Merging Position and Orientation Motion Primitives

Matteo Saveriano[1], Felix Franzel[2], and Dongheui Lee[1,2]

***Abstract*— In this paper, we focus on generating complex robotic trajectories by merging sequential motion primitives. A robotic trajectory is a time series of positions and orientations ending at a desired target. Hence, we first discuss the generation of converging pose trajectories via dynamical systems, providing a rigorous stability analysis. Then, we present approaches to merge motion primitives which represent both the position and the orientation part of the motion. Developed approaches preserve the shape of each learned movement and allow for continuous transitions among succeeding motion primitives. Presented methodologies are theoretically described and experimentally evaluated, showing that it is possible to generate a smooth pose trajectory out of multiple motion primitives.**

I. Introduction

Robots operating in everyday environments will execute a multitude of tasks ranging from simple motions to complex activities consisting of several actions performed on different objects. Hand programming of all these tasks is not feasible. Hence, researchers have investigated how to acquire novel tasks in an intuitive manner [1], [2]. A possible solution is to demonstrate the task to execute, for example by physically guiding the robot towards the task completion [3], [4]. Collected data are then used for motion planning.

Motion planning with dynamical systems has gained attention in the robot learning community and researchers have developed several approaches to represent demonstrations as dynamical systems [5]–[16]. Dynamical systems are used to plan in joint or Cartesian space, and, in Cartesian space, to encode both position and orientation [13]–[16]. Moreover, robots driven by stable systems are able to reproduce complex paths [7]–[10], to incrementally update a predefined skill [11], [12], and to avoid possible collisions [17]–[21].

Complex robotic tasks, consisting of several actions, can be obtained by sequencing multiple motion primitives [22]–[26]. As in [24]–[26], this work represents the motion primitives as Dynamic Movement Primitives (DMP) [5], but other choices are possible [22], [23]. Given a set of DMPs, the problem arises of how the DMPs can be merged to generate a unique and smooth trajectory without stopping at the end of each motion primitive. Pastor et al. [27] address this problem by activating the succeeding motion primitive when the velocity of the current primitive is smaller than a threshold. The succeeding primitive is initialized with the state reached by the previous one at the switching point. This avoids jumps in the velocity but it may cause jumps

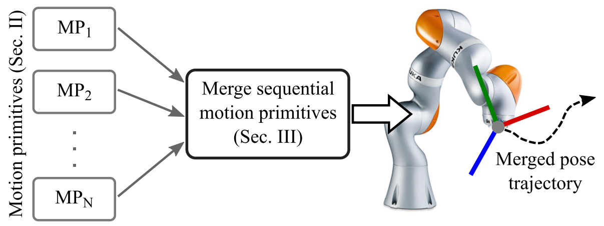

Motion primitives (Sec. II)
MP$_1$
MP$_2$
MP$_N$
Merge sequential motion primitives (Sec. III)
Merged pose trajectory

Fig. 1. Motion primitives are merged to generate a smooth robot trajectory.

in the acceleration. To avoid jumps in acceleration, [28] augments the DMP with a low-pass filter. Nevertheless, when positions and velocities of the consecutive primitives at the switching point are significantly different, the trajectory has to be filtered a lot introducing a delay with consequent large deviations from the demonstration. The approach in [29] has been proposed to learn hitting motions in table tennis, but it can be used to merge motion primitives. In [29], the DMP is augmented with a moving target and final velocity. Hence, each DMP reaches a certain position with a given velocity (different from zero) which are used to initialize the succeeding DMP. Instead of switching the DMPs, the approach in [30] creates a unique DMP by overlapping sequential movements. The unique DMP preserves the shape of each overlapped motion.

Aforementioned approaches are effective when the dynamical system is used to represent Cartesian positions or joint angles. However, they do not consider the orientation part of the motion. DMP formulations capable of encoding Cartesian orientation have been proposed in [14], [15], but without considering the problem of merging multiple movements. In this work, we first describe how DMP and unit quaternions are used to encode orientation trajectories and present a rigorous stability analysis that is missing in the related literature [14], [15]. We then extend the approaches in [27], [29], and [30] to merge sequential DMP representing both position and orientation (see Fig. 1). We use unit quaternions to represent the orientation and rely on quaternion algebra to define all the mathematical operations needed to merge the learned motions. Finally, we compare the presented approaches on simulated and real data in order to underline advantages and drawbacks of each approach.

II. Cartesian Pose Motion Primitives

In this section, we describe how Cartesian poses are represented in the dynamic movement primitives (DMP) framework [5] and provide a stability analysis.

A. Position DMP

Following the representation introduced by Park et al. [31], Cartesian positions are generated via the second-order dy-

[1]Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Weßling, Germany matteo.saveriano@dlr.de.

[2]Human-Centered Assistive Robotics, Technical University of Munich, Munich, Germany felix.franzel@tum.de, dhlee@tum.de.

This work has been supported by Helmholtz Association.

namical system (time dependency is omitted for simplicity)

$$\tau \dot{\boldsymbol{p}} = \boldsymbol{v}, \tag{1a}$$
$$\tau \dot{\boldsymbol{v}} = \boldsymbol{K}^p \left[(\boldsymbol{p}_d - \boldsymbol{p}) - \boldsymbol{d}_0^p(h) + \boldsymbol{f}^p(h) \right] - \boldsymbol{D}^p \boldsymbol{v}, \tag{1b}$$

where $\boldsymbol{p} \in \mathbb{R}^3$ is the position, $\dot{\boldsymbol{p}} = \boldsymbol{v} \in \mathbb{R}^3$ is the linear velocity, and $\dot{\boldsymbol{v}} \in \mathbb{R}^3$ is the linear acceleration. The time scaling factor τ can be adapted to change the duration of the movement without changing the path. The positive definite matrices $\boldsymbol{K}^p$, $\boldsymbol{D}^p \in \mathbb{R}^{3\times3}$ are linear stiffness and damping gains respectively. The scalar h is an exponentially decaying clock signal, obtained by integrating the so-called canonical system $\tau \dot{h} = -\gamma h$, with $\gamma > 0$. The clock signal is $h = 1$ at the beginning of the motion and it exponentially converges to zero. The term $\boldsymbol{d}_0^p(h) = (\boldsymbol{p}_d - \boldsymbol{p}_0)h$ in (1b) prevents a jump at the beginning of the motion and it vanishes for $h \to 0$. The forcing term $\boldsymbol{f}^p(h)$ in (1b) is defined as

$$\boldsymbol{f}^p(h) = \frac{\sum_{i=1}^N \boldsymbol{w}_i \psi_i(h)}{\sum_{i=1}^N \psi_i(h)} h, \quad \psi_i(h) = e^{-a(h-c_i)^2}. \tag{2}$$

Given the amplitude a and the centers c_i, the parameters $\boldsymbol{w}_i$ are learned from demonstration using weighted least square [5]. From (2), it is clear that $\boldsymbol{f}^p(h)$ vanishes for $h \to 0$.

B. Orientation DMP

Dynamic movement primitives, commonly used to represent Cartesian or joint position, have been extended to represent Cartesian orientation [14], [15]. The approaches in [14] and [15] use a different definition of the orientation error, as detailed later in this section. In this work, orientation is represented by a unit quaternion $\boldsymbol{q} = [\eta, \boldsymbol{\epsilon}^\top]^\top \in \mathcal{S}^3$, where $\mathcal{S}^3$ is the unit sphere in the 3D space. Unit quaternions have less parameters compared to rotation matrices (4 instead of 9). Compared to other representations, like Euler angles, they are uniquely defined and have no singularities if rotations are restricted to one hemisphere of $\mathcal{S}^3$ [32]. A DMP for the orientation is defined as

$$\tau \dot{\boldsymbol{q}} = \frac{1}{2} \tilde{\boldsymbol{\omega}} * \boldsymbol{q}, \tag{3a}$$
$$\tau \dot{\boldsymbol{\omega}} = \boldsymbol{K}^q \left[\boldsymbol{e}_o(\boldsymbol{q}_d, \boldsymbol{q}) - \boldsymbol{d}_0^q(h) + \boldsymbol{f}^q(h) \right] - \boldsymbol{D}^q \boldsymbol{\omega}, \tag{3b}$$

where[1] $\boldsymbol{q} \in \mathcal{S}^3$ is the unit quaternion, $\boldsymbol{\omega} \in \mathbb{R}^3$ and $\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ are the angular velocity and acceleration respectively, and τ is a temporal scaling factor. The symbol $*$ indicates the product of two quaternions defined in (13), $\boldsymbol{e}_o(\cdot,\cdot) \in \mathbb{R}^3$ is the error between two quaternions, and the quantity $\tilde{\boldsymbol{\omega}}$ is the angular velocity quaternion, i.e. $\tilde{\boldsymbol{\omega}} = [0, \boldsymbol{\omega}^\top]^\top$. In other words, $\tilde{\boldsymbol{\omega}}$ is a quaternion with zero as scalar part and the angular velocity as vector part. The positive definite matrices $\boldsymbol{K}^q$, $\boldsymbol{D}^q \in \mathbb{R}^{3\times3}$ are angular stiffness and damping gains respectively. The clock signal is the same used for the position ($\tau \dot{h} = -\gamma h$). The term $\boldsymbol{d}_0^q(h) = \boldsymbol{e}_o(\boldsymbol{q}_d * \boldsymbol{q}_0)h$ in (3b) prevents a jump at the beginning of the motion and it vanishes for $h \to 0$. The nonlinear forcing term $\boldsymbol{f}^q(h)$ in (3b) is defined as in (2), it is learned from demonstration, and it vanishes for $h \to 0$. The quaternion rate (3a) is integrated by means of (17).

There are two key differences between position DMP in (1a)–(1b) and orientation DMP in (3a)–(3b). First, the relationship between the time derivative of the quaternion $\dot{\boldsymbol{q}}$ and the angular velocity in (3a) is nonlinear, while the derivative of the position equals the linear velocity in (1a). Second, the error between two quaternions $\boldsymbol{e}_o(\cdot,\cdot)$ in (3b) is a nonlinear function and it has multiple definitions, while the error between two positions in (1b) is simply their difference. In robotics and control, the orientation error between quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ is typically defined as $\boldsymbol{e}_o = \text{vec}(\boldsymbol{q}_1 * \overline{\boldsymbol{q}}_2)$ [32], [33], where the function $\text{vec}(\boldsymbol{q})$ returns the vector part of $\boldsymbol{q}$. This definition of the orientation error is used in [14] for orientation DMP, while Ude et al. [15] propose to use the quantity $2\log(\boldsymbol{q}_1 * \overline{\boldsymbol{q}}_2)$ as orientation error, where the *logarithmic map* $\log(\cdot)$ is defined as in (15).

C. Stability analysis

The stability of the position DMP in (1a)–(1b) is trivially proved. Indeed, $\boldsymbol{d}_0^p(h)$ and $\boldsymbol{f}^p(h)$ vanish for the time $t \to +\infty$ and (1a)–(1b) become a linear system. Hence, the positive definiteness of $\boldsymbol{K}^p$ and $\boldsymbol{D}^p$ is sufficient to conclude that the dynamical system (1a)–(1b) asymptotically converges to $\boldsymbol{p}_d$ with zero velocity. For orientation DMPs, instead, it is interesting to prove the following stability theorem:

Theorem 1. *The orientation DMP in* (3a)–(3b)*, with clock signal h defined such that $h \to 0$ for $t \to +\infty$ and orientation error defined as* $\boldsymbol{e}_o(\boldsymbol{q}_1, \boldsymbol{q}_2) = \text{vec}(\boldsymbol{q}_1 * \overline{\boldsymbol{q}}_2)$*, globally asymptotically converges to $\hat{\boldsymbol{q}} = \boldsymbol{q}_d$ with $\hat{\boldsymbol{\omega}} = \boldsymbol{0}$.*

Proof. Recall that the non-linearities in (3a)–(3b) are smooth functions and that the time dependancy introduced by h vanishes for $t \to +\infty$. Hence, (3a)–(3b) are an *asymptotically autonomous differential system* and the stability can be proved analyzing its asymptotic behavior [34]. In other words, we have to prove the stability of

$$\dot{\boldsymbol{q}} = \frac{1}{2} \tilde{\boldsymbol{\omega}} * \boldsymbol{q}, \quad \dot{\boldsymbol{\omega}} = \boldsymbol{K}^q \text{vec}(\boldsymbol{q}_d * \overline{\boldsymbol{q}}) - \boldsymbol{D}^q \boldsymbol{\omega}, \tag{4}$$

where we set $\tau = 1$ without loss of generality. The stability of the non-linear system (4) is proved with the Lyapunov method [35], using the Lyapunov candidate

$$V(\boldsymbol{x}) = (\eta_d - \eta)^2 + \|\boldsymbol{\epsilon}_d - \boldsymbol{\epsilon}\|^2 + \frac{1}{2} \boldsymbol{\omega}^\top (\boldsymbol{K}^q)^{-1} \boldsymbol{\omega}, \tag{5}$$

where the state $\boldsymbol{x} = [\boldsymbol{q}^\top, \boldsymbol{\omega}^\top]^\top$, $\boldsymbol{q} = [\eta, \boldsymbol{\epsilon}^\top]^\top$, and $\boldsymbol{q}_d = [\eta_d, \boldsymbol{\epsilon}_d^\top]^\top$. The candidate Lyapunov function in (5) is positive definite and it vanishes only at the equilibrium $\hat{\boldsymbol{x}} = [\boldsymbol{q}_d{}^\top, \boldsymbol{0}^\top]^\top$. The time derivative of $V(\boldsymbol{x})$ is

$$\begin{aligned}\dot{V} &= -2\,(\eta_d - \eta)\,\dot{\eta} - 2\,(\boldsymbol{\epsilon}_d - \boldsymbol{\epsilon})\,\dot{\boldsymbol{\epsilon}} + \boldsymbol{\omega}^\top (\boldsymbol{K}^q)^{-1} \dot{\boldsymbol{\omega}} \\ &= (\eta_d - \eta)\,\boldsymbol{\epsilon}^\top \boldsymbol{\omega} + (\boldsymbol{\epsilon}_d - \boldsymbol{\epsilon})^\top (\eta \boldsymbol{I} - \boldsymbol{S}(\boldsymbol{\epsilon}))\,\boldsymbol{\omega} + \boldsymbol{\omega}^\top (\boldsymbol{K}^q)^{-1} \dot{\boldsymbol{\omega}}\end{aligned}$$

where we used $\dot{q}$ in (4) and the quaternion propagation (14). Considering the definition of $\dot{\boldsymbol{\omega}}$ in (4), we obtain that

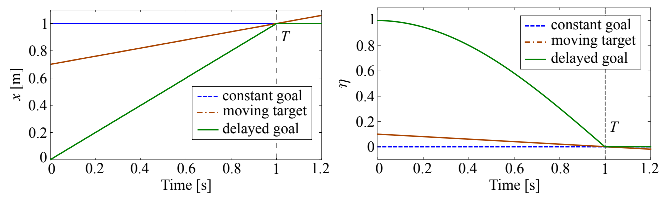
$$\begin{aligned}\dot{V} = {} & (\eta_d - \eta)\,\boldsymbol{\epsilon}^\top \boldsymbol{\omega} + (\boldsymbol{\epsilon}_d - \boldsymbol{\epsilon})\,(\eta \boldsymbol{I} - \boldsymbol{S}(\boldsymbol{\epsilon}))\,\boldsymbol{\omega} \\ & + \boldsymbol{\omega}^\top \cancel{(\boldsymbol{K}^q)^{-1} \boldsymbol{K}^q} \text{vec}(\boldsymbol{q}_d * \overline{\boldsymbol{q}}) - \boldsymbol{\omega}^\top (\boldsymbol{K}^q)^{-1} \boldsymbol{D}^q \boldsymbol{\omega}\end{aligned}$$

Considering the quaternion product in (13) and that $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{a} = \boldsymbol{0}$ if $\boldsymbol{S}(\cdot)$ is a *skew-symmetric* matrix, we obtain that

$$\dot{V} = -\boldsymbol{\omega}^\top (\boldsymbol{K}^q)^{-1} \boldsymbol{D}^q \boldsymbol{\omega} + \boldsymbol{\omega}^\top \cancel{\left(\text{vec}(\boldsymbol{q}_d * \overline{\boldsymbol{q}}) - \text{vec}(\boldsymbol{q}_d * \overline{\boldsymbol{q}})\right)}$$

The matrix $(\boldsymbol{K}^q)^{-1}\boldsymbol{D}^q$, where $(\boldsymbol{K}^q)^{-1}$ and $\boldsymbol{D}^q$ are positive definite matrices, is positive definite iff $(\boldsymbol{K}^q)^{-1}\boldsymbol{D}^q = \boldsymbol{D}^q(\boldsymbol{K}^q)^{-1}$. This property can be guaranteed, for example,

[1]The DMP formulation in (3a)–(3b) is also adopted in [14] but using $\boldsymbol{e}_o(\boldsymbol{q}, \boldsymbol{q}_d) = -\boldsymbol{e}_o(\boldsymbol{q}_d, \boldsymbol{q})$ as orientation error.



(a) Position (b) Quaternion

Fig. 2. The constant goal, moving target, and delayed goal obtained obtained with $\boldsymbol{p}(0) = [0, 0, 0]^\top$ m, $\boldsymbol{p}_d = [1, 0, 0]^\top$ m, $\boldsymbol{q}(0) = [1, 0, 0, 0]^\top$, $\boldsymbol{q}_d = [0, 1, 0, 0]^\top$, $\boldsymbol{v}_d = [0.3, 0.3, 0.3]^\top$ m/s, $\boldsymbol{v}_d = [0.3, 0.3, 0.3]^\top$ m/s, $\boldsymbol{\omega}_d = [0.2, 0.2, 0.2]^\top$ m/s, $\delta t = 0.01$ s, and $T = 1$ s. Only x for the position and η for the quaternion are shown for a better visualization.

by assuming that $\boldsymbol{K}^q$ and $\boldsymbol{D}^q$ are diagonal matrices. If $(\boldsymbol{K}^q)^{-1}\boldsymbol{D}^q$ is a positive definite matrix, then $\dot{V} \leq 0$ and $\dot{V}$ vanishes iff $\boldsymbol{\omega} = \boldsymbol{0}$. The LaSalle's invariance theorem [35] allows to conclude the stability of (4). □

In [15], authors use $\boldsymbol{e}_o = 2\log(\boldsymbol{g} * \overline{\boldsymbol{q}})$. With this choice, the stability can be shown using $V(\boldsymbol{x}) = (\eta_d - \eta)^2 + \|\boldsymbol{\epsilon}_d - \boldsymbol{\epsilon}\|^2 + 0.5\boldsymbol{\omega}^\top\boldsymbol{\omega}$ as Lyapunov function and selecting $\boldsymbol{K}^q = \frac{\|\text{vec}(\boldsymbol{q}_d * \overline{\boldsymbol{q}})\|}{2\arccos(\text{scal}(\boldsymbol{q}_d * \overline{\boldsymbol{q}}))}\boldsymbol{I}$ as stiffness gain. This non-linear stiffness gain has a singularity when $\boldsymbol{q}_d$ and $\boldsymbol{q}$ are aligned. In this work, we use the vector-based quaternion error $\boldsymbol{e}_o = \text{vec}(\cdot, \cdot)$ to avoid non-linearity and singularity in the gain matrices.

III. Merging Pose Motion Primitives

Motion primitives can be combined to execute complex robotics tasks [24]–[26]. In this section, we present three different approaches to merge pose DMPs. Each of them follows a different idea on how to smoothly transition between successive DMPs. We assume that L sequential pose DMPs are given. Each DMP converges to a certain position $\boldsymbol{p}_d^l$ and orientation $\boldsymbol{q}_d^l$ for $l = 1, \ldots, L$. In all the presented approaches the clock signal vanishes for $t \to +\infty$. As discussed in Sec. II-C, this is sufficient to guarantee the convergence to the last goal $\boldsymbol{p}_d^L$, $\boldsymbol{q}_d^L$. Note that blue text is used in the equations to highlight the differences between the approaches in this section and the pose DMP in Sec. II.

A. First Approach

The method described in [27] originates from the assumption that any DMP reaches the end position with zero velocity and zero acceleration. This means that once a motion is fully executed it will come to a full stop and that, close to the goal position, the robot moves with a decreasing velocity. In order to combine L motion primitives, one can stop the current motion when the norm of the velocity is smaller than a certain threshold and start the next primitive. The next primitive is initialized with the state of the previous one ($\boldsymbol{p}_{ne} = \boldsymbol{p}_{pr}$, $\boldsymbol{v}_{ne} = \boldsymbol{v}_{pr}$) to avoid discontinuities. This applies to orientation by initializing the state of the next DMP as $\boldsymbol{q}_{ne} = \boldsymbol{q}_{pr}$, $\boldsymbol{\omega}_{ne} = \boldsymbol{\omega}_{pr}$. Note that the approach applies to any second-order dynamical system including DMPs.

B. Second Approach

The approach in [29] allows to cross the goal position of a DMP with a non-zero velocity. This is achieved by allowing the DMP to track a position target that moves at a given velocity. Hence, the linear acceleration in (1b) becomes

$$\tau\dot{\boldsymbol{v}} = \boldsymbol{K}^p\left[(\boldsymbol{p}_m^l - \boldsymbol{p})(1-h) + \boldsymbol{f}^p(h)\right] + \boldsymbol{D}^p(\boldsymbol{v}_d^l - \boldsymbol{v})(1-h),$$

where $\boldsymbol{v}_d^l$ is the chosen final linear velocity of the l-th DMP and the moving target $\boldsymbol{p}_m^l$ is defined as $\boldsymbol{p}_m^l(t) = \boldsymbol{p}_m^l(0) - \boldsymbol{v}_d^l\frac{\tau\ln(h)}{\gamma}$, $\boldsymbol{p}_m^l(0) = \boldsymbol{p}_d^l - T^l\boldsymbol{v}_d^l$, where $\boldsymbol{p}_d^l$ is the goal position and T^l is the time duration of the l-th DMP. The moving target $\boldsymbol{p}_m^l$ is designed to reach the goal position at $\boldsymbol{p}_m^l(T^l) = \boldsymbol{p}_d^l$ (see Fig. 2(a)). This is because the term $-\tau\ln(h)/\gamma$ represents the time if h is defined by the canonical system $\tau\dot{h} = -\gamma h$. The initial position of the moving target $\boldsymbol{p}_m^l(0)$ is computed by moving the goal position $\boldsymbol{p}_d^l$ for T^l at constant velocity $-\boldsymbol{v}_d^l$. High accelerations at the beginning of the movement are avoided by the prefactor $(1-h)$, that replaces the term $\boldsymbol{d}_0^p(h)$ in (1b).

The presented idea is here extended to unit quaternions. The angular acceleration in (3b) is rewritten as

$$\tau\dot{\boldsymbol{\omega}} = \boldsymbol{K}^q\left[\boldsymbol{e}_o(\boldsymbol{q}_m^l, \boldsymbol{q})(1-h) + \boldsymbol{f}^q(h)\right] + \boldsymbol{D}^q(\boldsymbol{\omega}_d^l - \boldsymbol{\omega})(1-h),$$

where $\boldsymbol{\omega}_d^l$ is the chosen final angular velocity of the l-th DMP and $\boldsymbol{e}_o(\boldsymbol{q}_m^l, \boldsymbol{q}) = \text{vec}(\boldsymbol{q}_m^l * \overline{\boldsymbol{q}})$ as detailed in Sec. II-B. High angular accelerations at the beginning of the motions are prevented by the prefactor $(1-h)$ that replaces the term $\boldsymbol{d}_0^q(h)$ used in (3b). The moving target $\boldsymbol{q}_m^l$ is defined as

$$\begin{aligned}\boldsymbol{q}_m^l(t) &= \exp\left(-\frac{\tau\ln(h)}{2\gamma}\boldsymbol{\omega}_d^l\right) * \boldsymbol{q}_m^l(0),\\ \boldsymbol{q}_m^l(0) &= \exp\left(-\frac{T^l}{2}\boldsymbol{\omega}_d^l\right) * \boldsymbol{q}_d^l,\end{aligned} \tag{6}$$

where $\boldsymbol{q}_d^l$ is the goal orientation and T^l is the time duration of the l-th DMP. The initial orientation of the moving target $\boldsymbol{q}_m^l(0)$ is computed by moving the goal orientation $\boldsymbol{q}_d^l$ for T^l at constant velocity $-\boldsymbol{\omega}_d^l$. Considering the definitions of the exponential map $\exp(\cdot)$ and the quaternion product $*$ in (16) and (13) respectively, it is straightforward to verify that $\boldsymbol{q}_m^l$ reaches the goal quaternion at $\boldsymbol{q}_m^l(T^l) = \boldsymbol{q}_d^l$ (Fig. 2(b)).

Having now the ability to cross each goal after T^l with a non-zero velocity, we can combine multiple motion primitives. Given two consecutive DMPs l and $l+1$, we run l for T^l seconds and then switch to $l+1$. To avoid discontinuities, we initialize the state of $l+1$ with the final state of l [27].

C. Third Approach

The approach in [30] merges multiple DMPs into a single, more complex one. In [30], the canonical system is

$$\dot{h} = -\alpha_h e^{\frac{\alpha_h}{\delta_t}(\tau T - t)}/[1 + e^{\frac{\alpha_h}{\delta_t}(\tau T - t)}]^2, \tag{7}$$

where α_h defines the steepness of the sigmoidal decay function h centred at the time moment T. The value of h is $h = 1$ for $t < T - \delta$, where δ depends on the steepness α_h, and then it decays to $h = 0$. The linear acceleration in (1b) becomes

$$\tau\dot{\boldsymbol{v}} = \boldsymbol{K}^p(\boldsymbol{p}_m^l - \boldsymbol{p}) + \boldsymbol{K}^p\boldsymbol{f}^p(h) - \boldsymbol{D}^p\boldsymbol{v}, \tag{8}$$

while the linear velocity in (1a) is the same. The moving target $\boldsymbol{p}_m^l$, called delayed goal function in [30], is defined as

$$\tau\dot{\boldsymbol{p}}_m^l = \begin{cases}\frac{\delta t}{T^l}(\boldsymbol{p}_d^l - \boldsymbol{p}^l(0)), & \sum_{k=1}^{l-1} T^k \leq t \leq \sum_{k=1}^{l} T^k\\ [0,\ 0,\ 0]^\top, & \text{otherwise}\end{cases}, \tag{9}$$

where $\boldsymbol{p}^l(0)$ and $\boldsymbol{p}_d^l$ are the initial and goal position of the l-th DMP, T^l is the duration of l-th DMP, δt is the sampling rate, $l = 1, \ldots, L$, and L is the number of movement primitives to merge. Note that (9) generates a piecewise linear moving target $\boldsymbol{p}_m^l$ that reaches the goal $\boldsymbol{p}_d^l$ after T^l s (see Fig. 2(a)). Being $\boldsymbol{p}_m^l = \boldsymbol{p}^l(0)$, the acceleration (8) is smooth at the beginning of the motion. For this reason, the term $\boldsymbol{d}_0^p(s)$ used in (1b) is not needed in (8). The non-linear forcing term $\boldsymbol{f}^p(h)$ used in (8) slightly differs from the one in (2)

$$\boldsymbol{f}^p(h) = \frac{\sum_{i=1}^N \boldsymbol{w}_i \psi_i(t)}{\sum_{i=1}^N \psi_i(t)} h, \quad \psi_i(t) = e^{-(\frac{t}{\tau T} - c_i)^2 / 2\sigma_i^2}, \tag{10}$$

where σ_i is the width of the i-th kernel, c_i are their centres, and h is given by (7). The kernels ψ_i in (10) differ from those in (2) since the term $t/\tau T$ replaces the canonical system h. Note that, for $\tau = 1$, $0 \leq t/\tau T \leq 1$ and the kernels are equally spaced between 0 and 1. The kernel widths σ_i are constant and depend on the number of kernels.

Given L DMPs in the described form, one can obtain a single DMP by combining kernels and weights of the separately learned DMPs. In particular, the centers, originally equally spaced between 0 and 1, are replaced by

$$\overline{c}_i^l = \begin{cases} \frac{T^1(i-1)}{T(N-1)}, & l = 1 \\ \frac{T^l(i-1)}{T(N-1)} + \frac{1}{T}\sum\limits_{k=1}^{l-1} T^k, & \text{otherwise} \end{cases}, \tag{11}$$

where N is the number of kernels of each DMP, $i = 1, ..., N$, $l = 1, ..., L$, T^l is the duration of the l-th DMP, and $T = \sum_{k=1}^L T^k$ is the duration of the joint trajectory. The width of the kernels is scaled down by T^l/T, i.e. $\overline{\sigma}_l^l = \sigma_i^l T^l / T$. The weights of each DMP $\boldsymbol{w}_i^l$ remain unchanged. The N kernels and N weights of the L DMPs are stacked together to form a single DMP with NL kernels and NL weights. The combined kernels of succeeding DMPs now intersect at the transition points, resulting in smooth transitions.

We extend the described approach to unit quaternions. The angular acceleration in (3b) is rewritten as $\tau\dot{\boldsymbol{\omega}} = \boldsymbol{K}^q \boldsymbol{e}_o(\boldsymbol{q}_m^l, \boldsymbol{q}) + \boldsymbol{K}^q \boldsymbol{f}^q(h) - \boldsymbol{D}^q \boldsymbol{\omega}$. The quaternion goal function $\boldsymbol{q}_m^l$ ranges from $\boldsymbol{q}^l(0)$ to $\boldsymbol{q}_d^l$ in T^l seconds (see Fig. 2(b)). Hence, $\boldsymbol{q}_m^l$ is a geodesic on $\mathcal{S}^3$ and it is defined as $\boldsymbol{q}_m^l(t+1) = \exp\left(\frac{\tau \boldsymbol{\omega}_m^l}{2}\right) * \boldsymbol{q}_m^l(t)$, where

$$\boldsymbol{\omega}_m^l = \begin{cases} \frac{2}{T^l} \log(\boldsymbol{q}_d^l * \boldsymbol{q}^l(0)) & \sum\limits_{k=1}^{l-1} T^k \leq t \leq \sum\limits_{k=1}^{l} T^k \\ [0, 0, 0]^\top, & \text{otherwise} \end{cases}. \tag{12}$$

In (12), $\boldsymbol{q}_d^l$ is the goal and $\boldsymbol{q}^l(0)$ is the initial orientation of the l-th DMP, T^l is the time duration of the l-th DMP, $l = 1, \ldots, L$, and $2\log(\boldsymbol{q}_d^l * \boldsymbol{q}^l(0))$ is the angular velocity that rotates $\boldsymbol{q}^l(0)$ into $\boldsymbol{q}_d^l$ in a unitary time. The functions $\log(\cdot)$ and $\exp(\cdot)$ are defined in (15) and (16) respectively. Note that the described approach to calculate $\boldsymbol{q}_m^l$ corresponds to interpolate $\boldsymbol{q}^l(0)$ and $\boldsymbol{q}_d^l$ with the SLERP algorithm [36]. To reach the final orientation $\boldsymbol{q}_d^L$, the delayed goal function firstly reaches $\boldsymbol{q}_d^1$, then $\boldsymbol{q}_d^2$, and so on until $\boldsymbol{q}_d^L$ is reached.

IV. Experimental Results

A. Synthetic data

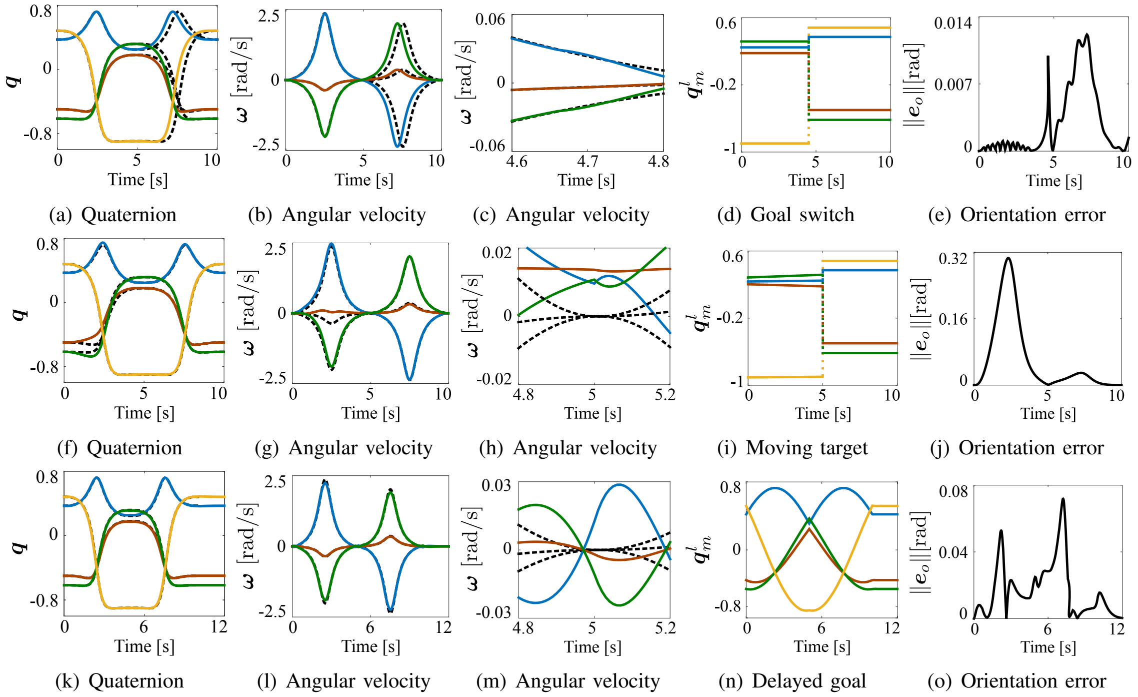
The aim of this experiment is to compare the behaviour of the proposed approaches when applied to generate an orientation trajectory. To this end, we pre-trained two orientation DMPs on synthetic data given by two minimum jerk trajectories connecting $\boldsymbol{q}(0) = [0.247, 0.178, 0.318, -0.897]^\top$ with $\boldsymbol{q}_d^1 = [0.372, -0.499, -0.616, 0.482]^\top$ (intermediate goal) and $\boldsymbol{q}_d^1$ with $\boldsymbol{q}_d^2 = \boldsymbol{q}(0)$ (final goal). Each trajectory lasts for $T^1 = T^2 = 5$ s (black dashed lines in Fig. 3(a)). Each DMP has $N = 15$ kernels, $\tau = 1$, and $\boldsymbol{K}^q = 10\boldsymbol{I}$. These values are empirically set, while $\boldsymbol{D}^q = 2\sqrt{\boldsymbol{K}^p}$ to have a critically damped system [37]. The sampling time is $\delta t = 0.01$ s. The two orientation DMPs are trained to reach the respective goals $\boldsymbol{q}_d^1$ and $\boldsymbol{q}_d^2$ with zero velocity. We apply the approaches presented in Sec. III to generate a smooth quaternion trajectory that starts and ends at $\boldsymbol{q}(0)$ while passing close to the "intermediate goal" $\boldsymbol{q}_d^1$, considered a via point. The performance of each approach is evaluated considering deformation, smoothness, and duration of the generated trajectory, as well as the distance to $\boldsymbol{q}_d^1$.

Results obtained with the three approaches are shown in Fig. 3. For the first approach, we switch to the second DMP when the distance from the intermediate goal (via point) $\boldsymbol{q}_d^1$ is below $d_1 = 0.01$ rad, i.e. after about 4.7 s (Fig. 3(d)). Alternatively, one can switch the primitives when the velocity is below a certain threshold as suggested in [27]. For the second approach, we run the first DMP for $T^1 = 5$ s and then switch to the second one. The desired intermediate velocity is $\boldsymbol{\omega}_d^1 = [0.01, 0.01, 0.01]^\top$ rad/s. The third approach does not require a switching rule between the DMPs and automatically treats $\boldsymbol{q}_d^1$ as a via point. As expected, all the generated trajectories converge to $\boldsymbol{q}_d^2$ (Fig. 3(a), (f), and (k)).

Error plots in Fig. 3(e), (j), and (o) show the deformation introduced by each approach. The first approach is the most accurate (maximum tracking error $e_{o,max} = 0.012$ rad), followed by the third approach ($e_{o,max} = 0.072$ rad). The second approach is the less accurate ($e_{o,max} = 0.307$ rad). The second approach partially sacrifices the accuracy to cross the via point $\boldsymbol{q}_d^1$ after T^1 s ($\boldsymbol{e}_o(T^1) = 0.001$ rad) with velocity $\boldsymbol{\omega}(T^1) \approx \boldsymbol{\omega}_d^1$ (Fig. 3(h)). On the other hand, the third approach favors the overall accuracy passing 0.025 rad away from $\boldsymbol{q}_d^1$. The trajectory pass "close" to the intermediate goal, but the distance depends on the weights of the merged DMPs and cannot be decided a priori. In the first approach, the distance from intermediate goals is a tunable parameter.

In the first approach, the distance to the goal affects the time duration of the generated trajectory. With the used distance $d_1 = 0.01$ rad, the generated trajectory converges to $\boldsymbol{q}_d^2$ (distance below 0.001 rad) after 9.5 s. Hence, the execution is faster than the demonstration ($T = 10$ s). Bigger values of d_1 result in shorter trajectories and vice versa. The second approach produces a trajectory that, as the training data, converges in 10 s. Finally, the third approach generates a trajectory of 11.7 s, that is 1.7 s longer than the demonstrated one. In general, the third approach produces a trajectory that lasts more than the demonstration. This is because the sigmoidal clock signal in (7)—and the effects of the forcing term—vanishes after $T + \delta$ s, where δ depends on the steepness α_h of the sigmoid ($\alpha_h = 1$ in this case). Bigger values of α_h result in shorter trajectories and vice versa.

All the tested approaches are able to generate smooth ori-



(a) Quaternion (b) Angular velocity (c) Angular velocity (d) Goal switch (e) Orientation error

(f) Quaternion (g) Angular velocity (h) Angular velocity (i) Moving target (j) Orientation error

(k) Quaternion (l) Angular velocity (m) Angular velocity (n) Delayed goal (o) Orientation error

Fig. 3. Results obtained by applying the first (a)–(e), second (f)–(j), and third (k)–(o) approach to merge two DMPs trained on synthetic data.

entation trajectories with continuous velocities. Nevertheless, the third approach is the only one capable of generating continuous accelerations, while the others may create discontinuous accelerations around the switching point.

B. Robot experiment

This experiment compares the merging approaches in a real case where a robot adds sugar into a cup (see Fig. 4). The task consists of three motion primitives, namely *1)* reach the sugar bowl and fill the spoon, *2)* put the sugar into the cup, and *3)* reach a final pose. The task is demonstrated by kinesthetic teaching and motion primitives are segmented using zero velocity crossing [38] with a velocity threshold empirically set to of 5 mm/s. The three DMPs are separately learned to reach the relative goal (last pose in the segment) with zero velocity. We use the same parameters as in the previous case. The robot is able to execute the task by stopping at each intermediate goal, but this takes 24.7 s that is 5 s longer than the demonstration. Depending on the task at end, the longer execution time may cause issues. Therefore, we also consider the accuracy of the executionmovements and the success of the task to compare the different merging approaches. For the first approach, we switch the DMP when the distance to the current intermediate goal is below 0.005 m (rad), allowing the robot to successfully execute the task in 19.4 s. The generated trajectory passes close to the intermediate goals (distance below 0.005 m (rad)) and accurately represents the demonstration (maximum errors are $e_{p,max} = 0.006$ m and $e_{o,max} = 0.035$ rad). For the second approach, we set the desired crossing velocity to 0.005 m/s (rad/s) along each direction. The robot is able to cross the goals (distance below 0.001 m and 0.002 rad) but it hits the sugar bowl and fails the task (maximum errors are $e_{p,max} = 0.052$ m and $e_{o,max} = 0.073$ rad). The reason is that when the robot reaches the desired z position it is outside the cup and then it touches the cup while reaching the desired x-y position (Fig. 4). It is worth noticing that the robot is able to execute the task if the crossing velocity is reduced, but this will increase the total execution time. The third approach allows the robot to successfully execute the task in 21.2 s. The generated trajectory passes close to the intermediate goals (0.001 m and 0.02 rad from the first goal, 0.002 m and 0.015 rad from the second goal) and accurately represents the demonstration ($e_{p,max} = 0.018$ m and $e_{o,max} = 0.05$ rad).

C. Discussion

Presented results on synthetic and real data have shown similarities and differences of the three merging approaches. Important features of each approach are summarized in Tab. I. The approach in Sec. III-A generates a trajectory that converges before the demonstration time. The faster convergence may represent a problem, for instance when multiple robots are executing a cooperative task. This issue can be alleviated by increasing the time scaling factor τ to match the demonstrated time. Among the three approaches, the first one is the easiest to implement since it does not require any

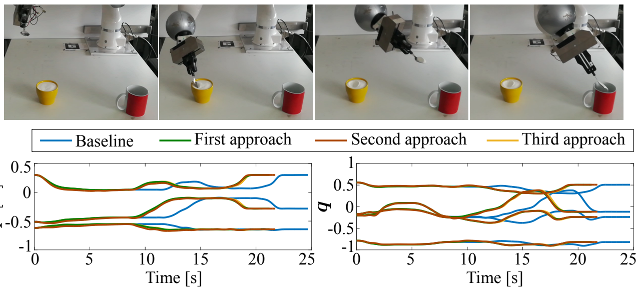


Fig. 4. (Top) Successful execution of the add sugar task. (Bottom) Pose trajectories executed by the robot.

TABLE I

Comparison of the proposed approaches for motion primitives merging.

	Intermediate goal crossing	Desired switch velocity	Change DMP structure	Smooth motion	Computational complexity wrt original DMP
First approach	No	No	No	Continuous velocity	Same
Second approach	Yes	Yes	Yes	Continuous velocity	Same
Third approach	No	No	Yes	Continuous acceleration	Linear with the number of DMPs

change in the DMP structure and in the learning process. On the other hand, approaches two and three requires a moving target and, for approach two, a goal velocity. Hence, the first approach is preferable if standard DMPs were trained and if the goal of a DMP corresponds to the start of the next one.

The second approach introduces a deformation in the generated trajectory. Depending on the desired final velocity, this deformation may not be negligible for the task at hand—as in the presented experiment where the robot touched the sugar bowl and failed the task. However, the second approach is the only capable of crossing the intermediate goals with a user defined velocity. As shown in Fig. 3(c), (h), and (m), the second approach is the only one capable of crossing the goal with a user defined velocity. This is of importance in dynamic tasks like hitting or batting. According to our analysis, the second approach is the best suited for such dynamic tasks.

The third approach stacks kernels and weights of L trained DMPs into one DMP. Assuming that each DMP has N kernels, the resulting DMP has NL kernels and NL weights. Therefore, the computational complexity of the third approach grows linearly with the number of DMPs, while the other two approaches have the same cost of a single DMP. From a certain value of L and N, that depends on the available hardware, the third approach is not able to generate the motion in real-time—typically 1 to 10 ms. To alleviate this issue, one can start generating the trajectory using only the kernels of the first two DMPs. The kernels overlaps only in a neighborhood of the intermediate goal. Hence, after passing the intermediate goal, the kernels of first DMP can be replaced with those of the third one, and so on until the last primitive is reached. The third approach is the only one that generates continuous accelerations. Compared to the first approach (Fig. 3(e)), the third approach slightly deviates from the demonstrated trajectory (Fig. 3(o)) because training data are smoothen to generate smooth accelerations. The third approach outperforms the first one if the velocity of the successive DMP is different from zero. In this case, the first approach starts the second DMP with a velocity close to zero which causes inaccuracies in reproducing the demonstration. The third approach, instead, generates a velocity at the switching point that is closer to the demonstrated one, resulting in a more accurate trajectory.

V. Conclusion

We presented three approaches to combine a set of motion primitives and generate a smooth trajectory for the robot. The approaches assume that each motion primitive is represented via second-order dynamical systems, the so-called dynamic movement primitives. In contrast to similar work in the field, we consider the orientation part of the motion. We represent the orientation via unit quaternions and exploit the mathematical properties of the quaternion space to rigorously define all the operations required to merge sequential movements. Presented approaches are evaluated both on synthetic and real data, showing that each approach has some distinctive features which make it well suited for specific tasks. In the future, we plan to integrate the motion primitives merging approaches with the symbolic task compression in [39] allowing a smooth execution of structured tasks.

Appendix I

Unit quaternions are elements of $\mathcal{S}^3$, the unit sphere in the 3D space. A unit quaternion has four elements $\mathcal{Q} \triangleq \{\eta, \boldsymbol{\epsilon}\} \in \mathcal{S}^3$, where η is the scalar and $\boldsymbol{\epsilon}$ is the vector part of the quaternion. The constraint $\eta^2 + \|\boldsymbol{\epsilon}\|^2 = 1$ relates the scalar and the vector parts. For implementation reasons, a quaternion is represented as a 4D vector $\boldsymbol{q} \triangleq [\eta, \boldsymbol{\epsilon}^\top]^\top = [\eta, \epsilon_1, \epsilon_2, \epsilon_3]^\top$. The product of two quaternions is

$$\boldsymbol{q}_1 * \boldsymbol{q}_2 = [\eta_1\eta_2 - \boldsymbol{\epsilon}_1^\top \boldsymbol{\epsilon}_2,\, (\eta_1\boldsymbol{\epsilon}_2 + \eta_2\boldsymbol{\epsilon}_1 + \boldsymbol{S}(\boldsymbol{\epsilon}_1)\boldsymbol{\epsilon}_2)^\top]^\top, \quad (13)$$

where $\boldsymbol{S}(\boldsymbol{\epsilon}) \in \mathbb{R}^{3\times 3}$ is a *skew-symmetric* matrix. The conjugate of a quaternion, i.e. the quaternion $\overline{\boldsymbol{q}}$ such that $\boldsymbol{q} * \overline{\boldsymbol{q}} = [1, 0, 0, 0]^\top$, is defined as $\overline{\boldsymbol{q}} \triangleq [\eta, -\boldsymbol{\epsilon}^\top]^\top \in \mathcal{S}^3$. The time derivative of a quaternion is related to the angular velocity by the so-called *quaternion propagation*

$$\dot{\boldsymbol{q}} = \frac{1}{2}\tilde{\boldsymbol{\omega}} * \boldsymbol{q} \rightarrow \dot{\boldsymbol{q}} = \begin{bmatrix}\dot{\eta}\\ \dot{\boldsymbol{\epsilon}}\end{bmatrix} = \begin{cases} \dot{\eta} = -\dfrac{1}{2}\boldsymbol{\epsilon}^\top\boldsymbol{\omega} \\ \dot{\boldsymbol{\epsilon}} = \dfrac{1}{2}\left(\eta\boldsymbol{I} - \boldsymbol{S}(\boldsymbol{\epsilon})\right)\boldsymbol{\omega} \end{cases}, \quad (14)$$

where $\tilde{\boldsymbol{\omega}} = [0, \boldsymbol{\omega}^\top]^\top$ is a quaternion with zero scalar part and the angular velocity as vector part. The *logarithmic map*

$$\boldsymbol{r} = \log(\boldsymbol{q}) \begin{cases} \arccos(\eta)\frac{\boldsymbol{\epsilon}}{\|\boldsymbol{\epsilon}\|}, & \|\boldsymbol{\epsilon}\| > 0 \\ [0, 0, 0]^\top, & \text{otherwise} \end{cases}. \quad (15)$$

transforms a unit quaternion into a rotation vector $\boldsymbol{r} \in \mathbb{R}^3$. The logarithmic map is uniquely defined and continuously differentiable if the domain is limited to $\mathcal{S}^3/\{-1, [0, 0, 0]^\top\}$. A rotation vector is mapped into a unit quaternion by the *exponential map*

$$\exp(\boldsymbol{r}) = \begin{cases} \left[\cos(\|\boldsymbol{r}\|), \sin(\|\boldsymbol{r}\|)\frac{\boldsymbol{r}^\top}{\|\boldsymbol{r}\|}\right]^\top, & \|\boldsymbol{r}\| > 0 \\ [1, 0, 0, 0]^\top, & \text{otherwise} \end{cases}. \quad (16)$$

The exponential map is uniquely defined and continuously differentiable if the domain is limited to $0 \leq \|\boldsymbol{r}\| < \pi$. The quaternion derivative (14) is integrated using the formula

$$\boldsymbol{q}(t+1) = \exp\left(\frac{\delta t}{2}\boldsymbol{\omega}(t)\right) * \boldsymbol{q}(t), \quad (17)$$

where δt is the sampling time.

References

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

[2] S. Calinon and D. Lee, "Learning control," in *Humanoid Robotics: a Reference*, P. Vadakkepat and A. Goswami, Eds. Springer, 2019.

[3] D. Lee and C. Ott, "Incremental kinesthetic teaching of motion primitives using the motion refinement tube," *Autonomous Robots*, vol. 31, no. 2, pp. 115–131, 2011.

[4] M. Saveriano, S. An, and D. Lee, "Incremental kinesthetic teaching of end-effector and null-space motion primitives," in *International Conference on Robotics and Automation*, 2015, pp. 3570–3575.

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

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

[7] C. Blocher, M. Saveriano, and D. Lee, "Learning stable dynamical systems using contraction theory," in *nternational Conference on Ubiquitous Robots and Ambient Intelligence*, 2017, pp. 124–129.

[8] S. M. Khansari-Zadeh and A. Billard, "Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Rob. And Auton. Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[9] K. Neumann and J. J. Steil, "Learning robot motions with stable dynamical systems under diffeomorphic transformations," *Robotics and Autonomous Systems*, vol. 70, pp. 1–15, 2015.

[10] N. Perrin and P. Schlehuber-Caissier, "Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems," *Systems & Control Letters*, vol. 96, pp. 51–59, 2016.

[11] M. Saveriano and D. Lee, "Incremental skill learning of stable dynamical systems," in *International Conference on Intelligent Robots and Systems*, 2018, pp. 6574–6581.

[12] K. Kronander, S. M. Khansari Zadeh, and A. Billard, "Incremental motion learning with locally modulated dynamical systems," *Robotics and Autonomous Systems*, vol. 70, pp. 52–62, 2015.

[13] E. Gribovskaya and A. Billard, "Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators," in *International Conference on Humanoid Robots*, 2009, pp. 472–477.

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

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

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

[17] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[18] M. Saveriano and D. Lee, "Point cloud based dynamical system modulation for reactive avoidance of convex and concave obstacles," in *International Conference on Intelligent Robots and Systems*, 2013, pp. 5380–5387.

[19] ——, "Distance based dynamical system modulation for reactive avoidance of moving obstacles," in *International Conference on Robotics and Automation*, 2014, pp. 5618–5623.

[20] M. Saveriano, F. Hirt, and D. Lee, "Human-aware motion reshaping using dynamical systems," *Pattern Recognition Letters*, vol. 99, pp. 96–104, 2017.

[21] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *International Conference on Robotics and Automation*, 2009, pp. 1534–1539.

[22] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012.

[23] M. Mühlig, M. Gienger, and J. J. Steil, "Interactive imitation learning of object movement skills," *Autonomous Robots*, vol. 32, no. 2, pp. 97–114, 2012.

[24] S. Manschitz, J. Kober, M. Gienger, and J. Peters, "Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations," *Robotics and Autonomous Systems*, vol. 74, pp. 97–107, 2015.

[25] R. Caccavale, M. Saveriano, G. A. Fontanelli, F. Ficuciello, D. Lee, and A. Finzi, "Imitation learning and attentional supervision of dual-arm structured tasks," in *International Conference on Development and Learning and on Epigenetic Robotics*, 2017, pp. 66–71.

[26] R. Caccavale, M. Saveriano, A. Finzi, and D. Lee, "Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction," *Autonomous Robots*, 2018.

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

[28] B. Nemec, M. Tamosiunaite, F. Woergoetter, and A. Ude, "Task adaptation through exploration and action sequencing," in *International Conference on Humanoid Robots*, 2009, pp. 610–616.

[29] J. Kober, K. Mülling, O. Krömer, C. H. Lampert, B. Schölkopf, and J. Peters, "Movement templates for learning of hitting and batting," in *International Conference on Robotics and Automation*, 2010, pp. 853–858.

[30] T. Kulvicius, K. Ning, M. Tamosiunaite, and F. Worgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2012.

[31] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *International Conference on Humanoid Robotics*, 2008, pp. 91–98.

[32] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st ed. Springer, 2008.

[33] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback," *Journal on Robotics and Automation*, vol. 4, no. 4, pp. 434–440, 1988.

[34] L. Markus, "Asymptotically autonomous differential systems," in *Contributions to the Theory of Nonlinear Oscillations III*, S. Lefschetz, Ed. Princeton University Press, 1956, pp. 17–30.

[35] J. Slotine and W. Li, *Applied nonlinear control*. Prentice-Hall Englewood Cliffs, 1991.

[36] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, 1985, pp. 245–254.

[37] R. Weitschat, A. Dietrich, and J. Vogel, "Online motion generation for mirroring human arm motion," in *International Conference on Robotics and Automation*, 2016, pp. 4245–4250.

[38] A. Fod, M. J. Matarić, and O. C. Jenkins, "Automated derivation of primitives for movement classification," *Autonomous Robots*, vol. 12, no. 1, pp. 39–54, 2002.

[39] M. Saveriano, M. Seegerer, R. Caccavale, A. Finzi, and D. Lee, "Symbolic task compression instructured task learning," in *International Conference on Robotic Computing*, 2019, pp. 171–176.