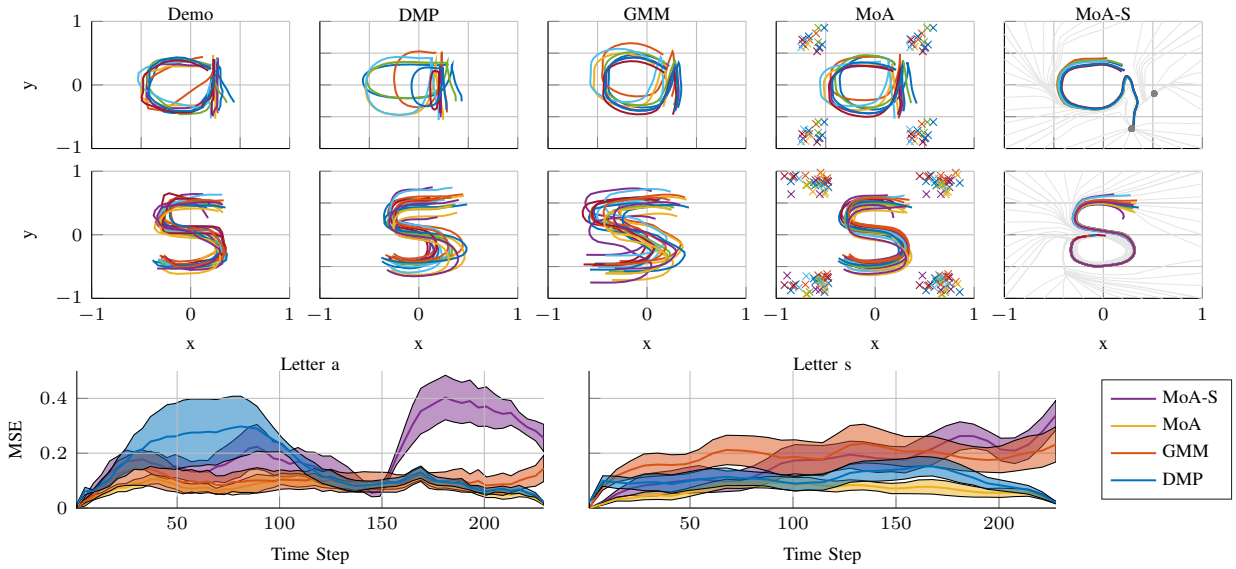


Fig. 3: Results on the Omniglot handwriting data set. The upper plots show the demonstrations and reproduced trajectories in $2D$ space for two letters of the data set. For MOA, additionally the attractor goals are shown. All methods except for MOA-S are time driven. For the spatially driven MOA-S, the gray lines show the attractor landscape. Here, the movements converge to the gray dots. The lower plots show the corresponding mean squared error over time (average and $\pm$ one standard deviation).

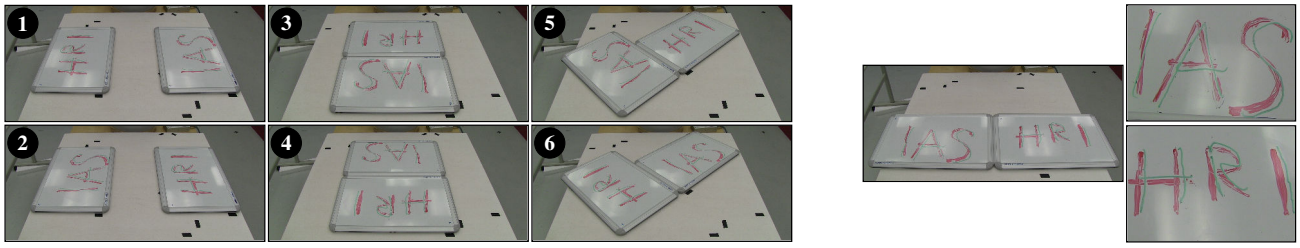
property to our MP representation, we associated a coordinate frame with the start and end points of each demonstration, respectively. As these points vary over the demonstrations, the attractor goals (transformed into the world frame) are automatically shifted together with the origins of the coordinate frames. As the MPs represent movements in $2D$ space, four attractors were used for each coordinate frame. The locations of the attractor goals were set according to the bounding box method suggested in Section III-A. In addition to the aforementioned MP representations, we also trained a second variant of MOA, where the weights of the basis functions were not conditioned on time, but on the current spatial position $\boldsymbol{x}_t$. We will refer to this variant as MOA-S. Here, we also used 20 support points. The centers of the basis functions were found by clustering the demonstrations with KMEANS and choosing the centers of the clusters as centers of the basis functions.

The results for two exemplary characters from the Latin alphabet are shown in Figure 3. The MSE between the drawn characters and the ones generated by the different time driven MP representations are in the same range with MOA slightly outperforming the other representations. DMPs perform worst for the letter 'a'. The reason is that the letter is not drawn very consistently. As DMPs depend linearly on the difference between start and end point, they do not seem to be robust against movements that have similar start and end points, but different shapes. The spatial variant MOA-S has the largest average error of all MP representations. While the generated movements reflect the general shape of the letters, they either converge to a spurious attractor ('a') or enter a cycle ('s'). Please note, however, that the focus of this paper are time driven movements. Therefore, the intention of MOA-S was to evaluate if it is in principle possible to learn spatially driven movements with our MP representation. We consider it future work to investigate in more detail MOA's applicability for spatially driven movements, for instance by analyzing important properties such as asymptotic stability (e.g., [23]). The conclusion from the Omniglot experiment is that the sequences of MP activations resulting from the MOA optimization process lead to movements which closely follow the demonstrations.

B. Robot Handwriting Evaluation

In a second experiment, we demonstrated a handwriting task on a Barrett WAM robot via kinesthetic teaching. As end-effector, a pen was attached to the robot, as shown in Figure 1. The task was to first write "IAS" on one whiteboard and subsequently write "HRI" on a second whiteboard. The intention of the experiment was to evaluate the generalization capabilities of our method. Therefore, the whiteboards were placed at different locations on the table for each demonstration (see Figure 4). The learned skill was then reproduced on a setup which was not seen in the demonstrations. For each demonstration, the $3D$ position of the tip of the pen was recorded in world coordinates and relative to each whiteboard. In order to generalize the skill to unseen whiteboard positions, the system has to learn to control the tip of the pen in the correct coordinate frame in each phase of the task. For instance, when writing "IAS", the pen has to be controlled in the coordinate frame of the corresponding whiteboard.

Overall, we performed six demonstrations of the task. The data was recorded with a frequency of 40 Hz. Before training, the demonstrations were aligned in time using DTW. The movement generated by our system was modulated temporally by the activation of 75 equally distributed support points. First, we trained our system using all six demonstrations. Subsequently, we put the whiteboards to positions that were different from the demonstrations and executed the skill on the real robot. The results are shown in Figure 4. The robot was able to generalize to the unseen setup and mastered the task for the new whiteboard positions. The system learned to control the pen in the correct coordinate frame for each phase of the task and therefore was able to generalize the skill to the



(a) Demonstration Setups (b) Reproduction Results

Fig. 4: Pictures of the six setups used for demonstrating the handwriting task (left). On the right, the setup for the reproduction is shown. For the demonstrations, we used a red pen and for the reproduction a green pen. The differences between the demonstrations and reproduction can be explained with the utilized controller and are not a result of the learning process.

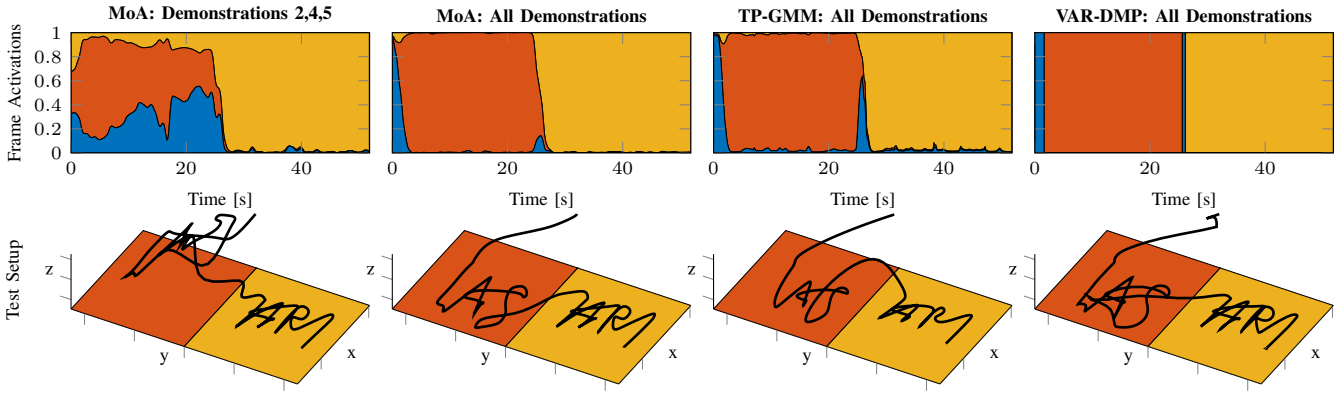


Fig. 5: Comparison of MOA with two other approaches. The upper plots show the frame activations over time. The colors correspond to the individual coordinate frames (● world frame, ● first whiteboard, ● second whiteboard). The lower plots show the predicted $3D$ paths for the test setup. The colored rectangles correspond to the two whiteboards. For the plots on the left, we trained MOA with a combination of demonstrations which did not lead to a proper discrimination of the coordinate frames. The other representations were able to generalize the skill to the test setup. Compared to VAR-DMP, MOA and TP-GMM also learn to blend smoothly between successive movements.

new situation. During execution, we control the orientation of the end-effector in null-space with compliance, as we focus on learning Cartesian positions in this paper.

Next, we trained two state-of-the-art-methods on all demonstrations and evaluated their generalization capabilities in simulation. The first method is the TP-GMM [17] introduced earlier. The method uses Gaussian distributions to model the data spatially in the different coordinate frames. For a fair comparison with our approach, we augmented the state-space with time. When reproducing a skill, we conditioned the desired position on the current time, so that the movement was also modulated temporally. We used 75 Gaussians for training, as more Gaussians did not lead to an improvement of the results anymore. The second method is an approach by Ureche et al. [15]. Here, the authors explicitly choose the coordinate frames over time based on the variance of the demonstrations. For a fixed time window and each frame, they compare the variance in this time window to the variance of the entire demonstration. The frame with the lowest value of the corresponding cost function is chosen as winner. For each resulting segment, we use DMPS to represent the movements in the corresponding frames. In the following, this approach will be referred to as VAR-DMP. Figure 5 shows the resulting coordinate frame selections of all three approaches. The resulting frame activations of all three approaches look quite similar and all approaches generalize the learned skill to the test setup. VAR-DMP successfully writes the two words, but does not learn the transition phase when changing the coordinate frame. As a consequence, the generated movement is less smooth and sometimes points in the wrong direction for a short period of time (e.g., in the beginning or after writing the S of IAS). TP-GMM performs slightly worse compared to MOA. One reason why MOA follows the shape of the movement more accurately is that it takes the attractor activation recursively into account, whereas TP-GMM treats successive data points as they were independent.

In order to evaluate if our system can learn the handwriting skill from fewer demonstrations, we additionally trained it individually on all 63 possible subsets of demonstrations (e.g., demonstration 1, 2, and 4) and evaluated the generalization capabilities of the learned skill in simulation. Figure 5 shows the frame activations and generated trajectory for one exemplary subset. While some of the learned skills were able to reproduce the task for the test setup, others were not. All skills were able to reproduce the task on the setups they were trained on. For the handwriting task, a good metric for quantifying the generalization capabilities is to measure the sparsity of the individual coordinate frame activations. Ideally, the pen should be controlled in the world frame in the beginning of the task, as the robot always started from the same initial joint configuration. When writing the two words IAS and HRI, the pen has to be controlled in the corresponding coordinate

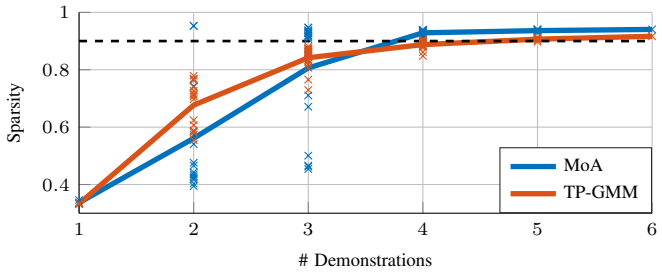


Fig. 6: Comparison of the sparsity of the frame activations for MOA and TP-GMMs. The solid lines show the mean of the sparsity, while the marks show the results for the individual combinations of the demonstrations. With an increasing number of demonstrations, the discrimination of the coordinate frames becomes better. We observed that a sparsity of at least 0.9 (dashed line) is required to generalize to the test setup.

frames of the whiteboards. Therefore, we measure the average sparsity of the activations as a metric for the generalization capabilities

$$J_{\text{sparsity}} = \frac{1}{N} \sum_{t=0}^{N-1} \sum_{k=1}^{K} \left\| \boldsymbol{a}_t^{(k)} \right\|_1^2, \tag{16}$$

where $\boldsymbol{a}_t^{(k)}$ are the activations of the attractors which are represented in coordinate frame k. In our case, $K = 3$. A fully sparse solution would result in $J_{\text{sparsity}} = 1$, while a equally distributed solution would result in $J_{\text{sparsity}} = 1/K$. Figure 6 shows the average sparsity for all training instances. The solutions become sparser if the skill is trained on more demonstrations. Depending on how dissimilar the whiteboard positions are in each demonstration, two demonstrations can be sufficient for learning a solution which is as sparse as the solution for all six demonstrations. We consider it future work to investigate why certain combinations lead to more sparse solutions than others and why the variance of the sparsity is larger compared to the TP-GMM.

V. Conclusion and Future Work

We presented the Mixture of Attractors (MOA) movement primitive representation. Due to its integration of multiple coordinate frames, MOA can be utilized for learning complex object-directed skills which generalize well to unseen object positions and orientations. In addition, the system blends smoothly between successive movements. Learning a skill is formalized as convex optimization problem. Therefore, in contrast to most other approaches, the quality of the skill does not depend on an initial estimate of parameter values. The evaluation showed that MOA outperforms two state-of-the-art approaches in terms of accuracy and/or generalization capabilities. In future work, we want to investigate MOA's capabilities of learning spatially-driven movements, as well as learning from non-optimal or partial demonstrations. In addition, we want to compare MOA to other MP representations such as TP-HSMM or PROMPs.

References

[1] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *IEEE Int. Conf. Robotics and Automation*, pp. 1398–1403, 2002.

[2] S. Calinon and A. Billard, "Active teaching in robot programming by demonstration," in *IEEE Int. Symp. on Robot and Human*, pp. 702–707, 2007.

[3] S. Khansari-Zadeh and A. Billard, "Learning stable non-linear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[4] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *IEEE Int. Conf. Robotics and Automation*, pp. 2831–2837, 2014.

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

[6] S. H. Lee, I. H. Suh, S. Calinon, and R. Johansson, "Autonomous framework for segmenting robot trajectories of manipulation task," *Autonomous Robots*, vol. 38, no. 2, pp. 107–141, 2015.

[7] S. Hangl, E. Ugur, S. Szedmak, and J. Piater, "Robotic playing for hierarchical complex skill learning," in *Int. Conf. Intelligent Robots and Systems*, pp. 2799–2804, 2016.

[8] B. Akgun and A. Thomaz, "Simultaneously learning actions and goals from demonstration," *Autonomous Robots*, vol. 40, no. 2, pp. 211–227, 2016.

[9] B. Huang, M. Li, R. L. De Souza, J. J. Bryson, and A. Billard, "A modular approach to learning manipulation strategies from human demonstration," *Autonomous Robots*, vol. 40, pp. 903–927, 2016.

[10] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Probabilistic decomposition of sequential force interaction tasks into movement primitives," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 3920–3927, 2016.

[11] D. Kappler, P. Pastor, M. Kalakrishnan, M. Wuthrich, and S. Schaal, "Data-driven online decision making for autonomous manipulation," in *Proceedings of Robotics: Science and Systems*, 2015.

[12] B. Kim, A.-m. Farahmand, J. Pineau, and D. Precup, "Learning from limited demonstrations," in *Int. Conf. Neural Information Processing Systems*, pp. 2859–2867, 2013.

[13] S. Niekum, S. Osentoski, G. D. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *International Journal of Robotics Research*, vol. 34, no. 2, pp. 131–157, 2015.

[14] J. Kober, M. Gienger, and J. Steil, "Learning movement primitives for force interaction tasks," in *IEEE Int. Conf. Robotics and Automation*, pp. 3192–3199, 2015.

[15] A. Ureche, K. Umezawa, Y. Nakamura, and A. Billard, "Task parameterization using continuous constraints extracted from human demonstrations," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1458–1471, 2015.

[16] M. Mühlig, M. Gienger, J. J. Steil, and C. Goerick, "Automatic selection of task spaces for imitation learning," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 4996–5002, 2009.

[17] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," in *IEEE Int. Conf. Humanoid Robots*, pp. 323–329, 2012.

[18] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell, "Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints," in *Int. Conf. Intelligent Robots and Systems*, pp. 1024–1030, 2015.

[19] M. J. A. Zeestraten, S. Calinon, and D. G. Caldwell, "Variable duration movement encoding with minimal intervention control," in *IEEE Int.Conf. Robotics and Automation*, pp. 497–503, May 2016.

[20] I. Havoutis and S. Calinon, "Supervisory teleoperation with online learning and optimal control," in *IEEE Int. Conf. Robotics and Automation*, pp. 1534–1540, May 2017.

[21] R. Sosnik, B. Hauptmann, A. Karni, and T. Flash, "When practice leads to co-articulation: the evolution of geometrically defined movement primitives," *Experimental Brain Research*, vol. 156, no. 4, pp. 422–438, 2004.

[22] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[23] N. Perrin and P. Schlehuber-Caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Systems & Control Letters*, vol. 96, no. Supplement C, pp. 51 – 59, 2016.