Abstract—Movement generation, and especially generalisation to unseen situations, plays an important role in robotics. Different types of movement generation methods exist such as spline based methods, dynamical system based methods, and methods based on Gaussian mixture models (GMMs). Using a large, new dataset on human manipulations, in this paper we provide a highly detailed comparison of five fundamentally different and widely used movement encoding and generation frameworks: dynamic movement primitives (DMPs), time based Gaussian mixture regression (tbGMR), stable estimator of dynamical systems (SEDS), Probabilistic Movement Primitives (ProMP) and Optimal Control Primitives (OCP). We compare these frameworks with respect to their movement encoding efficiency, reconstruction accuracy, and movement generalisation capabilities. The new dataset consists of nine object manipulation actions performed by 12 humans: pick and place, put on top/take down, put inside/take out, hide/uncover, and push/pull with a total of 7,652 movement examples.

Our analysis shows that for movement encoding and reconstruction DMPs and OCPs are the most efficient with respect to the number of parameters and reconstruction accuracy, if a sufficient number of kernels is used. In case of movement generalisation to new start- and end-point situations, DMPs, OCPs and task parameterized GMM (TP-GMM, movement generalisation framework based on tbGMR) lead to similar performance, which ProMPs only achieve when using many demonstrations for learning. All models outperform SEDS, which additionally proves to be difficult to fit. Furthermore we observe that TP-GMM and SEDS suffer from problems reaching the end-points of generalizations. These different quantitative results will help selecting the most appropriate models and designing trajectory representations in an improved task-dependent way in future robotic applications.

I. INTRODUCTION

Movement generation methods play an important role in robotics that is crucial for enabling robots to perform actions precisely, and especially to generalize movements to unknown situations. Apart from industrial applications, where fast and reliably repetitive movements are needed, static, non-adaptive trajectory representations such as interpolation between via-points using splines [1] are insufficient in most cases. In dynamic environments, trajectories need to be adapted, and it is infeasible to pre-program trajectories for each possible situation. Therefore, motion encoding frameworks need to be able to generalize movements to new situations, e.g., to new start- or end-points. Furthermore, this way, they are more easily allowing for learning by demonstration [2], [3], also called transfer learning or imitation learning. In imitation learning, instead of obtaining new movements by a theoretical description or optimization method, human demonstrations are encoded by some mathematical model and then used by a robot. Human demonstrations are usually obtained by manually moving the robot, called kinesthetic guidance, or via motion tracking of the human body.

To generalize trajectories to new situations, a motion encoding framework should therefore be able to generate human like movements for new situations from given demonstrations of known situations. As will be discussed below, various different approaches were developed for movement generalization, however the question arises, which of these frameworks performs best with respect to efficiency and accuracy of movement encoding and generalization to unseen situations.

a) Motion encoding frameworks: Dynamical Movement Primitives (DMPs, [4]) are one of the most frequently used encoding frameworks. There, a critically damped linear attractor system is forced to follow a desired trajectory by a nonlinear term that is approximated by a weighted sum of Gaussian kernels. The model is so widely used that it has been modified and extended in numerous ways. There have been alterations to the weight fitting process, for example to enable learning from multiple demonstrations [5], [6], extensions to include rotational orientation [7], [8], [9], or improving the generalization capabilities, for example by biologically motivated changes to the DMPs force field [10] or having via-points on the way [11]. There is an exhaustive review by Saveriano et al. [12] on that matter.

Via-Point Movement Primitives [13] also increase the capability of the system to adapt to new situations using via-points along the trajectory, but extend the DMP model further, by introducing components of Probabilistic Movement Primitives, which will be mentioned later.

A different approach that also uses a representation of a dynamical system is Optimal Control Primitives [14]. There, the trajectories are represented with Chebyshev polynomials instead of Gaussian kernels. A linear-quadratic regulator, derived from optimal control theory (hence the name), ensures the accurate reproduction of the trajectories and robustness against perturbations. All these frameworks are based on the common ground of dynamical systems with implicit time dependence, which means that the trajectories are obtained by integrating a specially designed dynamical system that then generates the desired trajectories.

Another class of approaches uses Gaussian Mixture Mod-
els (GMMs) to encode trajectories. Since GMMs in general are a model for regression of arbitrary distributions, they are commonly used for many other applications in statistics and data science. Moreover, they can also be used to estimate the parameters of frameworks with dynamical systems similar to the methods above [15], [16]. One key difference to the previously mentioned methods is that GMMs are able to encode multidimensional trajectories in one model, and thus, encode correlations between the dimensions, while most dynamical systems-based approaches use as many independent 1D systems as there are dimensions in the trajectory. GMM based models can also work completely without dynamical systems, for example by encoding the joint distribution of position and time and extracting the trajectory using the conditional distribution of position given time [17], [18]. Recently, a deep learning based approach was proposed to encode GMMs using a mixture density network (MDN, [19]).

Yet another different class of encoding frameworks, called transverse contracting dynamic system primitive (CDSP), uses contraction analysis [20] to estimate parameters of an autonomous dynamical system and store them either in a GMM [21] or a neural network [22].

Methods like Probabilistic Movement Primitives [23] exist, which can encode cross-correlations, too; but their working principle of modelling distributions over weight based representations of trajectories is closer to the GMM models than to DMPs. Recently, the TP-GMM approach used for generalizing GMMs was combined with ProMPs to further improve their generalization capabilities [24].

b) Own contribution: There exist several papers comparing different models either only theoretically, e.g., for DMPs [12], or with application on a common task, e.g., for DMPs, PMPs and OCPs [14] or for different kinds of GMM encodings [25]. Those papers compare relatively similar frameworks on small datasets and their respective tasks are not comparable between each other. There also exists a study on benchmarking of different models [26] which focuses on measures to judge the human-likeness of actions generated by several different frameworks, as well as their robustness to perturbations. This benchmarking was performed on a dataset of handwriting trajectories, however, there was no quantitative comparison of generalized trajectories (i.e., with respect to the changed end-point) to ground truth (human) trajectories. In the current work we aim to provide a thorough analysis and comparison of different type models on a larger dataset (9 different action classes; 7,652 movement trajectories in total) with the focus on generalization of point-to-point human hand movements to new start-/end-positions that plays an important role for generation of human-like robotic manipulation actions in unknown situations.

We compare five different frameworks, namely DMPs [4], task parameterized GMMs (TP-GMMs) with time based Gaussian Mixture Regression (tbGMR, [18]), stable estimator of dynamical systems (SEDS, [16]), Probabilistic Movement Primitives (ProMPs, [27]) and Optimal Control Primitives (OCPs, [14]). Those motion encoding frameworks have been widely used for robot motion control and demonstration learning [17], [28], [29], [2], [3], [30], [31].

We selected those five models, because they represent a wide variety of different encoding methods. These methods are either based on dynamical systems (DMPs, SEDS, OCPs) or not (GMMs, ProMPs), use different dimensions for movement encoding and a different number of encoding parameters, can encode single or multiple trajectories in one model, and generalize in different ways. We are aware that there exist multiple extensions and variations of these models as cited above, but in this work we compare the general basic models on a rather general movement encoding and generalization task. In some specific cases and applications, a more specialized sub-type of those models may lead to better performance.

We quantitatively analyze and compare the performance of the frameworks for movement encoding and reconstruction, and for generalization from demonstrations to new situations on a novel very large dataset of human manipulation actions. The main novel aspect compared to the previous work and the key focus of this study is on how well different movement generation frameworks are able to generalise to new situations in comparison to human movements.

First, we investigate how well the frameworks are able to encode and reconstruct single trajectories from the action dataset. Second, we also test the generalization capabilities of the frameworks by presenting subsets of the dataset as training demonstrations and testing the generalization capabilities on different situations obtained from the remaining demonstrations. Here we evaluate how similar trajectories generated by different models are to the human movements. Furthermore, we also compare how many free parameters the models need to achieve their respective performance.

II. DATASET

A. Recording setup

To test the performance of the selected models on human manipulation actions we recorded a dataset of simple manipulation actions using a multi camera recording setup (see Figure 1a). Movements were recorded with five calibrated and synchronized cameras at 100 Hz with a resolution of 1024 × 768 pixels. Movement trajectories were extracted as described in Section II.C below.

B. Manipulation actions

We selected nine different manipulation actions from [32] all of which included only one object, which was actively manipulated. All actions are listed with a short description in Table I (there also is a supplementary video with demonstrations).

We used five different objects in the experiments as shown in Figure 1c. A red wooden cube of dimensions 8 cm x 8 cm x 6 cm was used as the main manipulation object. It was open at the bottom and hollow inside, such that it could also be used to hide and uncover a smaller blue cube.

Most of the actions also included a second fixed object, called the “target object”. We used three bowl-like target
TABLE I: List of the selected movements and short descriptions.

| Movement               | Description                                                                 |
|------------------------|-----------------------------------------------------------------------------|
| pick and place         | Pick up the manipulation object and place it at the target position.        |
| put on top / take down | Put the manipulation object on top of the target object / take it down on   |
|                        | the table at the start position.                                            |
| put inside / take out  | Put the manipulation object inside the bowl-like target object / take it    |
|                        | out and place it on the table at the start position.                       |
| hide / uncover         | Put the manipulation object over the smaller target object / take it off    |
|                        | and place it on the table at the start position.                          |
| push / pull            | Push the manipulation object along the table without lifting it to the     |
|                        | target position / pull it back to the start position.                     |

(a) The multi-camera recording setup with one top-view and four side-view cameras (marked by red circles).

(b) Table setup view with the checkerboard mat.

(c) Objects used for the recordings.

Fig. 1: Recording setup.

objects of different height for the put on top/take down and put inside/take out actions. These objects were turned around to give raised bases of different height for the put on top and take down actions.

To control the start and end positions of the movements, we restricted them to a square grid with 10 cm grid spacing and chose the target positions to cover a range of different directions as shown by arrows in Figure 2. The movements in group 1 are frontal, in group 2 lateral, and groups 3 and 4 contain moves in mixed directions.

We obtained data from 12 adult participants, 4 females and 8 males. Each participant performed nine actions as detailed in Table 1 to the 13 different start-/end-points (see Figure 2). The movements in group 1 are frontal, in group 2 lateral, and groups 3 and 4 contain moves in mixed directions.

We obtained data from 12 adult participants, 4 females and 8 males. Each participant performed nine actions as detailed in Table 1 to the 13 different start-/end-points (see Figure 2). The movements in group 1 are frontal, in group 2 lateral, and groups 3 and 4 contain moves in mixed directions.

When using one of the bowl-like target objects, the moves to (3,0) and (4,1) were omitted due to space constraints, leaving only 11 start-/end-points. Each movement was repeated three times to accommodate for variability in the human demonstrations. Note that in some cases accidentally people performed two or four demonstrations instead of three. In total, our dataset consisted of 7,652 movements.

C. Extraction of the 3D trajectories

From the obtained movement recordings, we extracted the positions of the knuckles and wrist in the 2D images using DeepLabCut (DLC) [33], a neural network, which can be trained to track user-specified features in images and is publicly available as a python package [34], [35]. For the training of DLC’s neural network, we labeled 1000 frames out of all recordings. They were chosen by DeepLabCut using its internal clustering algorithm for extraction of the most differing frames in the respective videos. The error in feature tracking obtained from the test set is around three pixels. This corresponds to up to 5 mm distance, depending on the position of the hand in the camera frame. An error of 5 mm corresponds roughly to the size of the knuckles, which are inherently not very strictly defined features.

After 2D tracking, we reconstructed the 3D positions of those features using anipose [36], a publicly available 3D reconstruction library which is specifically built to work

3The dataset is published on Zenodo: https://doi.org/10.5281/zenodo.7351664
with DeepLabCut [37]. We used its spatiotemporally constrained triangulation method for the reconstruction of the 3D trajectories. It introduces additional terms in the triangulation loss function that penalize behavior that is possible in general, but not in the special case of hands. Tracked points may not accelerate arbitrarily fast (hands have mass) and their relative position cannot change arbitrarily, because hands have a given physical structure. To obtain the final trajectories used in this work, we averaged the trajectories of the four knuckles and the wrist into one general hand position. Some example motions are show in Figure 3.

III. MOVEMENT ENCODING FRAMEWORKS

The sections on DMPs, GMMs and SEDS were shortened to include less detail.

A. Dynamical Movement Primitives

1) Model description: The framework of Dynamical Movement Primitives (DMPs) [4], [38] is formalized by a system of second order differential equations. They consist of a critically damped linear attractor system and a nonlinear forcing term \( f \) to make the attractor follow a desired trajectory. The forcing term is learned from the presented data. The DMP model is a 1D system, so three separate models need to be trained for 3D trajectories, one for each dimension.

Here we use the DMP version introduced by [39]. It has been shown to have equivalent overall performance as compared to the original Ijspeert formulation [40], but better convergence at the end- and joining-point of the movement trajectory. Since in this work we are not concerned with the analysis of joining of movement trajectories, we removed the parameter \( v \) from the model (see Appendix A). The forcing term \( f \), that encodes the trajectory, is represented as a linear combination of \( K \) Gaussian kernels with weights \( w_i \). For this work, we chose the kernels to be equally spaced in time. Furthermore, we do not learn the individual standard deviations \( \sigma_i \), but set them to scale with the number of kernels \( K \) as

\[
\sigma = \frac{\kappa}{K},
\]

with some optimal \( \kappa \) determined from data (see Appendix C.1). This means, the more kernels there are, the narrower they get, hence the sum of the standard deviations \( \sigma \) remains fixed at \( \kappa \).

The only individually learned parameters in this model are the weights \( w_i \). To learn the weights, locally weighted regression [4] is used widely, but any other supervised learning rule can be used, too. Here we used the simple \( \delta \)-learning rule as explained in [39].

The number of free parameters in the model is just the number of weights, which is equal to the number of kernels \( K \). With one weight vector for each dimension \( D \) we get

\[
n_{\text{DMP}} = DK
\]

for height would just scale accordingly (approximately by the factor of three, since the highest point is about three times higher). This scaled trajectory would therefore rise around 9 cm higher above the goal point before going down to reach the end-point, which would deviate by far (instead of the observed 3 cm) from the human demonstration trajectory.

To achieve a generalization closer to the human trajectory, we instead take a weighted average of the DMP weight vectors \( \mathbf{w}_n \) of nearby demonstrations. This has been proposed before, but with weights only dependent on the distance to the demonstration targets [41]. We compare the performance of this method in Appendix B. For our comparison, the weights \( \alpha_n \), are obtained by the following process. First, we shift every trajectory to start point \( y_0 = 0 \). Then, we express the new goal \( g_{\text{new}} \) as a linear combination of the demonstration goals \( g_n \)

\[
g_{\text{new}} = \sum \alpha_n g_n
\]
with averaging weights $\alpha = (\alpha_1, \ldots, \alpha_n)$. This equation is underdetermined, so we add the additional constraints
\begin{equation}
\sum \alpha_n = 1 \text{ and } \|\alpha\| \text{ minimal}. \tag{4}
\end{equation}

Minimizing the norm of $\alpha$ forces the demonstrations to contribute as equally as possible to the weighted sum. This selects the one solution where all demonstrations contribute as much as individually possible to obtain the target trajectory (using the most of the available information), while still optimizing for giving more weight to the trajectories of highest relevance. Else, as the system is heavily underdetermined, it could be that only a few of the given demonstrations are considered at all.

Furthermore, we take an unweighted average over the dimensions that we do not change in the generalization. The variation in the $\eta_n$ of those dimensions only comes from the demonstration variance and does not imply that the actual situation changed and some trajectories should contribute more than others.

Because of this mode of generalization, there is no fixed number of free parameters. Instead it depends on the number of demonstration encodings used, which each contribute $3K$ parameters according to Equation (2). After averaging, the model again only contains $3K$ parameters, but cannot generalize to new situations on its own.

In addition to the main comparison between the three different models, we also performed a comparison between three different DMP generalization schemes: 1) change of the end-point, 2) weighted distance averaging, and 3) weighted goal averaging with constraints (as explained above). For the results of this comparison please see Appendix B.

B. Task Parameterized Gaussian Mixture Models

1) Model description: Gaussian Mixture Models (GMMs) are generally used for regression of arbitrary $D$-dimensional distributions. A GMM performs regression on a general distribution $P(\xi^I)$ of time $\xi^I$ (superscript $I$ for input) and the spatial coordinates $\xi^O$ (superscript $O$ for output) at that time is encoded. Note that, in contrast to DMPs, time is treated as an additional input dimension to the model, not as an independent variable.

To reconstruct the encoded trajectory, the conditional probability of having spatial coordinates $\xi^O$ at a given time $\xi^I$, $P(\xi^O|\xi^I)$, is used. Mathematical details on how to obtain it from $P(\xi^I, \xi^O)$ are given in [18] in Section 5.1. With some simplification, this distribution is again a multivariate Gaussian for each $\xi^I$. Thus, we can take the centers $\mu^O_I$ as the predicted trajectory of the model. Furthermore, the values of $\Sigma^O_I$ contain additional information about the uncertainties in the model, as they define uncertainty-ellipsoids around the predicted positions. Such information can, for example, be used in robotic applications, to indicate how strongly a motion controller should enforce following the predicted trajectory of the $\mu^O_I$.

To avoid overfitting, the model has to be regularized. The implementation we used in this work realizes this by adding a small constant $\epsilon$ to the main diagonal of the covariance matrices after each M-step in the fitting algorithm. This reduces their capability to get too restrictive in individual dimensions, smoothing out the obtained trajectories.

The parameters used by tbGMR are
\begin{equation}
\{\pi_i, \mu_i, \Sigma_i\}_{i=1}^K.
\end{equation}

The number of parameters depends on the number of Gaussians $K$ and the data dimension $D$. Since the $\Sigma_i$-matrices are symmetric, they have $(D^2 - D)/2 + D$ free parameters each (the upper right triangle including the diagonal). The $\mu_i$ each contain $D$ free parameters. From the $K$ linear combination coefficients $\pi_i$, only $K-1$ are free, because of the normalization condition $\sum \pi_i = 1$. Thus, we have
\begin{equation}
n_{\text{tbGMM}} = K \left( \frac{D^2 - D}{2} + 2D \right) + K - 1
\end{equation}
free parameters in total. In the case of 3D-trajectories, this model is 4-dimensional (one additional time dimension). Therefore, setting $D = 4$ into Equation (6), we get
\begin{equation}
n_{\text{tbGMM}} = 15K - 1
\end{equation}
for the number of free parameters in the model.

3) Task parameterized GMM: A TP-GMM considers the data $\xi$ in different reference frames $j = 1, \ldots, P$. They are obtained by affine linear transformations with some rotation matrices $A_j$ and translation vectors $b_j$. Those are called the “Task Parameters”, because they parameterize the relation of the different reference frames for each demonstration (task). GMMs in all reference frames are fitted to the data using the Expectation-Maximization-Algorithm (EM-Algorithm, [42]) in a modified version to fit all reference frames simultaneously (see Appendix A in [18]).

The models in the different reference frames $P$ can be combined to one model from which a trajectory can be retrieved by means of the chosen encoding type (for more
details see [18]). The ability to generalize to new situations is introduced by combining the models of the different reference frames with different task parameters than the ones used during learning. Note that the individual GMMs in the two different reference frames encode something conceptually different from a standard GMM with one reference frame. They cannot be used individually to retrieve the encoded trajectories, since they represent a combination of all learned trajectories as seen from one reference frame only.

It is possible to take different kinds of GMMs for encoding and reconstruction and enable generalization via the TP-GMM approach. Here we used time based Gaussian Mixture Regression as explained in Section III-B.2 to encode trajectories. The usage of $P$ different reference frames leads to $P$ sets of $\mu_i^{(j)}, \Sigma_i^{(j)}$ in the model instead of one. Thus, we have

$$n_{\text{TP-GMM}} = KP \left( \frac{D^2 - D}{2} + 2D \right) + K - 1$$

free parameters in total. This only differs from Equation (6) by the factor of $P$ in the first term reflecting the additional sets of $\mu_i^{(j)}, \Sigma_i^{(j)}$. In this work we will use $P = 2$, so putting in $D = 4$ from out tbgMR approach and $P = 2$ we get

$$n_{\text{TP-GMM}} = 29K - 1$$

free parameters for a TP-GMM.

4) Generalization using TP-GMM: In contrast to the DMP framework, where generalizations can be obtained by combining the weights of existing encodings for single trajectories, the TP-GMM approach operates on new encodings of multiple demonstration trajectories.

To generalize from a set of demonstrations to a new situation, we encode a TP-GMM with two reference frames $P = 2$. We chose the Task Parameters such that they rotate and translate all demonstration trajectories into two reference frames as defined as follows: In the first frame, identified by $j = 1$, all trajectories begin at 0. Furthermore, they are rotated such that their end-point lies in the $yz$-plane. In the second frame, denoted by $j = 2$, the roles of start- and end-point are switched. This yields $A_j$ and $b^O$ as needed by TP-GMM.

To generalize to a new trajectory, the recombination to one GMM is done with the Task Parameters of the target trajectory. These are obtained in exactly the same way as for the demonstration trajectories. The resulting GMM encodes a prediction for a trajectory with the new Task Parameters, which can be extracted by means of the encoding method used.

C. Stable Estimator of Dynamical Systems (SEDS)

1) Model description: The SEDS model [16], like the DMP approach, considers the trajectory to be the time evolution of a nonlinear dynamical system. In the case of SEDS it takes the simple form

$$\dot{\xi} = f(\xi)$$

where $\xi$ is the position in state space and $f$ some multidimensional nonlinear function. Trajectories cannot cross in state space, so care needs to be taken when choosing the state space variable $\xi$. In this work we used position and velocity. The function $f$ gets approximated by a GMM analogous to the time based GMR method from Section III-B.2. The joint distribution $P(\xi, \dot{\xi})$ is encoded with a GMM. For given $\xi, \dot{\xi}$ is then extracted as the maximum of the conditional probability $P(\xi|\dot{\xi})$.

In general, there exist several methods to estimate the parameters of the GMM (see Figure 3 in [16]), but they all have problems with stability. It is, thus, not guaranteed that a reconstructed trajectory will converge to the goal position in case of GMM.

The core of the SEDS approach is its way of estimating $f$. With SEDS, it is guaranteed that there is only one global attractor in the dynamic system, that is asymptotically stable, so trajectories from every point will converge to it. This property is achieved by reformulating the estimation of the GMMs parameters as a constrained nonlinear optimization problem optimizing model accuracy under the constraint of global asymptotic stability. The exact formulation of the optimization problem is given in Section V in [16]. As shown in Figure 3 in [16], the SEDS optimization leads to a smooth phase space with no secondary attractors. There, two different measures for model accuracy were proposed, the likelihood of the model and the mean squared error (MSE) of its prediction to the demonstration. In this work, we used the MSE. [16] also provides an implementation of the algorithms, which we used for our analysis.

While the resulting model is guaranteed to converge to the goal state, the nonlinear optimization to fit the model is not guaranteed to succeed. Our strategy of dealing with this problem is described in Appendix [3].

The number of free parameters in a SEDS model optimized with the MSE-loss is $K(1+\frac{3}{2}D(D+1))$ [16]. For our state space dimensionality of $D = 6$ (position and velocity) this yields

$$n_{\text{SEDS}} = 29K.$$  

2) Generalization using SEDS: Since SEDS optimization yields a globally stable attractor, one can simply change the position of the starting point of the integration to obtain a new trajectory that will converge to the target. While the model is designed to yield trajectories similar to the demonstrations in a vicinity around them, it is unclear whether its predictions will be accurate for generalization targets farther away. Therefore, the similarity between generalization and true human demonstrations will depend on the nature of the attractor landscape produced by the optimization. Furthermore, the duration of the movement cannot be set directly, but also depends on the properties of the attractor. In a first step we just integrate the dynamic system for as many timestamps as there are in the target trajectory (like we do with the other models). Additionally we also integrate the model for as many timesteps as needed to arrive within
5 mm of the target position, this distance corresponds to our estimated motion tracking accuracy in the dataset.

D. Probabilistic Movement Primitives

1) Model description: Probabilistic Movement Primitives (ProMPs) [23], [30], [31] are formulated as a Hierarchical Bayesian Model (HBM). A trajectory is modeled as a superposition of Gaussian basis functions with weights \( w \). In contrast to the general GMM used in tbGMR (see Section III-B.2), the kernel shapes are set (as hyperparameter), not fit. To encode the variance from multiple demonstrations, a distribution \( p(w; \theta) \) over the weights is introduced. By marginalizing \( w \) and assuming a Gaussian distribution of \( p(y_t; \theta) \) we arrive at the distribution \( p(y_t | \theta) \) as defined by Equation (4) in [23]. The parameters \( \theta = \{ \mu_w, \Sigma_w \} \) are fit by a specialized version of the EM-Algorithm for HBM [43]. We use the implementation provided by the movement_primitives python library [44].

As with the tbGMR approach, one can not only determine the mean positions for each time point, but also variance around them, providing additional information, for example for improving the controller used to follow the trajectory. Additionally, the model also encodes full cross-covariances between all the dimensions. The HBM nature of the encoding allows for changes to the distributions to be done by conditioning on new observations and propagating the results through to \( \theta \) by Bayes rule. This way, via points or even new start/end positions can be introduced.

Regularizing via the same method as for tbGMR proved to be ineffective, so we regularized the model by setting the widths of the basis functions, removing the tendency to oscillate like the tbGMR.

The parameters of a ProMP are the \( \mu_y \) and \( \Sigma_y \). Given \( K \) basis functions, we have \( DK \) free parameters from the means and \( ((DK)^2 - DK)/2 + DK \) from the covariances (symmetric \( DK \times DK \) matrix). With \( D = 3 \) this yields

\[
\text{n}_{\text{ProMP}} = \frac{9}{2} (K^2 + K).
\]

For the reconstruction case, where only one demonstration is presented and no variance is encoded, the covariance matrix is only very sparsely populated and the number of parameters decreases to \( \text{n}_{\text{ProMP rec}} = \frac{1}{2} (K^2 + 5K) \).

2) Generalization using ProMPs: The ProMP framework provides multiple ways of generalization. Different ProMPs can be joined or co-activated, to form new trajectories. In our case, generalization was achieved by conditioning the ProMPs on new start positions as explained below.

Similar to the TP-GMM method, we learn a ProMP from multiple demonstrations, but not only of the same motion, but of different ones. We shift all trajectories such that their end points are at 0. The resulting model encodes the positions of trajectories and their variance at once. Thus, like with the individual TP-GMM reference frames, evaluating the model would produce some “mean of all demonstrations” representation, not any single trajectory. Conditioning on the start point of the generalization target trajectory fixes the model to its prediction of that trajectory.

E. Optimal Control Primitives

1) Model description: Optimal Control Primitives (OCPs) [14] consist of two main parts, a trajectory representation and a controller derived from optimal control theory.

The trajectory representation could theoretically just be the trajectory itself, but to achieve some sort of encoding and compressed representation, the trajectory is expanded into a series of Chebyshev polynomials up to a given order. So, like with DMPs, the shape of the trajectory is encoded as a set of weights, just not for Gaussian kernels distributed through a phase variable (time), but as the summation coefficients of a series expansion.

The controller is used to follow the trajectory with perturbation resistance and enable adaptation to different scenarios. It is a linear-quadratic-regulator (LQR) derived from optimal control theory. The derivations and underlying models are given in [14]. It can be tuned with three parameters. \( R \) decides if to follow the reference trajectory more strictly or loosely, with high \( R \) meaning low control signals. Choosing \( R \) too high will result in the controller not properly following the trajectory. The other two parameters \( q_1 \) and \( q_2 \) weight the controllers reaction to spatial accuracy or correct velocity. \( q_1 > q_2 \) means more importance is placed on following the position profile than the velocity profile and vice-versa.

The number of free parameters in the model is equal to the number of weights for the trajectory representation. The Chebyshev polynomial series starts with order 0, so a representation of order \( o \) has \( o + 1 \) weights. Therefore, we chose the convention that an OCP with \( K = n \) uses a Chebyshev polynomial series up to order \( n - 1 \). This way, we have

\[
n_{\text{OCP}} = 3K
\]

2) Generalization using OCPs: Generalization using OCPs can be achieved by just changing the start and end points of the trajectories and letting the controller adapt to it. However, OCPs have no scaling properties like DMPs, so the resulting trajectories will be qualitatively different. For further analysis of this refer to Appendix B.

Still, the same considerations regarding using multiple models for generalization apply, as with the DMPs in Section III-A.2. So, to enable the OCPs to use all available demonstrations, we used the same generalization method as described for DMPs in Section III-A.2 and computed weighted averages of weights of the Chebyshev series.

As a result, there is again no fixed number of free parameters. Instead it depends on the number of demonstration used, but after averaging, the model again only contains \( 3K \) parameters.

IV. COMPARISON OF MOVEMENT FRAMEWORKS

A. Movement Reconstruction

1) Evaluation procedure: To test the reconstruction capabilities of the different models, we encoded every single trajectory from our dataset with every model, varying the number of kernels used from \( K = 3 \) to \( K = 11 \) for
B. Movement Generalization

different models.

to represent the average error in the reconstructions of the
reconstructions. Therefore we use medians instead of means
for every reconstruction as

$$d = \sqrt{\frac{1}{N} \sum_{i} (r_i - o_i)^2}.$$  \hspace{1cm} (14)

The resulting distributions of errors are not normally
distributed, but skewed and contain a tail of more inaccurate
reconstructions. Therefore we use medians instead of means
to represent the average error in the reconstructions of the
different models.

B. Movement Generalization

1) Evaluation procedure: To test the generalization ca-
pabilities of the models, we presented specific subsets of the
dataset (training set) as demonstrations for encoding and
then used the generalization methods described above
to generate trajectories for new situations obtained from
remaining trajectories in the dataset (test set). This way we
can directly compare the model predictions to the trajectories
of human demonstrations.

We tested two different generalization scenarios with dif-
ferent selection of demonstration trajectories. In the first case
we only used two trajectories of one action type to generate
a new trajectory. This represents the case where only few
demonstrations are available. For this, we selected triplets
of start-/end-points that were orthogonally connected (see
Figure 2). We then interpolated from the outer demonstra-
tions to the inner one and extrapolated from the inner and
one of the outer demonstrations to the respective other outer
demonstration. For example, using the indexing from Figure
2, we extrapolated from moves to positions (1,0) and (1,1)
to a move to (1,2). Because we have three repetitions of
each action type, this means the models get 6 demonstration
trajectories (two sets of three instances of the same class) to
generate the new target trajectory.

In the second case we used all available demonstrations
for a specific action type, but excluded the generalization
targets and everything that ends closer than 3 cm to the
generalization target’s end point. For the TP-GMM-model
we further excluded all demonstrations that were closer than
3 cm to the target position in the TP-GMM reference frame,
otherwise, trajectories for positions with the same length over
ground (for example (2,2) and (6,2), see Figure 2) would
have provided direct demonstrations for the generalization
target.

In both cases we did not use demonstrations from multiple
participants in one model, every generalization task was done
for each participant individually.

Again, model hyperparameters were chosen optimally, as
described in Section V-B.1 and Appendix C.

C. Performance measures

As in the case of trajectory reconstruction, the main mea-
sure to assess the generalization quality is the deviation of the
generalized trajectory from the human target demonstration.
We use the same measure as before, given in Equation (14).
To gain a more complete picture of the model performance,
we used two additional measures for further comparison.

We compared the deviation of the generalization from
the true demonstration with the general variance in the
dataset itself. This helps to qualify meaning of the deviation
generalized and ground truth (human) trajectory, since identical trajectories (as in reconstruction) are not
expected. It is possible to do this, because we have three
repetitions of every single action. So to estimate the variance
in the human execution of the actions, we measured the
mean squared distances $d_h$ between the individual human
repetitions $h(j)$ as

$$d_h^{(j,k)} = \sqrt{\frac{1}{N} \sum_{i} (h_i^{(j)} - h_i^{(k)})^2}.$$  \hspace{1cm} (15)

To compare the trajectories of different lengths, we down-
sampled the longer trajectory to match the shorter trajectory.

Note that $d$ and $d_h$ are conceptually different measures.
$d$ measures error between a human trajectory and the
and the corresponding generalized trajectory, whereas $d_h$ measures variance in the human repetitions. The end-points of the
human trajectories were not given, because humans grasped
the object slightly differently in different repetitions, re-
sulting in differences between the repetitions. The model
generalizations, however, are given exact start- and end- ends. Therefore, the inter-human variance for repetitions of a demonstration is conceptually different and expected
to be higher than the distance between a generalized target trajectory and the corresponding human demonstration. We
assume that the obtained generalization accuracy is reason-
able acceptable if it does not exceed the variance of the
human demonstrations.

The second measure we used for additional quantification
was the end-point deviation $d_e$, defined as

$$d_e = \|r_N - o_N\|$$  \hspace{1cm} (16)

with the predicted trajectory $r$ and the original trajectory $o$,
both of length $N$. Here we assessed how close the end-points
of the generalization come to the targets set by the human
demonstration, regardless of the trajectory shape. This is to
test the end-point convergence of the two GMM models.
Since the SEDS model does not always converge to the target position on time (see Section VI-A), we evaluated the generalization performance of SEDS for two cases: 1) under the condition with fixed trajectory duration (labelled SEDS in figures) as for the other models. Here, the fixed trajectory duration corresponds to the duration of the human demonstration. 2) under the relaxed condition on trajectory duration (labelled SEDS*). For this, we first did an estimation of acceptable time convergence. We measured how much longer the longest of the three repetitions of a human movement takes compared to the shortest one. We found that 95% of all longest human demonstrations completed in less than 38% more time than the shortest demonstration for the same task. Therefore, we counted a SEDS generalization successful under the relaxed time condition if it completed within 72.5% and 138% of the duration of the human demonstration and computed the generalization error only among those successful generalizations.

V. RESULTS

A. Movement Encoding and Reconstruction

1) Hyperparameter tuning: Choosing sub-optimal hyperparameters has different effects, depending on the type of model. The stability of the SEDS model is not considered here but in Section VI-A.3 because we don’t have direct influence on the kernel width in that model.

DMP, tbGMR and ProMP will produce oscillations in the velocity profile, if the kernel width is too narrow in case of DMPs and ProMPs or the regularization term is too small in case of tbGMR. This is not the case for the human demonstrations and should therefore be avoided. Choosing sub-optimal width for the DMP and ProMP kernels increases the reconstruction error (see Appendix C). For tbGMR, however, lowering the regularization, and therefore introducing oscillations in the velocity profile, reduces the reconstruction error of the position profile. This makes choosing hyperparameters more difficult, as the well defined reconstruction error cannot be the only criterion (see Appendix C.2). Therefore it is harder to determine the optimal regularization for accurate but still human-like trajectories with the tbGMR model.

2) Reconstruction accuracy: Figure 4 shows some examples of trajectory reconstructions using different numbers of kernels that are indicative of some general behaviour of the models.

With only three kernels, all models but SEDS struggle with accurate trajectory representation. Although OCP and DMP have larger overall error than the other models, they have zero error at the start- and end-point, which is a
Fig. 5: Comparison of movement encoding frameworks on the movement reconstruction task. Median reconstruction error vs. number of kernels is shown for each model. Failed SEDS encodings are not included in this statistics (see Section V-A.3). Note, there is no result for SEDS for $K = 11$ because the encoding time for high $K$ values using the repeated optimization method (see Section III-C.1) is impractically long.

direct outcome of their respective learning mechanisms. The tbGMR and ProMP models even miss the correct start- and end-point completely. This behaviour is inherent and can’t be avoided.

When increasing the number of kernels, the reconstructions become qualitatively identical to the demonstrations with errors dropping below the 5 mm of tracking error in all cases. The main source of error for tbGMR and SEDS at high $K$ is not the trajectory shape, but rather deviations in velocity. In contrast to all other models, tbGMR is not able to start the trajectory with near zero velocity, resulting in it being ahead of the target.

The median reconstruction error obtained from all reconstructions vs. the number of used kernels is shown in Figure 5.

We can see that the median errors of all models fall below our 5 mm tracking error estimate, if a sufficient number of kernels is used. However, there are qualitative differences among the five models. The DMP, ProMP and OCP models perform significantly worse than the others for few kernels but become the best for more than six kernels, greatly benefiting from more kernels. SEDS and tbGMR improve less with the use of more kernels. Furthermore, the error distributions of DMP, ProMP and OCP become much narrower, whereas the SEDS error only marginally decreases and the spread of the error distribution remains highest. This demonstrates that SEDS encoding can fail or be of insufficient reconstruction quality even with higher numbers of kernels. Moreover, increasing the number of kernels does not ensure convergence of the model. For the other models, however, using more kernels significantly improves the reconstruction accuracy. Especially the DMP, ProMP and OCP models become more and more accurate, all reaching sub-millimeter errors with $K = 11$, whereas the tbGMR error is 2.6 mm.

Considering the parameters used to encode the trajectories (see Table III), the difference in using 1D or multidimensional Gaussian kernels (see Section VI-A) becomes apparent. Both GMM based approaches need many more parameters than DMP or OCP at any number of kernels. With $K = 11$, DMP and OCP only need 33 parameters, whereas the tbGMR and SEDS models with $K = 3$ already need 44 and 87 parameters, respectively. ProMP without variance encoding (as only one demonstration does not define variance) uses more parameters than OCP and DMP, but not as many as the GMM based models. When comparing the two GMM based models, the tbGMR model uses fewer parameters, because the TP-GMM extension for generalization (that nearly doubles the parameters used) is not needed for reconstruction. Nevertheless, all encoding models compress the actual trajectory data (except for very short trajectories). The average length of a trajectory is 83 samples, which leads to $83 \times 3 = 249$ parameters. This number of parameters is surpassed by DMP, tbGMR, SEDS, ProMP, OCP at $K = 83$, 17, 9, 20, 83, respectively.

3) Convergence of SEDS: As the optimization process in the SEDS model is not guaranteed to converge for all initial conditions, there were failed encodings. As described in Section III-C, we tried to mitigate this by adding random noise to the initial conditions and changing a parameter affecting stability. Still we were not able to find an automatic way to make the optimization convergent for all trajectories in the dataset regardless of the number of kernels $K$. With $K = 3$, 84% of the optimizations converged. The number of successful encodings increased with the higher number of kernels up to 92% for $K = 7$. Out of all trajectories, only 28 could not be successfully encoded regardless of the number of kernels. In almost all of the cases these trajectories were take out actions from the highest target object, which are also much longer as compared to the other actions.

B. Movement Generalization

1) Hyperparameter tuning: As we only recombine existing DMP and OCP encodings for generalization, hyperparameter tuning does not have to be reconsidered for these models.

However, the TP-GMM\(^2\) and ProMP hyperparameters need to be tuned again, since the fitting process is different from the reconstruction case (see Sections III-B.4, III-D.2 and Appendix C.4). For TP-GMMs, we observed the same qualitative behaviour as in the reconstruction case with tbGMR. Lowering the regularization term reduces the error of position profile, but increases oscillations in the velocity profile. To obtain smooth trajectories, the regularization term had to be chosen an order of magnitude higher than in the

\(^2\)Note that TP-GMM consists of two tbGMR models fitted on two reference frames and and not one tbGMR model as in the reconstruction case.
Fig. 6: Examples of trajectories for different generalization situations. Position and velocity profiles are shown in the top and bottom rows, respectively, whereas plots at the bottom of each panel show the deviation from the human trajectory. Color coding for models is the same as in Figure 4. Additionally, human demonstration trajectories are shown in light gray.

reconstruction case. However, this lead on average to more than 5 mm error at the end-point of the trajectory regardless of the number of kernels.

To circumvent this issue, we chose to use only little regularization (10^{-6}) and removed the oscillations by filtering the high frequencies out with a low-pass filter at 3 Hz. While this is a post-processing step that is not part of the model, it is neither complex nor computationally expensive.

2) Generalisation using only few demonstration trajectories: Figure 6 shows several examples of trajectories for movement generalization to exemplify some behavioural traits of the different models.

We observe that the SEDS model has the highest errors mainly because the velocity profile is not well matched. Figure 6b shows a case where the SEDS attractor landscape is not formed appropriately. Although at the beginning of the movement, the generalised trajectory follows the target trajectory quite well, it does it at a very low speed, and therefore it does no reach the target position in the same time as the original demonstration.

In Figure 6c, we also see that the TP-GMM misses the start-point by a few millimeters. DMP and OCP are generally very close to each other in velocity and position profile.

Furthermore, the human variability in demonstrations becomes apparent. Especially, in Figure 6d we can see that in one case the manipulation object was likely grabbed differently in one demonstration, resulting in a completely different trajectory shape with start-/end- points about 3 cm higher than usual.

Statistics for the generalization error are shown in Figure 7, where the results for interpolation and extrapolation cases are shown in (7a) and (7b), respectively. Two cases for SEDS are shown: under the condition of a fixed trajectory duration (SEDS) as for the other models and under a relaxed condition on the trajectory duration (SEDS*) as described in Section IV-C.

Errors for the extrapolation cases are consistently higher than for the interpolation cases.

As in the reconstruction case, the DMP and OCP models shows a big increase in accuracy for growing $K$, but DMP needs fewer kernels than OCP to reach the same performance level.

The SEDS model performs the worst of the three models and has much wider error distributions as compared to the other two models.

In case of the relaxed time condition (SEDS*), the error is comparable to the other models for interpolation, but still higher for extrapolation as compared to DMP, OCP and TP-GMM.

The ProMP model performs better than SEDS, but worse than the other three models. Particularly for extrapolation, its error stays higher than the human variance regardless of kernel number.

One can also observe that the errors are not converging to zero if the number of kernels $K$ is increased, but saturate for larger $K$.

Considering the number of parameters, both SEDS and TP-GMM now use nearly the same number of parameters with $29K$ and $29K−1$, respectively. ProMP overtakes them on parameter usage at $K = 7$, increasing quadratically in $K$. Because the average number of demonstrations is 6, the DMP and OCP averaging procedure to obtain the generalised trajectory uses $18K$ parameters. This is still less than the other models, but significantly more (six times) than in reconstruction.
Fig. 7: Comparison of movement encoding frameworks on the movement generalisation task when only few demonstration trajectories are available in case of interpolation (a) and extrapolation (b). Median error vs. number of kernels is shown for each model. Failed SEDS encodings are not included in this statistics (see Section V-A.3). Note, there are no results for SEDS for $K > 6$ because the encoding time for high $K$ values using the repeated optimization method (see Section III-C.1) becomes impractically long. In case of human demo, error denotes the variance among human demonstrations whereas for models, error denotes deviation from human demonstration (see Section IV-C).

3) Generalisation using many demonstration trajectories: Results for the generalisation using many demonstration trajectories are shown in Figure 8. This set of generalisations contains all possible target positions with all available demonstrations, as the difference between inter- and extrapolation is not well defined in this case. There is no qualitative change in the shape of the error curves as compared to the case of generalisation with few demonstrations. The absolute error is comparable to the interpolation case. The number of SEDS reconstructions with large error is smaller than with few demonstrations. Notably, the ProMP model, which performed worse than DMP, TP-GMM and OCP with few demonstrations, performs on the same level as DMP.

SEDS, TP-GMM and ProMP still use the same number of parameters, since they encode everything into one model. Hence, the number of demonstrations has no effect on the number of parameters used. The DMP and OCP models, however, use more, because the set of demonstrations used is now even bigger that before. Since there are 13 possible start-/end-points (see Figure 2), one of which is the target, there are 12 positions in the demonstration set. With the three repetitions of each action, the average demonstration set contains 36 moves (there might be extra/forgotten repetitions or exclusions as defined in IV-B.1). This yields $108K$ DMP/OCP parameters that are used in the averaging procedure to obtain a generalized trajectory.

4) Behaviour and convergence of SEDS: The SEDS model gives no guarantee for end-point convergence within the specified time in case of generalised trajectories that are not similar to the demonstrations it had been fitted to. However, the smoothness of its state space leads to the reasonable assumption that it can indeed generalize to different positions. There are many cases where the model takes much longer (see Figure 6b) or is faster (see Figure 6e) than the target trajectory.

Figure 9 shows the generalization case presented in Figure 6b integrated for a longer time.

This specific case does not fulfill the requirement of converging to the target on time, since it takes 1.28 s instead of the original 0.74 s, which is more than a 38% increase of the movement duration.

Table III summarizes the success rates of SEDS encodings for the optimization procedure (Optim. success) as defined in Section III-C.1 and for the converge to the end-point under the relaxed time condition (End-point conv. success) as given in Section IV-C.

We can see that in summary over all $K$ (last row of Table...
TABLE II: Percentages of successful SEDS encodings. Optim. success is the percentage of cases where the optimization succeeds by the procedure defined in [III-C.1], regardless of the generalization quality. End-point conv. success is the percentage of cases where SEDS converge to the end-point under the relaxed time condition (SEDS*) given in Section [IV-C]. The last row contains the union over all $K$. The other four models have 100% optimization success rate.

| $K$ | Interpolation | Extrapolation | Many demos |
|-----|---------------|---------------|------------|
|     | Opt. success conv. | Opt. success conv. | Opt. success conv. | Opt. success conv. |
| 3   | 89%           | 58%           | 93%         | 44%         | 94%         | 51%         |
| 4   | 82%           | 54%           | 96%         | 42%         | 82%         | 54%         |
| 5   | 89%           | 61%           | 90%         | 39%         | 88%         | 55%         |
| 6   | 78%           | 49%           | 81%         | 35%         | 71%         | 46%         |
| all | 99.9%         | 88%           | 99.7%       | 73%         | 99.8%       | 80%         |

almost all demonstration trajectories can be encoded by the SEDS model, however, not all of those models represent generalized trajectories that converge to the target point even under the relaxed time condition. In general, the ratio of optimization success has a tendency to decrease with higher $K$, as optimizing more kernels is harder.

VI. DISCUSSION

A. Key Model Differences

The five models investigated in this study represent different approaches for encoding, reconstructing and generalization of movement trajectories. They all have different requirements and give different guarantees with respect to the solutions obtained by them. Here we briefly summarize the key differences of the models from a theoretical point of view.

DMP, SEDS and OCP are all based on a dynamical system with an attractor. Therefore, the convergence of the trajectories from any start-point to the specified end-point is always guaranteed. Without the occurrence of perturbations, DMP and OCP ensure convergence in a given time, while with SEDS the actual duration of a generalized movement depends on the obtained model. For ProMP and TP-GMM an additional controller is needed to provide perturbation resistance.

For the TP-GMM model, the start- and end-points can only indirectly be set by defining new task parameters. Whether the model will reach these points or not is determined by the quality of the model fit achieved from demonstration data, however there is no guarantee that the generalised trajectory will start and end at the points specified by the task parameters.

For the ProMP model, the end points can be set exactly by conditioning the model. However, the shape of the trajectory at the end points depends on the structure of the encoded distribution over the trajectory parameters. If there are insufficient demonstrations around the new points, the trajectory might not look human-like.

The GMM based models and ProMP contain cross correlations between the different dimensions represented by the off-diagonal entries of the covariance matrix, whereas DMPs and OCPs have fully independent models for each dimension. While this turned out to not be beneficial for movement encoding, reconstruction, and generalisation, it might be beneficial for other tasks such as movement recognition.

In Table III we provide a summary for the numbers of free parameters in the different models as derived in the previous sections.

TABLE III: Number of free model parameters $n_p$ for 3D trajectories. The numbers $K$ and $N$ correspond to the number of kernels and the number of demonstrations, respectively.

|               | DMP/OCP | bGMR | TP-GMM | SEDS | ProMP |
|---------------|---------|------|--------|------|-------|
| Reconstruction | $3K$    | $15K - 1$ | $29K$ | $\frac{1}{2}(K^2 + 5K)$ |       |
| Generalization| $3NK$   | $29K - 1$ | $29K$ | $\frac{9}{2}(K^2 + K)$ |       |

Because the non DMP or OCP models use multivariate Gaussian kernels instead of 1D Gaussian kernels with fixed center and variance, they use significantly more parameters. The parameter number in the covariance matrices of the GMMs and ProMP scales with $D^2$, while DMPs and OCPs only scale with $D$.

Note that the comparatively lower number of parameters in DMPs and OCPs does not directly imply that they are more efficient for trajectory compression. While this might be true for simple reconstruction without generalization, the GMM based models and ProMPs also retain information about the variance in the demonstrations and cross correlations between dimensions, that DMPs and OCPs do not encode at all. This information can be used by external controllers to modulate the response to perturbations based on the variance. If the variance is high in the current situation anyway, a perturbation can be accepted and slowly corrected. If not, quick action needs to be taken. This can be done for OCPs as well, but only globally. The parameters $q1$ and $q2$ define how important it is to keep the correct position or velocity, however they are not automatically adapted to the current situation.

ProMPs further have the additional benefit, that it is not
necessary to pre-define possible generalization cases, like with selecting the reference frames for TP-GMM. The model can theoretically be conditioned on any point at any time, the encoded model, and therefore the provided demonstration data, then decides if the resulting trajectory will be accurate or not.

One property of the GMM based encodings, the cross correlation between dimensions, was not assessed in this work. More research is needed to investigate the benefit of these off-diagonal elements of covariance matrices for other tasks, e.g., movement recognition based on GMM or DMP encoding parameters.

B. Influence of Hyperparameters

For generalization with TP-GMM, it proved to be impossible to use the regularization for keeping trajectories smooth, while also maintaining good accuracy. Especially in the case of using few demonstrations, where regularization is needed to keep the models from overfitting to the few available demonstrations, high regularization leads to higher errors and the generalized trajectories miss the target end-points in most of the cases. Lowering the regularization reduces the error in the position profiles, however, it introduces oscillations in the velocity profiles.

The more kernels are used, the smaller the absolute value of the oscillations becomes, still the qualitative behaviour does not change. To test if this behaviour is specific to our dataset, we reproduced the reconstruction and generalization experiments as in [18] and looked at their velocity profile. We found that also with their chosen regularization, the velocity profiles of their reconstructions and generalizations oscillate as observed in our case.

Since this undesired behaviour of the TP-GMM could not be fixed by means of the model itself, we lowered the regularization to reduce errors in position profiles, and solved the velocity oscillation problem by utilising a low-pass filter.

ProMPs also exhibited oscillations when using sub-optimal hyperparameters. For $K = 3$ and $K = 4$, it was even impossible to find hyperparameters that lowered the oscillations below our defined threshold (see Appendix C.4). However, the general reconstruction/generalization error was also high in that cases, such that the use of this model with low $K$ values becomes impractical.

C. Influence of the Number of Demonstration Trajectories

The difference between generalizing with few or many demonstrations is significant. In case of many demonstrations, the TP-GMM model improves its end-point convergence by more than 50% as compared to the case with few demonstrations.

The ProMP model produces worse generalizations than DMP, OCP and TP-GMM with few demonstrations. When more demonstrations are given, however, the generalization quality reaches the same level as DMP, OCP and TP-GMM. We suppose, the choice of modeling the distribution over the weights as a Gaussian distribution (see Section III-D) cannot accurately represent demonstrations concentrated on only two different action classes.

For SEDS, the number of encodings with high generalization error decreases with more demonstrations. Comparing Figures 7 and 8, we see that the third quartile of the error distributions when using many demonstrations is lower than for few demonstrations in most of the cases.

Furthermore, when looking at Table II, we see an additional dependence on nature of the presented demonstrations. The optimization success for individual $K$ values is higher for extrapolation than interpolation, indicating this case is easier. But, the generalization error for extrapolation is worse than for interpolation, (also see Figure 7a and Figure 7b), indicating it is harder. We suppose that encoding an extrapolation model as such is actually easier than encoding an interpolation model, because in case of extrapolations the demonstrations are closer together than for interpolations. However, obtaining an attractor landscape for accurate generalizations outside demonstrations is less probable for extrapolation than obtaining an attractor landscape within demonstrations for interpolation, raising the number of extrapolation cases that do not meet the relaxed time convergence criteria.

For DMPs and OCPs, the usage of many demonstrations as compared to few demonstrations did not change the results significantly. For a comparison with the simplest generalization variant with only one demonstration, which is possible with these two models, see Appendix B.

D. Extensions and specializations

As discussed in the introduction, there are countless extensions and improvements to the models presented in this study. However, all of the models but SEDS are able to produce generalizations within human variance, given enough kernels and demonstrations. Moreover, they all end up very close to each other (see Figure 8 at $K = 11$). Therefore we expect these results to be close to the optimum of what is possible given our demonstration dataset. We do not except any specialized variant to yield substantially better results, error-wise. However, there might be some other possible improvements regarding ease of use or additional capabilities.

The TP-GMM approach is agnostic of the underlying GMM used. The distributions and task parameters chosen in Section III-B.2 can be chosen differently, to represent the encoded trajectories in a different way. The paper on TP-GMMs by Calinon et al. [18] presents a lot more possibilities to encode and generalize trajectories with TP-GMMs. Future work could test more of these encoding variants to see if any of them lead to easier hyperparameter tuning for smooth trajectories without losing precise end-point convergence.

The SEDS model presents itself as difficult to use, because of its optimization issues and bad convergence. To improve the convergence of the SEDS model, a different solver could be used. The paper on SEDS [16] mentions a custom written solver that has “several advantages over general purpose solvers” (see Section V. in the paper), that was unfortunately

---

3Note that in [18] velocity profiles are not shown
not published with the rest of the code, so we had to use the general purpose solver. However, there is no guarantee that a better local optimum of the loss function would yield a better generalization.

Our DMP optimization algorithm could also be improved. The implications of the very simple δ-rule with its simple loss function can be clearly seen in the reconstructions with only few parameters (for example Figure 4a and 4b). For $K = 3$ the error is 0 at three places, the centers of the Gaussian kernels that matter for the δ-rule. To improve the encoding quality, theoretically, any other more sophisticated supervised learning rule could be used, for example one that would optimize the encoding based on the whole trajectory or adjust the centers and variances of the Gaussian kernels [40], [45], [46]. Using variable position kernels could improve the DMP results, but this would prevent the averaging of weight vectors to obtain generalized trajectories and, therefore, some other generalisation mechanism would be needed instead. Furthermore, using biologically inspired DMPs [10] might enable generalization from single trajectories without the problems caused by DMPs scaling properties (see Appendix B).

The ProMP model exhibited difficulties with generalizing in the presence of only few demonstrations. Recently, a method which combines the TP-GMM approach (which is to a certain extent agnostic of the underlying model) with ProMPs was published [24]. Using this method, to generalize in the few demonstration case, could improve the ProMP performance at the expense of having to pre-define task parameters again.

E. Encoding and Inference Time

We will only make relative statements about the inference and encoding times of our models, since it is always possible to take more or less potent hardware with which different run-times will be obtained.

The inference times of all three models are significantly faster then the respective encoding times. ProMPs and the GMM based models are slower than DMPs and OCPs, but still fast enough for online motion generation for robotic applications.

The differences of encoding times are more prominent. DMPs, as the simplest model, without matrix multiplications and with a simple loss function, are significantly faster than the GMM based models. OCPs and ProMPs are between them, but this is most definitely implementation dependent, as ProMPs also run an EM-Algorithm optimization, like the tbGMR approach. While the tbGMR approach remains on the order of seconds, a run-time of the SEDS optimization can take on the order of a minute.

F. Considerations beyond our dataset

Our dataset contains basic human manipulation actions. However, most of the general manipulations are built up from those simple building blocks [32]. In the following we discuss how well the models may cope with more complex actions. We cannot judge the similarity to human actions as we can with the actions in our dataset, but we can at least estimate how easy or hard it would be to adapt the models.

In more complex cases, DMPs may probably need to be augmented with via points [13] or be represented as chains of joined simpler actions [39], because our simple method of averaging based on goal positions would not be applicable anymore. As described in the introduction, there exist multiple different models for this.

OCPs would not work in more complex scenarios. Theoretically, the LQR-controller can be used to reach any point, but the trajectory would most definitely not be human-like. For this, a more sophisticated method to generalize the trajectory representation would be needed. To our knowledge, no such method on a series of Chebyshev Polynomials exists.

For TP-GMMs, it would be necessary to work with more reference frames to introduce additional constraints. However, these need to be fixed and manually designed prior to learning the models, again limiting the ability to react to any spontaneous generalization requirement.

ProMPs seem very suitable for more complex generalizations, since they can be conditioned on any via point without special preparation. However, our results for the difference in generalization error between the few and many generalization cases indicate that this would only work well if sufficient demonstrations around the possible generalizations are provided.

Because of its global state space and smoothing constraint, it does not seem that SEDS would be able to generalize well over more complex actions.

VII. Conclusions

In this work we compared five motions encoding frameworks including the most widely used encoding frameworks in robotics: Dynamic Movement Primitives (DMPs), time based Gaussian Mixture Regression (tbGMR) with Task Parameterized Gaussian Mixture models (TP-GMMs), stable estimator of dynamical systems (SEDS), Probabilistic Movement Primitives (ProMPs) and Optimal Control Primitives (OCPs).

We showed that the encoding of single trajectories from our dataset of human manipulation actions for reconstruction is possible for all models with an accuracy comparable to our tracking error estimate, given that enough kernels are used. However, the SEDS model does not always converge to the demonstration just by increasing the number of kernels.

The GMM based models and ProMP with their multidimensional Gaussian kernels use significantly more parameters than DMPs and OCPs, but do not achieve better results.

Tuning the hyperparameters of the system is easiest for DMPs, OCPs and ProMPs. When using the tbGMR approach, care has to be taken to not under-regularize the model. While low regularization improves the reconstruction error of the position profiles, it also introduces oscillations in the velocity profiles.
Generalization from demonstrations in our dataset to new targets is possible with all analysed models, but they all have different strengths and weaknesses. The DMP and OCP models are easy to operate, because they require no additional hyperparameter considerations, as they re-use existing encodings from reconstruction. They also produce generalizations comparable to the human inter-demonstration variation in every condition. The TP-GMM model achieves comparable results in terms of position profiles of the generalised trajectories, but fails to reach the target end-point point precisely without additional hyperparameter considerations and post processing steps. The ProMP model is the easiest to operate, given sufficiently many demonstrations. It is easy to regularize, requires no consideration regarding generalization (like weighting different contributions or pre-defining task parameters) and performs on the same level as the other models. However, it yields worse results when only few demonstrations are given.

The SEDS model was not primarily developed to perform this kind of generalization, so it performs worse as compared to other models, but generalization is still possible to some extent. However, the convergence problems with the non-linear optimization, the inferior generalization quality and the long encoding times make SEDS impractical for many situations.

In conclusion, we showed that all models but SEDS can be used to reliably generalize to new situations. The obtained generalizations indeed closely resemble human motion patterns. The choice of the best model, however, depends on the application. If only few demonstrations are available, DMP, OCP and TP-GMM can be used, with DMP needing the least parameters for sufficient accuracy, while TP-GMM should not be used if exact end-point convergence is necessary. Given many demonstrations that cover the state space more evenly, all models but SEDS can be used, with ProMP being easiest to operate.

VIII. DECLARATION OF CONFLICTING INTERESTS

The Authors declare that there is no conflict of interest.

IX. FUNDING

This work was supported by the the Volkswagen Foundation [“DeMoDia”, grant number ZN3543] and by the German Science Foundation Grant [DFG WO 388/16-1].

X. RESEARCH DATA

The dataset used in this work is published on Zenodo:
https://doi.org/10.5281/zenodo.7351664

REFERENCES

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. Robotics: Modelling, Planning and Control. Springer Publishing Company, 2009.

[2] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot Programming by Demonstration. In Bruno Siciliano and Oussama Khatib, editors, Springer Handbook of Robotics, pages 1371–1394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[3] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent Advances in Robot Learning from Demonstration. Annual Review of Control, Robotics, and Autonomous Systems, 3(1):297–330, May 2020.

[4] Auke J. Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamic Movement Primitives: Learning Attractor Models for Motor Behaviors. Neural Computation, 25(2):328–373, February 2013.

[5] Xiaochuan Yin and Qijun Chen. Learning nonlinear dynamical system for movement primitives. In 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 3761–3766, October 2014.

[6] Affan Pervez, Arslan Ali, Jee-Hwan Ryu, and Dongheui Lee. Novel learning from demonstration approach for repetitive teleoperation tasks. In 2017 IEEE World Haptics Conference (WHC), pages 60–65, Munich, Germany, June 2017.

[7] Aleš Ude, Bojan Nemeč, Tadej Petrič, and Jun Morimoto. Orientation in Cartesian space dynamic movement primitives. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 2997–3004, May 2014.

[8] Fares Abu-Dakka, Bojan Nemeč, Jimmy Jorgensen, Thiusius Savarinmuthu, Norbert Krüger, and Aleš Ude. Adaptation of manipulation skills in physical contact with the environment to reference force profiles. Autonomous Robots, 39, May 2015.

[9] Fares J. Abu-Dakka and Ville Kyrki. Geometry-aware Dynamic Movement Primitives. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 4421–4426, May 2020.

[10] Heiko Hoffmann, Peter Pastor, Dae-Hyung Park, and Stefan Schaal. Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In 2009 IEEE International Conference on Robotics and Automation, pages 2587–2592, May 2009.

[11] Roman Weitschat and Harald Aschemann. Safe and Efficient Human–Robot Collaboration Part II: Optimal Generalized Human-in-the-Loop Real-Time Motion Generation. IEEE Robotics and Automation Letters, 3(4):3781–3788, October 2018.

[12] Matteo Saveriano, Fares J Abu-Dakka, Aljaž Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. The International Journal of Robotics Research, 42(13):1133–1184, November 2023.

[13] You Zhou, Jianfeng Gao, and Tamim Asfour. Learning Via-Point Movement Primitives with Inter- and Extrapolation Capabilities. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4301–4308, Macau, China, November 2019.

[14] Sebastian Herzog, Florentin Wörgötter, and Tomas Kulpivc. Genera- tion of movements with boundary conditions based on optimal control theory. Robotics and Autonomous Systems, 94:1–11, August 2017.

[15] Seyed M. Khansari-Zadeh and Aude Billard. BM: An iterative algorithm to learn stable non-linear dynamical systems with Gaussian mixture models. In 2010 IEEE International Conference on Robotics and Automation, pages 2381–2388, Anchorage, AK, May 2010.

[16] Seyed M. Khansari-Zadeh and Aude Billard. Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. IEEE Transactions on Robotics, 27(5):943–957, October 2011.

[17] Sylvain Calinon, Florent Guenter, and Aude Billard. On Learning, Representing, and Generalizing a Task in a Humanoid Robot. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics), 37(2):286–298, April 2007.

[18] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. Intelligent Service Robotics, 9(1):1–29, January 2016.

[19] You Zhou, Jianfeng Gao, and Tamim Asfour. Movement primitive learning and generalization: Using mixture density networks. IEEE Robotics & Automation Magazine, 27(2):22–32, 2020.

[20] Ian R. Manchester and Jean-Jacques E. Slotine. Transverse contraction criteria for existence, stability, and robustness of a limit cycle. Systems & Control Letters, 63:32–38, January 2014.

[21] Harish chaandar Ravichandar and Ashwin Dani. Learning position and orientation dynamics from demonstrations via contraction analysis. Autonomous Robots, 43(4):897–912, April 2019.

[22] Harish chaandar Ravichandar, Pavan Kumar Thota, and Ashwin P. Dani. Learning periodic motions from human demonstrations using transverse contraction analysis. In 2016 American Control Conference (ACC), pages 4853–4858, July 2016.
[23] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic Movement Primitives. *Advances in Neural Information Processing Systems (NIPS)*, page 9, January 2013.

[24] Xueyang Yao, Yinghan Chen, and Bryan Tripp. Improved Generalization of Probabilistic Movement Primitives for Manipulation Trajectories. *IEEE Robotics and Automation Letters*, 9(1):287–294, January 2024.

[25] Sylvain Calinon, Tohid Alizadeh, and Caldwell G. Darwin. On improving the extrapolation capability of task-parameterized movement models. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 610–616, November 2013.

[26] A. Lemme, Y. Meirovitch, M. Khansari-Zadeh, T. Flash, A. Billard, and J. J. Steil. Open-source benchmarking for learned reaching motion generation in robotics. *Paladyn, Journal of Behavioral Robotics*, 6(1), March 2015.

[27] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[28] Pham N. Hung and Takashi Yoshimi. An approach to learn hand movements for robot actions from human demonstrations. In 2016 IEEE/SICE International Symposium on System Integration (SII), pages 711–716, December 2016.

[29] Ales Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives. *Robotics, IEEE Transactions on*, 26:800–815, November 2010.

[30] Alexandros Paraschos, Gerhard Neumann, and Jan Peters. A probabilistic approach to robot trajectory generation. *IEEE-RAS International Conference on Humanoid Robots*, 2015:477–483, February 2015.

[31] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42, March 2018.

[32] Florentin Wörgötter, Eren Erdal Aksoy, Norbert Krüger, Justus Piater, Ales Ude, and Minija Tamosiunaite. A Simple Ontology of Manipulation Actions Based on Hand-Object Relations. *IEEE Transactions on Autonomous Mental Development*, 5(2):117–134, June 2013.

[33] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281–1289, September 2018.

[34] Tannay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols*, 14(7):2152–2176, July 2019.

[35] Alexander Mathis and Mackenzie Mathis. *DeepLabCut/DeepLabCut*, 2018.

[36] Pierre Karashchuk, Katie L. Rupp, Evyn S. Dickinson, Elischa Sanders, Eiman Azim, Bingni W. Brunton, and John C. Tuthill. Anipose: A toolkit for robust markerless 3D pose estimation. *bioRxiv: the preprint server for biology*, page 2020.05.26.117325, May 2020.

[37] Pierre Karashchuk and Katie L. Rupp. LambdaLoop/anipose, 2019.

[38] Stefan Schaal, Peyman Mohajerian, and Auke Ijspeert. Dynamics systems vs. optimal control — a unifying view. In *Progress in Brain Research*, volume 165, pages 425–445. Elsevier, 2007.

[39] Tomas Kulvicius, KeJun Ning, Minija Tamosiunaite, and Florentin Wörgötter. Joining Movement Sequences: Modified Dynamic Movement Primitives for Robotics Applications Exemplified on Handwriting. *IEEE Transactions on Robotics*, 28(1):145–157, February 2012.

[40] A.J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, pages 1398–1403 vol.2, May 2002.

[41] Roman Weitschat, Sami Haddadin, Felix Huber, and Alin Albu-Schaffer. Dynamic optimality in real-time: A learning framework for near-optimal robot motions. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5636–5643, Tokyo, November 2013. IEEE.

[42] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, September 1977.

[43] Alessandro Lazaric and Mohammad Ghavamzadeh. Bayesian multitask reinforcement learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML ’10, pages 599–606, Madison, WI, USA, June 2010. Omnipress.

[44] Robotics Innovation Center DFKI GmbH. Dfki/movement_primitives, 2021.

[45] Stefan Schaal and Christopher G. Atkeson. Constructive Incremental Learning from Only Local Information. *Neural Computation*, 10(8):2047–2084, November 1998.

[46] Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, volume 1, pages 288–293. Morgan Kaufmann, 2000.
as well. Figure 10 compares the reconstruction accuracy computed all DMP encodings with the sigmoid function performance of the DMP formulation presented in [39], we without change the goal point V-A.2.

Fig. 10: Comparison of DMP framework with and without $v$ on the movement reconstruction task.

APPENDIX

A. DMP Model Simplification

In this work we simplified the DMP formulation presented in [39] by removing the function $v$ from the computation of the function $f$. The function $v$ is a sigmoid that drops from 1 to 0 in a very short time around the end-time $T$ of a trajectory ($v(t) = 0.5$ at $t = T$). This function was specifically designed for joining of movement trajectories which allows blending of DMP kernels at the joining point without affecting the influence of kernels at the joining point. If no joining is performed (or at the end of the joined motion), the function $v$ ensures that the trajectory converges to the goal point regardless of the shape of $f$ in time $t = T + \delta t$ (where $\delta t << T$).

To make sure this modification does not affect the performance of the DMP formulation presented in [39], we computed all DMP encodings with the sigmoid function as well. Figure 10 compares the reconstruction accuracy between the models with and without $v$, exactly as in Section V-A.2.

There is no significant difference to the DMP variant without $v$. Since all generalizations are done using the encodings of the single trajectories, leaving out the sigmoid will have the same (no) effect on generalizations as well.

B. DMP/OCP Generalization Variations

In addition to the DMP/OCP generalization method presented in Section III-A.2 here we also include two other approaches, namely, generalization by changing the end-point, and generalisation by using weighted averaging without constraints.

1) Generalization by changing end-point: The simplest way to generalize a DMP/OCP to a new situation is to just change the goal point $g$ and use the weights/trajectory-representation without any changes. Instead of averaging the available demonstrations, we select the closest trajectory with respect to the generalization target and change its goal point to the new end-point.

2) Generalisation using weighted distance average: Instead of computing averaging weights based on goals with constraints as explained in Section III-A.2 one can also think of a simpler way to determine averaging weights. Another way to weigh the contribution of demonstrations is just by distance to the generalization target ([41] Section IV-A). For this, we also translate all the trajectories to a common reference frame, but instead of obtaining the weights from expressing the new goal as a linear combination of the demonstration goals, we just weigh the trajectories by the distance $d_i$ of their goal to the new target goal. We want close trajectories to have more influence than far away ones, so we take the inverse of $d_i$ and obtain the weights $\alpha_i$ as

$$x_i = 1/d_i,$$  \hspace{1cm} (17)

$$\alpha_i = x_i/\sum x_i.$$  \hspace{1cm} (18)

The second equation normalizes the weights $\alpha_i$ such that $\sum \alpha_i = 1$.

3) DMP performance: We compared the generalization strategies on both the few and many demonstration tasks as defined in Section IV-B.1.

Figure 11 shows the results for the generalization error evaluated like in Section V-B.3. The general trend of extrapolation errors being higher than interpolation errors is visible again. For interpolation, goal and distance averaging perform the same, because our demonstration scenario of two bundles of demonstrations to either side of the generalization target makes them choose the same weights. In extrapolation, there is a difference between the two, as pure distance averaging has no means of accounting for the spatial distribution of demonstrations. So it produces DMP weights that would be suitable for a point between the demonstrations, but closer to the ones near the generalization target. The method is effectively still doing an interpolation, but to the wrong point. This results in errors even worse than simple end-point shifting. Generalisation by changing the end-point performs worse than the inter human variance in all situations. Additionally, there is less difference between interpolation and extrapolation than in the averaging cases. This could be explained by the fact that selecting the closest trajectory neglects the difference between the interpolation and extrapolation scenarios. There is always only one trajectory, so compared to averaging the spatial arrangement of the other available trajectories does not matter, leaving less difference between the two scenarios.

Figure 12 shows the results for the generalization error evaluated like in Section V-B.3. Again, goal point averaging performs the best of the three. The simplest method of changing the end point performs worse. Relative to the inter human variance, both averaging methods get lower errors at high enough $K$, whereas the end-point changing stays above.

In conclusion, there is more accuracy gained by using a more complex method for obtaining new DMP weights, especially choosing an averaging approach over simple end-
Fig. 11: Comparison of movement encoding frameworks on the movement generalisation task when only few demonstration trajectories are available in case of interpolation (a) and extrapolation (b). Median error vs. number of kernels is shown for each model. In case of human demo, error denotes the variance among human demonstrations whereas for models, error denotes deviation from human demonstration (see Section IV-C).

Fig. 12: Comparison of movement encoding frameworks on the movement generalisation task when many demonstration trajectories are available. Median error vs. number of kernels is shown for each model.

Fig. 13: Example OCP generalizations by end point shifting alone. The target is given in dark gray, the demonstration in light gray.

C. Hyperparameter Tuning

All the considered models contain hyperparameters. Choosing them optimally improves their performance. To find the optimal hyperparameters for our models, we conducted parameter searches over sensible parameter regions.
on a limited set of 250 randomly chosen actions for each individual number of kernels.

1) **DMP**: To find the optimal \( \kappa \), which determines the kernel variance \( \sigma \) via Equation (1), we considered \( \kappa \) between 0.6 and 1.2 in steps of 0.1. \( \kappa = 0.6 \) corresponds to relatively narrow Gaussian kernels, where the 1\( \sigma \)-intervals of kernels don’t overlap at all. With \( \kappa = 1.2 \), the kernels are much wider such that their 1\( \sigma \)-intervals overlap. The optimal \( \kappa \) for our data is 0.7 for \( K = [3 \ldots 10] \) and 1.1 for \( K > 10 \).

2) **tbGMR**: Choosing the regularization parameter \( \epsilon \) for the tbGMR models is more complicated than in the DMP case. We conducted a parameter search within the range \([0.01, 10^{-7}]\). This is from the same order of magnitude as the data, down to very little regularization that is only good for numerical stability. While the parameter scan clearly showed that it is best to use as little regularization as possible to achieve lowest reconstruction error, the shape of those trajectories was problematic. Different from human demonstrations, those tbGMR reconstructions looked like a piece-wise linear approximation and the velocity profile was qualitatively completely different, and contained oscillations. Therefore, we decided to not use the low regularization for the lowest reconstruction error, because we are comparing the frameworks on human actions and those trajectories were qualitatively completely different.

We estimated the necessary regularization to keep the model’s reconstructions from becoming to different from human demonstrations using two other metrics than only reconstruction error. We observed that there were as many oscillations in the velocity profile as kernels in the model, so we compared the corresponding frequency in the FFT of the velocity profile with the fundamental frequency of the motion. We considered a trajectory smooth enough if the oscillations caused by the kernels were lower than 20\% of the base frequency.

We also checked how far the end-points of the reconstructions are away for the end-points of human demonstrations. We considered 5 mm, our estimated error from the trajectories are away for the end-points of human demonstrations. So, we just made the controller as stiff as possible with a slight focus on position, rather than velocity \((q_1 = 1, q_2 = 0.1, R = 1e^{-12})\), to follow the trajectory representation most accurately. In actual robotic applications, this would have to be tuned depending on the specific hardware.

3) **SEDS**: With SEDS, there are not globally optimal hyperparameters, because of the nature of its nonlinear optimization algorithm. To deal with this, we optimize every individual fit. We start from an estimation obtained by conventional GMM regression with the EM-algorithm, modify it to fulfill the constraints, and then refine it with SEDS optimization. If the optimization fails, we add noise to the initial conditions and start the optimization again. Additionally, after five failed trials, we increase the value of the \( \text{tol_mat_bias} \) value in the SEDS optimizer, that is similar to the \( \epsilon \) of the TP-GMM models and is documented by the authors of the SEDS code as helping with instabilities. We consider an optimization as failed if the numeric solver reports a fail or if the obtained model is unable to reconstruct the first given demonstration trajectory with less than 2 cm distance anywhere along the trajectory or 5 mm at the end point. Note that we do not test the model against the generalization target to judge if the optimization should be repeated, which, unfairly, would use test data for training, but compare against data that was already used in training.

4) **ProMP**: ProMPs showed the same behaviour as the tbGMR approach, with oscillations in the velocity profile, directly related to the number of kernels. The python library we used [44] did not expose any hyperparameters, so we modified the code to apply regularization. Applying the same regularization as with tbGMR (additive constant in EM-Algorithm) did not change the outcome, because this only fits the distribution \( p(w, \theta) \) where the kernel properties are already set. The library internally already had a parameter for the width of the basis functions, that was hard coded to 0.7. We optimized this hyperparameter in the range \([0.75, 0.95]\), as the default of 0.7 was definitely to low and higher values would deteriorate the accuracy to much.

With only very few kernels (3, 4), it proved impossible to get below the 20\% threshold as defined above for tbGMR. But, in contrast to tbGMR, the reconstructions or generalizations with these kernel numbers were not on or below the level of human variation, so one would not use them anyway. Thus, we accept this without attempting any further fixes.

5) **OCP**: Since the shape of the Chebyshev polynomials themselves is fixed and the fit of the Chebyshev series always converges, there were no hyperparameters to fit regarding the trajectory representation. The LQR-controller itself was also easy to tune, because we do not actually need the perturbation resistance in the scenarios considered here. So, we just made the controller as stiff as possible with a slight focus on position, rather than velocity \((q_1 = 1, q_2 = 0.1, R = 1e^{-12})\), to follow the trajectory representation most accurately. In actual robotic applications, this would have to be tuned depending on the specific hardware.