Article:

Huang, Y, Abu-Dakka, FJ, Silvério, J et al. (1 more author) (2021) Towards Orientation Learning and Adaptation in Cartesian Space. IEEE Transactions on Robotics, 37 (1). pp. 82-98. ISSN 1552-3098

https://doi.org/10.1109/TRO.2020.3010633

Towards Orientation Learning and Adaptation in Cartesian Space

Yanlong Huang, Fares J. Abu-Dakka, João Silvério, and Darwin G. Caldwell

Abstract—**As a promising branch of robotics, imitation learning emerges as an important way to transfer human skills to robots, where human demonstrations represented in Cartesian or joint spaces are utilized to estimate task/skill models that can be subsequently generalized to new situations. While learning Cartesian positions suffices for many applications, the end-effector orientation is required in many others. Despite recent advances in learning orientations from demonstrations, several crucial issues have not been adequately addressed yet. For instance, how can demonstrated orientations be adapted to pass through arbitrary desired points that comprise orientations and angular velocities? In this paper, we propose an approach that is capable of learning multiple orientation trajectories and adapting learned orientation skills to new situations (e.g., via-points and end-points), where both orientation and angular velocity are considered. Specifically, we introduce a kernelized treatment to alleviate explicit basis functions when learning orientations, which allows for learning orientation trajectories associated with high-dimensional inputs. In addition, we extend our approach to the learning of quaternions with angular acceleration or jerk constraints, which allows for generating smoother orientation profiles for robots. Several examples including experiments with real 7-DoF robot arms are provided to verify the effectiveness of our method.**

Index Terms—**imitation learning, orientation learning, generalization, human-robot collaboration.**

I. Introduction

In many challenging tasks (e.g., robot table tennis [2] and bimanual manipulation [3]), it is non-trivial to manually define proper trajectories for robots beforehand, hence imitation learning is suggested in order to facilitate the transfer of human skills to robots [4]. The basic idea of imitation learning is to model consistent or important motion patterns that underlie human skills and, subsequently, employ these patterns in new situations. A myriad of imitation learning techniques have been reported in the past few years, such as dynamic movement primitives (DMP) [5], probabilistic movement primitives (ProMP) [6], task-parameterized Gaussian mixture model (TP-GMM) [7] and kernelized movement primitives (KMP) [8].

Yanlong Huang is with School of Computing, University of Leeds, Leeds LS29JT, UK (e-mail: y.l.huang@leeds.ac.uk).

Fares J. Abu-Dakka is with Intelligent Robotics Group at the Department of Electrical Engineering and Automation, Aalto University, Finland (e-mail: fares.abu-dakka@aalto.fi).

João Silvério is with Idiap Research Institute, CH-1920 Martigny, Switzerland (e-mail: joao.silverio@idiap.ch).

Darwin G. Caldwell is with Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163 Genoa, Italy (e-mail: darwin.caldwell@iit.it).

Preliminary results have been presented in IEEE International Conference on Robotics and Automation [1].

While the aforementioned skill learning approaches have been proven effective for robot trajectory generation at the level of Cartesian positions and joint angles [9], [10], [11], learning of orientation in task space still imposes great challenges. Unlike position operations in Euclidean space, orientation is accompanied by additional constraints, e.g., the unit norm of the quaternion representation or the orthogonal constraint of rotation matrices. In many previous work, quaternion trajectories are learned and adapted without considering the unit norm constraint (e.g., orientation TP-GMM [3] and DMP [12]), leading to improper quaternions and hence requiring an additional renormalization.

Instead of learning quaternions in Euclidean space, a few approaches that comply with orientation constraints have been proposed. One representative type of approach is built on DMP [13], [14], [15], where unit quaternions were used to represent orientation and different reformulations of DMP were developed to ensure proper quaternions over the course of orientation adaptation. However, [13], [14], [15] can only adapt quaternions towards a desired target with zero angular velocity as a consequence of the spring-damper dynamics inherited from the original DMP.

Another solution of learning orientation was proposed in [16], where GMM was employed to model the distribution of quaternion displacements so as to avoid the quaternion constraint. However, this approach only focuses on orientation reproduction without addressing the adaptation issue. In contrast to [16] that learns quaternion displacements, the Riemannian topology of the $\mathbb{S}^3$ manifold was exploited in [17] to probabilistically encode and reproduce distributions of quaternions. Moreover, an extension to task-parameterized movements was provided in [17], which allows for adapting orientation tasks to different initial and final orientations. However, adaptation to orientation via-points and angular velocities is not provided.

In addition to the above-mentioned issues, learning orientations associated with high-dimensional inputs is important. For example, in a human-robot collaboration scenario, the robot end-effector orientation is often required to react promptly and properly according to the user's state (e.g., hand poses). More specifically, the robot might need to adapt its orientation in accordance to dynamic environments. The results [12], [13], [14], [15] are built on time-driven[1] DMP, and hence it is non-straightforward to extend these works to learn demonstrations

[1]Despite that time is often transformed into a phase variable in DMP, we will refer to DMP as a time-driven approach, since time and phase are 1-dimensional and their mapping is bijective, which make them be equivalent in our argument.

TABLE I: Comparison Among the State-of-the-Art and Our Approach

	Probabilistic	Unit norm	Via-quaternion	Via-angular velocity	End-quaternion	End-angular velocity	Angular acc (or jerk) constraints	Multiple inputs†
Silvério *et al.* [3]	✓	-	-	-	✓	-	-	✓
Pastor *et al.* [13]	-	✓	-	-	✓	-*	-	-
Ude *et al.* [14]	-	✓	-	-	✓	-*	-	-
Abu-Dakka *et al.* [15]	-	✓	-	-	✓	-*	-	-
Kim *et al.* [16]	✓	✓	-	-	-	-	-	✓
Zeestraten *et al.* [17]	✓	✓	-	-	✓	-	-	✓
Saveriano *et al.* [18]	-	✓	✓	✓	✓	✓	-	-
Kramberger *et al.* [19]	-	✓	-	-	✓	-*	-	-‡
Our approach	✓	✓	✓	✓	✓	✓	✓	✓

* In these works, primitives end with zero angular velocity, i.e., one can not set a desired non-zero velocity.
† The multiple inputs are parts of the demonstrations, which shall be distinguished from external contextual variables. For example, in a human-robot handover task, a typical demonstration consists of a varying human hand trajectory (i.e., high-dimensional inputs) and a robot trajectory (i.e., outputs).
‡ This work considers the generalization of demonstrations towards different external contextual (or conditional) states (which could be high-dimensional). However, the demonstrations are composed of time sequences (i.e., 1-D input) and the corresponding robot trajectories (i.e., outputs).

consisting of high-dimensional varying inputs and deal with tasks where the robot should react to a high number of input variables, e.g., the hand position of a human partner in a collaborative task. In contrast, due to the employment of GMM, learning orientations with multiple inputs is feasible in [3], [16], [17]. However, extending these approaches to tackle adaptations towards via-points associated with multi-dimensional inputs is non-trivial.

While many imitation learning approaches focus on mimicking human demonstrations, the constrained skill learning is often overlooked. As discussed in [20], [21], trajectory smoothness (e.g., acceleration and jerk) will influence robot performance, particularly in time-contact systems (e.g., striking movement in robot table tennis). Thus, it is desirable to incorporate smoothness constraints into the process of learning orientations.

In summary, if we consider the problem of adapting quaternions and angular velocities to pass through arbitrary desired points (e.g., via-point and end-point) while taking into account high-dimensional inputs and smoothness constraints, no previous work in the scope of imitation learning provides an all-encompassing solution.

In this paper, we aim at providing an *analytical* solution that is capable of

(*i*) learning *multiple* quaternion trajectories,
(*ii*) allowing for orientation *adaptations* towards arbitrary desired points that consist of both *unit quaternions and angular velocities*,
(*iii*) coping with orientation learning and adaptations associated with *high-dimensional inputs*,
(*iv*) accounting for *smoothness constraints*.

For the purpose of clear comparison, the main contributions of the state-of-the-art approaches and our approach are summarized in Table I.

This paper is structured as follows. We first illustrate the probabilistic learning of multiple quaternion trajectories and derive our main results in Section II. Subsequently, we extend the obtained results to quaternion adaptations in Section III, as well as quaternion learning and adaptation with angular acceleration (or jerk) constraints in Section IV. After that, we take a typical human-robot collaboration case as an example to show how our approach can be applied to the learning of quaternions along with multiple inputs in Section V. We evaluate our method through several simulated examples (including discrete and rhythmic quaternion profiles) and real experiments (a painting task with time input on Barrett WAM robot and a handover task with a multi-dimensional input on KUKA robot) in Section VI. In Section VII, we discuss the related work as well as limitations and possible extensions of our approach. Finally, our work is concluded in Section VIII. Note that this paper has comprehensively extended our previous work [1] in terms of both theoretical parts (e.g., Sections IV and V) and evaluations (e.g., Sections VI-B, VI-C and VI-E).

II. Probabilistic Learning of Quaternion Trajectories

As suggested in [7], [22], the probability distribution of multiple demonstrations often encapsulates important motion features and further facilitates the design of optimal controllers [11], [23], [24], [25]. Nonetheless, the direct probabilistic modeling of quaternion trajectories is intractable as a result of the unit norm constraint. Similarly to [14], [16], [17], we propose to transform quaternions into Euclidean space, which hence enables the probabilistic modeling of transformed trajectories (Section II-A). Then, we exploit the distribution of transformed trajectories using a kernelized approach, whose predictions allow for the retrieval of proper quaternions (Section II-B). We summarize key notations used throughout the paper in Table II.

A. Probabilistic modeling of quaternion trajectories

For the sake of clarity, let us define quaternions $\mathbf{q}_1 = \begin{bmatrix} v_1 \\ \mathbf{u}_1 \end{bmatrix}$ and $\mathbf{q}_2 = \begin{bmatrix} v_2 \\ \mathbf{u}_2 \end{bmatrix}$, where $\mathbf{q}_i \in \mathbb{S}^3$, $v_i \in \mathbb{R}$ and $\mathbf{u}_i \in \mathbb{R}^3$, $i \in \{1, 2\}$. Besides, we write $\bar{\mathbf{q}}_2 = \begin{bmatrix} v_2 \\ -\mathbf{u}_2 \end{bmatrix}$ as the conjugation of $\mathbf{q}_2$ and, $\mathbf{q} = \mathbf{q}_1 * \bar{\mathbf{q}}_2 = \begin{bmatrix} v \\ \mathbf{u} \end{bmatrix}$ as the quaternion product of $\mathbf{q}_1$ and $\bar{\mathbf{q}}_2$. The function $\log(\cdot) : \mathbb{S}^3 \mapsto \mathbb{R}^3$ that can be used to determine the difference vector between $\mathbf{q}_1$ and $\mathbf{q}_2$ is defined as [14]

$$\log(\mathbf{q}_1 * \bar{\mathbf{q}}_2) = \log(\mathbf{q}) = \begin{cases} \arccos(v)\dfrac{\mathbf{u}}{||\mathbf{u}||}, \mathbf{u} \neq \mathbf{0} \\ [0 \;\; 0 \;\; 0]^\top, \qquad otherwise, \end{cases} \quad (1)$$

TABLE II: Description of Key Notations

$\mathbf{q}$, $\bar{\mathbf{q}}$	$\triangleq$	quaternion and its conjugation
$\mathbf{q}_a$	$\triangleq$	auxiliary quaternion
$\boldsymbol{\zeta}$	$\triangleq$	transformed state of quaternion
$\boldsymbol{\omega}$	$\triangleq$	angular velocity
$\mathbf{p}$	$\triangleq$	Cartesian position
C	$\triangleq$	number of Gaussian components in GMM
$\pi_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c$	$\triangleq$	parameters of c–th Gaussian component in GMM, see (3)
$\mathbf{w}$	$\triangleq$	unknown parametric vector
$\boldsymbol{\phi}(t)$, $\boldsymbol{\Theta}(t)$	$\triangleq$	B-dimensional basis function vector and its corresponding expanded matrix, see (10)
$\boldsymbol{\varphi}(t)$, $\boldsymbol{\Omega}(t)$	$\triangleq$	expanded matrices, see (27) and (30)
$k(\cdot,\cdot)$	$\triangleq$	kernel function
$\mathbf{D}_q = \{\{t_{n,m}, \mathbf{q}_{n,m}\}_{n=1}^N\}_{m=1}^M$	$\triangleq$	M demonstrations in terms of time and quaternion, where each demonstration has N datapoints
$\mathbf{D}_\zeta = \{\{t_{n,m}, \boldsymbol{\zeta}_{n,m}, \dot{\boldsymbol{\zeta}}_{n,m}\}_{n=1}^N\}_{m=1}^M$	$\triangleq$	transformed data obtained from $\mathbf{D}_q$, where $\boldsymbol{\zeta}_{n,m} = \log(\mathbf{q}_{n,m} * \bar{\mathbf{q}}_a)$, see (2)
$\mathbf{D}_\eta = \{\{t_{n,m}, \boldsymbol{\eta}_{n,m}\}_{n=1}^N\}_{m=1}^M$	$\triangleq$	compact form of $\mathbf{D}_\zeta$, where $\boldsymbol{\eta}_{n,m} = [\boldsymbol{\zeta}_{n,m}^\top\ \dot{\boldsymbol{\zeta}}_{n,m}^\top]^\top$
$\mathbf{D}_r = \{t_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$	$\triangleq$	probabilistic reference trajectory extracted from $\mathbf{D}_\eta$, with $\hat{\boldsymbol{\eta}}_n \vert t_n \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n)$
$\boldsymbol{\Phi}, \boldsymbol{\Sigma}, \boldsymbol{\mu}$	$\triangleq$	expanded matrices/vectors defined on $\mathbf{D}_r$, see (14)
$\mathbf{k}(t_i, t_j)$	$\triangleq$	expanded kernel matrix, see (16) or (31)
$\tilde{\mathbf{D}}_q = \{\tilde{t}_h, \tilde{\mathbf{q}}_h, \tilde{\boldsymbol{\omega}}_h\}_{h=1}^H$	$\triangleq$	H desired quaternion states
$\tilde{\mathbf{D}}_\zeta = \{\tilde{t}_h, \tilde{\boldsymbol{\zeta}}_h, \dot{\tilde{\boldsymbol{\zeta}}}_h\}_{h=1}^H$	$\triangleq$	transformed states obtained from $\tilde{\mathbf{D}}_q$, see (19) and (23)
$\tilde{\mathbf{D}}_\eta = \{\tilde{t}_h, \tilde{\boldsymbol{\eta}}_h\}_{h=1}^H$	$\triangleq$	compact form of $\tilde{\mathbf{D}}_\zeta$, where $\tilde{\boldsymbol{\eta}} = [\tilde{\boldsymbol{\zeta}}^\top\ \dot{\tilde{\boldsymbol{\zeta}}}^\top]^\top$
$\tilde{\mathbf{D}}_r = \{\tilde{t}_h, \tilde{\boldsymbol{\eta}}_h, \tilde{\boldsymbol{\Sigma}}_h\}_{h=1}^H$	$\triangleq$	additional reference trajectory to indicate the transformed desired points
$\mathbf{D}_r^U = \{t_l^U, \boldsymbol{\mu}_l^U, \boldsymbol{\Sigma}_l^U\}_{l=1}^{N+H}$	$\triangleq$	extended reference trajectory, see (26)
$\mathbf{D}^0 = \{\{\mathbf{s}_{n,m}, \boldsymbol{\xi}_{n,m}^0\}_{n=1}^N\}_{m=1}^M$	$\triangleq$	demonstration database with high-dimensional input $\mathbf{s}_{n,m}$ and output $\boldsymbol{\xi}_{n,m}^0 = \begin{bmatrix}\mathbf{p}_{n,m}\\ \mathbf{q}_{n,m}\end{bmatrix}$
$\mathbf{D}^s = \{\{\mathbf{s}_{n,m}, \boldsymbol{\xi}_{n,m}\}_{n=1}^N\}_{m=1}^M$	$\triangleq$	transformed data obtained from $\mathbf{D}^0$ with $\boldsymbol{\xi}_{n,m} = \begin{bmatrix}\mathbf{p}_{n,m}\\ \log(\mathbf{q}_{n,m} * \bar{\mathbf{q}}_a)\end{bmatrix}$
$\mathbf{D}_r^s = \{\mathbf{s}_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$	$\triangleq$	probabilistic reference trajectory extracted from $\mathbf{D}^s$
$\boldsymbol{\phi}(\mathbf{s})$, $\boldsymbol{\Theta}(\mathbf{s})$	$\triangleq$	basis function vector with high-dimensional inputs and its corresponding expanded matrix, see (33)
$\mathbf{k}(\mathbf{s}_i, \mathbf{s}_j)$	$\triangleq$	expanded kernel matrix, see (35)
$\tilde{\mathbf{D}}^0 = \{\tilde{\mathbf{s}}_h, \tilde{\boldsymbol{\xi}}_h^0\}_{h=1}^H$	$\triangleq$	H desired points associated with high-dimensional inputs
$\tilde{\mathbf{D}}^s = \{\tilde{\mathbf{s}}_h, \tilde{\boldsymbol{\xi}}_h\}_{h=1}^H$	$\triangleq$	transformed desired data from $\tilde{\mathbf{D}}^0$
$\tilde{\mathbf{D}}_r^s = \{\tilde{\mathbf{s}}_h, \tilde{\boldsymbol{\xi}}_h, \tilde{\boldsymbol{\Sigma}}_h\}_{h=1}^H$	$\triangleq$	additional reference trajectory for high-dimensional inputs

where $||\cdot||$ denotes ℓ_2 norm. By using this function, demonstrated quaternions can be projected into Euclidean space.

Let us assume that we can access a set of demonstrations $\mathbf{D}_q = \{\{t_{n,m}, \mathbf{q}_{n,m}\}_{n=1}^N\}_{m=1}^M$ with N being the time length and M the number of demonstrations, where $\mathbf{q}_{n,m}$ denotes a quaternion at the n-th time-step from the m-th demonstration. In addition, we introduce an auxiliary quaternion[2] $\mathbf{q}_a$, which is subsequently used for transforming demonstrated quaternions into Euclidean space, yielding new trajectories as $\mathbf{D}_\zeta = \{\{t_{n,m}, \boldsymbol{\zeta}_{n,m}, \dot{\boldsymbol{\zeta}}_{n,m}\}_{n=1}^N\}_{m=1}^M$ with

$$\boldsymbol{\zeta}_{n,m} = \log(\mathbf{q}_{n,m} * \bar{\mathbf{q}}_a) \tag{2}$$

and $\dot{\boldsymbol{\zeta}}_{n,m} \in \mathbb{R}^3$ being the derivative of $\boldsymbol{\zeta}_{n,m} \in \mathbb{R}^3$.

It is worth pointing out that $\mathbf{q}$ and $-\mathbf{q}$ denote the same orientation. In order to ensure that all demonstrations have no discontinuities, we assume that $\mathbf{q}_{n,m}^\top \mathbf{q}_{n+1,m} > 0$, $\forall n \in \{1,2,\ldots,N-1\}$, $\forall m \in \{1,2,\ldots,M\}$. Note that if this is not satisfied, we can simply multiply $\mathbf{q}_{n+1,m}$ by -1. In addition, from the definition of $\log(\cdot)$ in (1), we can see that $\log(\mathbf{q} * \bar{\mathbf{q}}_a)$ and $\log(-\mathbf{q} * \bar{\mathbf{q}}_a)$ are different, albeit that $\mathbf{q} * \bar{\mathbf{q}}_a$ and $-\mathbf{q} * \bar{\mathbf{q}}_a$ represent the same orientation. To avoid this issue, at the n-th time step with $n \in \{1,2,\ldots,N\}$, we assume $(\mathbf{q}_{n,i} * \bar{\mathbf{q}}_a)^\top (\mathbf{q}_{n,j} * \bar{\mathbf{q}}_a) > 0$, $\forall i,j \in \{1,2,\ldots,M\}$, implying that $\mathbf{q}_{n,1} * \bar{\mathbf{q}}_a, \mathbf{q}_{n,2} * \bar{\mathbf{q}}_a, \ldots, \mathbf{q}_{n,M} * \bar{\mathbf{q}}_a$ stay in the same hemisphere of $\mathbb{S}^3$. If we write quaternion product into the form of matrix-vector multiplication, we have $(\mathbf{q}_{n,i} * \bar{\mathbf{q}}_a)^\top (\mathbf{q}_{n,j} * \bar{\mathbf{q}}_a) = \big(\mathbf{A}(\bar{\mathbf{q}}_a)\mathbf{q}_{n,i}\big)^\top \big(\mathbf{A}(\bar{\mathbf{q}}_a)\mathbf{q}_{n,j}\big) = \mathbf{q}_{n,i}^\top \mathbf{q}_{n,j}$, where $\mathbf{A}(\bar{\mathbf{q}}_a) \in \mathbb{R}^{4\times 4}$ is an orthogonal matrix [3]. In summary, demonstrations $\mathbf{D}_q$ should satisfy the following assumption:

Assumption 1 $\mathbf{q}_{n,i}^\top \mathbf{q}_{n,j} > 0$, $\forall n \in \{1,2,\ldots,N\}$, $\forall i,j \in \{1,2,\ldots,M\}$. *Moreover,* $\mathbf{q}_{n,m}^\top \mathbf{q}_{n+1,m} > 0$, $\forall n \in \{1,2,\ldots,N-1\}$, $\forall m \in \{1,2,\ldots,M\}$.

For simplicity, we denote $\boldsymbol{\eta} = [\boldsymbol{\zeta}^\top\ \dot{\boldsymbol{\zeta}}^\top]^\top \in \mathbb{R}^6$ and accordingly $\mathbf{D}_\zeta$ becomes $\mathbf{D}_\eta = \{\{t_{n,m}, \boldsymbol{\eta}_{n,m}\}_{n=1}^N\}_{m=1}^M$. Now, we apply GMM [7] to model the joint probability distribution $\mathcal{P}(t, \boldsymbol{\eta})$ from $\mathbf{D}_\eta$, leading to

$$\mathcal{P}(t, \boldsymbol{\eta}) \sim \sum_{c=1}^{C} \pi_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{3}$$

where π_c denotes prior probability of the c-th Gaussian component whose mean and covariance are, respectively, $\boldsymbol{\mu}_c = \begin{bmatrix}\boldsymbol{\mu}_{t,c}\\ \boldsymbol{\mu}_{\eta,c}\end{bmatrix}$ and $\boldsymbol{\Sigma}_c = \begin{bmatrix}\boldsymbol{\Sigma}_{tt,c} & \boldsymbol{\Sigma}_{t\eta,c}\\ \boldsymbol{\Sigma}_{\eta t,c} & \boldsymbol{\Sigma}_{\eta\eta,c}\end{bmatrix}$[3]. Then, Gaussian mixture regression (GMR) [7], [26] is employed to retrieve the conditional probability distribution, i.e.,

$$\mathcal{P}(\boldsymbol{\eta}|t) = \sum_{c=1}^{C} h_c(t) \mathcal{N}(\bar{\boldsymbol{\mu}}_c(t), \bar{\boldsymbol{\Sigma}}_c) \tag{4}$$

[2] $\mathbf{q}_a$ should meet the constraint: $\mathbf{q}_{n,m} * \bar{\mathbf{q}}_a \neq [-1\ 0\ 0\ 0]^\top$, which shall be seen in the Assumption 2 explained later.

[3] In order to keep notations consistent, we still use notations $\boldsymbol{\mu}_{t,c}$ and $\boldsymbol{\Sigma}_{tt,c}$ to represent scalars.

with

$$h_c(t)=\frac{\pi_c\mathcal{N}(t|\boldsymbol{\mu}_{t,c},\boldsymbol{\Sigma}_{tt,c})}{\sum_{i=1}^{C}\pi_i\mathcal{N}(t|\boldsymbol{\mu}_{t,i},\boldsymbol{\Sigma}_{tt,i})}, \tag{5}$$

$$\bar{\boldsymbol{\mu}}_c(t)=\boldsymbol{\mu}_{\eta,c}+\boldsymbol{\Sigma}_{\eta t,c}\boldsymbol{\Sigma}_{tt,c}^{-1}(t-\boldsymbol{\mu}_{t,c}) \tag{6}$$

$$\text{and}\quad \bar{\boldsymbol{\Sigma}}_c=\boldsymbol{\Sigma}_{\eta\eta,c}-\boldsymbol{\Sigma}_{\eta t,c}\boldsymbol{\Sigma}_{tt,c}^{-1}\boldsymbol{\Sigma}_{t\eta,c}. \tag{7}$$

With the properties of multivariate Gaussian distributions, we can estimate $\mathbb{E}(\boldsymbol{\eta}|t)$ and $\mathbb{D}(\boldsymbol{\eta}|t)$ from (4), i.e., [7]

$$\begin{aligned}\hat{\boldsymbol{\mu}}_t&=\mathbb{E}(\boldsymbol{\eta}|t)=\sum_{c=1}^{C}h_c(t)\bar{\boldsymbol{\mu}}_c(t),\\ \hat{\boldsymbol{\Sigma}}_t&=\mathbb{D}(\boldsymbol{\eta}|t)=\mathbb{E}(\boldsymbol{\eta}\boldsymbol{\eta}^\top|t)-\mathbb{E}(\boldsymbol{\eta}|t)\mathbb{E}^\top(\boldsymbol{\eta}|t)\\ &=\sum_{c=1}^{C}h_c(t)\left(\bar{\boldsymbol{\mu}}_c(t)\bar{\boldsymbol{\mu}}_c^\top(t)+\bar{\boldsymbol{\Sigma}}_c\right)-\hat{\boldsymbol{\mu}}_t\hat{\boldsymbol{\mu}}_t^\top.\end{aligned} \tag{8}$$

Furthermore, we use $\mathcal{N}(\hat{\boldsymbol{\mu}}_t,\hat{\boldsymbol{\Sigma}}_t)$ to approximate (4), i.e.,

$$\mathcal{P}(\boldsymbol{\eta}|t)\approx\mathcal{N}(\hat{\boldsymbol{\mu}}_t,\hat{\boldsymbol{\Sigma}}_t). \tag{9}$$

Please refer to [7], [8], [26] for more details. Therefore, for a given time sequence[4] $\{t_n\}_{n=1}^N$ that spans the input space, we can obtain its corresponding trajectory $\{\hat{\boldsymbol{\eta}}_n\}_{n=1}^N$ with $\hat{\boldsymbol{\eta}}_n|t_n\sim\mathcal{N}(\hat{\boldsymbol{\mu}}_n,\hat{\boldsymbol{\Sigma}}_n)$, yielding a probabilistic reference trajectory $\mathbf{D}_r=\{t_n,\hat{\boldsymbol{\mu}}_n,\hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$. Here, we can view $\mathbf{D}_r$ as a representative of $\mathbf{D}_\eta$ since it encapsulates the distribution of trajectories in $\mathbf{D}_\eta$ in terms of mean and covariance. Therefore, we exploit $\mathbf{D}_r$ instead of the original demonstrations $\mathbf{D}_\eta$ in the next subsection.

B. Learning quaternions using a kernelized approach

As a recently developed framework, KMP [8] exhibits several advantages over state-of-the-art approaches:

(*i*) In comparison with DMP [5] and TP-GMM [7], that focus on target adaptations, KMP is capable of adapting trajectories towards arbitrary desired points (e.g., start/via/end- points).

(*ii*) Unlike DMP and ProMP [6] that rely on explicit definition of basis functions, KMP employs the kernel trick to alleviate the definition of basis functions and thus allows for convenient extensions to the learning and adaptation of demonstrations consisting of high-dimensional inputs.

(*iii*) In comparison to DMP and ProMP, KMP can learn complex non-linearity underlying demonstrations with fewer open parameters owing to the kernelized form.

Note that kernel approaches generally have some limitations [27], such as the storage of experience data and the increasing computation complexity with the size of training data. However, our approach learns the probabilistic reference trajectory instead of the raw demonstration data, thus the corresponding storage load and computation cost are alleviated.

[4]The size of time sequence is not necessarily the same as that of demonstrations.

We follow the treatment in KMP to learn the probabilistic reference trajectory $\mathbf{D}_r$. Formally, let us first write $\boldsymbol{\eta}$ in a parameterized way[5], i.e.,

$$\boldsymbol{\eta}(t)=\begin{bmatrix}\boldsymbol{\zeta}(t)\\ \hline \dot{\boldsymbol{\zeta}}(t)\end{bmatrix}=\underbrace{\begin{bmatrix}\boldsymbol{\phi}^\top(t)&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\boldsymbol{\phi}^\top(t)&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\boldsymbol{\phi}^\top(t)\\ \hline \dot{\boldsymbol{\phi}}^\top(t)&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\dot{\boldsymbol{\phi}}^\top(t)&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\dot{\boldsymbol{\phi}}^\top(t)\end{bmatrix}}_{\boldsymbol{\Theta}^\top(t)}\mathbf{w}, \tag{10}$$

where $\boldsymbol{\phi}(t)\in\mathbb{R}^B$ represents a B-dimensional basis function vector. In order to learn $\mathbf{D}_r$, we consider the problem of maximizing the posterior probability

$$J(\mathbf{w})=\prod_{n=1}^{N}\mathcal{P}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}|\hat{\boldsymbol{\mu}}_n,\hat{\boldsymbol{\Sigma}}_n), \tag{11}$$

whose optimal solution $\mathbf{w}^*$ can be computed as

$$\mathbf{w}^*=\mathrm{argmin}_{\mathbf{w}}\sum_{n=1}^{N}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}-\hat{\boldsymbol{\mu}}_n)^\top(\hat{\boldsymbol{\Sigma}}_n)^{-1}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}-\hat{\boldsymbol{\mu}}_n)+\lambda\mathbf{w}^\top\mathbf{w}, \tag{12}$$

where the objective to be minimized can be viewed as the sum of covariance-weighted squared errors[6]. Note that a regularization term $\lambda\mathbf{w}^\top\mathbf{w}$ with $\lambda>0$ is introduced in (12) so as to mitigate the over-fitting.

Similarly to the derivations of kernel ridge regression [28], [29], [30], the optimal solution $\mathbf{w}^*$ of (12) can be computed. Thus, for an inquiry point t^*, its corresponding output $\boldsymbol{\eta}(t^*)$ can be predicted as

$$\boldsymbol{\eta}(t^*)=\boldsymbol{\Theta}^\top(t^*)\mathbf{w}^*=\boldsymbol{\Theta}^\top(t^*)\boldsymbol{\Phi}(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}+\lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu} \tag{13}$$

where

$$\begin{aligned}\boldsymbol{\Phi}&=[\boldsymbol{\Theta}(t_1)\,\boldsymbol{\Theta}(t_2)\,\cdots\,\boldsymbol{\Theta}(t_N)],\\ \boldsymbol{\Sigma}&=blockdiag(\hat{\boldsymbol{\Sigma}}_1,\hat{\boldsymbol{\Sigma}}_2,\ldots,\hat{\boldsymbol{\Sigma}}_N),\\ \boldsymbol{\mu}&=[\hat{\boldsymbol{\mu}}_1^\top\,\hat{\boldsymbol{\mu}}_2^\top\,\cdots\,\hat{\boldsymbol{\mu}}_N^\top]^\top.\end{aligned} \tag{14}$$

Furthermore, (13) can be kernelized as

$$\boldsymbol{\eta}(t^*)=\mathbf{k}^*(\mathbf{K}+\lambda\boldsymbol{\Sigma})^{-1}\boldsymbol{\mu} \tag{15}$$

with $\mathbf{k}^*_{[i]}=\mathbf{k}(t^*,t_i)$ and $\mathbf{K}_{[i,j]}=\mathbf{k}(t_i,t_j)$, $i\in\{1,2,\ldots,N\}, j\in\{1,2,\ldots,N\}$, where $\mathbf{k}(\cdot,\cdot)$ is defined by

$$\mathbf{k}(t_i,t_j)=\boldsymbol{\Theta}^\top(t_i)\boldsymbol{\Theta}(t_j)=\begin{bmatrix}k_{t,t}(t_i,t_j)\mathbf{I}_3&k_{t,d}(t_i,t_j)\mathbf{I}_3\\ k_{d,t}(t_i,t_j)\mathbf{I}_3&k_{d,d}(t_i,t_j)\mathbf{I}_3\end{bmatrix} \tag{16}$$

[5]Similar parametric strategies were used in DMP [5] and ProMP [6].

[6]Similar variance-weighted scheme has also been exploited in trajectory-GMM [7], motion similarity estimation [22] and optimal control [24].

with[7]

$$k_{t,t}(t_i,t_j)=k(t_i,t_j),$$
$$k_{t,d}(t_i,t_j)=\frac{k(t_i,t_j+\delta)-k(t_i,t_j)}{\delta},$$
$$k_{d,t}(t_i,t_j)=\frac{k(t_i+\delta,t_j)-k(t_i,t_j)}{\delta},$$
$$k_{d,d}(t_i,t_j)=\frac{k(t_i+\delta,t_j+\delta)-k(t_i+\delta,t_j)-k(t_i,t_j+\delta)+k(t_i,t_j)}{\delta^2},$$

where $\delta>0$ is a small constant and $k(t_i,t_j)=\boldsymbol{\phi}(t_i)^\top\boldsymbol{\phi}(t_j)$ represents the kernel function.

By observing (10), we can find that both $\boldsymbol{\phi}(t)$ and $\dot{\boldsymbol{\phi}}(t)$ are used. If we would have used $\boldsymbol{\phi}(t)$ to parameterize $\boldsymbol{\zeta}(t)$ and $\dot{\boldsymbol{\zeta}}(t)$ independently of each other, i.e., $\boldsymbol{\eta}(t)=\begin{bmatrix}\boldsymbol{\zeta}(t)\\ \dot{\boldsymbol{\zeta}}(t)\end{bmatrix}=diag\big(\boldsymbol{\phi}^\top(t),\boldsymbol{\phi}^\top(t),\ldots,\boldsymbol{\phi}^\top(t)\big)\begin{bmatrix}\mathbf{w}_1\\ \mathbf{w}_2\end{bmatrix}$ with $\mathbf{w}_1\in\mathbb{R}^{3B}$ and $\mathbf{w}_2\in\mathbb{R}^{3B}$, a simpler kernel $\mathbf{k}(t_i,t_j)=k(t_i,t_j)\mathbf{I}_6$ can be obtained in (16). However, this treatment would ignore the derivative relationship between $\boldsymbol{\zeta}(t)$ and $\dot{\boldsymbol{\zeta}}(t)$. Consequently, in predictions (13) and (15), the derivative relationship between $\boldsymbol{\zeta}(t^*)$ and $\dot{\boldsymbol{\zeta}}(t^*)$ could not be guaranteed.

Once we have determined $\boldsymbol{\eta}(t^*)$ at a query point t^* via (15), we can use its component $\boldsymbol{\zeta}(t^*)$ to recover the corresponding quaternion $\mathbf{q}(t^*)$. Specifically, $\mathbf{q}(t^*)$ is determined by

$$\mathbf{q}(t^*)=\exp(\boldsymbol{\zeta}(t^*))*\mathbf{q}_a, \tag{17}$$

where the function $\exp(\cdot):\mathbb{R}^3\mapsto\mathbb{S}^3$ is [14], [15]

$$\exp(\boldsymbol{\zeta})=\begin{cases}\begin{bmatrix}\cos(||\boldsymbol{\zeta}||)\\ \sin(||\boldsymbol{\zeta}||)\frac{\boldsymbol{\zeta}}{||\boldsymbol{\zeta}||}\end{bmatrix}, & \boldsymbol{\zeta}\neq\mathbf{0}\\ [1\ \ 0\ \ 0\ \ 0]^\top, & otherwise.\end{cases} \tag{18}$$

It should be noted that the singularity issue exists in $\log(\cdot)$, thus an assumption is imposed throughout this paper:

Assumption 2 [14] *The input domain of the mapping* $\log(\cdot)$ *is restricted to* $\mathbb{S}^3$ *except for* $[-1\ \ 0\ \ 0\ \ 0]^\top$, *while the input domain of the mapping* $\exp(\boldsymbol{\zeta})$ *should fulfill the constraint* $||\boldsymbol{\zeta}||<\pi$.

Under this assumption, both $\log(\cdot)$ and $\exp(\cdot)$ are bijective, and $\exp(\cdot)$ can be viewed as the inverse function of $\log(\cdot)$, leading to $\exp\big(\log(\mathbf{q}*\bar{\mathbf{q}}_a)\big)*\mathbf{q}_a=\mathbf{q}$. Please refer to [14] for the discussion. Note that when we choose the auxiliary quaternion $\mathbf{q}_a$ in (2), it must obey the Assumption 2.

III. Adaptation of Quaternion Trajectories

While the approach in Section II-B is limited to orientation reproduction, we now consider the problem of adapting the reference trajectory in terms of desired quaternions and angular velocities. To do so, we propose to transform desired orientation states into Euclidean space (Section III-A), and subsequently we reformulate the kernelized learning approach to incorporate these transformed desired points (Section III-B). Finally, the adapted trajectory in Euclidean space is used to retrieve its corresponding adapted quaternion trajectory.

[7]Note that $\dot{\boldsymbol{\phi}}(t)$ is approximated by $\dot{\boldsymbol{\phi}}(t)\approx\frac{\boldsymbol{\phi}(t+\delta)-\boldsymbol{\phi}(t)}{\delta}$ in order to facilitate the following kernelized operations.

A. Transform desired quaternion states

Let us denote H desired quaternion states as $\tilde{\mathbf{D}}_q=\{\tilde{t}_h,\tilde{\mathbf{q}}_h,\tilde{\boldsymbol{\omega}}_h\}_{h=1}^H$, where $\tilde{\mathbf{q}}_h\in\mathbb{S}^3$ and $\tilde{\boldsymbol{\omega}}_h\in\mathbb{R}^3$ represent desired quaternion and angular velocity at time $\tilde{t}_h$, respectively. Similarly to (2), the desired quaternion $\tilde{\mathbf{q}}_h$ can be transformed as

$$\tilde{\boldsymbol{\zeta}}_h=\log(\tilde{\mathbf{q}}_h*\bar{\mathbf{q}}_a). \tag{19}$$

In order to incorporate the desired angular velocity $\tilde{\boldsymbol{\omega}}_h$, we resort to the relationship between derivatives of quaternions and angular velocities, i.e., [14], [15]

$$\dot{\mathbf{q}}=\frac{1}{2}\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix}*\mathbf{q}\Rightarrow\mathbf{q}(t+\delta_t)=\exp\left(\frac{\boldsymbol{\omega}}{2}\delta_t\right)*\mathbf{q}(t), \tag{20}$$

where $\delta_t>0$ denotes a small constant. By using (20), we can compute the desired quaternion at time $\tilde{t}_h+\delta_t$ as

$$\tilde{\mathbf{q}}(\tilde{t}_h+\delta_t)=\exp\left(\frac{\tilde{\boldsymbol{\omega}}_h}{2}\delta_t\right)*\tilde{\mathbf{q}}_h, \tag{21}$$

which is subsequently transformed into Euclidean space via (2), resulting in

$$\tilde{\boldsymbol{\zeta}}(\tilde{t}_h+\delta_t)=\log(\tilde{\mathbf{q}}(\tilde{t}_h+\delta_t)*\bar{\mathbf{q}}_a). \tag{22}$$

Thus, we can approximate the derivative of $\tilde{\boldsymbol{\zeta}}_h$ as

$$\begin{aligned}\dot{\tilde{\boldsymbol{\zeta}}}_h&\approx\frac{\tilde{\boldsymbol{\zeta}}(\tilde{t}_h+\delta_t)-\tilde{\boldsymbol{\zeta}}_h}{\delta_t}\\ &=\frac{\log\left(\left(\exp(\frac{\tilde{\boldsymbol{\omega}}_h}{2}\delta_t)*\tilde{\mathbf{q}}_h\right)*\bar{\mathbf{q}}_a\right)-\log(\tilde{\mathbf{q}}_h*\bar{\mathbf{q}}_a)}{\delta_t}.\end{aligned} \tag{23}$$

Now, $\tilde{\mathbf{D}}_q$ can be transformed into $\tilde{\mathbf{D}}_\zeta=\{\tilde{t}_h,\tilde{\boldsymbol{\zeta}}_h,\dot{\tilde{\boldsymbol{\zeta}}}_h\}_{h=1}^H$ via (19) and (23), which can be further rewritten in a compact way as $\tilde{\mathbf{D}}_\eta=\{\tilde{t}_h,\tilde{\boldsymbol{\eta}}_h\}_{h=1}^H$ with $\tilde{\boldsymbol{\eta}}_h=[\tilde{\boldsymbol{\zeta}}_h^\top\ \dot{\tilde{\boldsymbol{\zeta}}}_h^\top]^\top\in\mathbb{R}^6$. In addition, we can design a covariance $\tilde{\boldsymbol{\Sigma}}_h\in\mathbb{R}^{6\times6}$ for each desired point $\tilde{\boldsymbol{\eta}}_h$ to control the precision of adaptations. Thus, we can obtain an additional probabilistic reference trajectory $\tilde{\mathbf{D}}_r=\{\tilde{t}_h,\tilde{\boldsymbol{\eta}}_h,\tilde{\boldsymbol{\Sigma}}_h\}_{h=1}^H$ to indicate the transformed desired quaternion states.

B. Adapting quaternion trajectories towards desired points

Formally, the adaptation problem can be addressed by incorporating $\tilde{\mathbf{D}}_r$ into (12), i.e.,

$$\begin{aligned}\mathbf{w}^*=\mathrm{argmin}_{\mathbf{w}}&\underbrace{\sum_{n=1}^N(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}-\hat{\boldsymbol{\mu}}_n)^\top(\hat{\boldsymbol{\Sigma}}_n)^{-1}(\boldsymbol{\Theta}^\top(t_n)\mathbf{w}-\hat{\boldsymbol{\mu}}_n)}_{imitation}\\ &+\underbrace{\sum_{h=1}^H\big(\boldsymbol{\Theta}^\top(\tilde{t}_h)\mathbf{w}-\tilde{\boldsymbol{\eta}}_h\big)^\top(\tilde{\boldsymbol{\Sigma}}_h)^{-1}\big(\boldsymbol{\Theta}^\top(\tilde{t}_h)\mathbf{w}-\tilde{\boldsymbol{\eta}}_h\big)}_{adaptations}+\underbrace{\lambda\mathbf{w}^\top\mathbf{w}}_{regularizer},\end{aligned} \tag{24}$$

whose compact representation is

$$\begin{aligned}\mathbf{w}^*=\mathrm{argmin}_{\mathbf{w}}\sum_{l=1}^{N+H}&\Bigg(\left(\boldsymbol{\Theta}^\top(t_l^U)\mathbf{w}-\boldsymbol{\mu}_l^U\right)^\top(\boldsymbol{\Sigma}_l^U)^{-1}\\ &\left(\boldsymbol{\Theta}^\top(t_l^U)\mathbf{w}-\boldsymbol{\mu}_l^U\right)\Bigg)+\lambda\mathbf{w}^\top\mathbf{w},\end{aligned} \tag{25}$$

with

$$\begin{cases} t_l^U = t_l, \boldsymbol{\mu}_l^U = \hat{\boldsymbol{\mu}}_l, \boldsymbol{\Sigma}_l^U = \hat{\boldsymbol{\Sigma}}_l, & 1 \leq l \leq N \\ t_l^U = \tilde{t}_{l-N}, \boldsymbol{\mu}_l^U = \tilde{\boldsymbol{\eta}}_{l-N}, \boldsymbol{\Sigma}_l^U = \tilde{\boldsymbol{\Sigma}}_{l-N}, & N+1 \leq l \leq N+H. \end{cases} \tag{26}$$

It can be observed that the new objective (25) shares the same form with (12), except that the reference trajectory $\mathbf{D}_r^U = \{t_l^U, \boldsymbol{\mu}_l^U, \boldsymbol{\Sigma}_l^U\}_{l=1}^{N+H}$ in (25) is longer than that in (12), thus the solution of (25) can be determined in a similar way. Finally, $\boldsymbol{\eta}(t) = [\boldsymbol{\zeta}^\top(t)\, \dot{\boldsymbol{\zeta}}^\top(t)]^\top$ can be computed via (15) and, subsequently, $\mathbf{q}(t)$ is recovered from (17) by using $\boldsymbol{\zeta}(t)$. In this case, $\mathbf{q}(t)$ is capable of passing through the desired quaternions $\tilde{\mathbf{q}}_h$ with desired angular velocities $\tilde{\boldsymbol{\omega}}_h$ at time $\tilde{t}_h$, provided that $\tilde{\boldsymbol{\Sigma}}_h$ is small enough[8].

IV. Quaternion Adaptations with Angular Acceleration/Jerk Constraints

It is well known that robot trajectories should be smooth in order to facilitate the design of controllers as well as the execution of motor commands [20], [21]. For instance, in a striking task that needs fast striking motions, extremely high accelerations or jerks may degrade the final striking performance, given the physical limits of motors. It is possible to formulate this constraint as an optimization problem and search for the optimal trajectory via an iterative scheme, as done in [20]. In this section, we consider the problem of learning and adapting quaternion trajectories while taking into account angular acceleration or jerk constraints. Specifically, we aim to provide an *analytical* solution to the issue.

Formally, we consider the angular acceleration or jerk constraints as minimizing $\sum_{n=1}^{N} ||\dot{\boldsymbol{\omega}}(t_n)||^2$ or $\sum_{n=1}^{N} ||\ddot{\boldsymbol{\omega}}(t_n)||^2$. Note that the aforementioned imitation learning problem (12) is built on the trajectory $\boldsymbol{\zeta}(t)$, therefore we need to find the relationship between $\boldsymbol{\omega}(t)$ and $\boldsymbol{\zeta}(t)$. Here, we provide two main results:

Theorem 1 *Given the definition* $\boldsymbol{\zeta}(t) = \log(\mathbf{q}(t) * \bar{\mathbf{q}}_a)$, *if we let* $\mathbf{q}(t_1) = \mathbf{q}_a$, *then the optimal quaternion trajectory* $\{\mathbf{q}(t_n)\}_{t=1}^{N+2}$ *of minimizing* $\sum_{n=1}^{N} ||\ddot{\boldsymbol{\zeta}}(t_n)||^2$ *corresponds to the optimal solution of minimizing* $\sum_{n=1}^{N} ||\dot{\boldsymbol{\omega}}(t_n)||^2$.

Proof. The minimization of $\sum_{n=1}^{N} ||\ddot{\boldsymbol{\zeta}}(t_n)||^2$ implies that

$$\dot{\boldsymbol{\zeta}}(t_1) = \dot{\boldsymbol{\zeta}}(t_2) = \cdots = \dot{\boldsymbol{\zeta}}(t_N) = \dot{\boldsymbol{\zeta}}(t_{N+1}).$$

Consequently, we can write $\boldsymbol{\zeta}(t_{n+1}) = \boldsymbol{\zeta}(t_n) + \Delta$ with $\Delta \in \mathbb{R}^3$ being a constant. Given $\mathbf{q}(t_1) = \mathbf{q}_a$, we have $\boldsymbol{\zeta}(t_1) = \mathbf{0}$ and

$$\boldsymbol{\zeta}(t_n) = (n-1)\Delta, n = 1, 2, \ldots, N+2.$$

Using the definition of $\boldsymbol{\zeta}(t)$, we can obtain the optimal quaternion trajectory $\{\mathbf{q}(t_n)\}_{t=1}^{N+2}$ of minimizing $\sum_{n=1}^{N} ||\ddot{\boldsymbol{\zeta}}(t_n)||^2$, i.e.,

$$\mathbf{q}(t_n) = \exp\big((n-1)\Delta\big) * \mathbf{q}_a.$$

[8]In (24) both the 'imitation' and 'adaptation' have impacts on $\mathbf{w}^*$ and both terms rely on their own covariance matrices. Thus, if one needs precise adaptation, $\tilde{\boldsymbol{\Sigma}}_h$ should be set far smaller (e.g., at least 2-3 orders of magnitude) than the variance of the reference trajectory.

Now, we use the optimal quaternion trajectory to calculate the corresponding $\{\boldsymbol{\omega}(t_n)\}_{n=1}^{N+1}$. Specifically, we have[9]

$$\begin{aligned} \boldsymbol{\omega}(t_n) &= \frac{2}{\delta_t} \log\big(\mathbf{q}(t_{n+1}) * \overline{\mathbf{q}(t_n)}\big) \\ &= \frac{2}{\delta_t} \log\big(\exp(n\Delta) * \overline{\exp((n-1)\Delta)}\big) \\ &= \frac{2}{\delta_t} \log\big(\exp(\Delta)\big) = \frac{2}{\delta_t}\Delta \end{aligned}$$

with δ_t being the time interval between $\mathbf{q}(t_n)$ and $\mathbf{q}(t_{n+1})$, which implies

$$\boldsymbol{\omega}(t_1) = \boldsymbol{\omega}(t_2) = \cdots = \boldsymbol{\omega}(t_N) = \boldsymbol{\omega}(t_{N+1}).$$

Thus, we have $\dot{\boldsymbol{\omega}}(t_n) = \mathbf{0}, n = 1, 2, \ldots, N$, which corresponds to the optimal solution of minimizing $\sum_{n=1}^{N} ||\dot{\boldsymbol{\omega}}(t_n)||^2$. $\square$

Theorem 2 *Given the definition* $\boldsymbol{\zeta}(t) = \log(\mathbf{q}(t) * \bar{\mathbf{q}}_a)$, *if we let* $\mathbf{q}(t_1) = \mathbf{q}_a$ *and* $\boldsymbol{\omega}(t_1) = \mathbf{0}$, *then the optimal quaternion trajectory* $\{\mathbf{q}(t_n)\}_{t=1}^{N+3}$ *of minimizing* $\sum_{n=1}^{N} ||\dddot{\boldsymbol{\zeta}}(t_n)||^2$ *corresponds to the optimal solution of minimizing* $\sum_{n=1}^{N} ||\ddot{\boldsymbol{\omega}}(t_n)||^2$.

Proof. Minimizing $\sum_{n=1}^{N} ||\dddot{\boldsymbol{\zeta}}(t_n)||^2$ corresponds to

$$\ddot{\boldsymbol{\zeta}}(t_1) = \ddot{\boldsymbol{\zeta}}(t_2) = \cdots = \ddot{\boldsymbol{\zeta}}(t_N) = \ddot{\boldsymbol{\zeta}}(t_{N+1}).$$

Then, we have $\dot{\boldsymbol{\zeta}}(t_{n+1}) = \dot{\boldsymbol{\zeta}}(t_n) + \Delta$, where $\Delta \in \mathbb{R}^3$ is a constant. With the approximation $\dot{\boldsymbol{\zeta}}(t) = \frac{\boldsymbol{\zeta}(t+\delta_t) - \boldsymbol{\zeta}(t)}{\delta_t}$, we have $\boldsymbol{\zeta}(t_{n+2}) = 2\boldsymbol{\zeta}(t_{n+1}) - \boldsymbol{\zeta}(t_n) + \delta_t\Delta$. Note that we assume $\mathbf{q}(t_1) = \mathbf{q}_a$ and $\boldsymbol{\omega}(t_1) = 0$. Hence, $\mathbf{q}(t_1) = \mathbf{q}(t_2) = \mathbf{q}_a$ and $\boldsymbol{\zeta}(1) = \boldsymbol{\zeta}(2) = 0$. It can be further seen that

$$\boldsymbol{\zeta}(t_n) = \frac{(n-1)(n-2)}{2}\delta_t\Delta, n = 1, 2, \ldots, N+3.$$

Using the definition of $\boldsymbol{\zeta}(t)$, we have the optimal quaternion trajectory $\{\mathbf{q}(t_n)\}_{t=1}^{N+3}$ of minimizing $\sum_{n=1}^{N} ||\dddot{\boldsymbol{\zeta}}(t_n)||^2$, i.e.,

$$\mathbf{q}(t_n) = \exp\left(\frac{(n-1)(n-2)}{2}\delta_t\Delta\right) * \mathbf{q}_a.$$

The corresponding angular velocity $\{\boldsymbol{\omega}(t_n)\}_{n=1}^{N+2}$ of using the optimal quaternion trajectory is

$$\begin{aligned} \boldsymbol{\omega}(t_n) &= \frac{2}{\delta_t} \log\left(\mathbf{q}(t_{n+1}) * \overline{\mathbf{q}(t_n)}\right) \\ &= \frac{2}{\delta_t} \log\left(\exp\left(\frac{n(n-1)}{2}\delta_t\Delta\right) * \overline{\exp\left(\frac{(n-1)(n-2)}{2}\delta_t\Delta\right)}\right) \\ &= \frac{2}{\delta_t} \log\big(\exp\big((n-1)\delta_t\Delta\big)\big) = 2(n-1)\Delta, \end{aligned}$$

[9]The following results are used in Theorems 1 and 2:

(*i*) If $\Delta \neq \mathbf{0}$ and $i \neq j$,

$$\begin{aligned} \exp(i\Delta) * \overline{\exp(j\Delta)} &= \begin{bmatrix} cos(||i\Delta||) \\ sin(||i\Delta||)\frac{\Delta}{||\Delta||} \end{bmatrix} * \begin{bmatrix} cos(||j\Delta||) \\ -sin(||j\Delta||)\frac{\Delta}{||\Delta||} \end{bmatrix} \\ &= \begin{bmatrix} cos(||i\Delta|| - ||j\Delta||) \\ sin(||i\Delta|| - ||j\Delta||)\frac{\Delta}{||\Delta||} \end{bmatrix} = \begin{bmatrix} cos((i-j)||\Delta||) \\ sin((i-j)||\Delta||)\frac{\Delta}{||\Delta||} \end{bmatrix} \\ &= \begin{bmatrix} cos(|i-j|||\Delta||) \\ sin(|i-j|||\Delta||)\frac{(i-j)\Delta}{|i-j|||\Delta||} \end{bmatrix} = \exp((i-j)\Delta). \end{aligned}$$

(*ii*) If $\Delta = \mathbf{0}$ or $i = j$,

$$\exp(i\Delta) * \overline{\exp(j\Delta)} = \begin{bmatrix} 1 \\ \mathbf{0} \end{bmatrix} = \exp((i-j)\Delta).$$

which implies that $\boldsymbol{\omega}(t_{n+1}) - \boldsymbol{\omega}(t_n) = 2\Delta$. Thus,

$$\dot{\boldsymbol{\omega}}(t_1) = \dot{\boldsymbol{\omega}}(t_2) = \cdots = \dot{\boldsymbol{\omega}}(t_N) = \dot{\boldsymbol{\omega}}(t_{N+1}),$$

leading to $\ddot{\boldsymbol{\omega}}(t_n) = \mathbf{0}, n = 1, 2, \ldots, N$. So, we can conclude that the optimal trajectory $\{\mathbf{q}(t_n)\}_{n=1}^{N+3}$ that minimizes $\sum_{n=1}^{N} ||\dddot{\boldsymbol{\zeta}}(t_n)||^2$ yields the optimal solution $\{\boldsymbol{\omega}(t_n)\}_{n=1}^{N+2}$ of minimizing $\sum_{n=1}^{N} ||\ddot{\boldsymbol{\omega}}(t_n)||^2$. □

With Theorems 1 and 2, the problem of learning quaternions with angular acceleration or jerk constraints can be approximately tackled by incorporating $\sum_{n=1}^{N} ||\ddot{\boldsymbol{\zeta}}(t_n)||^2$ or $\sum_{n=1}^{N} ||\dddot{\boldsymbol{\zeta}}(t_n)||^2$ into the objective (12). For brevity, we take the angular acceleration constraints as an example, while the case of angular jerk constraints can be treated in a similar way. Following the parameterization form in (10), we have

$$\ddot{\boldsymbol{\zeta}}(t) = \underbrace{\begin{bmatrix} \ddot{\boldsymbol{\phi}}^\top(t) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddot{\boldsymbol{\phi}}^\top(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddot{\boldsymbol{\phi}}^\top(t) \end{bmatrix}}_{\boldsymbol{\varphi}^\top(t)} \mathbf{w}. \tag{27}$$

Thus, the problem of learning orientations with angular acceleration constraints becomes

$$\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}} \underbrace{\sum_{n=1}^{N} (\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n)^\top (\hat{\boldsymbol{\Sigma}}_n)^{-1} (\boldsymbol{\Theta}^\top(t_n)\mathbf{w} - \hat{\boldsymbol{\mu}}_n)}_{imitation}$$
$$+ \lambda_a \underbrace{\sum_{n=1}^{N} (\boldsymbol{\varphi}^\top(t_n)\mathbf{w})^\top (\boldsymbol{\varphi}^\top(t_n)\mathbf{w})}_{angular\ acceleration\ constraints} + \underbrace{\lambda \mathbf{w}^\top \mathbf{w}}_{regularizer}, \tag{28}$$

where $\lambda_a > 0$ acts as a trade-off regulator between orientation learning and angular acceleration minimization.

Let us re-arrange (28) into a compact form, resulting in

$$\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}} \sum_{n=1}^{N} (\boldsymbol{\Omega}^\top(t_n)\mathbf{w} - \bar{\boldsymbol{\mu}}_n)^\top (\bar{\boldsymbol{\Sigma}}_n)^{-1} (\boldsymbol{\Omega}^\top(t_n)\mathbf{w} - \bar{\boldsymbol{\mu}}_n) + \lambda \mathbf{w}^\top \mathbf{w}, \tag{29}$$

where

$$\boldsymbol{\Omega}(t_n) = [\boldsymbol{\Theta}(t_n)\ \boldsymbol{\varphi}(t_n)], \bar{\boldsymbol{\mu}}_n = \begin{bmatrix} \hat{\boldsymbol{\mu}}_n \\ \mathbf{0} \end{bmatrix}, \bar{\boldsymbol{\Sigma}}_n = \begin{bmatrix} \hat{\boldsymbol{\Sigma}}_n & \mathbf{0} \\ \mathbf{0} & \frac{1}{\lambda_a}\mathbf{I} \end{bmatrix}. \tag{30}$$

It can be observed that (29) shares the same formula as (12), and hence we can follow (13)–(18) to derive a kernelized solution for quaternion reproduction with angular acceleration constraints. Note that $\boldsymbol{\Omega}(t)$ comprises the second-order derivative of $\boldsymbol{\phi}(t)$, thus a new kernel matrix

$$\mathbf{k}(t_i, t_j) = \boldsymbol{\Omega}^\top(t_i)\boldsymbol{\Omega}(t_j) = \begin{bmatrix} \boldsymbol{\Theta}^\top(t_i)\boldsymbol{\Theta}(t_j) & \boldsymbol{\Theta}^\top(t_i)\boldsymbol{\varphi}(t_j) \\ \boldsymbol{\varphi}^\top(t_i)\boldsymbol{\Theta}(t_j) & \boldsymbol{\varphi}^\top(t_i)\boldsymbol{\varphi}(t_j) \end{bmatrix} \tag{31}$$

is required instead of (16). Please see the detailed derivations of kernelizing (31) in Appendix A.

Similarly, by analogy with (12) and (24), the adaptation issue with angular acceleration or jerk constraints can be addressed by reformulating (29) to include the desired points. It is noted that the assumptions in Theorem 1 (i.e., $\mathbf{q}(t_1) = \mathbf{q}_a$) and Theorem 2 (i.e., $\mathbf{q}(t_1) = \mathbf{q}_a$ and $\boldsymbol{\omega}(t_1) = \mathbf{0}$) are trivial since they can be guaranteed by simply specifying a desired point at time t_1.

V. Learning Quaternions Associated with High-Dimensional Inputs

While the aforementioned results focus on learning and adapting orientation trajectories associated with time input, we now consider the case of learning orientations with high-dimensional varying inputs. Specifically, we focus on the problem of learning demonstrations $\mathbf{D}^0 = \{\{\mathbf{s}_{n,m}, \boldsymbol{\xi}_{n,m}^0\}_{n=1}^N\}_{m=1}^M$ consisting of inputs $\mathbf{s}_{n,m}$ and outputs[10] $\boldsymbol{\xi}_{n,m}^0 = \begin{bmatrix} \mathbf{p}_{n,m} \\ \mathbf{q}_{n,m} \end{bmatrix}$, where $\mathbf{s}_{n,m} \in \mathbb{R}^{\mathcal{I}}$ denotes an $\mathcal{I}$-dimensional input vector and $\boldsymbol{\xi}_{n,m}^0$ stands for the concatenation of end-effector position $\mathbf{p}$ and quaternion $\mathbf{q}$ in Cartesian space. Please note that the varying high-dimensional input trajectories $\{\{\mathbf{s}_{n,m}\}_{n=1}^N\}_{m=1}^M$ are parts of demonstrations, which shall not be confused with external contextual variables describing the conditions under which demonstrations are recorded.

In order to illustrate the importance of learning demonstrations that consist of multiple varying inputs, we first motivate this problem in Section V-A. After that, we show the modeling of demonstrations $\mathbf{D}^0$ in Section V-B, which is later exploited to derive the kernelized approach for learning and adapting quaternions associated with high-dimensional inputs (Sections V-C and V-D).

A. *Why learning demonstrations comprising high-dimensional inputs?*

Let us take human-robot collaboration as an example, where the robot is demanded to react properly in response to the human states (e.g., human hand positions/orientations). In many previous work, human and robot motions are encoded by taking time as input (e.g., DMP was used in [31] and ProMP in [32]). Specifically, when the human trajectory is rescaled in time, the corresponding robot trajectory with respect to time will also be modified. However, this treatment will cause a synchronization issue, since human motions in the new evaluations could be significantly different (e.g., faster/slower velocity) from the demonstrated ones. For instance, assuming in a human-robot handover task where demonstrations (including human and robot motions) lasting for $10s$ are recorded, but in the evaluation stage human hand has a pause for more than $10s$ before moving. In this case, the robot will still keep moving as its trajectory is driven by time. As a consequence, before the human hand starts to move, the robot has finished its handover action, which violates the synchronization constraints. To avoid this issue, in [33] human movement duration in new evaluations is required to be the same as the one in the training demonstrations. However, as pointed out in [32], this restriction on human motion duration is impractical. In order to provide a generic solution for human-robot collaboration, various strategies of phase-estimation and time-alignment are designed towards synchronizing human and robot in [31], [32].

[10]Note that outputs that comprise multiple Cartesian positions, quaternions, rotation matrices and joint positions can be tackled similarly.

In contrast, we propose to consider high-dimensional varying signals (e.g., human hand positions/orientations) as inputs and predict robot motions (e.g., Cartesian positions and orientations) according to the sensed states of the human hand. We have successfully tested this solution in previous work on human-robot collaboration, namely in the collaborative hand task [8] and robot-assisted painting task [11], [34]. However, in none of those works we have considered orientation outputs, as the proposed tools were designed for Euclidean data, and only [8] considers adaptation to new, unseen inputs. Note that now robot trajectory is directly driven by the state of the human hand, thus time is not explicitly involved when predicting robot actions. Recalling the above-mentioned example and following our strategy, if the human hand has a pause (i.e., inputs are unchanged), the corresponding robot trajectory (i.e., outputs) will also remain unchanged. Therefore, the main advantage of learning demonstrations consisting of multiple inputs is that additional synchronization procedures are not needed, providing a straightforward solution for accomplishing complex collaboration tasks.

B. Modeling quaternions with high-dimensional inputs

Handling high-dimensional inputs requires a different treatment, when compared to the time-driven case described in Section II, due to the higher complexity of the input space. Since we are considering quaternions as outputs, we follow the procedure in Section II-A, transferring $\mathbf{D}^0$ into $\mathbf{D}^s = \{\{\mathbf{s}_{n,m}, \boldsymbol{\xi}_{n,m}\}_{n=1}^N\}_{m=1}^M$, where $\boldsymbol{\xi}_{n,m} = \begin{bmatrix} \mathbf{p}_{n,m} \\ \log(\mathbf{q}_{n,m} * \bar{\mathbf{q}}_a) \end{bmatrix} \in \mathbb{R}^6$. Then, we model the joint probability distribution $\mathcal{P}(\mathbf{s}, \boldsymbol{\xi})$ from $\mathbf{D}^s$ via GMM, i.e.,

$$\mathcal{P}(\mathbf{s}, \boldsymbol{\xi}) \sim \sum_{c=1}^{C} \pi_c \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{32}$$

with $\boldsymbol{\mu}_c = \begin{bmatrix} \boldsymbol{\mu}_{\mathbf{s},c} \\ \boldsymbol{\mu}_{\boldsymbol{\xi},c} \end{bmatrix}$ and $\boldsymbol{\Sigma}_c = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{ss},c} & \boldsymbol{\Sigma}_{\mathbf{s}\boldsymbol{\xi},c} \\ \boldsymbol{\Sigma}_{\boldsymbol{\xi}\mathbf{s},c} & \boldsymbol{\Sigma}_{\boldsymbol{\xi}\boldsymbol{\xi},c} \end{bmatrix}$.

However, unlike the generation of probabilistic reference trajectories from time-driven demonstrations (Section II-A), it is not straightforward to decide the input sequence $\{\mathbf{s}_n\}_{n=1}^N$ for retrieving the corresponding reference trajectory, due to the fact that $\mathbf{s}$ is high-dimensional.

Note that a proper input sequence should span the whole input space, in order to adequately encapsulate all shown robot behaviors. This is relatively straightforward when the input is time, since the inputs of all demonstrations lie on one axis and have roughly the same duration. When the input is high-dimensional, one intuitive solution is to use the input parts of all demonstrations, which, however, will lead to two issues: (*i*) if all training inputs are used, redundancy will often arise, leading to a data-inefficient solution, where multiple datapoints map to the same robot pose; (*ii*) if only parts of input trajectories are exploited, one risks failing to capture important input points.

Therefore, we propose to sample inputs from the marginal probability distribution[11] $\mathcal{P}(\mathbf{s})$. Specifically, we sample an indicator variable z_c with the probability π_c and, subsequently, we sample $\mathbf{s}$ from $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{s},c}, \boldsymbol{\Sigma}_{\mathbf{ss},c})$. By using this sampling strategy, the input sequence $\{\mathbf{s}_n\}_{n=1}^N$ can be determined. In this way, the input sequence captures the probabilistic properties of the input space of demonstrations, where data-dense regions (hence important) will lead to more sampled input points and vice-versa for more sparse regions of the input space. Accordingly, by using GMR we have the probabilistic reference trajectory with high-dimensional inputs, denoted by $\mathbf{D}_r^s = \{\mathbf{s}_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$. The resulting probabilistic reference trajectory $\mathbf{D}_r^s$ hence encapsulates the probabilistic features of demonstrations. It should be noted that, for sampling, N needs not be the same as the number of points in each demonstration, hence one has the freedom to sub-sample when higher computational efficiency is required.

[11]Readers are suggested to refer to [35] for GMM sampling.

C. Learning quaternions with high-dimension inputs

Similarly to (10), we formulate the parametric trajectory associated with high-dimensional inputs as

$$\boldsymbol{\xi}(\mathbf{s}) = \underbrace{\begin{bmatrix} \boldsymbol{\phi}^\top(\mathbf{s}) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\phi}^\top(\mathbf{s}) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\phi}^\top(\mathbf{s}) \end{bmatrix}}_{\boldsymbol{\Theta}^\top(\mathbf{s})} \mathbf{w}. \tag{33}$$

Note that $\frac{d\boldsymbol{\phi}(\mathbf{s})}{dt}$ is not included in (33) since it relies on $\frac{d\mathbf{s}}{dt}$, which is often unpredictable in real applications. Formally, we formulate the imitation learning problem with multiple inputs as maximizing

$$J(\mathbf{w}) = \prod_{n=1}^{N} \mathcal{P}(\boldsymbol{\Theta}^\top(\mathbf{s}_n)\mathbf{w} | \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n). \tag{34}$$

Consequently, we can follow (12)–(15) to derive the kernelized approach that is capable of learning high-dimensional inputs, except that the definition of $\boldsymbol{\phi}(\cdot)$ is different and thus the kernel definition in (16) becomes

$$\mathbf{k}(\mathbf{s}_i, \mathbf{s}_j) = \boldsymbol{\Theta}^\top(\mathbf{s}_i)\boldsymbol{\Theta}(\mathbf{s}_j) = k(\mathbf{s}_i, \mathbf{s}_j)\mathbf{I}_6. \tag{35}$$

Therefore, given a query input $\mathbf{s}^*$, we can employ (15) to predict the output $\boldsymbol{\xi}(\mathbf{s}^*) = \begin{bmatrix} \boldsymbol{\xi}_\mathbf{p} \\ \boldsymbol{\xi}_\mathbf{q} \end{bmatrix}$ and, subsequently, retrieve the corresponding Cartesian state through $\begin{bmatrix} \mathbf{p}(\mathbf{s}^*) \\ \mathbf{q}(\mathbf{s}^*) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\xi}_\mathbf{p} \\ \exp(\boldsymbol{\xi}_\mathbf{q}) * \mathbf{q}_a \end{bmatrix}$.

D. Adapting quaternions with high-dimensional inputs

Let us write H desired points as $\tilde{\mathbf{D}}^0 = \{\tilde{\mathbf{s}}_h, \tilde{\boldsymbol{\xi}}_h^0\}_{h=1}^H$ with $\tilde{\boldsymbol{\xi}}_h^0 = \begin{bmatrix} \tilde{\mathbf{p}}_h \\ \tilde{\mathbf{q}}_h \end{bmatrix}$. Then, we transform the desired points $\tilde{\mathbf{D}}^0$ into Euclidean space, leading to $\tilde{\mathbf{D}}^s = \{\tilde{\mathbf{s}}_h, \tilde{\boldsymbol{\xi}}_h\}_{h=1}^H$ with $\tilde{\boldsymbol{\xi}}_h = \begin{bmatrix} \tilde{\mathbf{p}}_h \\ \log(\tilde{\mathbf{q}}_h * \bar{\mathbf{q}}_a) \end{bmatrix}$. In order to incorporate the adaptation precision, we can assign covariance matrices $\tilde{\boldsymbol{\Sigma}}_h$ for various transformed points $\tilde{\boldsymbol{\xi}}_h$. Thus, we have an additional reference

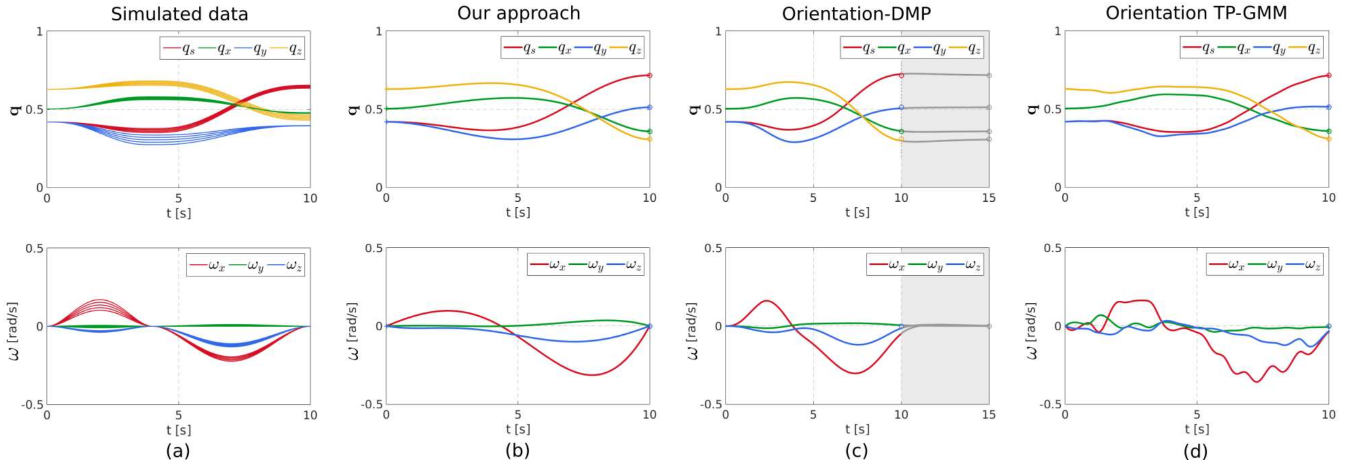


Fig. 1: Evaluations of various approaches on simulated examples. (*a*) shows simulated quaternion trajectories and their corresponding angular velocities. (*b*)–(*d*) display adapted quaternion trajectories towards new target (i.e., end-point) as well as the adapted angular velocities by using our approach (*b*), orientation-DMP [14], [15] (*c*) and orientation TP-GMM [3] (*d*). Note that for all approaches the desired movement duration is 10s, the shaded area in (*c*) denotes extra time required for DMP. The circles with bright colors denote desired quaternions and angular velocities, while the gray circles in (*c*) correspond to the delayed desired points.

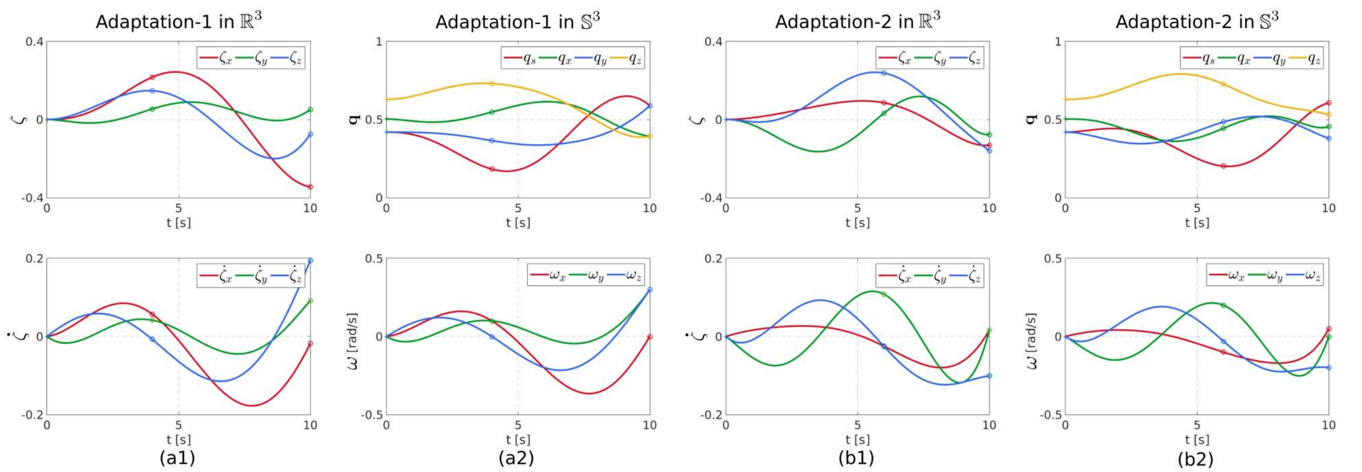


Fig. 2: Adaptations of quaternion and angular-velocity profiles with various constraints of desired points (depicted by circles), where (*a1*)–(*a2*) and (*b1*)–(*b2*) correspond to the first and second evaluations, respectively. Note that (*a1*) and (*b1*) represent adaptations in the transformed space $\mathbb{R}^3$ that is determined via (2), while (*a2*) and (*b2*) correspond to adaptations in $\mathbb{S}^3$ space.

trajectory that represents the transformed desired points in Euclidean space, i.e., $\tilde{\mathbf{D}}_r^s = \{\tilde{\mathbf{s}}_h, \tilde{\boldsymbol{\xi}}_h, \tilde{\boldsymbol{\Sigma}}_h\}_{h=1}^H$. According to the discussion in Section III-B, we can concatenate $\tilde{\mathbf{D}}_r^s$ with $\mathbf{D}_r^s = \{\mathbf{s}_n, \hat{\boldsymbol{\mu}}_n, \hat{\boldsymbol{\Sigma}}_n\}_{n=1}^N$, resulting in an extended reference trajectory $\mathbf{D}_r^{sU}$ that can be used to generate a 6-D trajectory[12] in Euclidean space and later recover the Cartesian trajectory (comprising Cartesian position and quaternion) that passes through various desired points defined by $\tilde{\mathbf{D}}^0$.

It is worth mentioning that, given the joint probability distribution $\mathcal{P}(\mathbf{s}, \boldsymbol{\xi})$ in (32), GMR can be employed to predict the corresponding output for a query input $\mathbf{s}^*$ through calculating $\mathcal{P}(\boldsymbol{\xi}|\mathbf{s}^*)$. However, GMR is only limited for task reproduction (i.e., reproducing demonstrations). When the adaptation problem is encountered, e.g., the predicted trajectory must pass through the desired points $\tilde{\mathbf{D}}^0$, GMR becomes inapplicable, since $\mathcal{P}(\mathbf{s}, \boldsymbol{\xi})$ (extracted from demonstrations) does not address the constraints from desired points. In contrast, within our framework, both reproduction and adaptation issues can be directly tackled by using $\mathbf{D}_r^s$ or $\mathbf{D}_r^{sU}$.

VI. Evaluations

In this section, we report several examples to illustrate the performance of our approach:

(*i*) orientation adaptation towards a desired target point (Section VI-A1), where orientation-DMP [14], [15] and orientation TP-GMM [3] are employed as comparisons[13];
(*ii*) orientation adaptations towards arbitrary desired points in terms of quaternions and angular velocities (Section VI-A2);
(*iii*) orientation adaptations with angular acceleration constraints (Section VI-B);
(*iv*) rhythmic orientation reproduction and adaptations (Section VI-C);
(*v*) concurrent adaptations of Cartesian position and orientation in a painting task (Section VI-D);
(*vi*) learning Cartesian trajectory with high-dimensional inputs in a human-robot collaboration scenario (Section VI-E).

[12]This trajectory passes through the transformed desired points $\tilde{\mathbf{D}}^s$.

[13]In this section, $\mathbf{q}$ is represented as $\mathbf{q} = [q_s \ q_x \ q_y \ q_z]^\top$.

TABLE III: Planned Errors of Our Approach and State-of-the-Art

	Quaternion distance error*		Angular velocity error	
	$t = 10s$**	$t = 15s$	$t = 10s$	$t = 15s$
Our approach	0	-	0.0017	-
Orientation–DMP [14], [15]	0.0285	0.0046	0.0513	0.0034
Orientation TP-GMM [3]	0.0085	-	0.0498	-

**Quaternion distance is calculated by* [14], [15]:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \begin{cases} 2\pi, & \mathbf{q}_1 * \bar{\mathbf{q}}_2 = [-1\,0\,0\,0]^\top \\ 2||\log(\mathbf{q}_1 * \bar{\mathbf{q}}_2)||, & otherwise. \end{cases}$$

***Note that the desired movement duration is* $10s$.

TABLE IV: Smoothness Costs of Our Approach and State-of-the-Art

	Quaternion smoothness cost c_q	Angular-velocity smoothness cost c_ω
Our approach	7.4326×10^{-4}	8.5820×10^{-4}
Orientation–DMP [14], [15]	7.7441×10^{-4}	9.3905×10^{-4}
Orientation TP-GMM [3]	7.8706×10^{-4}	1.6463×10^{-3}

**The smoothness costs (36)–(37) are evaluated between* $0s$ *and* $10s$.

The evaluations (*i*)–(*iv*) are verified in simulated examples while (*v*)–(*vi*) are carried out on real robots. Videos of the experimental evaluations as well as didactic codes are provided at *https://sites.google.com/view/quat-kmp*.

A. Evaluations on quaternion adaptations

We collected five simulated quaternion trajectories with time-length $10s$, as depicted in Fig. 1(a), where minimal jerk polynomial and renormalization are used to generate smooth and proper quaternion trajectories. In order to show the performance of our approach, we first compare it with orientation-DMP [14], [15] and orientation TP-GMM [3] in Section VI-A1. Subsequently, we evaluate our approach by adapting quaternions and angular velocities towards various desired points in Section VI-A2. The Gaussian kernel $k(t_i, t_j) = \exp(-\ell(t_i - t_j)^2)$ with $\ell = 0.01$ and the regularization factor $\lambda = 1$ are used in this section.

1) Comparison with state-of-the-art approaches: Since orientation-DMP is restricted to target (i.e., end-point) adaptation while having zero angular velocity at the ending point, we here consider an example with the desired point being $\tilde{t}_1 = 10s$, $\tilde{\mathbf{q}}_1 = [0.7172\ 0.3586\ 0.5123\ 0.3074]$, $\tilde{\boldsymbol{\omega}}_1 = [0\ 0\ 0]$. The auxiliary quaternion $\mathbf{q}_a$ is set as the initial value of simulated quaternion trajectories.

The evaluations of using our approach and orientation–DMP are provided in Fig. 1(b)–(c). It can be seen from Fig. 1(b) that our approach is capable of generalizing learned quaternion trajectories to the new target point $\tilde{\mathbf{q}}_1$ while having zero angular velocity at the ending time $\tilde{t}_1$. However, orientation–DMP needs extra time (depicted by shaded area) to converge to the desired point. Furthermore, we use orientation TP-GMM[14] to tackle the same target adaptation problem, whose adapted trajectories are shown in Fig. 1(d).

[14] A second-order linear dynamics model is employed together with TP-GMM towards obtaining smooth trajectories. Please refer to [3] for implementation details.

TABLE V: Angular Acceleration Costs Under Different λ_a

λ_a	10^1	10^2	10^3	10^4	10^5
$c_{\omega d}$	0.0307	0.0303	0.0273	0.0183	0.0140

The planned errors of three methods in comparison with the desired point is summarized in Table III, showing that our approach achieves the best performance in terms of adaptation precision. Note that the errors from DMP can possibly be further reduced by tuning the relevant parameters, e.g., the number of basis functions, bandwidth of basis functions and the length of time step, but here we present the best results we could obtain. In addition, we evaluate the smoothness of adapted quaternion and angular-velocity profiles, where the smoothness cost for quaternion is defined as

$$c_q = \frac{1}{N} \sum_{n=1}^{N-1} ||\mathbf{q}(t_{n+1}) - \mathbf{q}(t_n)|| \tag{36}$$

and the cost for angular velocity is

$$c_\omega = \frac{1}{N} \sum_{n=1}^{N-1} ||\boldsymbol{\omega}(t_{n+1}) - \boldsymbol{\omega}(t_n)||. \tag{37}$$

As can be seen in Table IV, our approach corresponds to the smallest costs in terms of both c_q and c_ω.

2) Adapting quaternion trajectory towards various desired points: Now, we consider a more challenging adaptation task that needs various desired points (i.e., via-/end- points) in terms of quaternion and angular velocity. Note that orientation-DMP [14], [15] and orientation TP-GMM [3] are not applicable in this case. Two groups of quaternion adaptations in $\mathbb{S}^3$, accompanied by the corresponding adaptations of the projected trajectories via (2) in $\mathbb{R}^3$, are shown in Fig. 2, showing that our approach indeed modulates quaternions and angular velocities to pass through various desired points (plotted by circles).

B. Evaluations on quaternion adaptations with angular acceleration constraints

In this section, we consider quaternion adaptations with angular acceleration constraints, where the same simulated demonstrations, as plotted in Fig. 1(a), are employed. Specifically, we aim to adapt quaternion profile, while taking into account the angular acceleration constraints. In order to quantitatively show the performance of our approach, we define the angular acceleration cost as

$$c_{\omega d} = \frac{1}{N} \sum_{n=1}^{N} ||\dot{\boldsymbol{\omega}}(t_n)||^2 \tag{38}$$

and meanwhile a group of penalty parameters λ_a are used. We use Gaussian kernel for the evaluations. Other relevant parameters are set as $\ell = 0.01$ and $\lambda = 1$.

The evolved trajectories of quaternion and angular velocity are depicted in Fig. 3, where the color changes from light to dark as λ_a increases. Note that the evolved trajectories pass through various desired points (depicted by circles) precisely. The corresponding angular acceleration costs with different λ_a

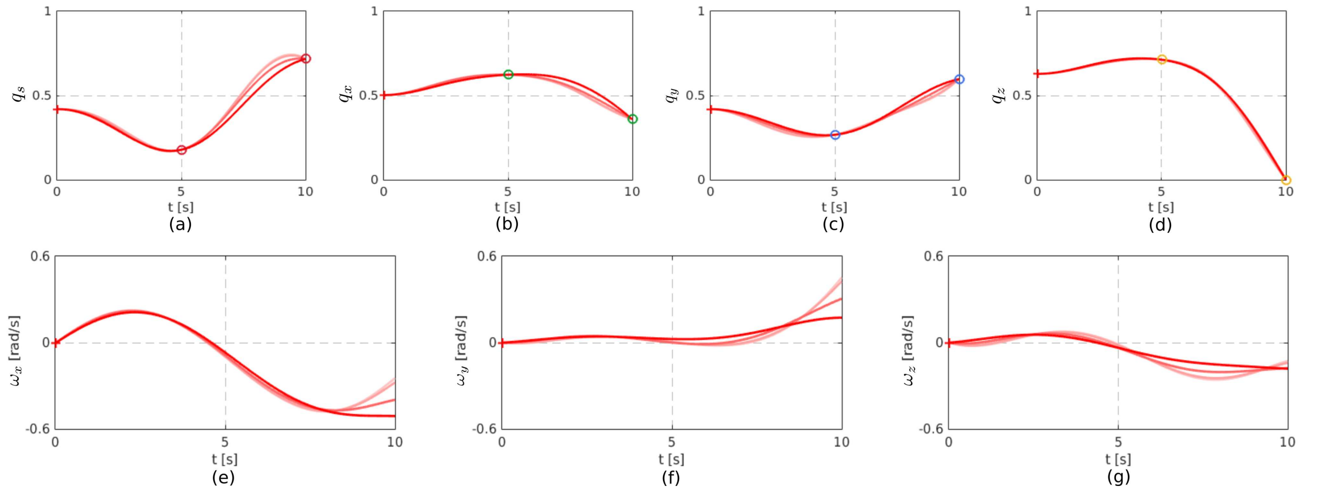


Fig. 3: Orientation adaptations with angular acceleration constraints, where quaternion profiles are adapted towards various desired points (depicted by colorful circles). The cross '+' represents the starting point of trajectories. The solid curves correspond to different values of λ_a, with color that switches from light red to dark red corresponding to the increasing direction of λ_a.

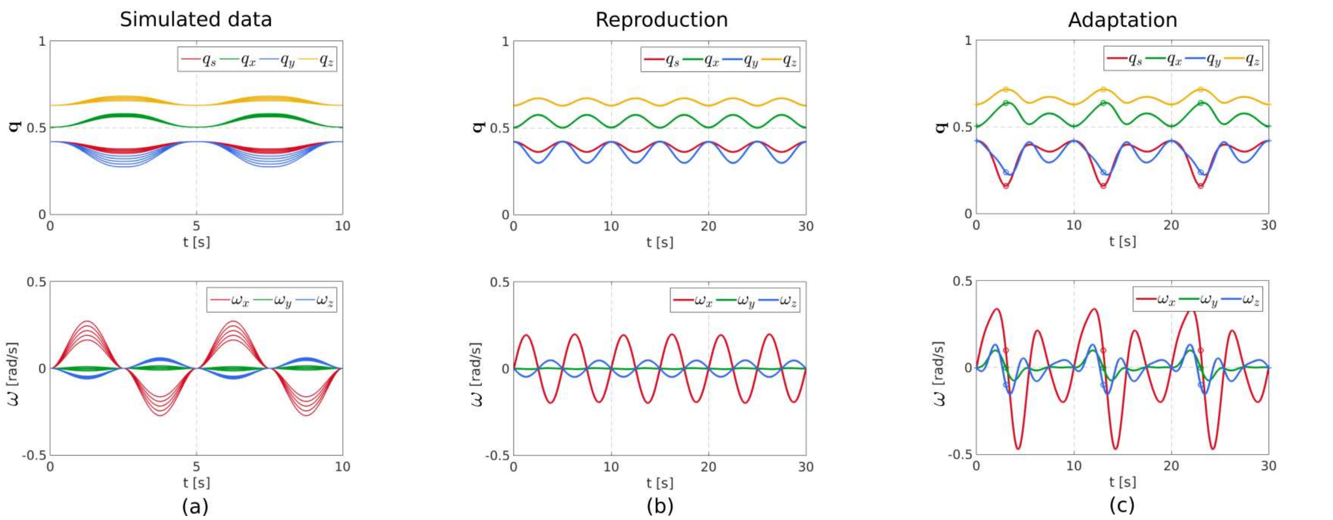


Fig. 4: Reproduction and adaptation of rhythmic orientation trajectories by using our approach. (a) plots simulated quaternions and their corresponding angular velocities, where the motion period is $10s$. (b) and (c) show orientation reproduction and adaptation over three periods (i.e., $30s$), respectively. Circles in (c) denote the desired quaternion and angular velocity.

are provided in Table V, representing that $c_{\omega d}$ decreases as λ_a increases (which indeed coincides with our interpretation of the penalty coefficient). Thus, we can conclude that our approach is capable of adapting quaternions towards various desired points while incorporating angular acceleration constraints.

C. Evaluations on rhythmic quaternion trajectories

Differing from the aforementioned examples on point-to-point quaternions, we here test our approach on rhythmic quaternion trajectories. Note that rhythmic quaternions are very important in many orientation-sensitive tasks, such as screwing a lid off the bottle and wiping a surface. Similarly to Section VI-A, we use polynomials to generate five demonstrations (each lasts for $10s$) for training our approach, as shown in Fig. 4(a). The periodic kernel [36] $k(t_i, t_j) = \exp(-\ell sin^2(\frac{t_i - t_j}{T}\pi))$ with $\ell = 0.4$ and $T = 10s$ is employed. The regularization factor is set to be $\lambda = 10$. In this section, the angular acceleration constraints are not considered, i.e., $\lambda_a = 0$, but one can easily incorporate these constraints into rhythmic movements.

We first consider the reproduction capability of our approach, where quaternion and angular-velocity profiles over three periods (i.e., $30s$) are generated. It can be seen from Fig. 4(b) that our method can reproduce trajectories that maintain the shape of demonstrations and meanwhile exhibit rhythmic properties. Second, we test the adaptation capability of our method by imposing a via-point constraint at $t = 3s$. The adapted trajectories over three periods are given in Fig. 4(c), where the quaternion and angular-velocity trajectories are modulated towards the via-point (depicted by circles) in each period. Moreover, the rhythmic property is kept in this adaptation case.

D. Evaluations of learning time-driven Cartesian trajectories on real robot

We here consider a painting task that requires the real Barrett WAM robot to paint different areas with proper orientations. Through kinesthetic teaching (*first row* in Fig. 5),

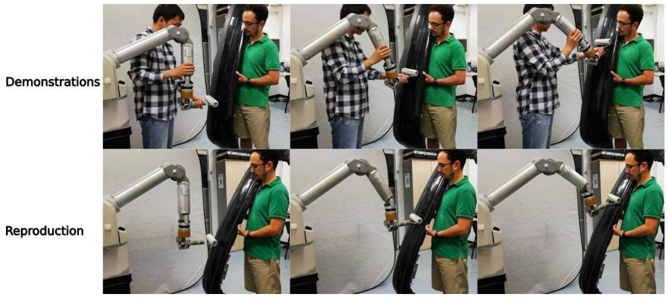


Fig. 5: Painting task on the real Barrett WAM robot. *First row* shows kinesthetic teaching of the painting task. *Second row* represents the task reproduction.

six demonstrations comprising time, Cartesian position and quaternion are recorded, as shown in Fig. 6(a). We first apply our approach to task reproduction, i.e., without incorporating the desired points. The auxiliary quaternion $\mathbf{q}_a$ is set as the initial value of demonstrations. Gaussian kernel with $\ell = 0.001$ is used and $\lambda = 1$. The reproduced Cartesian trajectory (solid curves) and its corresponding measured trajectory (dashed curves) are shown in Fig. 6(b). Snapshots of task reproduction are provided in Fig. 5 (*second row*), where we can see that the robot is capable of reproducing a similar task to that demonstrated by the human.

Now, we consider two groups of adaptation evaluations and, in each group, demonstrated Cartesian trajectories are modulated towards two unseen desired points (i.e., via-point and end-point). The adapted Cartesian trajectories are shown in Fig. 6(c)-(d), where the planned trajectories (solid curves) and real measured trajectories (dashed curves) are provided. It can be seen that the planned trajectories are capable of meeting various constraints, i.e., Cartesian position and quaternion constraints. More explanations of the adaptation evaluations are provided in our previous work [1].

E. Evaluations of learning Cartesian trajectories with high-dimensional inputs on real robot

In this section, we consider a human-robot handover task, where the robot moves towards the human user in order to accomplish the handover task. Specifically, we consider human hand position[15] as inputs while robot end-effector position and orientation as outputs. It is worth emphasizing that we aim to predict robot Cartesian state (6-D[16]) in accordance to human hand state (3-D) directly, without any additional operations like phase-estimation [31], [32].

We collect five demonstrations in terms of human hand position (red curves), as well as robot end-effector position (black curves) and quaternion, as show in Fig. 7. Note that the transformed trajectories of quaternions (yellow curves) via (2) are plotted for the sake of visualization. Then, by following the description in Section V-B, we can generate a reference trajectory (associated with high-dimensional inputs) for training our approach. The auxiliary quaternion $\mathbf{q}_a$ is set to be $\mathbf{q}_a = [1\,0\,0\,0]^{\top}$. The Gaussian kernel is used and the related parameters are $\ell = 1$ and $\lambda = 2$.

[15] An optical tracker is used to measure human hand position.

[16] We here refer to quaternion as a 3-D variable due to the norm constraint, albeit that our approach predicts four elements of quaternion simultaneously.

In order to evaluate our approach, we first consider a reproduction task and subsequently an adaptation task where a new handover location is needed. Figure 8 depicts human hand trajectory (dashed red curve) and the corresponding robot Cartesian trajectory planned by our approach (dashed black and yellow curves) in the reproduction case, where Fig. 8(a) and Fig. 8(b) correspond to positions and transformed quaternion data, respectively. In addition, in Fig. 9(a) the human hand positions, robot Cartesian positions and orientations (in terms of frames) are depicted together. It can be seen that the robot accomplishes the handover task when human hand trajectory resembles the demonstrated ones.

Now, we apply our approach to the adaptation situation, where the handover takes place at a new point that is unseen in demonstrations. This adaptation can be achieved by adding a desired point $\{\tilde{\mathbf{s}}_1, \tilde{\mathbf{p}}_1, \tilde{\mathbf{q}}_1\}$ into the original reference trajectory, where $\tilde{\mathbf{s}}_1 = \tilde{\mathbf{p}}_1 = \mathbf{p}^{new}$ and $\tilde{\mathbf{q}}_1 = \mathbf{q}^{new}$, ensuring that the robot reaches the new handover location $\mathbf{p}^{new}$ with desired quaternion $\mathbf{q}^{new}$ when the human hand arrives at $\mathbf{p}^{new}$. The adaptation evaluation is provided in Fig. 8, where the solid red curve denotes the user hand trajectory while the solid black and yellow curves correspond to the planned Cartesian trajectory for the robot. Again, we here only provide the transformed data of quaternions for the sake of easy observation. Similarly to the reproduction case, we represent human hand positions, robot Cartesian positions and orientations (in terms of frames) into a single plot, as shown in Fig. 9(b). By observing Fig. 8 and 9(b), we can find that robot trajectory is indeed modulated according to the user hand position, leading to a successful handover at the new location. Snapshots of kinesthetic teaching of handover task as well as reproduction and adaptation evaluations are shown in Fig. 10. Thus, our approach is effective in both reproduction and adaptation cases while considering high-dimensional inputs.

VII. Discussion

In this section, we discuss some related work on learning time-driven demonstrations with via-point constraints, as well as learning demonstrations comprising high-dimensional inputs (Section VII-A). Then, we discuss limitations and possible extensions of our approach (Section VII-B).

A. Related work

The topic of via-point adaptation has been the focus of a few works in the imitation learning literature, e.g., [6], [37], [38]. In [6], Gaussian conditioning operation was used in ProMP to address the via-point issue. In [37], the task-parameterized DMP was studied, where the via-point constraint was handled as a task parameter vector. However, both [6] and [37] did not take into account the Cartesian orientation. Please note that [37] in essence focuses on learning time-driven demonstrations (i.e., demonstrations comprising 1-D time input), albeit that the task parameters that describe the condition of demonstrations could be high-dimensional. Similarly, within the DMP framework, a time-varying target function was formulated in [38] to incorporate via-points in terms of Cartesian position, while the orientation was missing.

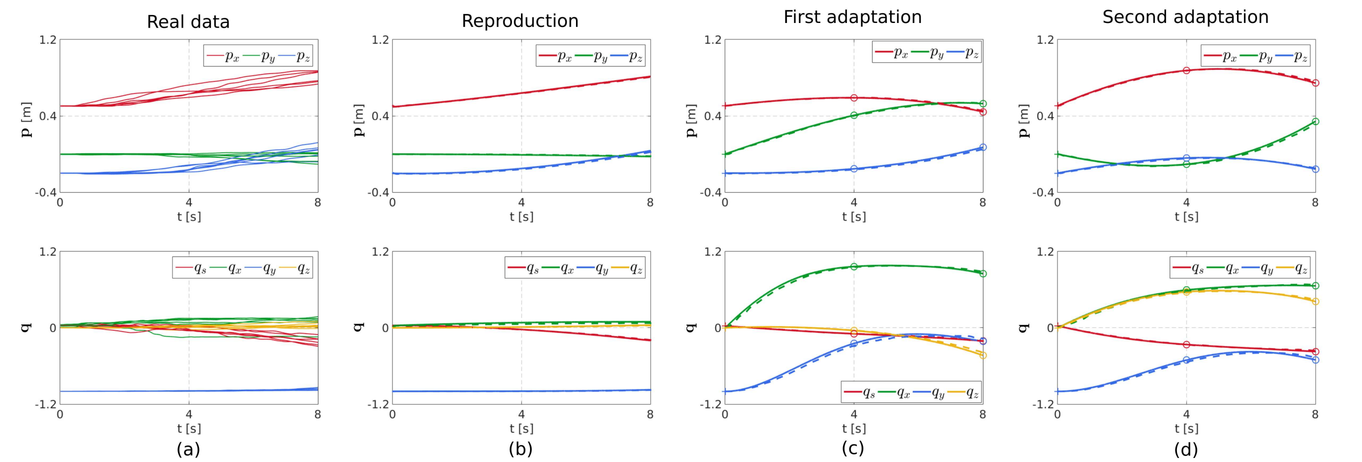


Fig. 6: Evaluations of our approach through a painting task on the real Barrett WAM robot. (*a*) shows demonstrated Cartesian positions and quaternions in the painting task. (*b*) represents Cartesian trajectory in task reproduction. (*c*)–(*d*) correspond to adapted Cartesian trajectories with various desired points. In plots (*b*)–(*d*), solid curves represent planned trajectories by using our approach while dashed curves denote real measured trajectories. Circles depict desired Cartesian positions and quaternions.

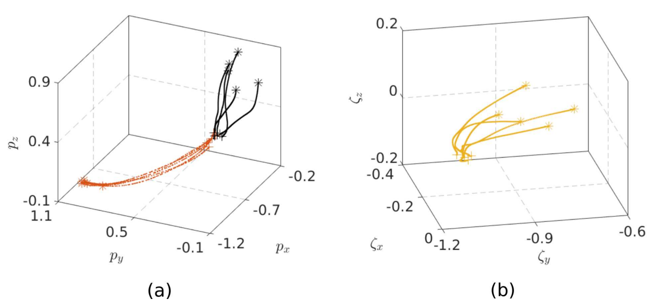


Fig. 7: Demonstrated human hand trajectories and robot Cartesian trajectories in a human-robot handover task. (*a*) shows human hand positions (red curves) and robot Cartesian positions (black curves). For the sake of visualization, the transformed trajectories of robot quaternions are plotted in (*b*).

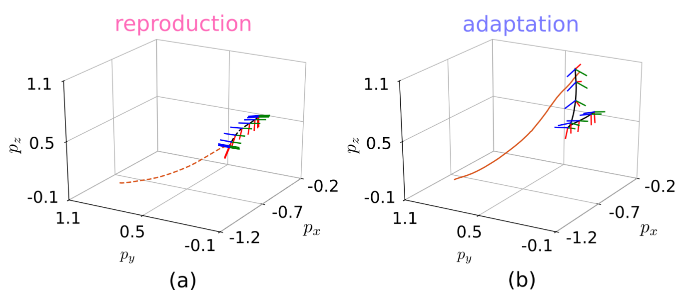


Fig. 9: Illustration of human hand trajectory and robot Cartesian trajectory in reproduction (*a*) and adaptation (*b*) evaluations, where the red curves denote human hand positions, the black curves denote the end-effector positions and frames correspond to its orientations.

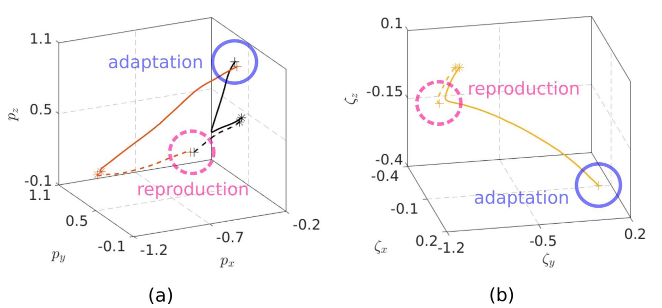


Fig. 8: Reproduction (dashed curves) and adaptation (solid curves) evaluations in the handover task. (*a*) represents human hand positions (red curves) and robot Cartesian positions (black curves). (*b*) plots the transformed trajectories of robot quaternions via (2).

In order to cope with the via-quaternion and via-angular velocity constraints, the strategy of sequencing different DMPs was proposed in [18]. To take the sequence of two DMPs as an example, in [18] the first DMP was used to plan a trajectory from the starting quaternion to the via-quaternion, and subsequently the second DMP was used to generate a trajectory from the via-quaternion to the target quaternion. Note that these two DMPs were trained by using different parts of a demonstration. Differing from [18] where the demonstration needs to be segmented for each movement primitive, we propose to learn and adapt the entire demonstrations using a single movement primitive, where the segmentation of demonstrations and the sequence of multiple motion primitives are not required. Specifically, in contrast to [37], [38], [18] where each DMP corresponds to a single training trajectory, we study imitation learning from a probabilistic perspective and exploit the consistent features underlying multiple demonstrations. As a result, including via-points in our approach can be done in a rather straightforward way by simply defining the via-point, its associated input and the desired precision in the form of a covariance matrix.

Various imitation learning approaches (e.g., DMP [5] and ProMP [6]) have been employed in human-robot collaboration [31], [32]. However, the majority of these works model human motion and robot motion with time (i.e., learning demonstrations with time input), which will lead to the synchronization issue (see Section V-A). Note that DMP and ProMP explicitly depend on basis functions, which are non-trivial to extend to learn demonstrations comprising high-dimensional inputs, due to the *curse of dimensionality*. As discussed in [27], the number of basis functions often increases significantly as the dimension of inputs increases. In addition, the process of defining proper parameters for basis functions over high-dimensional state is cumbersome. For instance, the definition of a multivariate Gaussian basis function requires a center vector and a covariance matrix. In contrast, our approach introduces the kernel trick (see (35)) and thus basis functions are not needed, yielding a non-parametric solution.

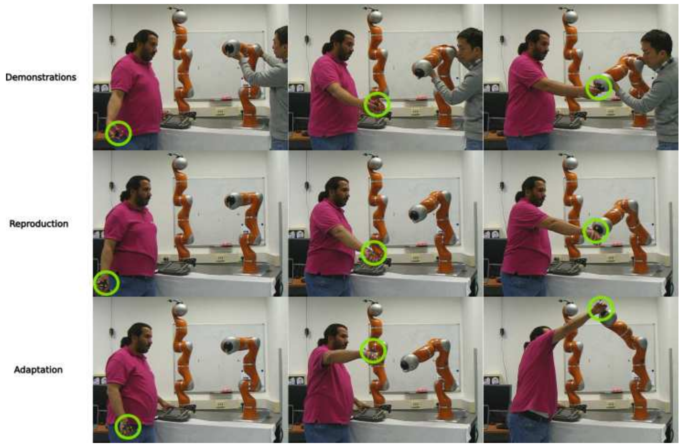


Fig. 10: Handover task on the real KUKA robot. *First row* shows kinesthetic teaching of the handover task. *Second row* and *third row* represent the reproduction and adaptation evaluations, respectively. Green circle corresponds to the optical tracker which measures the user's hand position on-line.

Moreover, note that GMM/GMR based approaches [16] are capable of learning demonstrations comprising high-dimensional inputs, whereas the adaptation feature is not provided. In order to endow GMM/GMR with the adaptation capability, the task parameterized treatment (i.e., TP-GMM) was studied in [3], [17]. However, TP-GMM suffers from two main limitations: (*i*) for the case of adapting demonstrations with multiple inputs, TP-GMM is restricted to target adaptation, where the via-point adaptation is not allowed; (*ii*) for the case of adapting demonstrations with time input, TP-GMM is unable to deal with the via-point and angular-velocity constraints (see also the discussion in Section VI-A2). Unlike TP-GMM, our approach can handle the adaptation issue with arbitrary desired points (including via-/end- points), as well as the angular velocity constraints.

Finally, it is worth mentioning that, as an extension of [14], [15], in [19] the orientation DMP was used for generalizing demonstrations towards different contextual (or conditional, task-specific) states. Specifically, in the training phase, each fixed contextual state[17] corresponds to one demonstration that consists of a time sequence (i.e., 1-D input) and its associated robot trajectory (i.e., outputs), and in the evaluation phase, the new contextual state is used to predict the corresponding DMP parameters that, subsequently, can be used to generate the entire robot trajectory. Thus, the work in [19] can be viewed as learning of demonstrations with time input, while considering additional contextual states. Differing from [19], in our framework (see Section V), each demonstration comprises a high-dimensional varying input trajectory and a robot trajectory, and in the evaluation phase, the high-dimensional inputs are employed to directly predict the corresponding robot actions (see the hand-over experiment, where the 3-D user's hand position is utilized to predict the 6-D robot Cartesian state). As we have shown, our approach alleviates the synchronization issue, allowing the robot to react in run-time to the human behavior that changes arbitrarily (e.g. with different speed). The adaptation capabilities of our approach complement this feature by allowing the robot to react, even to human behaviors that were not explicitly shown.

[17]The contextual state in [19] can be interpreted as the condition under which the whole time-driven demonstration has taken place.

B. Limitations and extensions of our approach

As explained in Section II, the learning and adaptation of quaternions are carried out in Euclidean space, where the $\log(\cdot)$ mapping in (1) is used to transform quaternions into Euclidean space. In order to guarantee that similar quaternions $\mathbf{q}$ correspond to similar $\boldsymbol{\zeta}$ in Euclidean space, we have imposed the Assumption 1. This assumption may restrict the applications of our method. For example, when the demonstrated quaternion trajectories differ from each other dramatically, the Assumption 1 could be violated, potentially invalidating the teaching of highly dynamic motions.

It is noted that we only focus on the prediction of quaternion profile in this paper. In fact, we can also predict the covariance (i.e., 3×3 matrix) of $\boldsymbol{\zeta}$ in $\mathbb{R}^3$, similarly to [8]. Indeed, specifying a covariance matrix directly in $\mathbb{S}^3$ is not possible, with other state-of-the-art approaches following the direction of representing variability/correlation in Euclidean spaces with 3×3 matrix [16], [17]. Interestingly, some recent works on exploiting the covariance of trajectories and optimal control were reported, e.g., uncertainty-aware controller [11] and minimum intervention controller [39]. Thus, it would be useful to integrate these ideas with our framework so as to perform orientation tasks in a safe and user-friendly way. In addition, we set kernel parameters experimentally in our evaluations. Despite the definition of the parameters being relatively straightforward (intuition about the kernel width can be derived from the order magnitude of the inputs), as an extension of our work, we plan to provide a theoretical guidance for choosing kernel parameters.

VIII. Conclusions

In this paper, we proposed an analytical approach for adapting quaternion and angular velocity towards arbitrary desired points. In addition, our method is capable of incorporating angular acceleration or jerk constraints. In comparison with previous works (e.g., [14], [15]) that mostly focus on orientation adaptation towards target points, our work allows for broader applications, particularly when both quaternion and angular velocity need to be modulated. Moreover, our approach is capable of learning quaternions associated with high-dimensional inputs (e.g., 3-D inputs were used in the real handover task), which is a quite desirable property in human-robot collaboration.

Acknowledgement

We thank anonymous reviewers for their constructive and helpful comments on this paper.

Appendix A
Kernel Derivation Under Angular Acceleration Constraints

According to the definitions of $\boldsymbol{\Theta}(t)$ and $\boldsymbol{\varphi}(t)$, i.e., (10) and (27), we have

$$
\begin{aligned}
\boldsymbol{\Omega}^{\top}(t_i)\boldsymbol{\Omega}(t_j) &= \begin{bmatrix} \boldsymbol{\Theta}^{\top}(t_i)\boldsymbol{\Theta}(t_j) & \boldsymbol{\Theta}^{\top}(t_i)\boldsymbol{\varphi}(t_j) \\ \boldsymbol{\varphi}^{\top}(t_i)\boldsymbol{\Theta}(t_j) & \boldsymbol{\varphi}^{\top}(t_i)\boldsymbol{\varphi}(t_j) \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{\phi}^{\top}(t_i)\boldsymbol{\phi}(t_j)\mathbf{I}_3 & \boldsymbol{\phi}^{\top}(t_i)\dot{\boldsymbol{\phi}}(t_j)\mathbf{I}_3 & \boldsymbol{\phi}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j)\mathbf{I}_3 \\ \dot{\boldsymbol{\phi}}^{\top}(t_i)\boldsymbol{\phi}(t_j)\mathbf{I}_3 & \dot{\boldsymbol{\phi}}^{\top}(t_i)\dot{\boldsymbol{\phi}}(t_j)\mathbf{I}_3 & \dot{\boldsymbol{\phi}}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j)\mathbf{I}_3 \\ \ddot{\boldsymbol{\phi}}^{\top}(t_i)\boldsymbol{\phi}(t_j)\mathbf{I}_3 & \ddot{\boldsymbol{\phi}}^{\top}(t_i)\dot{\boldsymbol{\phi}}(t_j)\mathbf{I}_3 & \ddot{\boldsymbol{\phi}}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j)\mathbf{I}_3 \end{bmatrix}
\end{aligned} \tag{39}
$$

It is well known that we can write $\boldsymbol{\phi}^{\top}(t_i)\boldsymbol{\phi}(t_j) = k(t_i, t_j)$ [27]. However, when we calculate $\boldsymbol{\Omega}^{\top}(t_i)\boldsymbol{\Omega}(t_j)$ in (39), the terms $\boldsymbol{\phi}^{\top}(t_i)\dot{\boldsymbol{\phi}}(t_j)$, $\boldsymbol{\phi}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j)$, $\dot{\boldsymbol{\phi}}^{\top}(t_i)\boldsymbol{\phi}(t_j)$, $\dot{\boldsymbol{\phi}}^{\top}(t_i)\dot{\boldsymbol{\phi}}(t_j)$, $\dot{\boldsymbol{\phi}}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j)$, $\ddot{\boldsymbol{\phi}}^{\top}(t_i)\boldsymbol{\phi}(t_j)$, $\ddot{\boldsymbol{\phi}}^{\top}(t_i)\dot{\boldsymbol{\phi}}(t_j)$, $\ddot{\boldsymbol{\phi}}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j)$ are also encountered. We here propose to approximate $\dot{\boldsymbol{\phi}}(t)$ and $\ddot{\boldsymbol{\phi}}(t)$, i.e.,

$$
\dot{\boldsymbol{\phi}}(t) \approx \frac{\boldsymbol{\phi}(t+\delta) - \boldsymbol{\phi}(t)}{\delta} \text{ and}
$$

$$
\ddot{\boldsymbol{\phi}}(t) \approx \frac{\dot{\boldsymbol{\phi}}(t+\delta) - \dot{\boldsymbol{\phi}}(t)}{\delta} \approx \frac{\boldsymbol{\phi}(t+2\delta) - 2\boldsymbol{\phi}(t+\delta) + \boldsymbol{\phi}(t)}{\delta^2},
$$

where $\delta > 0$ denotes a small constant. By using these approximations, (39) can be kernelized.

To take $\ddot{\boldsymbol{\phi}}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j)$ as an example, we have

$$
\begin{aligned}
\ddot{\boldsymbol{\phi}}^{\top}(t_i)\ddot{\boldsymbol{\phi}}(t_j) &= \left(\frac{\boldsymbol{\phi}^{\top}(t_i+2\delta) - 2\boldsymbol{\phi}^{\top}(t_i+\delta) + \boldsymbol{\phi}^{\top}(t_i)}{\delta^2}\right) \\
&\quad \left(\frac{\boldsymbol{\phi}(t_j+2\delta) - 2\boldsymbol{\phi}(t_j+\delta) + \boldsymbol{\phi}(t_j)}{\delta^2}\right) \\
&= \frac{\boldsymbol{\phi}^{\top}(t_i+2\delta)\boldsymbol{\phi}(t_j+2\delta) - 2\boldsymbol{\phi}^{\top}(t_i+2\delta)\boldsymbol{\phi}(t_j+\delta) + \boldsymbol{\phi}^{\top}(t_i+2\delta)\boldsymbol{\phi}(t_j)}{\delta^4} \\
&\quad - \frac{2\boldsymbol{\phi}^{\top}(t_i+\delta)\boldsymbol{\phi}(t_j+2\delta) - 4\boldsymbol{\phi}^{\top}(t_i+\delta)\boldsymbol{\phi}(t_j+\delta) + 2\boldsymbol{\phi}^{\top}(t_i+\delta)\boldsymbol{\phi}(t_j)}{\delta^4} \\
&\quad + \frac{\boldsymbol{\phi}^{\top}(t_i)\boldsymbol{\phi}(t_j+2\delta) - 2\boldsymbol{\phi}^{\top}(t_i)\boldsymbol{\phi}(t_j+\delta) + \boldsymbol{\phi}^{\top}(t_i)\boldsymbol{\phi}(t_j)}{\delta^4} \\
&= \frac{k(t_i+2\delta, t_j+2\delta) - 2k(t_i+2\delta, t_j+\delta) + k(t_i+2\delta, t_j)}{\delta^4} \\
&\quad - \frac{2k(t_i+\delta, t_j+2\delta) - 4k(t_i+\delta, t_j+\delta) + 2k(t_i+\delta, t_j)}{\delta^4} \\
&\quad + \frac{k(t_i, t_j+2\delta) - 2k(t_i, t_j+\delta) + k(t_i, t_j)}{\delta^4}.
\end{aligned}
$$

Therefore, (39) can be kernelized as

$$
\mathbf{k}(t_i, t_j) = \begin{bmatrix} k_{tt}(t_i,t_j)\mathbf{I}_3 & k_{td}(t_i,t_j)\mathbf{I}_3 & k_{ta}(t_i,t_j)\mathbf{I}_3 \\ k_{dt}(t_i,t_j)\mathbf{I}_3 & k_{dd}(t_i,t_j)\mathbf{I}_3 & k_{da}(t_i,t_j)\mathbf{I}_3 \\ k_{at}(t_i,t_j)\mathbf{I}_3 & k_{ad}(t_i,t_j)\mathbf{I}_3 & k_{aa}(t_i,t_j)\mathbf{I}_3 \end{bmatrix}, \tag{40}
$$

where

$$
\begin{aligned}
k_{tt}(t_i,t_j) &= k(t_i,t_j), \\
k_{td}(t_i,t_j) &= \big(k(t_i,t_j+\delta) - k(t_i,t_j)\big)/\delta, \\
k_{ta}(t_i,t_j) &= \big(k(t_i,t_j+2\delta) - 2k(t_i,t_j+\delta) + k(t_i,t_j)\big)/\delta^2, \\
k_{dt}(t_i,t_j) &= \big(k(t_i+\delta,t_j) - k(t_i,t_j)\big)/\delta, \\
k_{dd}(t_i,t_j) &= \big(k(t_i+\delta,t_j+\delta) - k(t_i,t_j+\delta) - k(t_i+\delta,t_j) \\
&\quad + k(t_i,t_j)\big)/\delta^2, \\
k_{da}(t_i,t_j) &= \big(k(t_i+\delta,t_j+2\delta) - 2k(t_i+\delta,t_j+\delta) \\
&\quad + k(t_i+\delta,t_j) - k(t_i,t_j+2\delta) + 2k(t_i,t_j+\delta) \\
&\quad - k(t_i,t_j)\big)/\delta^3, \\
k_{at}(t_i,t_j) &= \big(k(t_i+2\delta,t_j) - 2k(t_i+\delta,t_j) + k(t_i,t_j)\big)/\delta^2, \\
k_{ad}(t_i,t_j) &= \big(k(t_i+2\delta,t_j+\delta) - 2k(t_i+\delta,t_j+\delta) \\
&\quad + k(t_i,t_j+\delta) - k(t_i+2\delta,t_j) + 2k(t_i+\delta,t_j) \\
&\quad - k(t_i,t_j)\big)/\delta^3, \\
k_{aa}(t_i,t_j) &= \big(k(t_i+2\delta,t_j+2\delta) - 2k(t_i+2\delta,t_j+\delta) \\
&\quad + k(t_i+2\delta,t_j) - 2k(t_i+\delta,t_j+2\delta) \\
&\quad + 4k(t_i+\delta,t_j+\delta) - 2k(t_i+\delta,t_j) \\
&\quad + k(t_i,t_j+2\delta) - 2k(t_i,t_j+\delta) + k(t_i,t_j)\big)/\delta^4.
\end{aligned}
$$

REFERENCES

[1] Y. Huang, F. J. Abu-Dakka, J. Silvério, and D. G. Caldwell, "Generalized orientation learning in robot task space," in *Proc. International Conference on Robotics and Automation*, 2019, pp. 2531–2537.

[2] Y. Huang, B. Schölkopf, and J. Peters, "Learning optimal striking points for a ping-pong playing robot," in *Proc. International Conference on Intelligent Robots and Systems*, 2015, pp. 4587–4592.

[3] J. Silvério, L. Rozo, S. Calinon, and D. G. Caldwell, "Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems," in *Proc. International Conference on Intelligent Robots and Systems*, 2015, pp. 464–470.

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

[5] 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.

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

[7] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[8] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[9] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters, "Demonstration based trajectory optimization for generalizable robot motions," in *Proc. International Conference on Humanoid Robots*, 2016, pp. 515–522.

[10] Y. Zhou and T. Asfour, "Task-oriented generalization of dynamic movement primitive," in *Proc. International Conference on Intelligent Robots and Systems*, 2017, pp. 3202–3209.

[11] J. Silvério, Y. Huang, F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, "Uncertainty-aware imitation learning using kernelized movement primitives," in *Proc. International Conference on Intelligent Robots and Systems*, 2019, pp. 90–97.

[12] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 763–768.

[13] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *Proc. International Conference on Intelligent Robots and Systems*, 2011, pp. 365–371.

[14] A. Ude, B. Nemec, T. Petrić, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *Proc. International Conference on Robotics and Automation*, 2014, pp. 2997–3004.

[15] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, and A. Ude, "Adaptation of manipulation skills in physical contact with the environment to reference force profiles," *Autonomous Robots*, vol. 39, no. 2, pp. 199–217, 2015.

[16] S. Kim, R. Haschke, and H. Ritter, "Gaussian mixture model for 3-dof orientations," *Robotics and Autonomous Systems*, vol. 87, pp. 28–37, 2017.

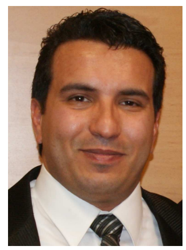
[17] M. J. Zeestraten, I. Havoutis, J. Silvério, S. Calinon, and D. G. Caldwell, "An approach for imitation learning on riemannian manifolds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1240–1247, 2017.

[18] M. Saveriano, F. Franzel, and D. Lee, "Merging position and orientation motion primitives," in *Proc. International Conference on Robotics and Automation*, 2019, pp. 7041–7047.

[19] A. Kramberger, A. Gams, B. Nemec, D. Chrysostomou, O. Madsen, and A. Ude, "Generalization of orientation trajectories and force-torque profiles for robotic assembly," *Robotics and Autonomous Systems*, vol. 98, pp. 333–346, 2017.

[20] O. Koç and J. Peters, "Learning to serve: an experimental study for a new learning from demonstrations framework," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1784–1791, 2019.
[21] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. International Conference on Robotics and Automation*, 2009, pp. 489–494.
[22] M. Muhlig, M. Gienger, S. Hellbach, J. J. Steil, and C. Goerick, "Task-level imitation learning using variance-based movement optimization," in *Proc. International Conference on Robotics and Automation*, 2009, pp. 1177–1184.
[23] E. Todorov and M. I. Jordan, "Optimal feedback control as a theory of motor coordination," *Nature Neuroscience*, vol. 5, no. 11, pp. 1226–1235, 2002.
[24] J. R. Medina, D. Lee, and S. Hirche, "Risk-sensitive optimal feedback control for haptic assistance," in *Proc. International Conference on Robotics and Automation*, 2012, pp. 1025–1031.
[25] Y. Huang, J. Silvério, and D. G. Caldwell, "Towards minimal intervention control with competing constraints," in *Proc. International Conference on Intelligent Robots and Systems*, 2018, pp. 733–738.
[26] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.
[27] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[28] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proc. International Conference on Machine Learning*, 1998, pp. 515–521.
[29] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.
[30] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *Proc. International Joint Conference on Artificial Intelligence*, 2011, pp. 2650–2655.
[31] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction primitives for human-robot cooperation tasks," in *Proc. International Conference on Robotics and Automation*, 2014, pp. 2831–2837.
[32] G. Maeda, M. Ewerton, G. Neumann, R. Lioutikov, and J. Peters, "Phase estimation for fast action recognition and trajectory generation in human–robot collaboration," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1579–1594, 2017.
[33] M. Ewerton, G. Neumann, R. Lioutikov, H. B. Amor, J. Peters, and G. Maeda, "Learning multiple collaborative tasks with a mixture of interaction primitives," in *Proc. International Conference on Robotics and Automation*, 2015, pp. 1535–1542.
[34] J. Silvério, Y. Huang, L. Rozo, S. Calinon, and D. G. Caldwell, "Probabilistic learning of torque controllers from kinematic and force constraints," in *Proc. International Conference on Intelligent Robots and Systems*, 2018, pp. 1–8.
[35] J. R. Hershey and P. A. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *Proc. International Conference on Acoustics, Speech and Signal Processing*, 2007, pp. 317–320.
[36] D. Duvenaud, "Automatic model construction with gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.
[37] F. Stulp, G. Raiola, A. Hoarau, S. Ivaldi, and O. Sigaud, "Learning compact parameterized skills with a single regression," in *Proc. International Conference on Humanoid Robots*, 2013, pp. 417–422.
[38] R. Weitschat and H. Aschemann, "Safe and efficient human–robot collaboration part ii: Optimal generalized human-in-the-loop real-time motion generation," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3781–3788, 2018.
[39] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *Proc. International Conference on Robotics and Automation*, 2014, pp. 3339–3344.

Yanlong Huang is a university academic fellow at the school of computing, University of Leeds. He received his BSc degree (2008) in Automatic Control and MSc degree (2010) in Control Theory and Control Engineering, both from Nanjing University of Science and Technology, Nanjing, China. After that, he received his PhD degree (2013) in Robotics from the Institute of Automation, Chinese Academy of Sciences, Beijing, China. From 2013 to 2019, he carried out his research as a postdoctoral researcher in Max-Planck Institute for Intelligent Systems and Italian Institute of Technology. His interests include imitation learning, optimal control, reinforcement learning, motion planning and their applications to robotic systems.



Fares J. Abu-Dakka received his B.Sc.degree in mechanical engineering from Birzeit University, Palestine, in 2003, and his M.Sc. and Ph.D. degrees in robotics motion planning from the Polytechnic University of Valencia, Spain, in 2006 and 2011, respectively. Currently, he is a senior researcher at Intelligent Robotics Group at the Department of Electrical Engineering and Automation (EEA), Aalto University, Finland. Before that he was researching at ADVR, Istituto Italiano di Tecnologia (IIT). Between 2013 and 2016 he was holding a visiting professor position at the Department of Systems Engineering and Automation of the Carlos III University of Madrid, Spain. His research activities include robot control and learning, human-robot interaction, impedance control, and robot motion planning.

João Silvério is a postdoctoral researcher at the Idiap Research Institute since July 2019. He received his M.Sc in Electrical and Computer Engineering (2011) from Instituto Superior Técnico (Lisbon, Portugal) and Ph.D in Robotics (2017) from the University of Genoa (Genoa, Italy) and the Italian Institute of Technology, where he was also a postdoctoral researcher until May 2019. He is interested in machine learning for robotics, particularly imitation learning and control. Webpage: `http://joaosilverio.eu`

Darwin G. Caldwell received his BSc. and PhD in Robotics from the University of Hull in 1986 and 1990 respectively. In 1996 he received an MSc in Management from the University of Salford. He is or has been an Honorary Professor at the University of Manchester, the University of Sheffield, the University of Bangor, the Kings College University of London, all in the U.K., and Tianjin University, and Shenzhen Academy of Aerospace Technology in China. The cCub, COMAN, WalkMan, HyQ, HyQ2Max, HyQ-Real and Centauro, Humanoid and quadrupedal robots were all developed in his Department. Prior to this he had worked on the development of the iCub. He is the author or co-author of over 500 academic papers, 20+ patents, and has received over 40 awards and nominations from leading journals and conferences. Prof. Caldwell has been a fellow of the Royal Academy of Engineering since 2015.