Generalization of Orientation Trajectories and Force-Torque Profiles for Robotic Assembly

Aljaž Kramberger[a,*], Andrej Gams[a], Bojan Nemec[a], Dimitrios Chrysostomou[b], Ole Madsen[b], Aleš Ude[a]

[a] *Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[b] *Robotics & Automation Group, Dept. of Mechanical and Manufacturing Engineering, Aalborg University, Fibigerstraede 14, 9220 Alborg, Denmark*

Abstract

A typical robot assembly operation involves contacts with the parts of the product to be assembled and consequently requires the knowledge of not only position and orientation trajectories but also the accompanying force-torque profiles for successful performance. To learn the execution of assembly operations even when the geometry of the product varies across task executions, the robot needs to be able to adapt its motion based on a parametric description of the current task condition, which is usually provided by geometrical properties of the parts involved in the assembly. In our previous work we showed how positional control policies can be generalized to different task conditions. In this paper we propose a complete methodology to generalize also the orientational trajectories and the accompanying force-torque profiles to compute the necessary control policy for a given condition of the assembly task. Our method is based on statistical generalization of successfully recorded executions at different task conditions, which are acquired by kinesthetic guiding. The parameters that describe the varying task conditions define queries into the recorded training data. To improve the execution of the skill after generalization, we combine the proposed approach with an adaptation method, thus enabling the refinement of the generalized assembly operation.

Keywords: Trajectory generalization, trajectory adaptation, programming by demonstration

1. Introduction

Robot assembly is one of the prime robotic tasks that can be found in industrial applications. It typically involves contacts between the robot and the objects involved in the task. The execution thus needs references for desired positions, orientations, and forces and torques (wrench). All of these can be provided for known, previously explored task conditions. However, that makes the execution possible only for a very

*Aljaž Kramberger
Email address: aljaz.kramberger@ijs.si (Aljaž Kramberger)

small set of explored situations. In this paper we propose a method of generating the required referential trajectories through statistical generalization. While generalization of positions has been previously discussed in the literature [1, 2], the novelty of this paper is: 1) generalization of orientation trajectories in unit quaternion space and 2) the generalization of force-torque profiles to provide referential forces and torques for assembly execution at previously unexplored task conditions.

As reported in the 2016 edition of A Roadmap for US Robotics [3], mass customization, which is an opposite to mass production, is ever more present in small and medium enterprises, but also in big manufacturers, such as the automotive industry. The impact of this trend is that the production systems have to adapt to handle more product variations, shorter life cycles, and smaller batch sizes. One of the main enablers of this transition is robotics and specifically robotic systems that are able to be reprogrammed quickly by non-robotics experts or reconfigure automatically when a new task in a factory arises [4]. Furthermore, such systems need to be able to take into account uncertainty in interactions with humans [5] and work in a collaborative manner [6]. Algorithms for autonomous adaptation of robots to the current state of the robotic workcell, i. e., the desired customized production, are a step further in the evolution of adaptive robotic cells.

In this paper we introduce methods that allow autonomous generation of trajectories from a database of related, but different situations. The approach is rooted in Programming by Demonstration (PbD) and statistical learning of robot trajectories. Recent work on learning by demonstration [7, 8], e. g., through kinesthetic guiding of robotic arms [9, 10, 11, 12, 13], has surpassed the traditional application of programming by demonstration, making it more suitable for industrial applications. Kinesthetic guiding has provided the necessary framework for knowledge transfer in industrial settings, using the new generation robotic arms such as the Kuka LWR series robot manipulators. The major advantage is that with kinesthetic guiding we can avoid the need for additional sensors to enable motion transfer and for transformation of human motion to robot motion.

A widely accepted approach in PbD is to encode the trajectories into a form that allows for easy application of various adaptations. Often, this is done with the use of dynamic movement primitives (DMP) [14]. This framework enables efficient modulation of trajectories while they are being executed, both spatially and temporally, because they are not explicitly time dependent. Schaal et al. [15] explain that direct time dependence is often inappropriate, since it does not allow to change the speed of execution. Furthermore, it does not allow stopping or restarting robotic movements in the event of unforeseen disturbances during execution of the desired task. However, the ability to adapt speed is important, because speed changes during execution of contact tasks are often required in order to raise the success ratio. Nevertheless, adaptation of a single trajectory is unlikely to provide an appropriate solution for more general situations, where positions, orientations, and force-torque profiles need to change significantly.

1.1. Generalization of position and orientation trajectories

Generalization from a database of previously explored situations has been applied to adapt to new situations. Locally weighted regression is one of several statistical gen-

eralization methods that were successfully applied in robotics within the DMP framework. Besides the locally weighted regression (LWR) [16], locally weighted projection regression (LWPR) [17], and Gaussian process regression (GPR) [18] have been applied most frequently. For example, Matsubara et al. [19] proposed an algorithm for the generation of new control policies from existing knowledge, thereby achieving an extended scalability of DMPs, while mixture of motor primitives was used for generation of table tennis swings in [20]. On the other hand, generalization of DMPs was combined with model predictive control by Krug and Dimitrov [21] or applied to DMP coupling terms [22], which were learned and later added to a demonstrated trajectory to generate new joint space trajectories. Furthermore, Stulp et al. [23] proposed learning a function approximator with one regression in the full space of phase and tasks parameters, bypassing the need for two consecutive regressions. Finally, generalization using GPR was applied over combined joint position trajectories and torque commands in the framework of compliant movement primitives [24], extending the DMP framework beyond the kinematic trajectory properties.

An important contribution of this paper is the generalization of orientation trajectories. Planning of trajectories in Cartesian space does not allow direct application of aforementioned generalization methods (per each degree of freedom), because there exists no minimal, singularity-free representation of orientation [25]. For example, the quaternion representation as a singularity-free but non-minimal representation of orientation with only 4 parameters (compared to 9 parameters of rotation matrices) must fulfill an additional constraint in 4-D space (unit norm) to describe a manifold of all orientations. Such constraints are not taken into account by general learning methods, and thus they are not preserved when learning from parameters that are constrained to a specific manifold. In this paper we provide a formal solution to applying the aforementioned LWR for generalization of orientation trajectories in such a way that that the generalized trajectory is guaranteed to lie on the orientation constraint manifold. To achieve this, we apply it to the differences between trajectories, which are expressed as 3-D vectors and are therefore not constrained to the unit manifold. We use Cartesian space DMPs [25] as the underlying representation for orientation trajectories. This approach integrates the equations of motion directly on the orientation manifold and is therefore guaranteed to generate quaternions with unit norm.

1.2. Generalization of contact tasks

For in-contact task, such as assembly, it has been shown that adaptation of the trajectories is extremely important [26] because even small positional deviations can result in high forces acting on the robot, consequently leading to a failed execution or even damaging the robot. Consequently, the robot control algorithm should consider the arising forces and torques [27]. Abu-Dakka et al. [28] proposed a method for assembly task execution from human demonstration with adaptation to reference force-torque profiles. Their method uses a single demonstration to provide the reference positions and orientations, which the robot had to change with respect to the desired force-torque profile in order to respond to noise and other small task changes.

In this work we propose a method to generalize position and orientation trajectories as well as the accompanying force-torque profiles based on a set of previously demonstrated assembly tasks at significantly different task conditions. Unlike [28], our newly

proposed method allows for adaptation to significant task changes. Approaches that adapt both positions/orientations and force-torque profiles have been proposed before. A method for real-time adaptation of learned trajectories depending on measured sensory data, where a force controller modified the accelerations of the DMP to comply with the previously measured force profile was proposed by Pastor et al. [26]. The approach was also expanded to provide a library of stereotypical movements associated with experienced sensory information [29]. On the other hand, we previously proposed methods where arbitrary desired force-torque profiles could be tracked using iterative learning control [30, 28]. Koropouli et al. [31] have progressed beyond mere adaptation and employed generalization approaches for motion-based force control policies. By learning both the policy and the policy difference data using locally weighted regression (LWR), they could estimate the policy at new inputs through superposition of the training data. A hybrid position-force control concept based on sequencing of movement primitives that contain kinematic and force modalities and are learned by kinesthetic guiding was proposed in [32]. In contrast to these approaches, Khansari et al. [33] did not use the demonstrated forces and torques directly but analyzed the data to identify higher level features of the human strategy and then incorporated these features into the controller.

Our proposed approach is applicable to generalization of contact-rich robotic skills, where we generate complete movement and force-torque profiles for new queries (inputs). We tested the proposed approach on a classical, contact-rich robotics assembly task, i. e. peg-in-hole (PiH). Model-based approaches have been extensively studied in the past [34, 35, 36]. PiH has also been used to demonstrate the abilities of learning neural network policies [37]. Similar to the latter, in this paper PiH is used to demonstrate the applicability of the proposed learning-based approach. The gist of our approach is in adapting to an external condition, which defines the task and can result in significant changes in the required robot motion, which cannot be handled by feedback control and fine adaptation only. In the PiH case, this external condition – used as a query into a library of demonstrated movements – can for example be the depth and diameter of the hole. The major novelty of the proposed method is that it allows the transfer of the learned skill that involves position and force control from multiple known conditions of the task (acquired by kinesthetic guiding), to new, previously unknown conditions, thereby providing the required trajectory and force-torque profile. Previously proposed adaptation methods can then be applied to refine the generalized motion [27, 38, 28]. The combination of generalization and subsequent adaptation provides us with a very versatile solution, which can be applied to different robotic tasks and platforms.

1.3. Overview

This paper is organized as follows. In Section 2 we explain what data is needed and how it is collected. In Section 3 we then provide a theoretical description of the proposed orientation trajectory and force-torque profile generalization method, followed by an explanation on the subsequent adaptation of generalized trajectories in Section 4. Experimental evaluation of the orientational motion generalization method based on the valve turning experiment and the force-torque profile generalization with adap-

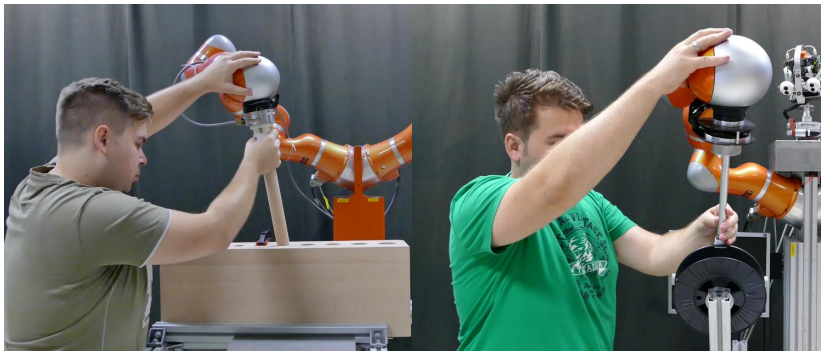


Figure 1: Kinesthetic guiding of KUKA LWR-4 robot arm, which was used to capture the training data for generalization of force-torque profiles (left) and orientational trajectories (right).

tation based on the peg-in-hole task follows in Section 5, with conclusions given in Section 6.

2. Recording the example movement library

Robot task execution can be captured in several ways. In this work we focus on kinesthetic guiding, shown in Fig. 1. 6-D Cartesian space movement trajectories are captured together with the forces and torques acting on the end-effector. These forces and torques are thus given in the robot tool frame. To record the training data, a human operator physically guides the robot along the desired trajectory and thus receives the same feedback from the environment as the robot. Consequently, the initial trajectories are from the human perspective optimal. If necessary they could be refined by means of reinforcement learning [38] or similar methods. Since such a refinement is not among the contributions of this paper, we omit further discussions of this topic.

To learn assembly skills by kinesthetic guiding, the following training data (trajectories) should be captured

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

where $\boldsymbol{p}_{i,j}$, $\boldsymbol{q}_{i,j}$ are the measured Cartesian space positions and orientations (represented as unit quaternions). $\dot{\boldsymbol{p}}_{i,j}$, $\boldsymbol{\omega}_{i,j}$, $\ddot{\boldsymbol{p}}_{i,j}$, and $\dot{\boldsymbol{\omega}}_{i,j}$ are the associated linear and angular velocities and accelerations, usually estimated from the recorded positions and orientations by numerical differentiation. See reference [39] and Appendix A for more details about quaternion representation of orientation. In addition, the resulting forces and torques arising at the robot's end-effector are needed

$$\mathscr{F}_d, = \{\boldsymbol{F}_{i,j}, \boldsymbol{M}_{i,j}\}_{j=1,\ i=1}^{T_i,\ NumEx}. \tag{2}$$

All these data are recorded at times $t_{i,j},\ j = 1,\ldots,T_i$, where T_i denotes the number of measurements in the i-th training set, $i = 1,\ldots,NumEx$, is the index of the training set, and $NumEx$ is the number of training sets (example task executions demonstrated by kinesthetic guiding).

Note that some robots use joint torques to estimate the end-effector forces and torques. In such cases the measured forces and torques are influenced by the human demonstrator. To obtain net forces and torques that are not corrupted by the influence of human demonstrator, we play back the recorded motion in exactly the same configuration of the workcell without human intervention. The forces and torques arising during such play back are measured and stored instead of the forces and torques arising during human demonstration. This procedure could be avoided if a force-torque sensor is used to capture the sensory feedback directly at the end-effector. However, even in such cases it is sometimes necessary to repeat the demonstrated movement because a better demonstration by kinesthetic guiding can often be generated when holding the manipulated object (Fig. 1) instead of holding the robot.

Training trajectories and force-torque profiles are recorded at different external conditions (queries) $\boldsymbol{s}$, which can be one or multi-dimensional. In PiH experiments described in Section 5.3, the queries $\boldsymbol{s}_i$ are defined by the depth of the hole h_i and the diameter of the hole d_i, $\boldsymbol{s}_i = [h_i, d_i]^T$. When generalizing the orientational trajectories as discussed in Section 5.1, the queries were one dimensional and given as the desired rotation angle of the valve, $s_i = \varphi_i$. These queries must be saved with the rest of the training data

$$\mathscr{S}_d = \{\boldsymbol{s}_i\}_{i=1}^{NumEx}. \tag{3}$$

Note that the distribution of example trajectories and the accompanying queries influences the success of generalization. Best results are usually obtained if the queries are uniformly distributed. Furthermore, as explained in [2], the recorded data (position and orientation trajectories and force-torque profiles) must transition smoothly between queries.

3. Generalization of Assembly Skills

3.1. Cartesian space DMPs – CDMPs

Since it is important for the understanding of the paper, we start our derivations by explaining the basics of Cartesian space dynamic movement primitives, abbreviated as CDMPs in this paper. CDMPs were originally proposed in [25]. In standard DMPs, trajectories are represented by nonlinear dynamic equations that can be flexibly adjusted to represent complex trajectories without the danger of instability of the equations [15]. In CDMPs, the positional part of the trajectory is treated as in standard DMPs, whereas the orientational part of the trajectory is represented by unit quaternions that require special treatment, both in the nonlinear dynamic equations and when integrating these equations.

A CDMP consists of the following free parameters: weights $\boldsymbol{w}_k^p,\ \boldsymbol{w}_k^o \in \mathbb{R}^3,\ k = 1,\ldots,N$, to respectively represent the positional and orientational part of the trajectory, trajectory duration τ and the final desired position $\boldsymbol{g}^p$ and orientation $\boldsymbol{g}^o$ at which the robot comes to rest. Parameter N defines the number of basis functions that are needed to approximate the trajectory. In the CDMP the orientation is represented by a unit quaternion $\boldsymbol{q} = v + \boldsymbol{u} \in \mathrm{S}^3$, where S^3 is a unit sphere in $\mathbb{R}^4$. The following formulation

has been proposed in [25] to encode position ($\boldsymbol{p}$) and orientation ($\boldsymbol{q}$) trajectories

$$\begin{aligned}
\tau\dot{\mathbf{z}} &= \alpha_z(\beta_z(\mathbf{g}^p - \mathbf{p}) - \mathbf{z}) + \mathbf{f}_p(x), && (4)\\
\tau\dot{\mathbf{p}} &= \mathbf{z}, && (5)\\
\tau\dot{\boldsymbol{\eta}} &= \alpha_z\left(\beta_z 2\log\left(\boldsymbol{g}^o * \overline{\boldsymbol{q}}\right) - \boldsymbol{\eta}\right) + \mathbf{f}_o(x), && (6)\\
\tau\dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{\eta} * \boldsymbol{q}, && (7)\\
\tau\dot{x} &= -\alpha x. && (8)
\end{aligned}$$

Here $\boldsymbol{z}$ and $\boldsymbol{\eta}$ denote the scaled linear and angular velocity ($\boldsymbol{z} = \tau\dot{\boldsymbol{p}}$, $\boldsymbol{\eta} = \tau\boldsymbol{\omega}$). The quaternion product $*$, conjugation $\overline{\boldsymbol{q}}$, and the quaternion logarithm $\log(\boldsymbol{q})$ are explained in the appendix. The forcing terms $\mathbf{f}_p$ and $\mathbf{f}_o$ in the CDMP are nonlinear and defined as

$$\begin{aligned}
\mathbf{f}_p(x) &= \boldsymbol{D}_p \frac{\sum_{k=1}^N \mathbf{w}_k^p \Psi_k(x)}{\sum_{k=1}^N \Psi_k(x)} x, && (9)\\
\mathbf{f}_o(x) &= \boldsymbol{D}_o \frac{\sum_{k=1}^N \boldsymbol{w}_k^o \Psi_k(x)}{\sum_{k=1}^N \Psi_k(x)} x. && (10)
\end{aligned}$$

Note that the aforementioned free parameters $\boldsymbol{w}_k^p$, $\boldsymbol{w}_k^o \in \mathbb{R}^3$ are contained in the forcing terms. They should be adjusted to reproduce any given smooth Cartesian trajectory $\{\boldsymbol{p}_j, \boldsymbol{q}_j, \dot{\boldsymbol{p}}_j, \boldsymbol{\omega}_j, \ddot{\boldsymbol{p}}_j, \dot{\boldsymbol{\omega}}_j, t_j\}_{j=1}^T$. The scaling matrices $\boldsymbol{D}_p$, $\boldsymbol{D}_o \in \mathbb{R}^{3\times 3}$ can be set to $\boldsymbol{D}_p = \boldsymbol{D}_o = \boldsymbol{I}$. Other possibilities are described in [25]. The nonlinear forcing terms are defined as a linear combination of radial basis functions Ψ_k

$$\Psi_k(x) = \exp\left(-h_k\left(x - c_k\right)^2\right). \quad (11)$$

In the above equation c_k are the centers and h_k the widths of RBFs. The distribution of weights can be defined in various ways, but here we follow the proposal from [1] where $c_k = \exp\left(-\alpha_x \frac{k-1}{N-1}\right)$, $h_k = \dfrac{1}{(c_{k+1} - c_k)^2}$, $h_N = h_{N-1}$, $k = 1, \ldots, N$. The time constant τ is set to the desired duration of the trajectory, i. e. $\tau = t_T - t_1$. The goal position and orientation are usually set to the final position and orientation on the desired trajectory, i. e. $\boldsymbol{g}^p = \boldsymbol{p}_{t_T}$ and $\boldsymbol{g}^o = \boldsymbol{q}_{t_T}$. Auxiliary math for the calculation and integration of Cartesian space DMPs is given in Appendix A. More details about CDMPs can be found in [25].

In the following we explain how to exploit this representation of Cartesian trajectories for the generalization of position and orientation trajectories.

3.2. Generalization functions

Assembly skills for new, previously unexplored task parameters (queries) $\boldsymbol{s}$ can be computed using the available training data (1) – (3). We apply locally weighted regression (LWR) [16] for statistical generalization of these data. LWR is a non-parametric method for statistical approximation, which uses raw data stored in memory, to determine new movements. In this paper we differentiate between three utilizations for generalization: for position, for orientation, and for force-torque profiles.

LWR was used for the generalization of position trajectories in [1], where it was applied to generalize throwing, reaching and drumming movements. The second utilization and the first novelty of this paper is the generalization of orientation trajectories in unit quaternion space. In this paper we show how we can apply LWR for generalization of orientation motion so that the generalized trajectory is guaranteed to lie on the orientation constraint manifold, i. e. so that the outcome of the generalization is a unit quaternion trajectory. The third utilization and the second major novelty of this paper is that we show how to apply LWR for forces and torques, thus generating a desired profile for an action at a previously unexplored query.

As explained above, the Cartesian position and orientation trajectories are encoded as CDMPs (see Section 3.1) and thus contain the following free parameters: weights $\{\boldsymbol{w}_k^p,\ \boldsymbol{w}_k^o\}_{k=1}^N$, trajectory duration τ, and the final desired position $\boldsymbol{g}^p$ and orientation $\boldsymbol{g}^o$. On the other hand, force-torque profiles are encoded as linear combinations of radial basis functions (RBFs) defined in Eq. (11)

$$\boldsymbol{F}(x) = \frac{\sum_{k=1}^N \boldsymbol{v}_k^F \Psi_k(x)}{\sum_{k=1}^N \Psi_k(x)}, \tag{12}$$

$$\boldsymbol{M}(x) = \frac{\sum_{k=1}^N \boldsymbol{v}_k^M \Psi_k(x)}{\sum_{k=1}^N \Psi_k(x)}. \tag{13}$$

Here $\{\boldsymbol{v}_k^F,\ \boldsymbol{v}_k^M\}_{k=1}^N$ are the free parameters that can be adjusted to approximate the reference force-torque profiles. Both position and orientation trajectories, encoded as CDMPs, and forces and torques, encoded with RBFs, share the same phase variable x. Note that unlike in CDMPs, where radial basis functions Ψ_k are multiplied with phase x in Eq. (10) to ensure faster convergence to the desired goal after the end of the training interval, it is not necessary to add this multiplication to Eq. (12) – (13) because the reference forces and torques have no meaning beyond the training interval. The weights $\boldsymbol{v}_k^F$, $\boldsymbol{v}_k^M$ need to be computed from the training data to generate the generalized force-torque profiles.

As the name locally weighted regression suggests, LWR computes local models by putting more emphasis on the data with training queries $\boldsymbol{s}_i$ close to the given query point $\boldsymbol{s}$. Formally, we compute the following mappings

$$\mathbf{G}_p(\mathscr{A}_d, \mathscr{S}_d) : \boldsymbol{s} \rightarrow \left[\boldsymbol{w}_1^{p\mathrm{T}}, \ldots, \boldsymbol{w}_N^{p\mathrm{T}},\ \tau,\ \boldsymbol{g}^{p\mathrm{T}}\right]^{\mathrm{T}}, \tag{14}$$

$$\mathbf{G}_o(\mathscr{A}_d, \mathscr{S}_d) : \boldsymbol{s} \rightarrow \left[\boldsymbol{w}_1^{o\mathrm{T}}, \ldots, \boldsymbol{w}_N^{o\mathrm{T}},\ \tau,\ \boldsymbol{g}^{o\mathrm{T}}\right]^{\mathrm{T}}, \tag{15}$$

to synthesize a Cartesian space DMP and

$$\mathbf{G}_{fm}(\mathscr{F}_d, \mathscr{S}_d) : \boldsymbol{s} \rightarrow \left[\boldsymbol{v}_1^{F\mathrm{T}}, \ldots, \boldsymbol{v}_N^{F\mathrm{T}},\ \boldsymbol{v}_1^{M\mathrm{T}}, \ldots, \boldsymbol{v}_N^{M}\right]^{\mathrm{T}} \tag{16}$$

to synthesize reference force-torque profiles.

The generation of generalization function $\mathbf{G}_p$ is the same as in [1] and we refer the reader to this paper for implementation details. In the following we explain the generation of the mapping $\mathbf{G}_o$. The generation of function $\mathbf{G}_{fm}$ for the generalization of force-torque profiles is explained in Section 3.4.

3.3. Generalization of orientation trajectories

Generalization of position and force-torque profiles, as explained in the previous section, cannot be analogously applied to the generalization of unit quaternion trajectories because the resulting generalized orientation would not preserve the unit norm, thus requiring an additional normalization step. Here we propose an approach that allows us to apply locally weighted regression without normalization. The proposed procedure generalizes among the differences between orientation trajectories instead of quaternion trajectories directly.

Our approach starts by searching for a training query point $\boldsymbol{s}_k$ closest to the new given query $\mathbf{s}$, at which the new orientation trajectory has to be synthesized

$$k = \arg\min_i \{\|\boldsymbol{s} - \boldsymbol{s}_i\|\}. \tag{17}$$

In our experiments we used the Euclidean norm, but other metrics could be used if required by the selection of query points. We continue by computing unit quaternion DMP $\mathbf{q}_k^{\mathrm{DMP}}$ for orientation trajectory closest to the given query point $\boldsymbol{s}$. The following training data are used for this purpose

$$\{\boldsymbol{q}_{k,j}, \boldsymbol{\omega}_{k,j}, \dot{\boldsymbol{\omega}}_{k,j}, t_{k,j}\}_{j=1}^{T_k}. \tag{18}$$

Next, differences between the training trajectories (1) and the estimated $\mathbf{q}_k^{\mathrm{DMP}}$ are calculated along the phase of the training data. These differences define a new training data set

$$\mathscr{A}_d' = \{\boldsymbol{r}_{i,j}, t_{i,j}\}_{i=1,\ j=1}^{\mathrm{NumEx},\ T_i}, \tag{19}$$

where

$$\boldsymbol{r}_{i,j} = \log\left(\boldsymbol{q}_{i,j} * \overline{\mathbf{q}_k^{\mathrm{DMP}}(x_{i,j})}\right), \tag{20}$$

$$x_{i,j} = \exp\left(-\frac{\alpha_x}{\tau_i} t_{i,j}\right). \tag{21}$$

Note that $\tau_i = t_{i,T_i} - t_{i,1}$. Unlike the unit quaternions $\boldsymbol{q}_{i,j}$, the difference vectors $\boldsymbol{r}_{i,j} \in \mathbb{R}^3$ are unconstrained. Hence locally weighted regression can be applied to these data without constraints. Another important point is that data in (20) correspond to a rotation vector representation of orientation, often called exponential coordinates. This representation is minimal and thus as every minimal representation of orientation contains singularities. However, due to the properties of exponential and logarithmic maps defined in the appendix, these singularities only arise at large rotation angles. Below we explain why our generalization approach is guaranteed to avoid such rotation angles.

Given training data (19), we can compute the generalized difference trajectory $\mathbf{r}$ associated with the given query point $\boldsymbol{s}$. Just like for forces and torques, we write this difference trajectory as a linear combination of radial basis functions

$$\mathbf{r}(x) = \sum_{i=1}^{N} \frac{\boldsymbol{v}_i^r \Psi_i(x)}{\sum_{j=1}^{N} \Psi_j(x)}. \tag{22}$$

For each dimension of $\boldsymbol{v}_i^r = \left[v_{i,1}^r, v_{i,2}^r, v_{i,3}^r\right]^{\mathrm{T}}$, the application of locally weighted regression results in the following least-squares optimization problem

$$\min_{\boldsymbol{v}_l^r} \sum_{i=1}^{NumEx} \|\boldsymbol{X}_i \boldsymbol{v}_l^r - \boldsymbol{r}_{i,l}\|^2 \mathrm{K}(\boldsymbol{s},\boldsymbol{s}_i),\ l = 1,2,3, \tag{23}$$

where K is the weighting kernel (see below) and

$$\boldsymbol{r}_{i,l} = \left[\ r_{i,1,l}, \quad \ldots, \quad r_{i,T_i,l}\ \right]^{\mathrm{T}}, \tag{24}$$

$$\boldsymbol{v}_l^r = \left[\ v_{1,l}^r, \quad \ldots, \quad v_{N,l}^r\ \right]^{\mathrm{T}}, \tag{25}$$

$$\boldsymbol{X}_i = \begin{bmatrix} \frac{\Psi_1(x_{i,1})}{\sum_{j=1}^N \Psi_j(x_{i,1})} & \cdots & \frac{\Psi_N(x_{i,1})}{\sum_{j=1}^N \Psi_k(x_{i,1})} \\ \vdots & \ddots & \vdots \\ \frac{\Psi_1(x_{i,T_i})}{\sum_{j=1}^N \Psi_j(x_{i,T_i})} & \cdots & \frac{\Psi_N(x_{i,T_i})}{\sum_{j=1}^N \Psi_j(x_{i,T_i})} \end{bmatrix}. \tag{26}$$

To put more emphasis on the data associated with queries closer to the current query, we chose the tricube weighting kernel K [16] for locally weighted regression

$$\mathrm{K}(d) = \begin{cases} (1-|l|^3)^3 & \text{if } |l| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

The tricube kernel has finite support and continuous first and second derivatives. Thus, the first two derivatives of the generalization function $\mathbf{G}_{fm}$ are also continuous. Furthermore, the computational complexity of the optimization problem (23) is reduced through this choice of K because the force-torque profiles for which K vanishes do not influence matrices $\boldsymbol{X}_i$. This makes the system matrix associated with the objective function (23) banded.

Many possible choices for the weighting kernel K exist, see [16] for other possibilities. As discussed in this paper [16], the choice of the weighting kernel is usually not critical for generalization accuracy. In our experiments we obtained good performance with the selected kernel.

Note that in the above LWR formulation only those i for which $\mathrm{K}(\boldsymbol{s},\boldsymbol{s}_i) > 0$ affect the result. Since $\mathrm{K}(\boldsymbol{s},\boldsymbol{s}_i) > 0$ is true only in the local neighborhood of $\boldsymbol{s}$, the relevant difference vectors $\boldsymbol{r}_{i,j}$ remain small, assuming that the orientation trajectories smoothly transition between each other. Since the rotation vector representation contains no singularities in the neighborhood of $\boldsymbol{r} = 0$, the optimization problem (23) avoids any critical areas where the rotation vector representation becomes discontinuous. Thus the optimization problem (23) remains well defined.

To control the robot, we need to transform the generalized difference trajectory back to the orientational part of a CDMP. This transformation can be calculated by multiplying the generalized rotation difference trajectory (22) with the quaternion DMP closest to the query point, i. e. $\mathbf{q}_k^{\mathrm{DMP}}$

$$\mathbf{q}^{\mathrm{DMP}}(x) = \exp(\mathbf{r}(x)) * \mathbf{q}_k^{\mathrm{DMP}}(x). \tag{28}$$

While we could apply Eq. (28) directly to control a robot, it is usually advantageous to first sample the resulting orientation trajectory, typically at robot servo rate, and compute new weights $\boldsymbol{w}^o$ from the sampled data using Cartesian DMP estimation techniques developed in [25]. This way we can exploit all advantages of DMPs, which is not possible with representation (28).

The example goal orientations as well as the duration of movements are directly available in the data. They are given as

$$\boldsymbol{g}_i^o = \boldsymbol{q}_{i,T_i},\ \tau_i = t_{i,T_i} - t_{i,1},\ i = 1,\ldots,NumEx. \tag{29}$$

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

$$\tau = \sum_{i=1}^{NumEx} \frac{\mathrm{K}(\boldsymbol{s},\boldsymbol{s}_i)\tau_i}{\sum_{j=1}^{NumEx}\mathrm{K}(\boldsymbol{s},\boldsymbol{s}_j)}. \tag{30}$$

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

$$\min_{\boldsymbol{g}^o\in\mathrm{S}^3} \sum_{i=1}^{NumEx} \mathrm{d}(\boldsymbol{g}^o,\boldsymbol{g}_i^o)\mathrm{K}(\boldsymbol{s},\boldsymbol{s}_i), \tag{31}$$

where d is a metric on a 4-D unit sphere defined in (48). Optimization problem (31) is nonlinear and can be solved using iterative methods such as Newton's method. The iteration process can be initialized with the following approximation

$$\boldsymbol{q}_0 = \frac{\sum_{i=1}^{NumEx}\mathrm{K}(\boldsymbol{s},\boldsymbol{s}_i)\boldsymbol{g}_i^o}{\|\sum_{i=1}^{NumEx}\mathrm{K}(\boldsymbol{s},\boldsymbol{s}_i)\boldsymbol{g}_i^o\|}. \tag{32}$$

3.4. Generalization of force-torque profiles

The computation of generalized weights $\boldsymbol{v}_k^F$, $\boldsymbol{v}_k^M$ is based on the training data set (2) – (3). It is equivalent for all six force-torque dimensions, therefore we here focus on one dimension of force profile (12), with the corresponding weights denoted as $\boldsymbol{v} = [v_1,\ldots,v_N]^T$. Using (2) and (12), we obtain the following expression for the data stemming from the i-th demonstration:

$$F(x_{i,j}) = \frac{\sum_{k=1}^{N} v_k\Psi_k(x_{i,j})}{\sum_{k=1}^{N}\Psi_k(x_{i,j})}, j = 0,\ldots,T_i, \tag{33}$$

where the phases $x_{i,j}$ are defined by the phase equation (8). Thus they are given by $x_{i,j} = \exp(-\alpha_x t_{i,j}/\tau_i)$. For each training set i, Eq. (33) is a system of linear equations in v_k. Therefore it can be written in a matrix form

$$\boldsymbol{f}_i = \boldsymbol{X}_i\boldsymbol{v}, \tag{34}$$

with the system matrix $\boldsymbol{X}_i$ defined as

$$\boldsymbol{X}_i = \begin{bmatrix} \frac{\Psi_1(x_{i,1})}{\sum_{k=1}^{N}\Psi_k(x_{i,1})} & \cdots & \frac{\Psi_N(x_{i,1})}{\sum_{k=1}^{N}\Psi_k(x_{i,1})} \\ \vdots & \ddots & \vdots \\ \frac{\Psi_1(x_{i,T_i})}{\sum_{k=1}^{N}\Psi_l(x_{i,T_i})} & \cdots & \frac{\Psi_N(x_{i,T_i})}{\sum_{k=1}^{N}\Psi_k(x_{i,T_i})} \end{bmatrix}. \tag{35}$$

Note that these matrices are dependent only on the distribution of phase, not on the recorded force and torques. The left and right hand side vectors in Eq. (34) are defined as

$$\boldsymbol{f}_i = \begin{bmatrix} F(x_{i,1}) \\ \vdots \\ F(x_{i,T_i}) \end{bmatrix}, \boldsymbol{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix}. \tag{36}$$

By applying locally weighted regression [16] we can generalize the training data (2) and (3) to a new query point $\boldsymbol{s}$ by solving the following least squares optimization problem for each dimension of the force-torque profile

$$\min_{\boldsymbol{v}} \sum_{i=1}^{NumEx} \|\boldsymbol{X}_i\boldsymbol{v} - \boldsymbol{f}_i\|^2 \mathrm{K}(\|\boldsymbol{s} - \boldsymbol{s}_i\|), \tag{37}$$

where K is the weighting kernel that defines the influence of each training set. Just like in case of orientations, we chose a tricube weighting kernel to put more emphasis on the nearby data points.

4. Adaptation of Generalized Contact Trajectories

While executing the generalized contact movements represented by CDMPs and the accompanying force-torque profiles, the resulting forces and torques can differ from the ones computed by the generalization method. If these discrepancies are significant, they could cause the task execution, in our experiments PiH, to fail or even damage the workpieces or the robot. Therefore, the generalized trajectories have to be adapted during the execution. As proposed in the work of Abu-Dakka et al. [28], an error feedback calculated from the actual and demonstrated forces and torques can be used to modify the robot movement, thus reducing the discrepancies between the desired and actual forces and torques. In our work, we apply this method with the modification of how error feedback is calculated. Instead of using the demonstrated forces and torques as references for adaptation, the error feedback is calculated using the discrepancies between the *generalized* and actual forces and torques that arise during the execution. In other words, the generalized forces and torques are used as references for adaptation of the generalized position and orientation trajectories. The error feedback $\boldsymbol{e}_p(x) \in \mathbb{R}^3$ for positions and $\boldsymbol{e}_q(x) \in \mathbb{R}^3$ for orientations is thus calculated as follows

$$\boldsymbol{e}_p(x) = \boldsymbol{q}(x) * (\boldsymbol{F}^{\mathrm{gen}}(x) - \boldsymbol{F}^{\mathrm{mes}}) * \overline{\boldsymbol{q}(x)} \tag{38}$$

$$\boldsymbol{e}_q(x) = \boldsymbol{q}(x) * (\boldsymbol{M}^{\mathrm{gen}}(x) - \boldsymbol{M}^{\mathrm{mes}}) * \overline{\boldsymbol{q}(x)} \tag{39}$$

where $\boldsymbol{F}^{\mathrm{gen}}$ and $\boldsymbol{M}^{\mathrm{gen}}$ are the generalized force and torque at phase x, respectively, $\boldsymbol{F}^{\mathrm{mes}}$ and $\boldsymbol{M}^{\mathrm{mes}}$ the current measured force and torque, respectively, while $\boldsymbol{q}(x)$ is the unit quaternion specifying the current orientation of the robot's tool. Note that $*$ denotes a quaternion product and that 3-D vectors are interpreted as quaternions with zero scalar component in (38) and (39). Using this error feedback, the underlying CDMP specifying the desired position and orientation can be modified and optimized using iterative learning control [28], which ensures successful and optimal execution of the contact skill.

5. Experimental Evaluation

We evaluated the developed methods in several experiments with simulated and real data. The real experiments were conducted with KUKA LWR-4 robot arm. We used the KUKA provided Fast Research Interface [40] to realize real-time robot control.

5.1. Generalization of CDMPs in simulation

We first describe the results of orientation DMP generalization using simulated data. We synthesized an example set of 21 minimum jerk SLERP trajectories [41], with 10 of them shown in red in Fig. 2. All orientation trajectories started at the same initial orientation and finished at different, but evenly distributed end orientations. The rotation angles of the final orientations with respect to the initial orientation were used as query points for generalization. In our simulation test, the 11 orientation trajectories shown in blue were used as training data for generalization at intermediate queries (rotation angles) using the generalization approach described in Section 3.3. The intermediate queries were the same rotation angles at which the other 10 orientation trajectories (not used to define a training data set) were calculated. The results are shown in red in Fig. 2.

For testing the accuracy of the generalization method, we calculated the difference between the LWR generalized trajectories (red in Fig. 2) and the simulated minimum

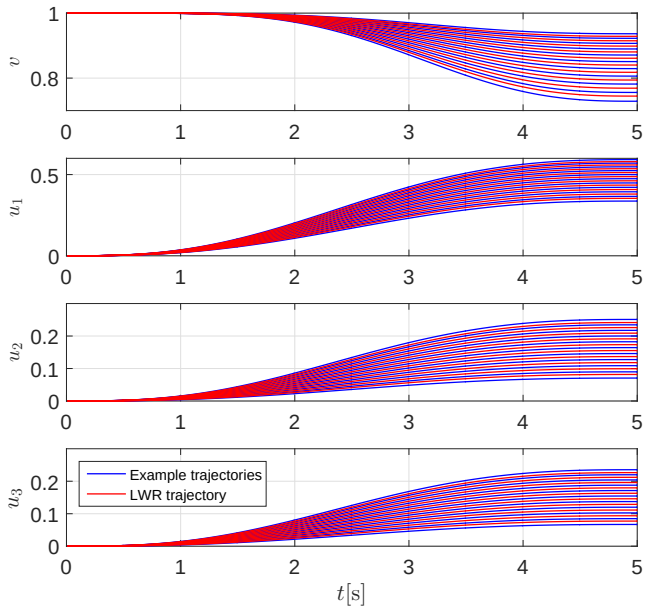


Figure 2: Synthesized database for generalization between minimum jerk SLERP trajectories, represented as quaternions $\boldsymbol{q} = v + \boldsymbol{u}$. The simulated trajectories are shown in blue, while the resulting generalized trajectories are shown in red.

jerk SLERP trajectories synthesized at the same intermediate query points as the generalized orientation trajectories. The results can be seen in Fig. 3. The calculated difference using formula (48) is very small for most of the trajectories (represented in blue), although the generalization accuracy closer to the edge of the database is lower (represented in red and green). This outcome is consistent with the results presented in the paper of Ude et al. [1].

5.2. Generalization of CDMPs with a real robot

After the initial results from simulation had turned out promising, we carried out an experiment on a real robot. The challenge was to turn a valve, shown in Fig. 4, to the desired final configuration from any starting angle. The recorded trajectories compared to the simulated ones were not as smooth, which made the task of generalization more difficult.

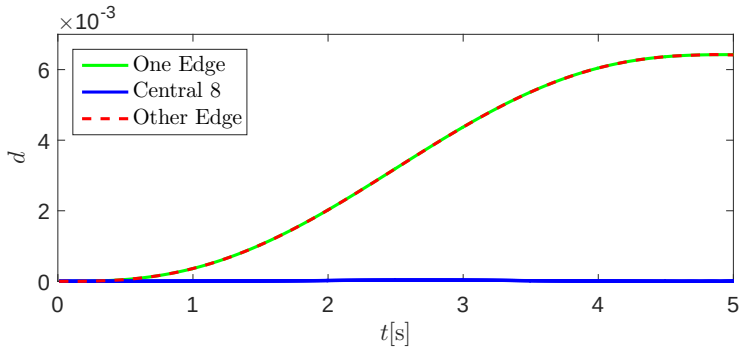


Figure 3: The difference between the LWR generalized trajectories and the corresponding minimum jerk SLERP trajectories at the same queries, calculated using formula (48). The errors of the two trajectories generalized close to the edge of the database are depicted in green and red, while all other 8 are depicted in blue.

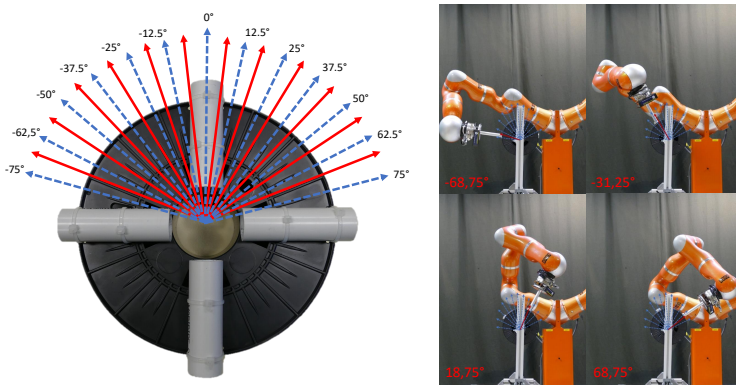


Figure 4: Valve (left) for testing the proposed orientation generalization method on a real robot. Blue color indicates the demonstrated example queries, while the red arrows show the example queries for generalization. The snapshots on the right show the robot arm at angles $\varphi_i = [-68.75°, -31.25°, 18.75°, 68.75°]$

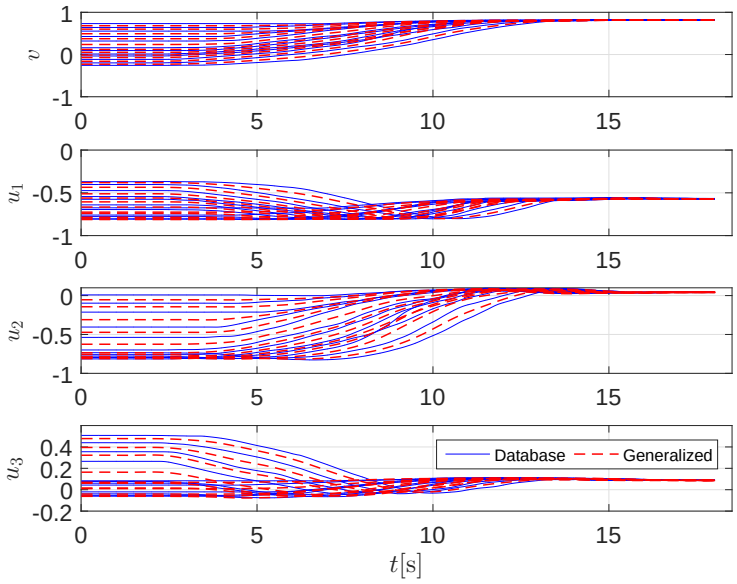


Figure 5: The database of real orientation trajectories (blue) and the generalized trajectories (red).

To build the training set for generalization, a human operator demonstrated 13 movements, all starting at different initial angles and ending in the same configuration. The starting angles $[-75°, -62.5°, -50°, -37.5°, -25°, -12.5°, 0°, 12.5°, 25°, 37.5°, 50°, 62.5°, 75°]$ are shown with blue colored arrows in Fig. 4. They were saved as queries $s_i = \varphi_i$.

The recorded database of valve turning movements, shown in blue in Fig. 5, was used to synthesize new movements with the desired initial angle of the valve used as query for generalization. The generalization results for all the intermediate queries $[-68.75°, -56.25°, -43.75°, -31.25°, -18.75°, 0°, 18.75°, 31.25°, 43.75°, 56.25°, 68.75°]$ are shown in red in Fig. 5.

The real-world generalization results are comparable to simulation results.

Just as in simulation, we confirmed the accuracy of the generalization process on the real robot. The results are presented in Fig. 6. The differences were calculated using leave-one-out cross-validation method, where each trajectory saved in the database was left out and generalized to that specific query point. With this method we calculated the difference between the generalized trajectory and the demonstrated trajectory stored in the training set using Eq. (48). The human demonstrated trajectories are not as uniformly distributed and smooth as the ones generated in simulation, therefore the discrepancies are larger compared to the results from Fig. 3. Note, however, that the differences remain small and most of the time below 0.01 rad $\approx 0.57°$.

Another issue that affects the accuracy of generalization is the density of the training data. The results of [1] show that the accuracy of generalization in general increases with the density of the training data. However, since the transformations $\mathbf{G}_p$ and $\mathbf{G}_o$, which are respectively provided in Eqs. (14) and (15), between the query points and

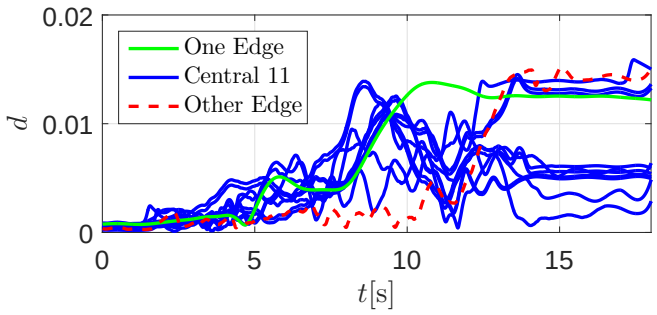


Figure 6: Difference between the LWR generalized trajectories and the demonstrated database trajectories at the same query points, calculated using leave-one-out cross-validation and distance metrics (48). Difference of the edge trajectories are depicted in green and red, all other trajectories are shown in blue.

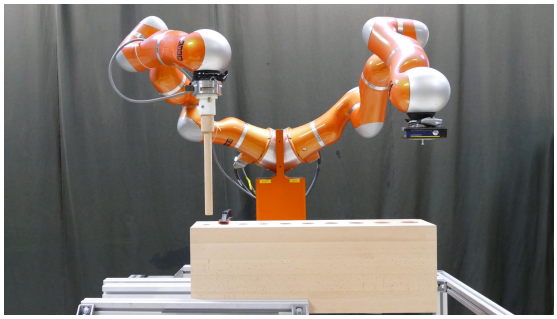


Figure 7: Experimental setup for testing the generalized contact skills.

the generalized trajectories are highly nonlinear, it is not possible to provide general guidelines for the required density of training data because it is highly task-specific.

5.3. Generalization of force-torque profiles

For implementation and testing of the force-torque generalization approach on a real robot, we equipped KUKA LWR-4 robot arm with a specialized gripper for grasping of round pegs. Although KUKA LWR-4 has a torque sensor at every joint, we mounted ATI Gamma SI-130-10 force-torque sensor on the wrist of the robot to obtain more accurate measurements of net forces and torques acting on the robot's end-effector during task demonstrations (see Fig. 7).

As a prototypical assembly task, Peg-in-hole (PiH) assembly was selected for experimental evaluation. We tested the PiH assembly with round pegs and holes, where the diameter of the peg d_i and the depth of the hole h_i varied. Thus a two-dimensional query space was formed, $\boldsymbol{s}_i = [d_i, h_i]^{\mathsf{T}}$, $i = 1, \ldots, NumEx$. A wooden base object comprising 10 holes of different diameters and a constant depth of 250 mm was constructed to carry out experiments. The diameter of the holes varied from 24 mm to 51 mm in 3 mm increments with a tolerance between the hole and the peg of less than 1 mm. In order to obtain holes with different depths, each hole with small pegs was filled with small pegs of height 20 mm.

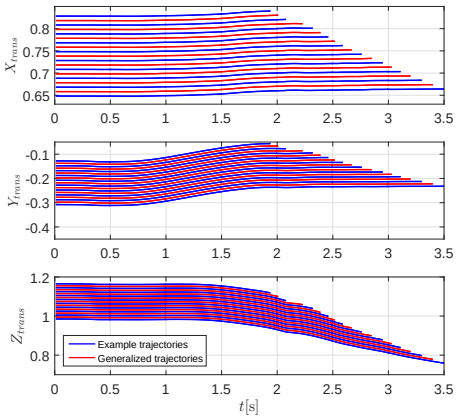


Figure 8: Position trajectory database for a single diameter of the hole at 10 depths shown in blue and the generalized trajectories shown in red. The trajectories are offset for a clear comparison.

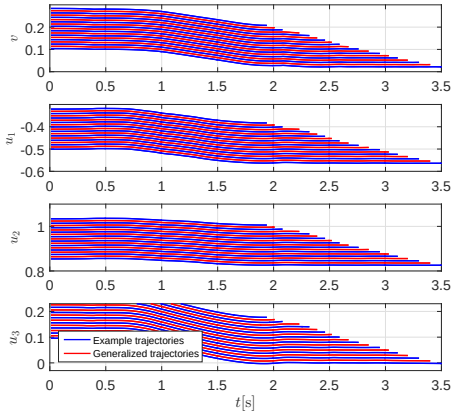


Figure 9: Graph shows the quaternion $\boldsymbol{q} = v + \boldsymbol{u}$ orientation trajectory database for a single diameter of the hole at 10 depths shown in blue and the generalized trajectories shown in red. The trajectories are offset for a clear comparison.

Using kinesthetic guiding we obtained trajectories at 10 different hole diameters $d_i \in [24, 27, 30, 33, 36, 39, 42, 45, 48, 51]$ mm and for each diameter at 10 different depths of the hole $h_i \in [0, -20, -40, -60, -80, -100, -120, -140, -160, -180] + 250$ mm. An example database for a single hole at 10 depths is presented in Fig. 8 for position and Fig. 9 for orientation trajectories. Thus altogether we acquired 100 training trajectories. The measured forces and torques were in general increasing according to the diameter of the hole (see Fig. 10 and 11). It is evident from these figures that the measured force-torque profiles continuously transition between each other, which is important for generalization.

The generated training data were used to generalize the PiH assembly to new position and orientation trajectories (see Section 3.3) and new force-torque profiles (see Section 3.4) at the desired intermediate queries. The resulting movements were executed using the adaptation algorithm described in Section 4 and reference [28], with generalized force-torque profiles used as a reference for adaptation. Without adaptation the PiH operation often cannot be performed successfully. To evaluate the effectiveness of the procedure for the generalization of force-torque profiles, we also executed the adaptation algorithm with the nearest neighbor force-torque profile as reference instead of the generalized force torque-profile. Here criterion (17) was used to determine nearest neighbor force-torque profiles. Note that it is not possible to use the nearest neighbor position and orientation trajectories as in this case the PiH assembly cannot be successfully accomplished. On the other hand, it is possible to use the nearest neighbor force-torque profiles if they are synchronized to the generalized position and orientation trajectories through the common phase. Thus for evaluation purposes, each PiH assembly was executed with the proposed adaptation algorithm twice; once with the generalized and once with the nearest neighbor force-torque profile as reference.

The following values were estimated for analysis:

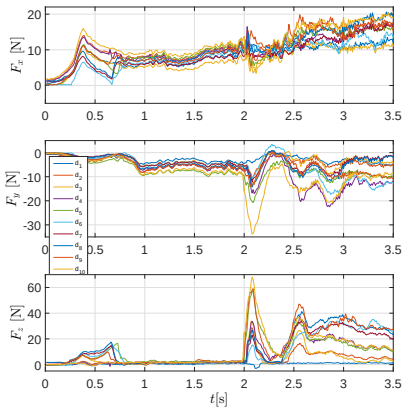


Figure 10: Force profile database recorded at the maximal depth and 10 diameters of hole represented in color.

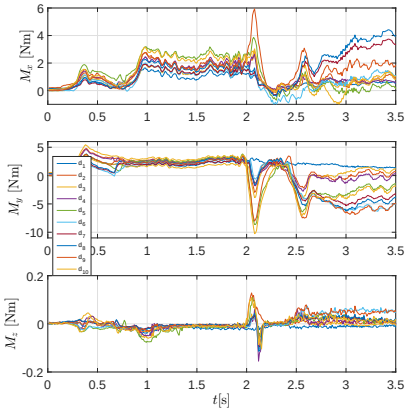


Figure 11: Torque profile database recorded at the maximal depth and 10 diameters of hole represented in color.

Table 1: Comparison of median magnitude of force, torque, position, and orientation differences between the values measured during the adaptation and reference values (see Eqs. (40) – (43)). The last row shows the average difference between the generalized execution time (30) and the execution time of the PiH operation with adaptation, where the generalized and nearest neighbor force-torque profiles where respectively used as reference for adaptation.

PiH with	Generalized force-torque reference profiles	Nearest neighbor force-torque reference profiles
Median magnitude of force differences [N]	3.714	4.294
Median magnitude of torque differences [Nm]	0.5892	0.6977
Median magnitude of position difference [m]	0.0027	0.0092
Median magnitude of orientation difference [rad]	0.00083	0.00084
Average time difference [s]	0.3792	0.5242

- The difference between the generalized execution time (30) and the execution time of PiH assembly with adaptation (Fig. 12).

- The magnitude of differences between the generalized / nearest neighbor force-torque profiles and the measured forces and torques (Fig. 13 and 14) during adaptation.
- The magnitude of differences between the generalized and adapted position and orientation trajectories.

In total 90 experiments at intermediate queries (depicted with star in Fig. 12, 13, and 14) were conducted. The results are presented in Table 1 and show the medians of average magnitude of differences calculated from all executions with adaptation at intermediate queries. For each query, the average magnitudes were calculated as follows

$$e^f = \frac{1}{T}\sum_{i=1}^{T} \|\boldsymbol{F}_i^{\text{ref}} - \boldsymbol{F}_i^{\text{mes}}\|, \tag{40}$$

$$e^m = \frac{1}{T}\sum_{i=1}^{T} \|\boldsymbol{M}_i^{\text{ref}} - \boldsymbol{M}_i^{\text{mes}}\|, \tag{41}$$

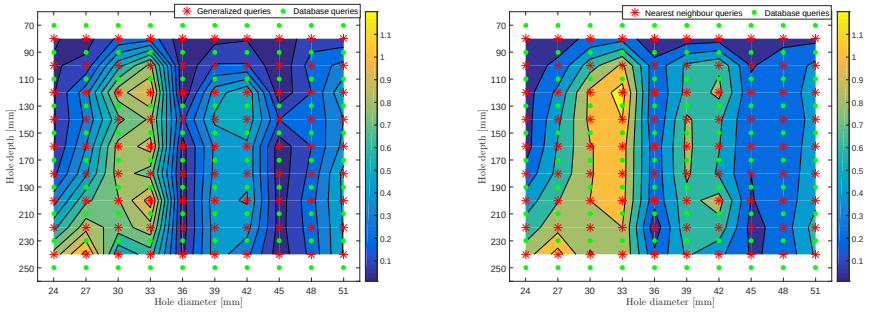
$$e^p = \frac{1}{T}\sum_{i=1}^{T} \|\boldsymbol{p}_i^{\text{ref}} - \boldsymbol{p}_i^{\text{adapt}}\|, \tag{42}$$

$$e^q = \frac{1}{T}\sum_{i=1}^{T} \|\mathrm{d}(\boldsymbol{q}_i^{\text{ref}}, \boldsymbol{q}_i^{\text{adapt}})\|. \tag{43}$$

where $\boldsymbol{F}_i^{\text{ref}}$ and $\boldsymbol{M}_i^{\text{ref}}$ represent the reference force-torque profiles (i. e. generalized and nearest neighbor) and $\boldsymbol{F}_i^{\text{mes}}$ and $\boldsymbol{M}_i^{\text{mes}}$ the measured forces and torques during adaptation. $\boldsymbol{p}_i^{\text{ref}}$ and $\boldsymbol{q}_i^{\text{ref}}$ respectively represent the generalized positions and orientations and $\boldsymbol{p}_i^{\text{adapt}}$ and $\boldsymbol{q}_i^{\text{adapt}}$ the adapted positions and orientations. T is the number of sampling times. Distance metrics d is defined in Eq. (48).

All differences between the execution times of the PiH assembly with adaptation and the generalized execution time according to Eq. (30) at intermediate queries are shown in Fig. 12. Note that if the PiH assembly is executed with generalized force-torque profiles as reference for adaptation, it is most cases executed faster than when nearest neighbor force-torque profiles are used. The reference trajectory duration in these experiments was around 2 seconds, thus the differences in execution time are quite considerable. Table 1 provides more evidence that PiH execution with adaptation using generalized force-torque profiles as reference is faster than the PiH execution with adaptation using the nearest neighbor force-torque profiles as reference.

Fig. 13a shows all average magnitudes of force differences (40) acquired at intermediate queries by DMP adaptation with the generalized force-torque profiles as reference for adaptation. When compared to Fig. 13b, where the same values acquired by adaptation with the nearest neighbor force-torque profiles are shown, it can be seen that the average magnitudes of force differences are smaller in the former case. The same can be observed with torques depicted in Fig. 14a and 14b, where Eq. (41) was used to calculate the average differences. Average magnitude of position and orientation adaptation is shown in Table 1, showing that the average amount of adaptation was smaller when executing trajectories with generalized forces and torques used as



(a) Differences between the generalized execution time according to Eq. (30) and the actual execution time at intermediate queries, where the generalized force-torque profiles were used as reference for adaptation.

(b) Differences between the generalized execution time according to Eq. (30) and the actual execution time at intermediate queries, where the nearest neighbor force-torque profiles were used as reference for adaptation.

Figure 12: Comparison of execution times differences. The training queries are shown in green, the intermediate queries in red.

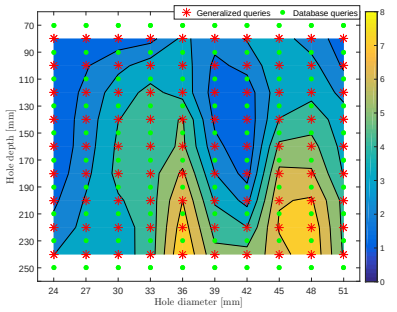
reference. Hence the generalization procedure provides force-torque profiles that require less adaptation. The adaptation amounted to less than 1 millimeter in most cases, which was within the tolerances of the objects used in the experiments. The imperfections in the wooden experimental objects caused the two maximums in Figs. (13a and 13b), but the generalization and adaptation procedure could cope with this issue.

Fig. 15 illustrates the differences between when performing the PiH assembly with adaptation at one example query. As in the previous results, the task was performed twice, once with the nearest neighbor forces and torques and once with the generalized forces and torques used as reference for adaptation. The first three graphs show that the differences between the reference forces and the forces measured during adaptation are far greater when the nearest neighbor forces were used as reference. A significant difference can also be noted in the phase evolution, shown in lower right plot in Fig. 15. The DMP phase stopping mechanism, which causes the movement to slow down when required for successful adaptation, was much more active in the execution with the nearest neighbor force-torque profile compared to when the generalized force-torque profile was used as reference.

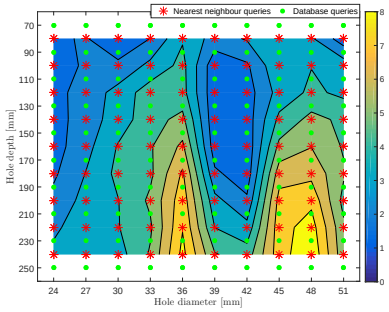
These results show that in most cases the generalization of force-torque profiles with locally weighted regression leads to a faster execution of the task with less adaptation than the simpler nearest neighbor approach.

6. Conclusion

In this paper we extended our previous work on task-specific generalization of position and joint space DMPs [1]. The first contribution of the paper is a new approach for task-specific generalization of orientation trajectories represented by unit quaternions, which was not considered in [1]. The proposed approach also extends our pre-

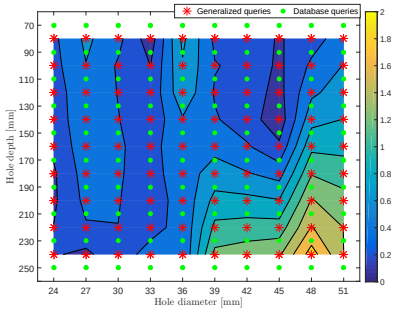


(a) Average magnitude of differences between the reference and measured forces during adaptation at intermediate queries, where generalized force-toque profiles were taken as references for adaptation.

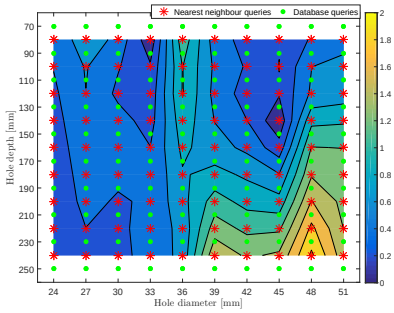


(b) Average magnitude of differences between the reference and measured forces during adaptation at intermediate queries, where nearest neighbor force-toque profiles were taken as references for adaptation.

Figure 13: Comparison of magnitudes of force differences. The training queries are shown in green, the intermediate queries in red.



(a) Average magnitude of differences between the reference and measured torques during adaptation at intermediate queries, where generalized force-toque profiles were taken as references for adaptation.



(b) Average magnitude of differences between the reference and measured torques during adaptation at intermediate queries, where nearest neighbor force-toque profiles were taken as references for adaptation.

Figure 14: Comparison of magnitudes of torque differences. The training queries are shown in green, the intermediate queries in red.

vious work on programming by demonstration of assembly tasks [28] and provides an entire solution for the generalization of assembly tasks involving contact with the environment. In the proposed approach, besides position and orientation trajectories also the demonstrated forces and torques are generalized to enable effective execution and adaptation of contact skills in different external conditions.

Locally weighted regression (LWR) was utilized as an underlying method for syn-

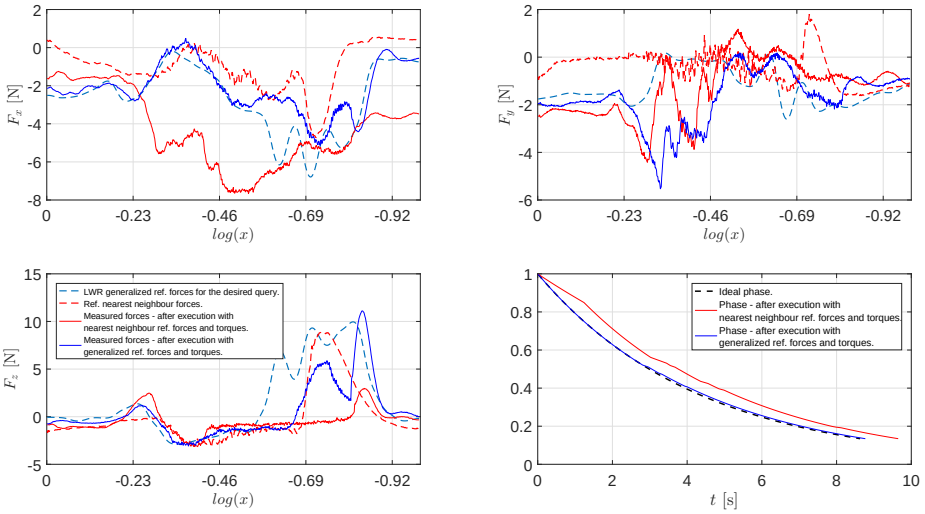


Figure 15: Comparison of peg-in-hole executions with adaptation, where nearest neighbor and generalized forces were respectively used as reference for adaptation. The upper two and the lower left graph show forces measured during the PiH execution with generalized forces and torques used as reference for adaptation (solid blue line) and with nearest neighbor forces and torques used as reference for adaptation (solid red line). The corresponding reference forces are depicted with red and blue dashed lines. The lower right graph depicts the phase evolution during the execution of the generalized trajectory, first using the generalized force-torque profile (blue solid line) and second the nearest neighbor force-torque profile (red solid line) as reference for adaptation. The dashed black line corresponds to an ideal DMP phase without adaptation, which causes the movement to slow down.

thesizing new Cartesian space DMPs and force-torque profiles. In LWR nearby data points are given higher emphasis than the distant ones, which results in local models. This is often advantageous because global models are in general more difficult to compute and can lead to complex optimization problems. The proposed generalization approach for orientation trajectories resolves the problem of maintaining the norm of unit quaternions in all computational steps.

In stiff environments generalized trajectories might still require additional adaptations in order to provide a better match between the generalized force-torque profiles and the actually measured forces and torques. If these discrepancies are too large, the task execution can fail. By applying a suitable trajectory adaptation method, the generalized trajectory can be modified so that the generalized forces and torques match the ones arising during the task execution better. We confirmed in our experiments that the execution time of the generalized contact skill can be improved by on-line adaptation of the generalized position and orientation trajectories.

The proposed approach was tested with the KUKA LWR-4 arm, which is impedance controlled. This was important to support our experiments. In simulation the orientation training data was perfectly smooth and the generalization method produced good results. When testing the approach with the real training trajectories obtained by kinesthetic guiding, the results were comparable. Thus the proposed approach can cope with human-generated training data. The experimental results of PiH assembly show that a

generalized force-torque profile provides an effective reference profile for adaptation at intermediate query points within the training space.

In our future work we will investigate the possibility of enhancing the initial training set with automatically acquired trajectories and force-torque profiles using approaches like iterative learning control and reinforcement learning. This way we expect to automate the construction of the training data and the robot will become able to autonomously improve its performance when executing assembly tasks.

Appendices

A. Auxiliary Math for Cartesian Space DMPs

A quaternion $\boldsymbol{q} = v + \boldsymbol{u}$ consists of a scalar part $v \in \mathbb{R}$ and vector part $\boldsymbol{u} \in \mathbb{R}^3$. A quaternion multiplication (denoted by $*$) is defined by

$$\boldsymbol{q}_1 * \boldsymbol{q}_2 \quad = \quad (v_1 + \boldsymbol{u}_1) * (v_2 + \boldsymbol{u}_2) = (v_1 v_2 - \boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{u}_2) + (v_1 \boldsymbol{u}_2 + v_2 \boldsymbol{u}_1 + \boldsymbol{u}_1 \times \boldsymbol{u}_2). \quad (44)$$

The set of all quaternions with the above multiplication forms a non-commutative division algebra. Conjugation of quaternions is denoted by a bar and defined as $\overline{\boldsymbol{q}} = \overline{v + \boldsymbol{u}} = v - \boldsymbol{u}$. The norm of a quaternion is defined as

$$\|\boldsymbol{q}\| = \sqrt{\boldsymbol{q} * \overline{\boldsymbol{q}}} = \sqrt{v^2 + \|\boldsymbol{u}\| 2} \quad (45)$$

The set of quaternions with unit norm forms a sphere S^3 in $\mathbb{R}^4$. It can be shown that the product of two unit quaternions is a unit quaternion, thus unit quaternions form a multiplicative group. They can be used to represent the orientation in Cartesian space. This is a 2-to-1 representation as unit quaternions $\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same orientation.

To derive CDMP equations in Section 3.1, we need to connect quaternion derivative $\dot{\boldsymbol{q}}(t)$ and angular velocity $\boldsymbol{\omega}(t)$. This relation is given by

$$\dot{\boldsymbol{q}} = \frac{1}{2} \boldsymbol{\omega} * \boldsymbol{q}. \quad (46)$$

In the above equation angular velocity $\boldsymbol{\omega}$ is treated like a quaternion with a zero scalar component. By comparing (46) and (7) we obtain $\boldsymbol{\eta} = \tau \boldsymbol{\omega}$, thus $\boldsymbol{\eta}$ is a scaled angular velocity.

The quaternion logarithm $\log : \mathrm{S}^3 \mapsto \mathbb{R}^3$, which is one of the operations in (6), is defined as

$$\log(\boldsymbol{q}) = \log(v + \boldsymbol{u}) = \begin{cases} \arccos(v) \dfrac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, \ \boldsymbol{u} \neq 0 \\ [0,0,0]^{\mathrm{T}}, \ \text{otherwise} \end{cases}. \quad (47)$$

The quaternion logarithm $\log(\boldsymbol{q}_2 * \overline{\boldsymbol{q}}_1)$ can be interpreted as a difference vector between two unit quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$. It can be used to define a distance metrics on S^3 [42]

$$\mathrm{d}(\boldsymbol{q}_1, \boldsymbol{q}_2) = \begin{cases} 2\pi, & \boldsymbol{q}_2 * \overline{\boldsymbol{q}}_1 = -1 + [0,0,0]^{\mathrm{T}} \\ 2\|\log(\boldsymbol{q}_2 * \overline{\boldsymbol{q}}_1)\|, & \text{otherwise} \end{cases} \quad (48)$$

The logarithmic map (47) is injective if we limit its domain to $S^3/\{-1+[0,0,0]^T\}$. Its inverse, the exponential map $\exp : \mathbb{R}^3 \mapsto S^3$, is defined as

$$\exp(\boldsymbol{r}) \quad = \quad \begin{cases} \cos(\|\boldsymbol{r}\|) + \sin(\|\boldsymbol{r}\|) \dfrac{\boldsymbol{r}}{\|\boldsymbol{r}\|}, \boldsymbol{r} \neq 0 \\ \\ 0, \text{ otherwise} \end{cases} \tag{49}$$

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

The exponential map is used to integrate Eq. (7). Given the orientation $\boldsymbol{q}$ and angular velocity $\boldsymbol{\omega}(t)$ at time t and assuming that angular velocity is constant on time interval $[t, t+\Delta t]$, we can calculate the orientation at the next integration time $t+\Delta t$ as follows

$$\boldsymbol{q}(t+\Delta t) = \exp\left(\frac{\Delta t}{2}\boldsymbol{\omega}\right) * \boldsymbol{q}(t) = \exp\left(\frac{\Delta t}{2\tau}\boldsymbol{\eta}\right) * \boldsymbol{q}(t). \tag{50}$$

Other equations defining a CDMP, i. e. Eq. (4) – (6) and (8), should be integrated using standard Euler integration [25]. A more in-depth theoretical description of quaternions and orientation representation by unit quaternions is provided in the book of Morais et al. [39].

Acknowledgements

Funding: This work was supported by EU Horizon 2020 Programme [grant number 680431] ReconCell; and Slovenian Research Agency [grant number J2-7360].

References

[1] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-specific generalization of discrete and periodic dynamic movement primitives, IEEE Transactions on Robotics 26 (5) (2010) 800–815.

[2] D. Forte, A. Gams, J. Morimoto, A. Ude, On-line motion synthesis and adaptation using a trajectory database, Robotics and Autonomous Systems 60 (10) (2012) 1327–1339.

[3] H. I. Christensen (Ed.), A Roadmap for US Robotics: From Internet to Robotics, 2016 Edition, University of California, San Diego, 2016.

[4] G. Michalos, S. Makris, N. Papakostas, D. Mourtzis, G. Chryssolouris, Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach, CIRP Journal of Manufacturing Science and Technology 2 (2) (2010) 81–91.

[5] P. Tsarouchi, S. Makris, G. Chryssolouris, Human–robot interaction review and challenges on task planning and programming, International Journal of Computer Integrated Manufacturing 29 (8) (2016) 916–931.

[6] N. Papakostas, G. Michalos, S. Makris, D. Zouzias, G. Chryssolouris, Industrial applications with cooperating robots for the flexible assembly, International Journal of Computer Integrated Manufacturing 24 (7) (2011) 650–660.

[7] B. D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration, Robotics and Autonomous Systems 57 (5) (2009) 469–483.

[8] R. Dillmann, Teaching and learning of robot tasks via observation of human performance, Robotics and Autonomous Systems 47 (2) (2004) 109–116.

[9] M. Hersch, F. Guenter, S. Calinon, A. Billard, Dynamical system modulation for robot learning via kinesthetic demonstrations, IEEE Transactions on Robotics 24 (6) (2008) 1463–1467.

[10] D. Lee, C. Ott, Incremental kinesthetic teaching of motion primitives using the motion refinement tube, Autonomous Robots 31 (2) (2011) 115–131.

[11] P. Kormushev, S. Calinon, D. G. Caldwell, Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input, Advanced Robotics 25 (5) (2011) 581–603.

[12] J. J. Steil, C. Emmerich, A. Swadzba, R. Grünberg, A. Nordmann, S. Wrede, Kinesthetic teaching using assisted gravity compensation for model-free trajectory generation in confined spaces, in: Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe:, Springer, 2014, pp. 107–127.

[13] S. Manschitz, J. Kober, M. Gienger, J. Peters, Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations, Robotics and Autonomous Systems 74 (2015) 97–107.

[14] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, S. Schaal, Dynamical movement primitives: learning attractor models for motor behaviors, Neural computation 25 (2) (2013) 328–373.

[15] S. Schaal, P. Mohajerian, A. Ijspeert, Dynamics systems vs. optimal control - a unifying view, Progress in brain research 165 (2007) 425–445.

[16] C. G. Atkeson, A. W. Moore, S. Schaal, Locally Weighted Learning, Artificial Intelligence Review 11 (1997) 11–73.

[17] S. Vijayakumar, A. D'Souza, T. Shibata, J. Conradt, S. Schaal, Statistical Learning for Humanoid Robots, Autonomous Robots 12 (1) (2002) 55–69.

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

[19] T. Matsubara, S.-H. Hyon, J. Morimoto, Learning parametric dynamic movement primitives from multiple demonstrations, Neural Networks 24 (5) (2011) 493–500.

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

[21] R. Krug, D. Dimitrov, Model predictive motion control based on generalized dynamical movement primitives, Journal of Intelligent & Robotic Systems 77 (1) (2015) 17–35.

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

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

[24] M. Deniša, A. Gams, A. Ude, T. Petrič, Learning compliant movement primitives through demonstration and statistical generalization, IEEE/ASME Transactions on Mechatronics, 2016 21 (5) (2016) 2581–2594.

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

[26] P. Pastor, L. Righetti, M. Kalakrishnan, S. Schaal, Online movement adaptation based on previous sensor experiences, in: International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, 2011, pp. 365–371.

[27] L. Rozo, P. Jiménez, C. Torras, A robot learning from demonstration framework to perform force-based manipulation tasks, Intelligent Service Robotics 6 (1) (2013) 33–51.

[28] F. J. Abu-Dakka, B. Nemec, J. A. Jørgensen, T. R. Savarimuthu, N. Krüger, A. Ude, Adaptation of manipulation skills in physical contact with the environment to reference force profiles, Autonomous Robots 39 (2) (2015) 199–217.

[29] P. Pastor, M. Kalakrishnan, L. Righetti, S. Schaal, Towards associative skill memories, in: IEEE-RAS International Conference on Humanoid Robots (Humanoids), Osaka, Japan, 2012, pp. 309–315.

[30] A. Gams, B. Nemec, A. Ijspeert, A. Ude, Coupling movement primitives: Interaction with the environment and bimanual tasks, IEEE Transactions on Robotics 30 (4) (2014) 816–830.

[31] V. Koropouli, S. Hirche, D. Lee, Generalization of force control policies from demonstrations for constrained robotic motion tasks, Journal of Intelligent & Robotic Systems 80 (1) (2015) 133–148.

[32] J. Kober, M. Gienger, J. J. Steil, Learning movement primitives for force interaction tasks, in: IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 3192–3199.

[33] M. Khansari, E. Klingbeil, O. Khatib, Adaptive human-inspired compliant contact primitives to perform surface–surface contact under uncertainty, The International Journal of Robotics Research (2016) 1651–1675.

[34] H. Bruyninckx, S. Dutre, J. De Schutter, Peg-on-hole: a model based solution to peg and hole alignment, in: IEEE International Conference on Robotics and Automation (ICRA), Nagoya, Japan, 1995, pp. 1919–1924.

[35] Y. Fei, X. Zhao, An assembly process modeling and analysis for robotic multiple peg-in-hole, Journal of Intelligent & Robotic Systems 36 (2) (2003) 175–189.

[36] A. Stemmer, A. Albu-Schaffer, G. Hirzinger, An analytical method for the planning of robust assembly tasks of complex shaped planar parts, in: IEEE International Conference on Robotics and Automation (ICRA), Rome, Italy, 2007, pp. 317–323.

[37] S. Levine, P. Abbeel, Learning neural network policies with guided policy search under unknown dynamics, in: Advances in Neural Information Processing Systems 27, 2014, pp. 1071–1079.

[38] M. Kalakrishnan, L. Righetti, P. Pastor, S. Schaal, Learning force control policies for compliant manipulation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, 2011, pp. 4639–4644.

[39] J. P. Morais, S. Georgiev, W. Sprößig, Real quaternionic calculus handbook, Birkhäuser, Basel, 2014.

[40] G. Schreiber, A. Stemmer, R. Bischoff, The Fast Research Interface for the KUKA lightweight robot, in: ICRA Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications – How to Modify and Enhance Commercial Controllers, Anchorage, Alaska, 2010.

[41] K. Shoemake, Animating rotation with quaternion curves, ACM Transactions on Graphics 19 (3) (1985) 245–254.

[42] A. Ude, Filtering in a unit quaternion space for model-based object tracking, Robotics and Autonomous Systems 28 (2) (1999) 163–172.