Generalization of Orientational Motion in Unit Quaternion Space

Aljaž Kramberger, Andrej Gams, Bojan Nemec and Aleš Ude*

***Abstract*— A requirement for the humanoid robot's operation in natural environments is that a humanoid robot can effectively adapt to new configurations of the external world. In this paper we address the problem of adaptation, where new robot movements are generated based on data accumulated in related but different situations. Our solution to this challenge is to apply statistical learning, which provides a method to generate robot responses in new situations. Building on our previous work on learning motor primitives [1], we propose a new methodology for task-specific generalization of orientation trajectories, which we encode as Cartesian space Dynamic Movement Primitives. Example trajectories are generalized by applying Locally Weighted Regression in unit quaternion space, using the parameters describing the task as query points into the trajectory database. We show on real-world and simulated tasks that the proposed methodology can be used for statistical learning of orientation trajectories.**

I. INTRODUCTION

Statistical learning has extensively been used in robotics to synthesize humanoid robot trajectories, appropriate for a new task within the training space, from a set of recorded movements [1]. Methods such as locally weighted regression (LWR) [2] and Gaussian process regression (GPR) [3] have been previously applied for generalization of joint space trajectories in the framework of dynamic movement primitives [1]. However, planning of trajectories in Cartesian space does not allow direct application of these methods, because there exists no minimal, singularity-free representation of orientation [4]. For example, the quaternion representation as a singularity free but non-minimal representation of orientation with only 4 parameters (compared to 9 parameters of rotation matrices) fulfils an additional constraint in 4-D space (unit norm) to describe a manifold of all orientations. Such constraints are not taken into account by general learning methods, and thus they are not preserved when learning from parameters that are constrained to a specific manifold.

In this paper we show how to apply the aforementioned LWR for generalization of orientation trajectories in such a way that that the generalized trajectory is guaranteed to lie on the orientation constraint manifold. We use Cartesian space DMPs – CDMPs originally proposed by Ude et al. [4] – as the underlying representation for orientation trajectories. This approach integrates the equations of motion directly on the orientation manifold and is therefore guaranteed to generate

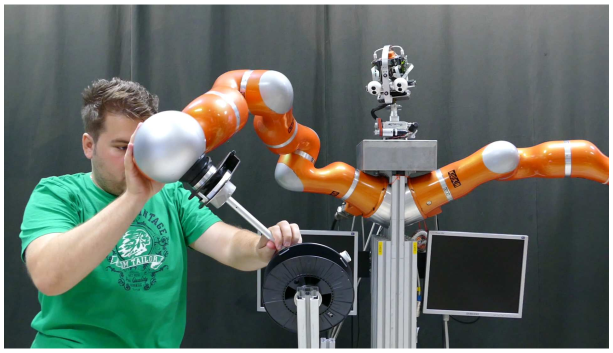
Fig. 1. Operator teaching an example movement by physically guiding the humanoid robot arm along the desired trajectory. The demonstrated motion is recorded in Cartesian space.

quaternions with unit norm. A related approach for learning orientation trajectories was proposed by Pastor et al. [5].

The use-case example in this paper focuses on the generalization of orientation trajectories, acquired through kinesthetic guiding, as shown in Fig. 1.

This paper is organized as follows. After the related work in Section II, we review the details of the Cartesian space DMPs and the required mathematical tools in Section III. LWR based generalization is presented in Section IV. The experimental evaluation follows in Section V, with conclusions given in Section VI.

II. RELATED WORK

Two main topics characterize related work: trajectory representation and generalization methods. The choice of one typically also affects the other.

Several statistical methods were successfully applied in robotics within the framework of dynamic movement primitives (DMP) [6], for example locally weighted regression (LWR) [1], locally weighted projection regression (LWPR) [7], and Gaussian process regression (GPR) [3]. Furthermore, generation of new control policies from existing knowledge, i. e. generalization, was demonstrated by Matsubara et al. [8], showing extended scalability of DMPs. Mixture of motor primitives was used for generation of table tennis swings in [9]. Generalization of DMPs was also combined with model predictive control by Krug and Dimitrov [10]. Similarly, Stulp et al. proposed learning a function approximator with one regression in the full space of phase and tasks parameters, bypassing the need for two consecutive regressions [11]. Generalization using GPR was applied over combined joint position torque trajectories in the framework of compliant movement primitives [12], extending the DMP framework beyond the kinematic trajectory properties. Generalization was also applied to DMP coupling terms [13], which were

*This work was supported by EU Horizon 2020 Programme grant 680431, ReconCell, and Slovenian Research Agency grant J2-7360.

All authors are with the Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. aljaz.kramberger@ijs.si

Published in the proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids), Cancun, Mexico, pp. 808-813, 2016.

learned and later added to a demonstrated trajectory to generate new joint-space trajectories.

DMPs were not the only trajectory representation used in generalization. For example [14] used Gaussian mixture models. These were extensively applied by Calinon for generalization, with an excellent tutorial and an overview provided in [15]. A collection of task-parameterized models provides a representation of movement / behavior that can adapt to a set of task parameters describing the current situation encountered by a robot, such as location of objects or landmarks in its workspace. These task parameters can be interpreted as queries for generalization as used by the approach presented in this paper. We relate the reader to [15] for a broader review of generalization methods from a set of learned robot movements.

III. Cartesian space DMPs – CDMPs

For the sake of clarity and completeness of the paper, we first review the orientation part of Cartesian space DMPs, which uses unit quaternions, and the required mathematical formulation for the generalization. We relate the reader to [4] for a more complete description of CDMPs.

A CDMP has the following free parameters: weights $\boldsymbol{w}_i^p, \boldsymbol{w}_i^o \in \mathbb{R}^3, i = 1,\ldots,N$, where N is the number of basis functions used to approximate the movement, trajectory duration τ and the final position $\boldsymbol{g}^p$ and orientation $\boldsymbol{g}^o$ of the robot. The orientation is represented as the unit quarternion $\boldsymbol{q} = v + \boldsymbol{u} \in \mathrm{S}^3$, where S^3 is a unit sphere in $\mathbb{R}^4$, with the scalar part $v \in \mathbb{R}$ and the vector part $\boldsymbol{u} \in \mathbb{R}^3$. In this paper we use the quaternion CDMP formulation to encode the orientations, given by

$$\begin{aligned} \tau\dot{\boldsymbol{\eta}} &= \alpha_z\left(\beta_z 2\log\left(\boldsymbol{g}_o * \bar{\boldsymbol{q}}\right) - \boldsymbol{\eta}\right) + \mathbf{f}_o(x), && (1)\\ \tau\dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{\eta} * \boldsymbol{q}, && (2)\\ \tau\dot{x} &= -\alpha x. && (3) \end{aligned}$$

The nonlinear forcing term of the CDMP $\mathbf{f}_o$ is defined as

$$\mathbf{f}_o(x) = \boldsymbol{D}_o \frac{\sum_{i=1}^N \boldsymbol{w}_i^o \Psi_i(x)}{\sum_{i=1}^N \Psi_i(x)} x, \qquad (4)$$

and contains free parameters $\boldsymbol{w}_i^o \in \mathbb{R}^3$, which need to be determined to encode any given orientation trajectory $\{\boldsymbol{q}_j, \boldsymbol{\omega}_j, \dot{\boldsymbol{\omega}}_j, t_j\}_{j=1}^T$. $\boldsymbol{D}_o \in \mathbb{R}^{3\times 3}$ is a scaling matrix that can be set to $\boldsymbol{D}_o = \boldsymbol{I}$, but see [4] for other possibilities. The Gaussian kernel functions are defined by

$$\Psi_i(x) = \exp\left(-h_i\left(x - c_i\right)^2\right), \qquad (5)$$

where c_i are their centers, distributed along the phase of the movement and h_i their widths. For a given N and setting the time constant τ equal to the total duration of the desired movement, we can define $c_i = \exp\left(-\alpha_x \frac{i-1}{N-1}\right)$, $h_i = \dfrac{1}{(c_{i+1} - c_i)^2}$, $h_N = h_{N-1}$, $i = 1,\ldots,N$. For each Cartesian degree of freedom, the weights w_i should be adjusted so that the desired behavior is achieved. Goal orientation and time duration are given as $\boldsymbol{g}_o = \boldsymbol{q}_T$ and $\tau = t_T - t_1$, respectively.

Equation (2) is derived from the equation that connects quaternion derivative $\dot{\boldsymbol{q}}(t)$ and angular velocity $\boldsymbol{\omega}(t)$. This relation is given by

$$\dot{\boldsymbol{q}} = \frac{1}{2}(0, \boldsymbol{\omega}) * \boldsymbol{q}. \qquad (6)$$

From (2) and (6) we obtain $\boldsymbol{\eta} = \tau\boldsymbol{\omega}$. Note that in (2), $\boldsymbol{\eta} \in \mathbb{R}^3$ is treated as quaternion with 0 scalar part. Conjugation of quaternions is denoted by bar and defined as $\bar{\boldsymbol{q}} = \overline{v + \boldsymbol{u}} = v - \boldsymbol{u}$. The asterisk $*$ denotes the quaternion multiplication on S^3, defined as

$$\begin{aligned} \boldsymbol{q}_1 * \boldsymbol{q}_2 &= (v_1 + \boldsymbol{u}_1) * (v_2 + \boldsymbol{u}_2) && (7)\\ &= (v_1 v_2 - \boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{u}_2) + (v_1 \boldsymbol{u}_2 + v_2 \boldsymbol{u}_1 + \boldsymbol{u}_1 \times \boldsymbol{u}_2). \end{aligned}$$

The above product of two unit quaternions is always another unit quaternion.

The quaternion logarithm $\log : \mathrm{S}^3 \mapsto \mathbb{R}^3$, which is one of the operations in (1), is defined as

$$\log(\boldsymbol{q}) = \log(v + \boldsymbol{u}) = \begin{cases} \arccos(v)\dfrac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, & \boldsymbol{u} \neq 0 \\ [0,0,0]^{\mathrm{T}}, & \text{otherwise} \end{cases}. \qquad (8)$$

The quaternion logarithm $\log(\boldsymbol{q}_2 * \bar{\boldsymbol{q}}_1)$ can be interpreted as a difference vector between two unit quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$. It can be used to define a distance metrics on S^3

$$\mathrm{d}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \begin{cases} 2\pi, & \boldsymbol{q}_2 * \bar{\boldsymbol{q}}_1 = -1 + [0,0,0]^{\mathrm{T}} \\ 2\|\log(\boldsymbol{q}_2 * \bar{\boldsymbol{q}}_1)\|, & \text{otherwise} \end{cases} \qquad (9)$$

The logarithmic map (8) is bijective if we limit its domain to $\mathrm{S}^3 / \{(-1, [0,0,0]^{\mathrm{T}})\}$. Its inverse, the exponential map $\exp : \mathbb{R}^3 \mapsto \mathrm{S}^3$, is defined as

$$\exp(\boldsymbol{r}) = \begin{cases} \cos\left(\|\boldsymbol{r}\|\right) + \sin\left(\|\boldsymbol{r}\|\right)\dfrac{\boldsymbol{r}}{\|\boldsymbol{r}\|}, & \boldsymbol{r} \neq 0 \\ 0, & \text{otherwise} \end{cases} \qquad (10)$$

If we limit the domain of the exponential map to $\|\boldsymbol{r}\| < \pi$ and of the logarithmic map to $\mathrm{S}^3 / \{(-1, [0,0,0]^{\mathrm{T}})\}$, then both mappings become one-to-one, continuously differentiable and inverse to each other.

Lets represent an orientation trajectory as a time dependent unit quaternion function $\boldsymbol{q}(t)$. Given the starting orientation $\boldsymbol{q}(t)$ and angular velocity $\boldsymbol{\omega}(t)$ (assumed constant on time interval $[t, t + \Delta t]$), we can calculate the orientation at the next integration time $t + \Delta t$ as follows

$$\boldsymbol{q}(t + \Delta t) = \exp\left(\frac{\Delta t}{2}\boldsymbol{\omega}\right) * \boldsymbol{q}(t) = \exp\left(\frac{\Delta t}{2\tau}\boldsymbol{\eta}\right) * \boldsymbol{q}(t). \qquad (11)$$

This formula is used to numerically integrate Eq. (2).

In the next section we show how we can exploit this representation for orientation trajectories when generalizing CDMPs.

IV. GENERALIZATION

In this paper we focus on generalization of orientation trajectories recorded as unit quaternions and encoded as DMPs, through locally weighted regression (LWR). The challenge is to ensure that the generalization method acts in the correct space and preserves the norm of unit quaternions.

A. Database for Generalization

The database is composed of the data recorded with kinesthetic guiding [16], with the collection for the presented use case shown in Fig. 1. The acquired movement trajectories are recorded in Cartesian space. To record the database, the operater grabs the robotic arm and guides it along the desired trajectory.

The following data is recorded: Cartesian space positions and orientations, their first and second derivatives (linear/angular velocities and accelerations)

$$\mathscr{A}_d = \{\boldsymbol{p}_{ij}, \boldsymbol{q}_{ij}, \dot{\boldsymbol{p}}_{ij}, \boldsymbol{\omega}_{ij}, \ddot{\boldsymbol{p}}_{ij}, \dot{\boldsymbol{\omega}}_{ij}, t_{ij}\}_{i=1,j=1,}^{\mathrm{NumEx},\ T_i}, \tag{12}$$

all recorded at times t_{ij}, $j = 0, ..., T_i$. T_i is the number of samples in the i-th recording, with $i = 1, \dots, \mathrm{NumEx}$, denoting the index of the recording, and NumEx the number of recordings (example trajectories in the library). Orientations are recorded as unit quaternions $\boldsymbol{q} = v + \boldsymbol{u} \in \mathbb{R}^4$. An external condition of the task (query) is recorded with each demonstration,

$$\mathscr{S}_d = \{\boldsymbol{s}_i\}_{i=1}^{\mathrm{NumEx}}. \tag{13}$$

In the real robot experiment presented in Section V, the external condition of the task $\boldsymbol{s}_i$ was one dimensional and equal to the starting angle ϑ_i of the valve turning motion. However, in general the dimension of the query point can be higher than one.

The generalization process applies LWR to compute new trajectories, using the data associated with query points $\boldsymbol{s}_i$ close to the desired query point $\boldsymbol{s}$

$$\mathbf{G}(\mathscr{A}_d, \mathscr{S}_d) : \boldsymbol{s} \to [\boldsymbol{w}^{p\mathrm{T}},\ \boldsymbol{w}^{o\mathrm{T}},\ \tau,\ \boldsymbol{g}^{p\mathrm{T}},\ \boldsymbol{g}^{o\mathrm{T}}]^{\mathrm{T}}. \tag{14}$$

Here $\mathbf{G}(\mathscr{A}_d, \mathscr{S}_d)$ is the generalization function that maps a given query point $\boldsymbol{s}$ into the new position $\boldsymbol{w}^p$, τ, $\boldsymbol{g}^p$, and orientation $\boldsymbol{w}^o$, τ, $\boldsymbol{g}^o$, DMP parameters, separately for the position (using the method described in [1]) and orientation part (see below). Here $\boldsymbol{w}_p = [\boldsymbol{w}_1^{p\mathrm{T}}, \dots, \boldsymbol{w}_N^{p\mathrm{T}}]^{\mathrm{T}}$ and $\boldsymbol{w}_o = [\boldsymbol{w}_1^{o\mathrm{T}}, \dots, \boldsymbol{w}_N^{o\mathrm{T}}]^{\mathrm{T}}$. Note that the generalized τ is shared among position and orientation part of the trajectories.

B. Generalization of Shape Parameters using LWR

Locally weighted regression (LWR) [2] is a non-parametric method for statistical approximation, which uses raw data stored in memory to determine its output. This approach was used for generalization of joint position trajectories in [1], where it was applied to generalize throwing, reaching, and drumming movements. In contrast to our previous work, we here focus on generalization of *orientation*, which was not solved before.

As explained in [1], the generalization of position trajectories can take place separately for each dimension. Therefore, three instances of LWR are executed to generalize to the new query for each dimension. This cannot be analogously applied to the generalization of unit quaternion trajectories because the resulting generalized orientation will not preserve the norm, and will require normalization. This reduces the quality of approximation.

To apply locally weighted regression to unit quaternion trajectories, we designed a new procedure that generalizes among the differences between orientation trajectories as defined by Eq. (19) instead of quaternion trajectories (12) directly. The main advantage of this approach is that the difference trajectories are not constrained to unit norm as the quaternion trajectories are. Once a new query point $\boldsymbol{s}$ has been received, we search for the closest query point in the training data

$$k = \arg\min_i \{\|\boldsymbol{s} - \boldsymbol{s}_i\|\}. \tag{15}$$

We then calculate the unit quaternion DMP $\mathbf{q}_k^{\mathrm{DMP}}$ for orientation trajectory closest to the given query point $\boldsymbol{s}$, i. e.

$$\{\boldsymbol{q}_{kj}, \boldsymbol{\omega}_{kj}, \dot{\boldsymbol{\omega}}_{kj}, t_{kj}\}_{j=1}^{T_k}. \tag{16}$$

Lets select the tricube weighting kernel K [2] for locally weighted regression

$$\mathrm{K}(\boldsymbol{s}, \boldsymbol{s}') = \begin{cases} (1 - (\|\boldsymbol{s} - \boldsymbol{s}'\|/h)^3)^3 & \text{if } \|\boldsymbol{s} - \boldsymbol{s}'\|/h < 1 \\ 0 & \text{otherwise} \end{cases}. \tag{17}$$

Only training data with query points $\boldsymbol{s}_i$, for which the distance to the current query $\boldsymbol{s}$ is smaller than h, are considered when performing locally weighted regression as $\mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i)$ is equal to zero for all other i. We now compute a new training data set

$$\mathscr{A}_d' = \{\boldsymbol{r}_{ij}, t_{ij}\}_{i=1,\ j=1}^{\mathrm{NumEx},\ T_i}, \tag{18}$$

where

$$\boldsymbol{r}_{ij} = \log\left(\boldsymbol{q}_{ij} * \overline{\boldsymbol{q}}_k^{\mathrm{DMP}}(x_{ij})\right) \in \mathbb{R}^3, \tag{19}$$

$$x_{ij} = \exp\left(-\frac{\alpha_x}{\tau_i} t_{ij}\right). \tag{20}$$

Note that $\tau_i = t_{iT_i} - t_{i1}$. More importantly, unlike the unit quaternion data $\boldsymbol{q}_{ij}$, the difference vectors are unconstrained, $\boldsymbol{r}_{ij} \in \mathbb{R}^3$. Hence locally weighted regression can be applied to these data. Another important point is that data in (19) correspond to a rotation vector representation of orientation, also called exponential coordinates [17], which as every minimal, i. e. 3-D representation, contains singularities. However, as explained below these singularities are not a problem because locally weighted regression uses only nearby data to calculate a new orientation trajectory.

Instead of directly computing the orientation trajectory $\mathbf{q}^{\mathrm{DMP}}$ associated with query point $\boldsymbol{s}$, we first compute the generalized difference trajectory $\mathbf{r}$ associated with that query point. Lets write this difference trajectory as a linear combination of radial basis functions (RBF)

$$\mathbf{r}(x) = \sum_{i=1}^{N} \frac{\boldsymbol{v}_i \Psi_i(x)}{\sum_{j=1}^{N} \Psi_j(x)} x. \tag{21}$$

For each dimension l of $\boldsymbol{r}$ and $\boldsymbol{v}$, $l = 1,2,3$, the application of locally weighted regression results in the following least-squares optimization problem

$$\min_{\boldsymbol{v}^l} \sum_{i=1}^{NumEx} \|\boldsymbol{X}_i \boldsymbol{v}^l - \boldsymbol{r}_i^l\|^2 \mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i), \tag{22}$$

where

$$\boldsymbol{r}_i^l = \left[\begin{array}{ccc} r_{i1}^l & \dots & r_{iT_i}^l \end{array}\right]^{\mathrm{T}}, \tag{23}$$

$$\boldsymbol{v}^l = \left[\begin{array}{ccc} v_1^l & \dots & v_N^l \end{array}\right]^{\mathrm{T}}, \tag{24}$$

$$\boldsymbol{X}_i = \left[\begin{array}{ccc} \frac{\Psi_1(x_{i1})}{\sum_{j=1}^N \Psi_j(x_{i1})} x_{i1} & \cdots & \frac{\psi_N(x_{i1})}{\sum_{j=1}^N \psi_k(x_{i1})} x_{i1} \\ \vdots & \ddots & \vdots \\ \frac{\psi_1(x_{iT_i})}{\sum_{j=1}^N \Psi_j(x_{iT_i})} x_{iT_i} & \cdots & \frac{\psi_N(x_{iT_i})}{\sum_{j=1}^N \Psi_j(x_{iT_i})} x_{iT_i} \end{array}\right] \tag{25}$$

Note that in the above LWR formulation only those i for which $\mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i) > 0$ affect the result. This is important because the orientation representation with rotation vector $\boldsymbol{r}$ is minimal and therefore contains singularities. But since $\mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i) > 0$ is true only in the neighbourhood of $\boldsymbol{s}$, the relevant difference vectors $\boldsymbol{r}_{ij}$ remain small, assuming that the orientation trajectories in (12) smoothly transition between each other. Since the rotation vector representation contains no singularities in the neighborhood of $\boldsymbol{r} = 0$, the optimization problem (22) avoids any critical areas where the rotation vector representation becomes discontinuous. Thus the optimization problem (22) remains well defined.

To control the orientation of the robot, we need to calculate a Cartesian space DMP. This is accomplished by transforming the optimal rotation trajectory (21) back to quaternion representation using the formula

$$\mathbf{q}^{\mathrm{DMP}}(x) = \exp(\mathbf{r}(x)) * \mathbf{q}_k^{\mathrm{DMP}}(x). \tag{26}$$

While it is possible to use this formula directly to control a robot, it is better to sample the resulting orientation trajectory, typically at robot servo rate, and compute new shape parameters $\boldsymbol{w}^o$ from the sampled data using Cartesian DMP estimation techniques developed in [4]. For these calculations we first need to generalize duration τ and goal orientation $\boldsymbol{g}^o$, which we do using methods described in Section IV-C. This way we can exploit all advantages of DMPs, which is not possible with representation (26).

C. *Generalization of Goal Orientation and Time Duration*

The goal positions and orientations as well as durations are directly available in the data. They are given as

$$\boldsymbol{g}_i^p = \boldsymbol{p}_{iT_i},\ \boldsymbol{g}_i^o = \boldsymbol{q}_{iT_i},\ \tau_i = t_{iT_i} - t_{i1},\ i = 1, \dots, NumEx. \tag{27}$$

Given a new query point $\boldsymbol{s}$ and using LWR, the goal position and time duration can be generalized as follows

$$\boldsymbol{g}^p = \sum_{i=1}^{NumEx} \frac{\mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i) \boldsymbol{g}_i^p}{\sum_{j=1}^{NumEx} \mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_j)}, \tag{28}$$

$$\tau = \sum_{i=1}^{NumEx} \frac{\mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i) \tau_i}{\sum_{j=1}^{NumEx} \mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_j)}. \tag{29}$$

Since we cannot add unit quaternions, a different method has to be used to generalize goal orientations. One possibility is to solve the following optimization problem

$$\min_{\boldsymbol{g}^o \in \mathrm{S}^3} \sum_{i=1}^{NumEx} \mathrm{d}(\boldsymbol{g}^o, \boldsymbol{g}_i^o) \mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i), \tag{30}$$

where d is a metric on a 4-D unit sphere defined in (9). Optimization problem (30) is nonlinear and can be solved using iterative methods such as Newton's method. The iteration process can be initialized with the following approximation

$$\boldsymbol{q}_0 = \frac{\sum_{i=1}^{NumEx} \mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i) \boldsymbol{g}_i^o}{\| \sum_{i=1}^{NumEx} \mathrm{K}(\boldsymbol{s}, \boldsymbol{s}_i) \boldsymbol{g}_i^o \|}. \tag{31}$$

V. EXPERIMENTAL EVALUATION

A. *In Simulation*

We first tested the proposed method in simulation. For the database of orientation trajectories, we synthesized an example set of 11 minimum jerk SLERP trajectories, depicted with blue in Fig. 2. The synthetic trajectories all start at the same orientation and finish at different ones. The trajectories transition smoothly between each other with an even distribution. The final orientation values of the trajectories were used as query points $\boldsymbol{s}$.

For generalization we first calculated the set of rotation differences vectors (18), shown in blue in Fig. 3. These data are used to compute generalized difference trajectories, shown in red in Fig. 3. These are then used to obtain the generalized orientation trajectories using formula (26), which guarantees to generate unit quaternions. The final outcome of generalization is shown in red in Fig. 2.

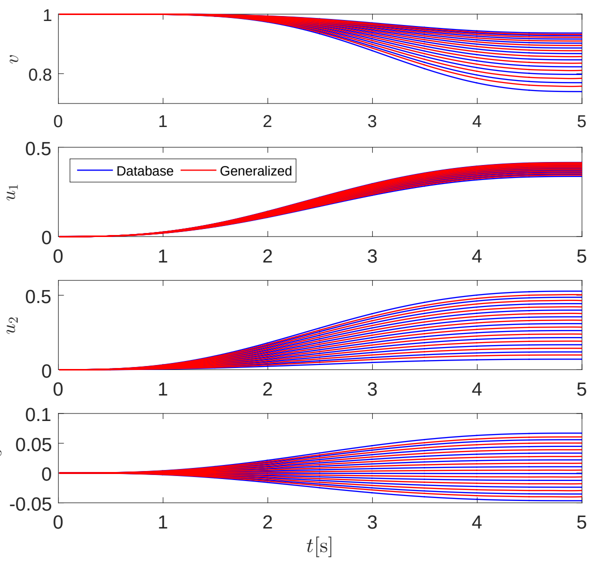

Fig. 2. Synthesized database for generalization between minimum jerk SLERP trajectories, represented as quaternions $\boldsymbol{q} = v + \boldsymbol{u}$. The simulated trajectories are shown in blue, while the resulting LWR generalized trajectories are red.

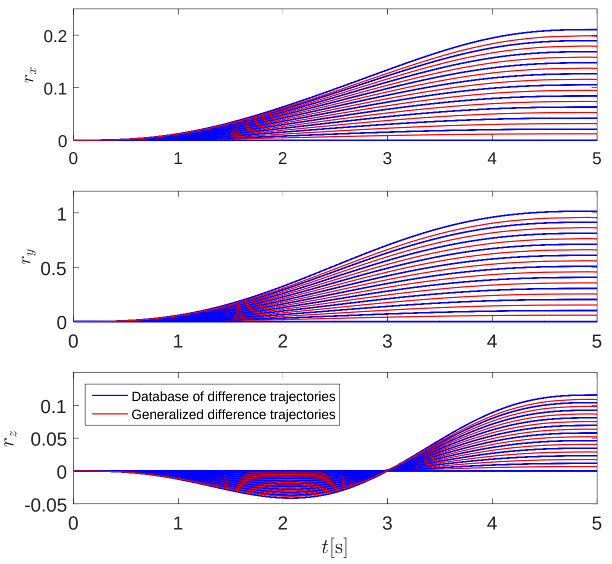

Fig. 3. Generalization of rotation difference vectors calculated from example trajectories in blue. Red color represents the generalized rotation difference vectors, which are used to calculate the new generalized orientation trajectories.

Using (9) we calculated the differences between the LWR generalized orientation trajectories depicted in Fig. 2 and the corresponding minimum jerk SLERP trajectories at the same queries. The results are shown in Fig. 4. The differences for the two trajectories at the edge of the database are shown in red and green, and the other 8 differences in blue. All the differences are very small, but generalization close to either edge of the database increases the error, in this particular case for an order of magnitude. This is consistent with what was reported for joint trajectories [1].

B. On a Real System

After the initial results from simulation turned out to be promising, we carried out experiments to confirm the performance of the proposed approach on a real robot. The experiment was performed with a 7 DOF Kuka LWR-4 robot, where the challenge was to turn a valve, shown in Fig. 5, from any starting angle (inside the work space of the robot). An additional challenge was to generalize human demonstrated trajectories which are not as refined and smooth as the synthesized ones.

5 example movements were demonstrated with kinesthetic

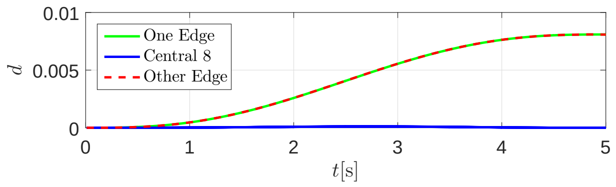

Fig. 4. The difference between the LWR generalized trajectories and the corresponding minimum jerk SLERP trajectories at the same queries, calculated using (9). The errors of the two trajectories generalized close to the edge of the database are depicted in green and red, while all other 8 are depicted in blue.

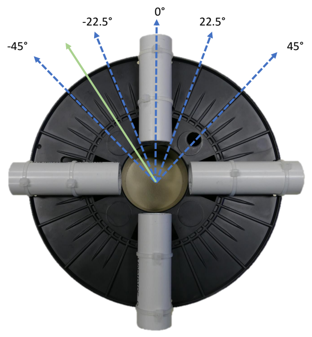

Fig. 5. Valve for testing the proposed method on a real system. Blue color indicates the demonstrated example queries (angles), while the green arrow shows an example query for generalization.

guiding, as shown in Fig. 1. Each started at a different starting angle of the valve and this angle was used as a query. The database query values were given as $\mathscr{S}_d = \{-45°, -22.5°, 0°, 22.5°, 45°\}$. The database query points, depicted in the plane of the valve, are shown in blue in Fig. 5. All trajectories ended in the same configuration of the robot. Note that the plane of the valve was at an arbitrary angle towards the robot. See also the accompanying video, which depicts both the database generation and the execution of the generalized trajectories.

The recorded database of movements, shown in blue in Fig. 6, was used to generalize to new movements according to the rotation angle of the valve (query), which was determined with a rotation sensor. The results, shown in red in Fig. 6, demonstrate successful generalization with LWR for query points $\boldsymbol{s}_i = \{-40°, -30°, -20°, -10°, 0°, 5°, 25°, 35°\}$. Fig. 8 shows the database (blue) and the generalized (red) rotation difference vectors. The real-world generalization results are comparable to simulation results. To generate the real robot motion, we also had to generalize position trajectories using the method proposed in [1]. The generalized position trajectories are depicted in Fig. 9. As evident from the attached video and Fig. Fig. 7, the robot was able to successfully

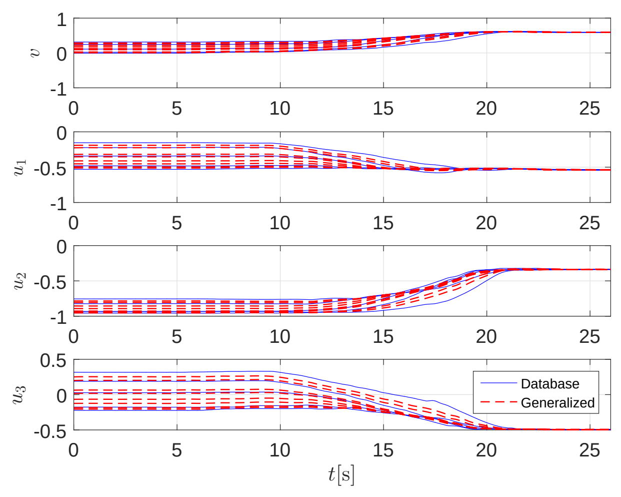

Fig. 6. The database of real orientation trajectories (blue) and the generalized trajectories (red).

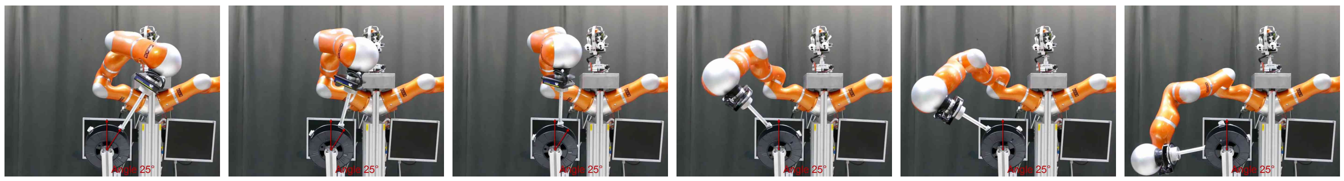

Fig. 7. Still images of the humanoid robot performing a valve-turning motion for the starting angle at 25°.

perform the task using the proposed approach.

VI. Conclusion

In this work we proposed a new approach for statistical generalization of orientation trajectories represented as unit quaternion trajectories [4]. The proposed approach complements the methods for task-specific generalization of position DMPs, which were initially proposed by Ude et al. [1]. The proposed method supports generic task descriptors, which are used as query points into a database of orientation trajectories. Locally weighted regression was used for generating new orientation trajectories. In LWR nearby data points are given higher weights than the distant ones, providing for good generalization performance. The proposed approach resolves the problem of maintaining the norm of unit quaternions after generalization.

We tested the developed methods both in simulation and for learning real humanoid robot trajectories, proving its efficiency. As future work we plan to extend the proposed methods for generalization of Cartesian space trajectories to movements that involve contact with the environment.

References

[1] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally Weighted Learning," *Artificial Intelligence Review*, vol. 11, pp. 11–73, 1997.

[3] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: The MIT Press, 2006.

[4] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in Cartesian space dynamic movement primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Hong Kong), pp. 2997–3004, 2014.

[5] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (San Francisco, California), pp. 365–371, 2011.

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

[7] S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, and S. Schaal, "Statistical Learning for Humanoid Robots," *Autonomous Robots*, vol. 12, no. 1, pp. 55–69, 2002.

[8] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[9] K. Muelling, J. Kober, and J. Peters, "Learning table tennis with a mixture of motor primitives," in *2010 10th IEEE-RAS International Conference on Humanoid Robots*, (Nashville, TN), pp. 411–416, 2010.

[10] R. Krug and D. Dimitrov, "Model predictive motion control based on generalized dynamical movement primitives," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 1, pp. 17–35, 2015.

[11] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud, "Learning compact parameterized skills with a single regression," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, (Atlanta, Georgia), pp. 417–422, 2013.

[12] M. Deniša, A. Gams, A. Ude, and T. Petrič, "Learning compliant movement primitives through demonstration and statistical generalization," *IEEE/ASME Transactions on Mechatronics*, vol. PP, 2016.

[13] A. Gams, M. Deniša, and A. Ude, "Learning of parametric coupling terms for robot-environment interaction," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 304–309, 2015.

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

[15] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2015.

[16] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.

[17] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

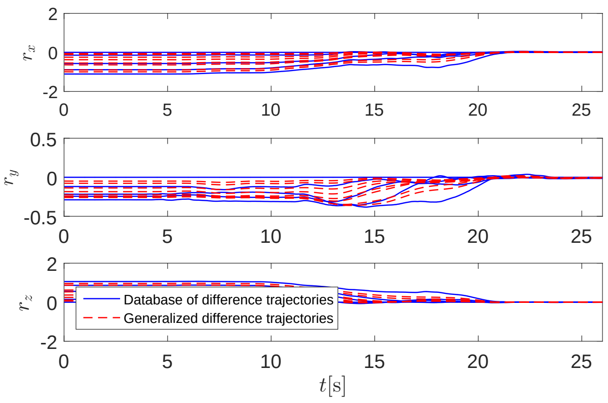

Fig. 8. Generalization of rotation difference vectors calculated from example trajectories (shown in blue). Red color represents the generalized rotation difference vectors, which were used to calculate the new generalized orientation trajectories.

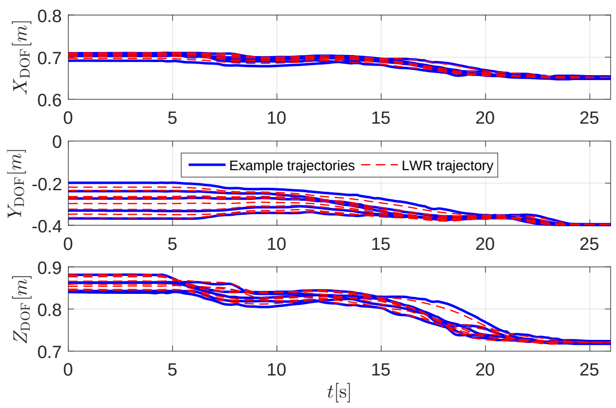

Fig. 9. Database of real position trajectories and the generalized position trajectories, which complement the orientation trajectories in Fig. 6.