REGULAR PAPER

Dynamic Via-points and Improved Spatial Generalization for Online Trajectory Generation with Dynamic Movement Primitives

Antonis Sidiropoulos[1] · Zoe Doulgeri[1]

Abstract

Dynamic Movement Primitives (DMP) have found remarkable applicability and success in various robotic tasks, which can be mainly attributed to their generalization, modulation and robustness properties. However, the spatial generalization of DMP can be problematic in some cases, leading to excessive overscaling and in turn large velocities and accelerations. While other DMP variants have been proposed in the literature to tackle this issue, they can also exhibit excessive overscaling as we show in this work. Moreover, incorporating intermediate points (via-points) for adjusting the DMP trajectory to account for the geometry of objects related to the task, or to avoid or push aside objects that obstruct a specific task, is not addressed by the current DMP literature. In this work we tackle these unresolved so far issues by proposing an improved online spatial generalization, that remedies the shortcomings of the classical DMP generalization, and moreover allows the incorporation of dynamic via-points. This is achieved by designing an online adaptation scheme for the DMP weights which is proved to minimize the distance from the demonstrated acceleration profile to retain the shape of the demonstration, subject to dynamic via-point and initial/final state constraints. Extensive comparative simulations with the classical and other DMP variants are conducted, while experimental results validate the practical usefulness and efficiency of the proposed method.

Keywords Dynamic movement primitives · Via-points · Programming by demonstration · Trajectory generation

1 Introduction

Dynamic Movement Primitives (DMP) [1, 2] have emerged as a promising method for encoding a desired trajectory and generalizing it to new situations. It is particularly favoured thanks to its ease of training, generalization, robustness and on-line modulation properties and has been applied in a plethora of practical robotic tasks and applications [3–8]. The trajectory to be encoded is typically provided through PbD (Programming by Demonstration) [9], as an intuitive and efficient means of teaching human-skills to robots.

In DMP, the objective of generalization is to generate a trajectory from a new initial position to a new, possibly time-varying, target position, while also preserving the shape of the demonstrated trajectory. To this end, different DMP variations and/or spatial generalization mechanisms have been proposed with the most well-known being the classical DMP formulation from [1]. However, the classical DMP spatial scaling has some important shortcomings, i.e. over-scaling of the generalized trajectory when the demonstrated end-points (initial and final position) are close, failure to generate a motion when the new end-points during execution coincide and mirroring of the trajectory when the sign of the difference between the new end-points is different from the sign of the difference of demonstrated ones. To remedy these issues, the bio-inspired DMP formulation is proposed in [10]. Nevertheless, it exhibits poor generalization for targets that are not close to the demonstration, as highlighted in [11], where an improved version of the bio-inspired DMP is proposed to enhance its spatial generalization. A novel generalization to tackle the scaling issues of classical DMP is also proposed in [12]. While [11, 12] resolve the classical DMP scaling issues, they introduce over-scaling in trajectories with amplitudes bigger than the distance between the initial and final position, which is typical in many practical applications, like in packing applications where lifting an object and placing it

✉ Antonis Sidiropoulos
   antosidi646@gmail.com

   Zoe Doulgeri
   doulgeri@ece.auth.gr

[1] Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Panepistimioupoli, Thessaloniki 54124, Greece

down in a box is required. A different offline spatial generalization approach is proposed in [13] based on minimizing a norm, that is learned from multiple demonstrations, solving an iterative optimization problem.

Few DMP works consider via points on-line in their spatial generalization. Via-points can be particularly useful to adjust a trajectory on-line, e.g. to avoid obstacles in a predictable way by specifying through the via-points how the robot should circumvent them [14] or to push aside other objects that obstruct reaching pathways to a desired target [15]. Moreover, via-points offer a more flexible way of adjusting online a DMP trajectory, in contrast to other approaches that propose to join separate DMPs [16, 17]. In the latter case, the trajectory segments need to be pre-specified, the complexity increases since multiple DMPs have to be defined to connect the via-points and also extra adjustments are required to transition smoothly from one DMP segment to the next without halting the motion [6]. Via-points are considered off-line in a DMP-like variant in [18] through offline optimization and in [19], which can include via-point constraints. So far, online incorporation of via-points is performed with probabilistic DMP [20], by online adapting the DMP weights to the new forcing term values, based on the provided via-points. Calculating the new forcing term value at via-points requires estimates of velocity and acceleration at the via-point, which can result in significant noise that gets even more adverse the sparser the via-points are arranged, as shown in [21]. Another approach to include via-points is proposed in [22] by introducing a time-varying attractor that shifts smoothly from one via-point to the next until the final target [22]. Nevertheless, this does not ensure that the via-point will be reached and the distortions in the trajectory shape that this time-varying attractor incurs are not predictable.

Other movement primitives that can include via-points are the Probabilistic Movement Primitives (ProMP) [23, 24] and the Kernelized Movement Primitives (KMP) [25]. However, multiple demonstrations are required for training such primitives. Moreover, they cannot generalize well away from the demonstrations and have higher computational complexity with multiple DoFs. Of course, these primitives have other advantages compared to DMP and vice versa; choosing the most appropriate depends on the target application. In [14] the Via-point Movement Primitives (VMP) are introduced, that adopt ideas from DMP and ProMP, to achieve adaptation to via-points, and are shown to have better extrapolation capabilities compared to ProMP, being also able to be trained from a single demonstration.

All aforementioned works are either off-line, or those that are online, consider static via-points, and do not address the case of via-points that may change dynamically. Dynamic changes of via-points occur if via-points are defined with respect to the position of the target or an obstacle, which is perturbed during execution. For instance in a human-robot collaborative packing scenario a via-point may be defined on top of the target box which may be displaced by the human to a more ergonomic position for him. The robot has to adjust on-line to this change.

In this work we propose an online adaptation scheme for the DMP weights based on minimizing the distance from the learned acceleration profile, that allows incorporation of dynamic via points in the DMP generated trajectory as compared to VMP that consider only static ones [14] and spatial generalization to new targets without the classical DMP scaling problems and the over-scaling that may appear in the variants [11, 12]. We achieve this by exploiting the novel DMP formulation presented in our previous work [26]. Moreover, we show how the proposed novel generalization can also be combined with our previous work [27] to impose kinematic inequality constraints in order to generate feasible trajectories in the presence of kinematic bounds related to the robot and the task environment (obstacles and via points).

The rest of this paper is organized as follows: In Section 2 we provide the preliminaries for the novel DMP formulation from [26]. Section 3 is devoted to the proposed novel online generalization. Simulations that demonstrate how the proposed method performs in comparison with the classical and other SoA DMP variants are provided in Section 4. The practical usefulness and efficacy of the proposed method is further showcased in the experimental Section 5. Insights and discussion on the results are provided in Section 6 and finally conclusions are drawn in Section 7. The source code for all conducted simulations and experiments can be found at https://github.com/Slifer64/novel_dmp_generalization.git.

2 DMP Preliminaries

In this section we briefly present the DMP formulation introduced in our previous work [26], which we exploit in this work to derive an improved on-line spatial generalization that can also incorporate dynamic via-points. As shown in [26], this formulation is mathematically equivalent to the classical DMP formulation from [1], retaining all properties of the latter.

Consider the state $\boldsymbol{y} \in \mathbb{R}^n$, which can be joint positions, Cartesian position or Cartesian orientation expressed using the quaternion logarithm (see Appendix A for preliminaries on unit quaternions). The DMP consists of two sets of differential equations, the transformation and the canonical system. A DMP can encode a desired trajectory t_j, $\boldsymbol{y}_{d,j}$ for $j = 1...m$ where m is the total number of points and $T_{f,d} = t_m - t_1$ the time duration of the demo, and through its transformation system it can generalize this trajectory spatially from an initial position $\boldsymbol{y}_0$ to a desired target position $\boldsymbol{g}$. Temporal scaling, i.e. execution of this trajectory with a different time duration or speed, is achieved through the

canonical system, which provides the clock s (time substitute) for the transformation system, to avoid direct time dependency. The **transformation system** is given by:

$$\ddot{y} = \ddot{y}_s - D(\dot{y} - \dot{y}_s) - K(y - y_s) \tag{1}$$

$$y_s(s) = K_s(f_p(s) - \hat{y}_{d,0}) + y_0 \tag{2}$$

$$f_p(s) = W^T \phi(s) \tag{3}$$

$$K_s = \text{diag}\left((g - y_0)./(\hat{g}_d - \hat{y}_{d,0})\right) \tag{4}$$

with K, $D \in \mathcal{S}_{++}^n$, with $\mathcal{S}_{++}^n$ denoting the set of symmetric $n \times n$ positive definite matrices, y_s is the scaled learned trajectory whose derivatives can be calculated analytically in closed form [26], the matrix K_s achieves the spatial scaling from the new initial position y_0 to a new target g, while $\hat{g}_d \triangleq f_p(s(T_{f,d}))$, $\hat{y}_{d,0} \triangleq f_p(s(0))$ are the learned initial and target demo positions. The demonstrated position trajectory is encoded through the weighted sum of Gaussians in $f_p(s)$, where the DMP weights $W \in \mathbb{R}^{K \times n}$ can be determined using Least Square, i.e. $\min_W \sum_{j=1}^m ||y_{d,j} - W^T \phi_j||_2^2$, or Locally Weighted Regression (LWR) [1]. The Gaussian kernels are $\phi(s)^T = [\psi_1(s) \cdots \psi_K(s)]/\sum_{i=1}^K \psi_i(s)$, with $\psi_i(s) = exp(-h_i(s - c_i)^2)$ with c_i being the centers and h_i the inverse widths. A reasonable heuristic is to set the centers c_i equally spaced in $[0, 1]$ and set $h_i = 1/(a_h(c_{i+1} - c_i))^2$ where a_h controls the standard deviation of the Gaussian kernels. Choosing $a_h \in [1.2, 1.5]$ produces empirically good approximation results.

Given $s(0) \triangleq s_0$, $s(\infty) \triangleq s_f$, and considering without loss of generality $s_0 = 0$ and $s_f = 1$, the **canonical system** is given by:

$$\ddot{s} = \begin{cases} d_1(\dot{s}_d - \dot{s}) & , s < s_f \\ -d_2\dot{s} - k_2(s - s_f) & , s \geq s_f \end{cases} \tag{5}$$

with $d_1, d_2, k_2 > 0$, $d_2 \geq 2\sqrt{k_2}$, $s(0) = 0$ and $\dot{s}(0) = \dot{s}_d = 1/T_f$, where T_f is the desired motion duration. In the general case $\dot{s}_d$ can also be time-varying. Notice that Eq. 5 guarantees that the phase s convergences from $s(0) = 0$ to $s \to 1$, $\dot{s}, \ddot{s} \to 0$ as $t \to \infty$.

As shown in [26], the novel DMP formulation Eqs. 1-4 is mathematically equivalent to the classical DMP. In particular, Eqs. 1-4 can be mathematically manipulated to be written in a form similar to the classical one:

$$\dot{z} = K(g - y) - Dz + (g - y_0)f_s(s) \tag{6}$$

$$\dot{y} = z \tag{7}$$

$$f_s(s) = \text{diag}(1./(g - y_0))(\ddot{y}_s + D\dot{y}_s - K(g - y_s)) \tag{8}$$

The system is globally asymptotically stable at g, since for $t \to \infty$, $f_s \to 0$, following from $\dot{s}, \ddot{s} \to 0$ implying $\dot{y}_s, \ddot{y}_s \to 0$ and $s \to 1 \Rightarrow y_s \to g$.

Spatial scaling is achieved through K_s from Eq. 4 and temporal scaling by changing $\dot{s}_d$ in Eq. 5. These are equivalent to changing $\{g, y_0\}$ or the temporal scaling τ according to $\dot{\tau} = d_1(\tau_d - \tau)$ in the classical DMP [1], resulting in the exact same trajectory with the same spatial and temporal scaling [26].

Notice that to achieve temporal scaling, in contrast to the classical DMP formulation, there is no need to pre-multiply Eqs. 6 - 7 by τ, as this is handled by $\dot{s}, \ddot{s}$ which are included in $\dot{y}_s, \ddot{y}_s$ in $f_s(s)$ (this can be better understood by the mathematical equivalence analysis in [26]).

For encoding Cartesian orientation, as shown in [28], the correct way is to formulate the DMP in the tangent space of $\mathcal{S}^3$ (unit quaternion manifold) and use the quaternion logarithm Eq. 18 to map points from $\mathcal{S}^3$ to its tangent space and the quaternion exponential Eq. 19 for the inverse mapping. Therefore, for Cartesian orientation, we define the state $y = \log(Q * \bar{Q}_0)$, which maps the unit quaternion Q on the tangent space of $\mathcal{S}^3$ at Q_0. Then, from the reference trajectory $y, \dot{y}, \ddot{y}$ obtained from Eq. 1, we can retrieve the unit quaternion, rotational velocity and acceleration using the mappings Eqs. 19, 20, and 24 from Appendix A.

3 Proposed Generalization

Here we present a novel spatial generalization scheme that does not suffer from the drawbacks of the classical DMP generalization, reported in [10] and further allows the on-line incorporation of dynamic via-points[1], while preserving the shape of the demonstrated trajectory, ensuring a smooth trajectory generation even for abrupt target or via-point changes. To this end, we remove the classical DMP scaling K_s, and redefine Eq. 2 as

$$y_s = f_p(s) \tag{9}$$

and achieve spatial generalization by optimizing online the DMP weights under the initial and final state constraints i.e. start at $t = 0$ from y_0 and reach the target g[2] at $t = T_f$ with zero velocity/acceleration. In order to generate a motion profile similar to the demonstration we minimize the distance from the learned (from the demonstration) acceleration profile. This online optimization allows also the incorporation of via-points in the spatial generalization. The above is translated to the following optimization problem, that optimizes

---

[1] The term "point" can refer either to joint positions, Cartesian position, Cartesian orientation or Cartesian pose.

[2] We assume that g can change between 0 and T_f and is constant afterwards.

the DMP weights W and has to be solved at each time-step i:

$$\min_W \sum_{j=1}^{m} \left\| \frac{\partial^2 \hat{y}_{d,j}}{\partial^2 s} - W^T \frac{\partial^2 \phi_j}{\partial^2 s} \right\|_2^2 \tag{10a}$$

$$\text{s.t.} \quad W^T A(0) = Y_0 \tag{10b}$$

$$W^T A(1) = G_i \tag{10c}$$

$$W^T \Phi_{v,i} = Y_{v,i} \tag{10d}$$

$$W^T C_j = Y_j , \quad j = 1...i \tag{10e}$$

where the learned acceleration profile is[3] $\frac{\partial^2 \hat{y}_{d,j}}{\partial^2 s} = W_0^T \frac{\partial^2 \phi_j}{\partial^2 s}$ with $W_0 = \text{argmin}_W \sum_{j=1}^{m} \left\| y_{d,j} - W^T \phi_j \right\|_2^2$. The matrices in the first two equality constraints, Eqs. 10b, 10c, with $A(s) = [\phi(s) \ \dot{\phi}(s) \ \ddot{\phi}(s)]$, enforce the initial state $Y_0 = [y_0 \ 0_{n \times 1} \ 0_{n \times 1}]$ and final state $G_i = [g_i \ 0_{n \times 1} \ 0_{n \times 1}]$ conditions. Via-point constraints, which can be used to locally modify the DMP trajectory, are defined in Eq. 10d, where $Y_{v,i} = [y_{v,1} \ ... \ y_{v,L_i}]$ contains the via-points, with L_i their number at the current time-step i, and $\Phi_{v,i} = [\phi(s_{v,1}) \ ... \ \phi(s_{v,L_i})]$.[4] The constraints in Eq. 10e with $C_j = C(s_j) = [\phi(s_j) \ \dot{\phi}(s_j) \ \ddot{\phi}(s_{j-1})]$ and $Y_j = [y_j \ \dot{y}_j \ \ddot{y}_{j-1}]$ encode the current and all previous state constraints up to timestep i. The default option is to set $Y_j = W_{i-1}^T C_j$. An alternative option is to set Y_j to the actual robot's state at step j if it is also desirable to adapt online the DMP to changes of the robot's state induced by external signals, like measured forces. Regardless of the choice of Y_j, the current state constraint for $j = i$, ensures that a smooth trajectory is generated by Eq. 1 even in the presence of abrupt target changes. The previous state constraints $Y_{1:i-1}$ guarantee that if the DMP is run in reverse as in [26], using the final adapted weights from the forward execution, the same trajectory (in reverse) will be executed. This could be expedient for safely retracting after the forward execution of a task, especially if the task has geometric constraints (like an insertion) which will have to be respected in the retraction phase.

3.1 Online DMP Weights Adaptation

Solving the problem in Eq. 10 on-line at each control cycle would be too slow and hence prohibitive for real-time usage. Instead we can employ the following recursive 2-step update

at each time-step $i > 0$, which has computational complexity $O(K^2)$ and can be carried out in real-time:

step 1: Remove the effect of the previous target g_{i-1} and via-points:

$$\hat{W}_i = W_{i-1} + \hat{K}_i (\hat{Z}_i - W_{i-1}^T \hat{H}_i)^T \tag{11a}$$

$$\hat{P}_i = P_{i-1} - \hat{K}_i \hat{H}_i^T P_{i-1} \tag{11b}$$

$$\hat{K}_i = P_{i-1} \hat{H}_i (\hat{R}_i + \hat{H}_i^T P_{i-1} \hat{H}_i)^{-1} \tag{11c}$$

where

$$\begin{aligned} \hat{Z}_i &= [G_{i-1} \ Y_{v,i}^-], \ \hat{H}_i = [A(1) \ \Phi_{v,i}^-], \\ \hat{R}_i &= -\epsilon I_{3+b_i^-} \end{aligned} \tag{12}$$

with $\epsilon \approx 0^+$ and $Y_{v,i}^- \in \mathbb{R}^{n \times b_i^-}$ containing in columns the via-points that exist in the columns of $Y_{v,i-1}$ and not in $Y_{v,i}$ (i.e. are present at time-step $i-1$ and not at i)[5] and $\Phi_{v,i}^- = [\phi(s_k)]_{k=1:b_i^-}$ with s_k being the phase corresponding to each via-point in $Y_{v,i}^-$.

step 2: Update to new target and via-points:

$$W_i = \hat{W}_i + K_i (Z_i - \hat{W}_i^T H_i)^T \tag{13a}$$

$$P_i = \hat{P}_i - K_i H_i^T \hat{P}_i \tag{13b}$$

$$K_i = \hat{P}_i H_i (R_i + H_i^T \hat{P}_i H_i)^{-1} \tag{13c}$$

where

$$\begin{aligned} Z_i &= [Y_i \ G_i \ Y_{v,i}^+], \ H_i = [C_i \ A(1) \ \Phi_{v,i}^+], \\ R_i &= \epsilon \ I_{6+b_i^+} \end{aligned} \tag{14}$$

with $Y_{v,i}^+ \in \mathbb{R}^{n \times b_i^+}$ containing in columns the via-points that are present at time-step i and not at $i-1$ and $\Phi_{v,i}^+ = [\phi(s_k)]_{k=1:b_i^+}$ with s_k being the phase corresponding to each via-point in $Y_{v,i}^+$.

For initialization at step $i = 0$ we set $Z_0 = [Y_0 \ G_0]$, $H_0 = [A(0) \ A(1)]$, $R_0 = \epsilon I_6$, $\hat{W}_0 = W_0$ and $\hat{P}_0 = P_0$ where

$$W_0 = \text{argmin}_W \sum_{j=1}^{m} \left\| y_{d,j} - W^T \phi_j \right\|_2^2$$

$$P_0 = \left(\sum_{j=1}^{m} \frac{\partial^2 \phi_j}{\partial^2 s} \frac{\partial^2 \phi_j}{\partial^2 s}^T \right)^{-1}$$

Remark 1 At the beginning of each execution, the DMP weights are initialized to the values that have been calculated

---

[3] Notice from Eq. 5 that, for nominal execution, $\ddot{s} = 0$ and hence the acceleration, i.e. the 2nd time derivative of Eq. 2, is just $\partial^2 \hat{y}_{d,j}/\partial s^2$ scaled by the constant $1/T_f^2$.

[4] If the phase $s_{v,l}, l = 1...L_i$, is not provided explicitly, a reasonable heuristic is to calculate it as $s_v = \text{argmin}_{s_k} ||y_s(s_k) - y_v||$, where s_k in uniformly sampled in $(s, 1)$. Empirically, taking approximately 80 samples is sufficient, as the minimization need not be exact.

[5] Such is the case for instance if they are defined relative to a varying target or an object, whose position has changed or has been completely removed.

offline once during the DMP training from the demonstration. Alternatively, if at a previous execution the DMP was modified, e.g. from via-points or the robot's actual state, resulting to some final weights W_f, and it is desirable to retain this adjusted pattern at subsequent executions, one could set $W_0 = W_f$.

Remark 2 When g is constant and dynamic changes are not expected (for the target or via-points), it suffices to run the initialization step ($i = 0$) from above to obtain W_1 and then keep the DMP weights constant, i.e. $W_i = W_1$.

Remark 3 In practice, due to finite numerical precision, setting ϵ in $\hat{R}_i$, R_i very close to zero can make the matrix being inverted in Eqs. 11c and 13c ill-conditioned. We have found that in practice choosing ϵ approximately in the range $(10^{-10}, 10^{-6})$ does not create any numerical issues and the constraints are satisfied within an error tolerance of the order of 10^{-4} or even less depending on how small ϵ is.

Remark 4 Despite using the same value ϵ in the above analysis for simplicity, it is easy to verify that different values of ϵ can be chosen for each type of constraint. The selected ϵ value can facilitate finding a solution in the optimization problem Eq. 10 in case constraints cannot be strictly satisfied (i.e. infinite accuracy). Then, the value of ϵ essentially relaxes equality constraints, by penalizing the cost function (see Theorem 3 from Appendix B). Choosing different values of ϵ for each type of constraint, allows to prioritize which of these constraints should be satisfied with greater accuracy. Moreover, when adapting to the actual robot's state, which may be perturbed by noisy external signals, higher values of ϵ can be used to suppress that noise.

Remark 5 The proposed recursive algorithm for updating the DMP weights to dynamically changing targets and/or via-points while ensuring a smooth trajectory generation can also be employed with other objective functions as well. For instance, it can be easily verified that optimizing the position (instead of the acceleration) results in the same equations with $P_0 = (\sum_{j=1}^{m} \phi_j \phi_j^T)^{-1}$. More general objective functions can also be specified, like the one from [13], which translates to defining $P_0 = \sum_{j=1}^{m} (\phi_j M \phi_j^T)^{-1}$, where M is a metric matrix that is learned from multiple demonstrations.

Remark 6 The proposed recursive algorithm to include dynamic via-points can also be applied with the spatial scaling from [1, 12] or [11]. In particular, one can use y_s as defined in Eq. 2 with the corresponding scaling matrix K_s, resulting from [1, 12] or [11], and employ the same recursive updates Eqs. 11, 13 by replacing W with $K_s W$ and transforming each position y (be it the current position, target,

via-point etc.) according to $y := y - y_0 + K_s \hat{y}_{d,0}$. This can also be combined with a different objective cost as explained in Remark 5.

3.2 Derivation/Proof of the Weights Adaptation

In the following, we prove that Eqs. 11, 13 solve the initial optimization problem given in Eq. 10. To this end, we will utilize the theorems provided in Appendix B. Problem Eq. 10 can be written compactly as:

$$\min_W \text{tr}\{(\ddot{\bar{Y}}_d - W^T \ddot{\bar{\Phi}})^T (\ddot{\bar{Y}}_d - W^T \ddot{\bar{\Phi}})\} \tag{15}$$
$$\text{s.t.} \quad W^T \bar{H}_i = \bar{Z}_i$$

where $\ddot{\bar{Y}}_d = [\frac{\partial^2 \hat{y}_{d,1}}{\partial^2 s} \ ... \ \frac{\partial^2 \hat{y}_{d,m}}{\partial^2 s}] \in \mathbb{R}^{n \times m}$, $\ddot{\bar{\Phi}} = [\frac{\partial^2 \phi_1}{\partial^2 s} \ ... \ \frac{\partial^2 \phi_m}{\partial^2 s}] \in \mathbb{R}^{K \times m}$, $\bar{Z}_i = [Y_0 \ G_i \ Y_{1:i} \ Y_{v,i}] \in \mathbb{R}^{n \times m_1}$ and $\bar{H}_i = [A(0) \ A(1) \ C_{1:i} \ \Phi_{v,i}] \in \mathbb{R}^{K \times m_1}$ with $m_1 = 6 + 3i + L_i$. Based on Theorem 4 problem Eq. 15 is equivalent to:

$$\min_W f_i(W) \triangleq \text{tr}\{(\ddot{\bar{Y}}_d - W^T \ddot{\bar{\Phi}})^T (\ddot{\bar{Y}}_d - W^T \ddot{\bar{\Phi}})\}$$
$$+ \text{tr}\{(\bar{Z}_i - W^T \bar{H}_i)^T \bar{R}_i^{-1} (\bar{Z}_i - W^T \bar{H}_i)\} \tag{16}$$

for $\bar{R}_i = \epsilon I_{m_1}$, $\epsilon \approx 0^+$. Given now the solution at timestep $i - 1$ for problem Eq. 16 we want to find the solution for timestep i, for which the cost function can be written as:

$$f_i = \hat{f}_{i-1} + \text{tr}\{(Z_i - W^T H_i)^T R_i^{-1} (Z_i - W^T H_i)\}$$

where $\hat{f}_{i-1} = f_{i-1} - \text{tr}\{(\hat{Z}_i - W^T \hat{H}_i)^T \hat{R}_i^{-1} (\hat{Z}_i - W^T \hat{H}_i)\}$. Invoking Theorem 5 the solution to $\hat{f}_{i-1}$ is given by Eq. 11. Given now $\hat{W}_i$, $\hat{P}_i$ for $\hat{f}_{i-1}$ and using Theorem 3 the solution to f_i is equivalent to the solution of $\text{tr}\{(W - \hat{W}_i)^T \hat{P}_i (W - \hat{W}_i)\} + \text{tr}\{(Z_i - W^T H_i)^T R_i^{-1} (Z_i - W^T H_i)\}$, which based on Theorem 2 is given by Eq. 13.

4 Simulations

In the following we compare the DMP produced with the proposed spatial generalization, henceforth referred as DMP^{++}, with the classical DMP, as well as other SoA variations that have been proposed in the literature. We further demonstrate and comment on the effects of adaptation to dynamic via-points and also including kinematic limits (inequality constraints) using the proposed generalization with [27]. In the following simulations and experiments, the DMP parameters (Section 2) were set to the following values: for the weighted sum of Gaussians in Eq. 3, the number of weights per DoF in W is $K = 25$; for the canonical system in Eq. 5 $d_1 = 30$ for tracking the desired speed of the trajectory evolution, and $d_2 = 100$, $k_2 = 1000$ so that after the desired

time duration T_f, fast convergence of the DMP to the target g is ensured; for the DMP dynamical system Eq. 1, we set for position $K = 40$, $D = 10$ and for orientation $K = 2$, $D = 0.6$ to ensure compliance in the presence of external forces. For all other DMP variants, against which we compare, the same canonical system and DMP parameters where used.

Regarding the proposed optimization, based on Remark 4 and in the spirit that higher priority should be placed on the initial and final position constraints followed by the via-point constraints and then the previous state constraints, we chose $\epsilon = 10^{-9}$ for the initial and final position, 10^{-7} for the initial and final velocity and acceleration constraints, 10^{-7} for via-points and $10^{-6}, 10^{-6}, 10^{-4}$, for the previous state constraint. Notice that violation of the previous state constraint generates a discontinuous acceleration in the DMP Eq. 1, which will be larger the bigger the error in the equality constraint is. For via-points, the phase variable assigned at a via-point $\boldsymbol{y}_v$ is determined as $s_v = \text{argmin}_{s_k} ||\boldsymbol{y}_s(s_k) - \boldsymbol{y}_v||$, where s_k in uniformly sampled in $(s, 1]$ taking 80 points, where s is the current value of the phase variable.

4.1 Comparison with Classical DMP

We start off by examining the three basic cases where the classical DMP scaling is problematic [10]. For simplicity we consider an 1 DoF demonstration starting at position zero. Simulation results for each case are plotted in Fig. 1, where the demo is plotted with green dashed line, the proposed DMP^{++} with blue and the classical DMP with magenta dotted line. In the first case (Fig. 1, top subplot) the demonstrated target g_d is quite close to the initial position $y_{0,d} = 0$. In this case, even a new target that is close to that of the demo, results in over-scaling for the classical DMP. In the figure, the resulting trajectory for the classical DMP is scaled by 400 for visualization purposes (the position actually converges to g but in the plot due to the scaling by 400 it is $g/400 \approx 0$). In the second case (Fig. 1, middle subplot) the demo initial and target positions are different but the new target is the same with the initial position i.e. $g = y_0$. In this case, no motion is generated by the classical DMP. Finally, in the third case (Fig. 1, bottom subplot), if the new target is chosen below the initial position, or more generally when $\text{sign}(g - y_0) = -\text{sign}(g_d - y_{0,d})$, the classical DMP produces a mirrored trajectory. Unlike the classical DMP, the proposed DMP^{++} retains the shape of the demonstrated trajectory in all cases, without exhibiting any side-effects.

4.2 Comparison with other DMP Variants

We further compare the proposed generalization against other SoA DMP variants that propose a different generalization. In particular, we compare against [12] (we will call it DMP-rot),
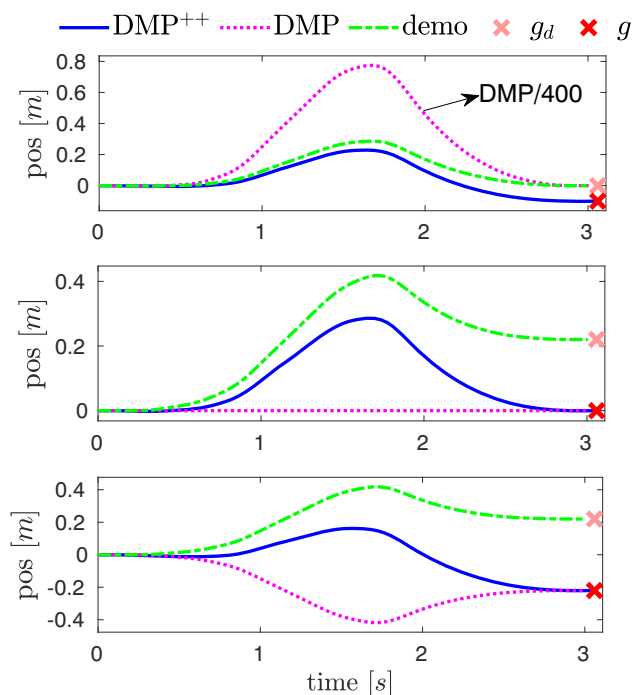

Fig. 1 Comparison with classical DMP generalization. Top: The demo start and goal positions are close. Middle: New goal set at the start position. Bottom: Mirroring

which proposes a different spatial scaling that remedies the 3 aforementioned drawbacks of classical DMP. We also compare against the bio-inspired DMP variant from [11] (we will call it DMP-bio$^+$), which remedies the scaling problem of the bio-inspired DMP [10] for new target positions that are not close to the demo target position.

Notice that both [12], that assumes the classical DMP formulation, and [11], that assumes the bio-inspired DMP formulation, use for spatial generalization the scaling matrix $\boldsymbol{K}_s = \boldsymbol{R} \frac{||g - y_0||}{||g_d - y_{d,0}||}$, where $\boldsymbol{R} \in SO(3)$ is such that $\frac{g - y_0}{||g - y_0||} = \boldsymbol{R} \frac{g_d - y_{d,0}}{||g_d - y_{d,0}||}$. It becomes obvious from such a $\boldsymbol{K}_s$, and in particular from $\frac{||g - y_0||}{||g_d - y_{d,0}||}$, that displacement in one axis can affect the scaling in all axes, which in turn can result in over-amplification of the amplitude of an axis even if there is no displacement there. This effect is more pronounced for trajectories where in a specific axis the amplitude of the demo (i.e. the distance between the min and max position) is bigger than the distance between the original and final position. Such trajectories are typical in many real practical scenarios like packing.

To highlight these issues, we consider for the comparisons such a trajectory, with a higher amplitude than the distance between the initial and final demo position and examine the spatial generalization for 4 different displacements of the new target g from g_d. The results are plotted in Fig. 2, where each column depicts the results of a target displacement, with the
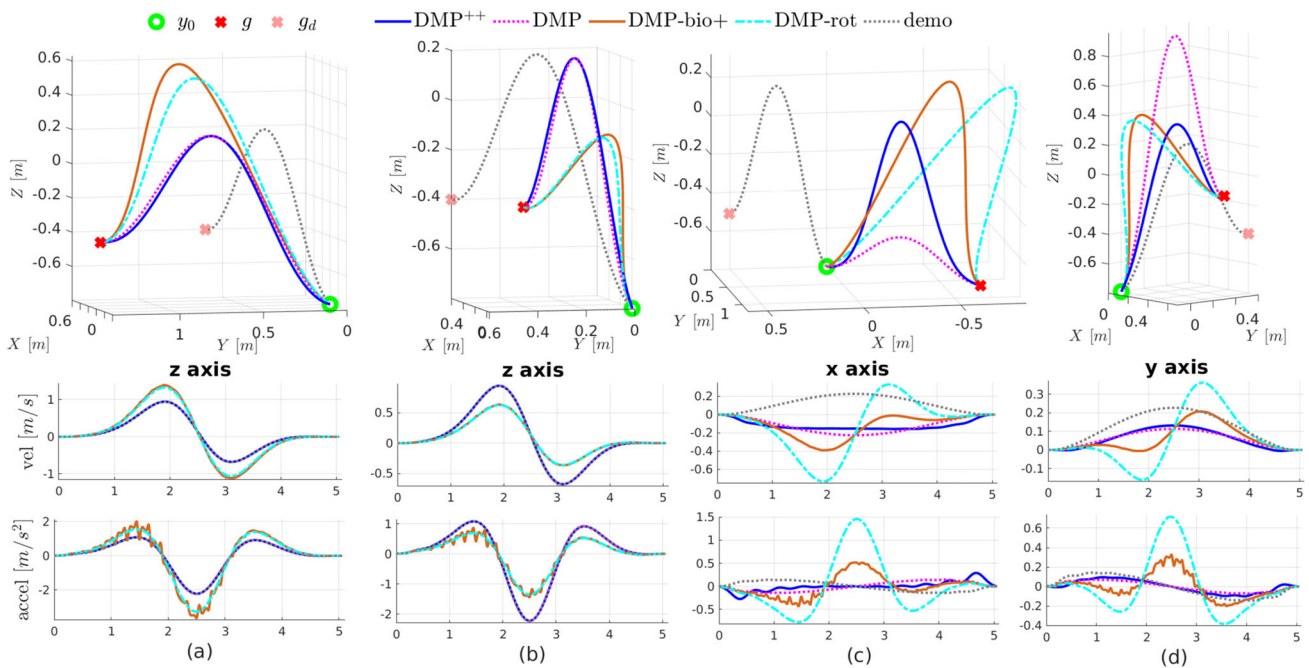
Fig. 2 Spatial generalization comparison between the proposed DMP^{++} and other DMP variants, for displacements of the new target g different from the demo target g_d. In each column the top subplot depicts the $3D$ paths and the two bottom subplots the corresponding velocities and accelerations for an indicative axis. (a)/(b): Same z and bigger/smaller displacement on the xy plane from y_0. (c)/(d) Displacement in all 3 axis, with lower/higher z

$3D$ paths of each DMP variant on top and the corresponding velocities and acceleration of an indicative axis at the bottom. In the first case (Fig. 2-(a)), we can observe that although both g and g_d have the same z, the bigger displacement of g from y_0 on the xy plane results in a large scaling for DMP-bio$^+$ and DMP-rot. This behaviour in practical pick and place tasks is arguably undesirable. For instance, when changing the xy position of a box where an object is to be placed, but the height of the box is the same, there is no good reason why the height of the DMP trajectory should alter. Moreover, this magnification of the amplitude generates much larger velocities and accelerations, as can be seen in the bottom subplots of Fig. 2-(a)[6]. In contrast, DMP^{++} does not alter the amplitude of the trajectory and exhibits a behaviour close to the classical DMP. Notice also that the velocities and acceleration of DMP^{++} and DMP coincide with the demo, as one would expect given that the height of the target is unaltered. Analogous conclusions can be drawn in Fig. 2-(b), where the height of the target is the same, but this time the displacement on the xy plane is smaller. Finally, in Fig. 2-(c),(d) the target is displaced in all axes, with lower or higher z respectively for g. Here, the scaling drawbacks of the classical DMP are clearly manifested. Notice also, that although DMP-rot and

DMP-bio$^+$ do not generate large scalings and the resulting path retains the demo shape, it is nevertheless rotated in such a way that such a behaviour would not be very natural in practical tasks (e.g. in a pick and place).

4.3 Generalization to Dynamic Via-Points

Here we test the spatial generalization in the presence of dynamic via-points. We consider a single 2D demonstration for placing an object inside a box. During simulation we consider a tighter box that is also higher than that of the demonstration, hence via-points are used to ensure the proper placement inside the box. A rectangular obstacle is also present in the scene, with via-points specified relative to it, so as to ensure its avoidance in a predictable manner. To emulate dynamic changes in the scene, which typical occur in real environments, the position of the box and the target are displaced at different time instances.

We compare DMP^{++} with VMP [14][7]. VMP are presented and tested in [14] only for static via-points. In order to conduct a comparison with dynamic via-points, we extend

---

[6] Note that DMP-bio$^+$ produces a jerky acceleration which is owning to the use of the mollifier kernel which have finite support (instead of Gaussian kernels) [11].

[7] Other methods like [11, 29] are able to adapt only to a new continuous trajectory segment and not via-points. Also, ProMP [24] and KMP [25] require multiple demonstrations and do not address the case of dynamic via-points.

VMP so as to account for dynamic via-points too. Specifically, a VMP is given by:

$$\boldsymbol{y}(s) = \boldsymbol{f}_p(s) + \boldsymbol{A}^T \boldsymbol{\phi}_2(s) \qquad (17)$$

where $\boldsymbol{f}_p(s)$ is given by Eq. 3, $\boldsymbol{A} \in \mathbb{R}^{6 \times n}$ contains in each column the coefficients of a 5th order polynomial for each DoF and $\boldsymbol{\phi}_2(s) = [1 \; s \; s^2 \; s^3 \; s^4 \; s^5]^T$. The weights $\boldsymbol{W}$ in $\boldsymbol{f}_p$ are fitted to the demo, while $\boldsymbol{A}$ is adapted according to the via-points, with the initial and target position treated as the first and final via-point. In particular, given the current phase value s, and the previous and next via-points $\boldsymbol{y}_v(s_1)$, $\boldsymbol{y}_v(s_2)$ with $s_1 \leq s < s_2$, $\boldsymbol{A}$ is determined by solving the system of equations:

$$\begin{bmatrix} \boldsymbol{\phi}_2(s_j)^T \\ \dot{\boldsymbol{\phi}}_2(s_j)^T \\ \ddot{\boldsymbol{\phi}}_2(s_j)^T \end{bmatrix} \boldsymbol{A} = \begin{bmatrix} (\boldsymbol{y}_v(s_j) - \boldsymbol{f}_p(s_j))^T \\ (\dot{\boldsymbol{y}}(s_j) - \dot{\boldsymbol{f}}_p(s_j))^T \\ (\ddot{\boldsymbol{y}}(s_j) - \ddot{\boldsymbol{f}}_p(s_j))^T \end{bmatrix}, \; j \in \{1, 2\}$$

Similar to [14], we set the velocity and acceleration at each via-point equal to that of the demonstration. Since via-points may alter on the fly, to ensure continuity we consider as the first via-point the current VMP point, i.e. we set $s_1 = s$ and $\boldsymbol{y}_v(s_1) = \boldsymbol{y}(s)$ (except for the initial via-point, which has position $\boldsymbol{y}_0$). If the via-points are static, this approach generates the same trajectory as in [14], since once $\boldsymbol{A}$ is adapted from a via-point $\boldsymbol{y}_v(s_1)$ to a next via-point $\boldsymbol{y}_v(s_2)$, then continuously updating $\boldsymbol{A}$ between $\boldsymbol{y}_v(s) = \boldsymbol{y}(s)$ and $\boldsymbol{y}_v(s_2)$, with $s \in (s_1, s_2)$ obviously yields the same $\boldsymbol{A}$.

The simulated scenario and the results with DMP^{++} are depicted in Fig. 3. At the top subplot, the initial target and obstacle are shown, with the predicted DMP trajectory (i.e. the one that would be produced) without the via-points plotted with dashed cyan line, and after the adaptation to the via-points with light blue dashed line. At $t = 1.8$ sec, the obstacle is abruptly displaced (Fig. 3, middle subplot), and the previous predicted DMP path and the new predicted DMP path, based on the updated via-points, are plotted again. The DMP path that has been executed so far is also shown with solid blue line. Finally, at $t = 3$ sec, the target is abruptly displaced (Fig. 3, bottom subplot). In all simulation snapshots, comparing the cyan with the light blue dashed line, it's obvious that without updating the DMP to the new via-points, either a collision with the obstacle would occur or failure to reach the target due to bumping at the box's boundaries.

We carried out the same simulation with VMP and plot the final results against the DMP^{++} in Fig. 4, where on the top subplot the 2D paths are drawn and on the bottom subplots the velocity and acceleration norms against the demonstrated ones. It can be observed from the velocities and accelerations that VMP generates more abrupt and higher velocities and accelerations, that differ significantly compared to the demo.
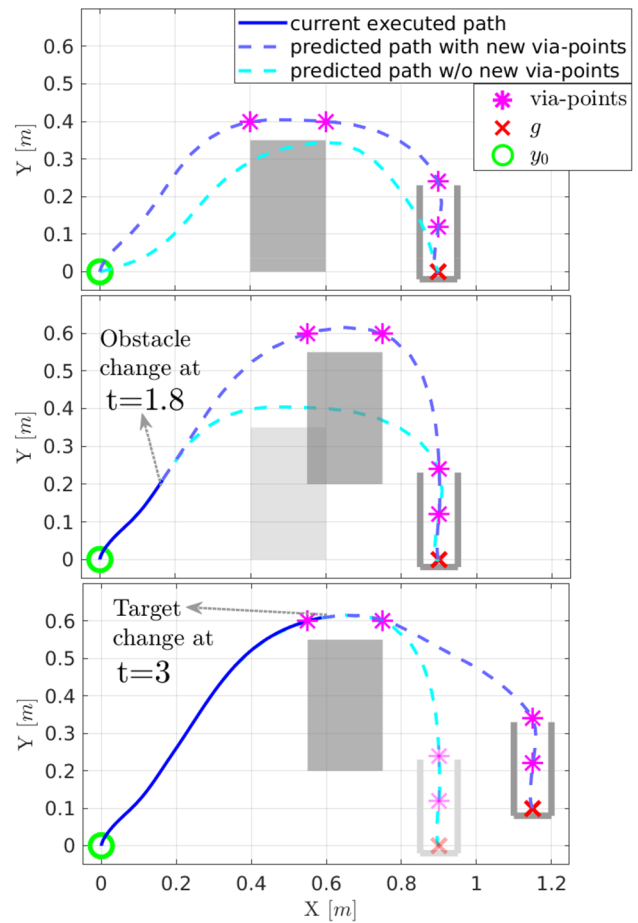

Fig. 3 DMP^{++} simulation with dynamic via-points. 1st subplot: Initial predicted DMP path (i.e. the one that would be produced) with and w/o the via-points. 2nd subplot: The obstacle is displaced and the current executed DMP path as well as the previous and updated predicted DMP path w/o and with the new via-points is shown. 3rd subplot: The target is displaced

This shortcoming is due to the fact that the VMP adaptation essentially overrides the demo with the only criterion being the satisfaction of the two currently active via-point constraints. In contrast, DMP^{++} performs a global optimization (see Eq. 10) that considers all via-points and adapts in a way as consistent as possible with the demo.

4.4 Adding Kinematic Inequality Constraints

Here we compare with the framework from [27] which uses the DMP formulation [26] with the classical DMP scaling to generate the generalized trajectory $\{\boldsymbol{y}_d, \dot{\boldsymbol{y}}_d, \ddot{\boldsymbol{y}}_d\}_i$ over a horizon of N future time-steps which is then optimized to respect kinematic limits like position velocity and acceleration bounds as well as passing from via-points. We show that by modifying the framework from [27] according to Fig. 5, where the DMP with the classical scaling is

Fig. 4 DMP^{++} vs VMP. Top plot: 2D paths. Bottom plots: velocity and acceleration L2 norms
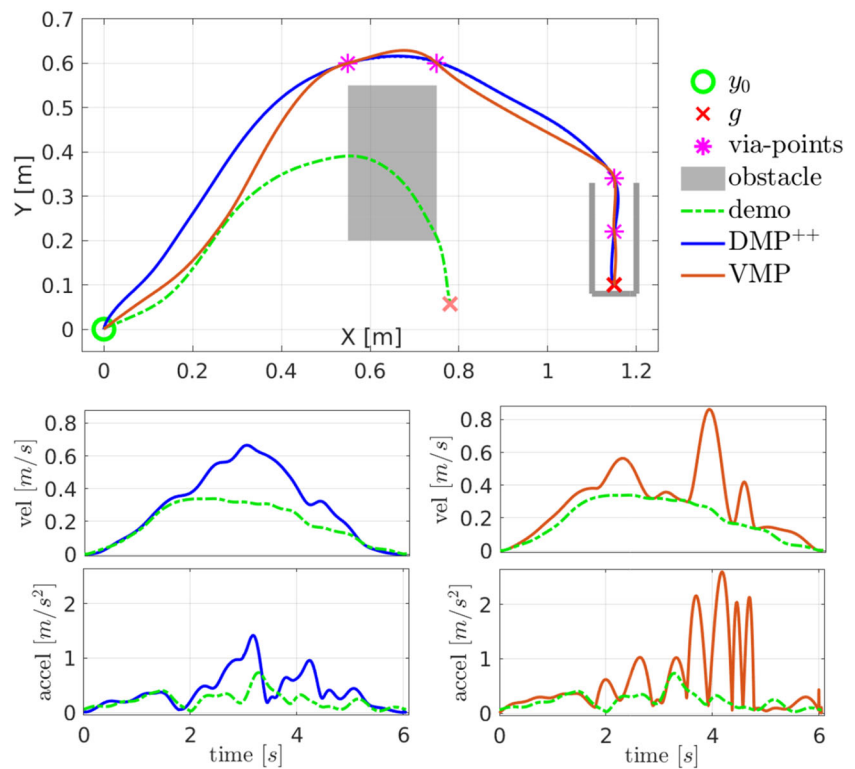
replaced by DMP^{++} we achieve more efficient generalization that accounts also for via-points. Notice that via-points are included both in DMP^{++} and in the optimization module of [27] (Fig. 5 brown rectangle) via the respective equality constraints so as to guarantee that the constrained within the kinematic inequality limits trajectory will also pass from the via-points.

Simulation results of a 1 DoF example with two via points and position, velocity and acceleration limits are shown in Fig. 6, where the method proposed in [27] is used to optimize the velocity profile with all relevant parameters chosen as in [27]. To endow the optimizer with greater flexibility in finding feasible solutions we consider the kinematic limits as hard limits (grey dashed lines) and introduce the lower and upper soft limits (magenta dashed lines) within the hard limits. We want to preferably operate within the soft limits and only exceed them if feasibility would be inevitable otherwise [27]. It can be observed that the DMP^{++} (blue line) retains the shape of the demonstration (green dash dotted line), even in the presence of kinematic limits and via-points (red asterisks). On the other hand, the DMP with the classical scaling (mustard dotted line) can induce large scalings that generate velocities and accelerations that violate considerably the limits, leading to saturation (see the 2nd and 3rd subplots of Fig. 6) and consequently the distortion of the demonstrated shape, as can be observed comparing the mustard with the green trajectory in the first subplot. This distortion would be even more adverse for cases in which the scaling is problematic like those presented in Fig. 1, where a feasible solution may even not be found.

Fig. 5 Combination of the proposed spatial generalization with [27] to further impose kinematic inequality constraints. The image is taken from [27], where the differences/additions are highlighted with magenta. DMP^{++} replaces the DMP with the classical spatial scaling. Via-points are also considered during the generation of the generalized (unconstrained) trajectory $\boldsymbol{y}_d$
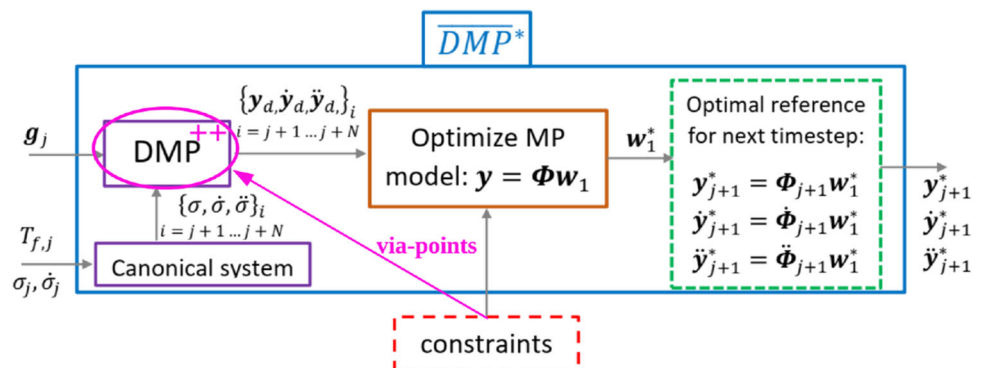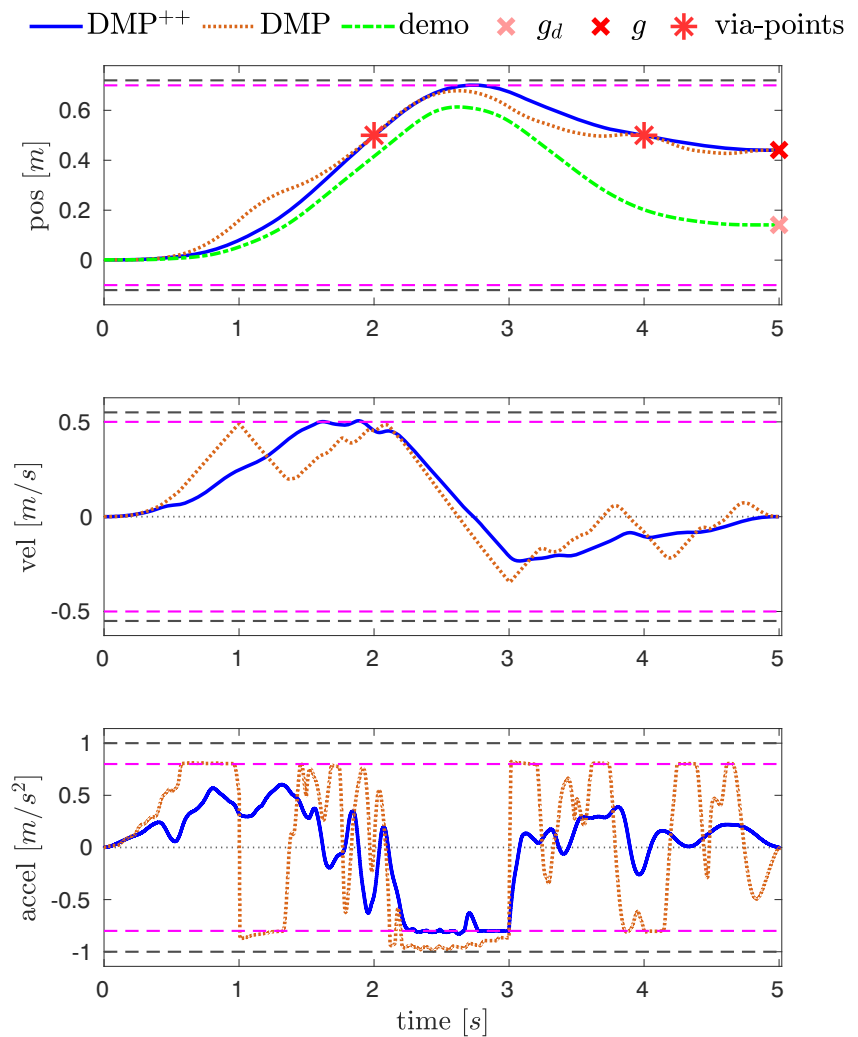
Fig. 6 Comparison of the classical DMP spatial generalization against the proposed DMP^{++} in combination with [27] to enforce kinematic inequality constraints

5 Experimental Validation

In this section, we further validate and demonstrate experimentally the efficiency of DMP^{++} in practical pick and place tasks involved in packing scenarios. In particular, we showcase our method's spatial generalization properties in a dynamic environment and incorporation of dynamic via-points.

5.1 Packing Scenario

The robot has to pick carton products and place them inside boxes, where the size of the product and associated packing box can vary. A single demonstration is provided with a small carton ($5 \times 12 \times 5$ cm) and a short box (10 cm height) (Fig. 7a). This demonstration is encoded in a DMP. Then we use this DMP to execute three scenarios: 1) packing the last carton product used in the demo in a dynamically changed box pose, 2) packing a larger carton product ($12 \times 12 \times 20$ cm) to its associated box (28 cm height) and 3) the latter scenario

in the presence of dynamic obstacles. In all three scenarios, depending on the height of the carton and box, via-points are specified above the target pose to ensure a proper insertion of the carton in the box. Moreover, in all cases the box's target pose is altered on the fly by a human, thus the associated target via-points change also dynamically. In the 3rd scenario, via-points are specified w.r.t. the obstacle to avoid it in a predictable way. As the obstacle is introduced dynamically, the associated via-points also change dynamically.

5.2 Technical Details

To carry out these experiments a velocity controlled ur5e robot is used, with 2 ms control cycle, which takes as reference the velocity produced by the DMP transformation system from Eq. 1, where y is either the Cartesian position or for orientation $y = \log(Q * \bar{Q}_0)$. The DMP weights are updated online according to Eqs. 11, 13, while ϵ is set as in the simulations. The current state constraint in the optimization is set equal to the last state generated by DMP^{++}. The

(a) Demonstration: Placing a carton inside a box.

(b) Scenario 1: small carton/box

(c) Scenario 2: large carton/box

(d) Scenario 2: large carton/box (NO via-points)

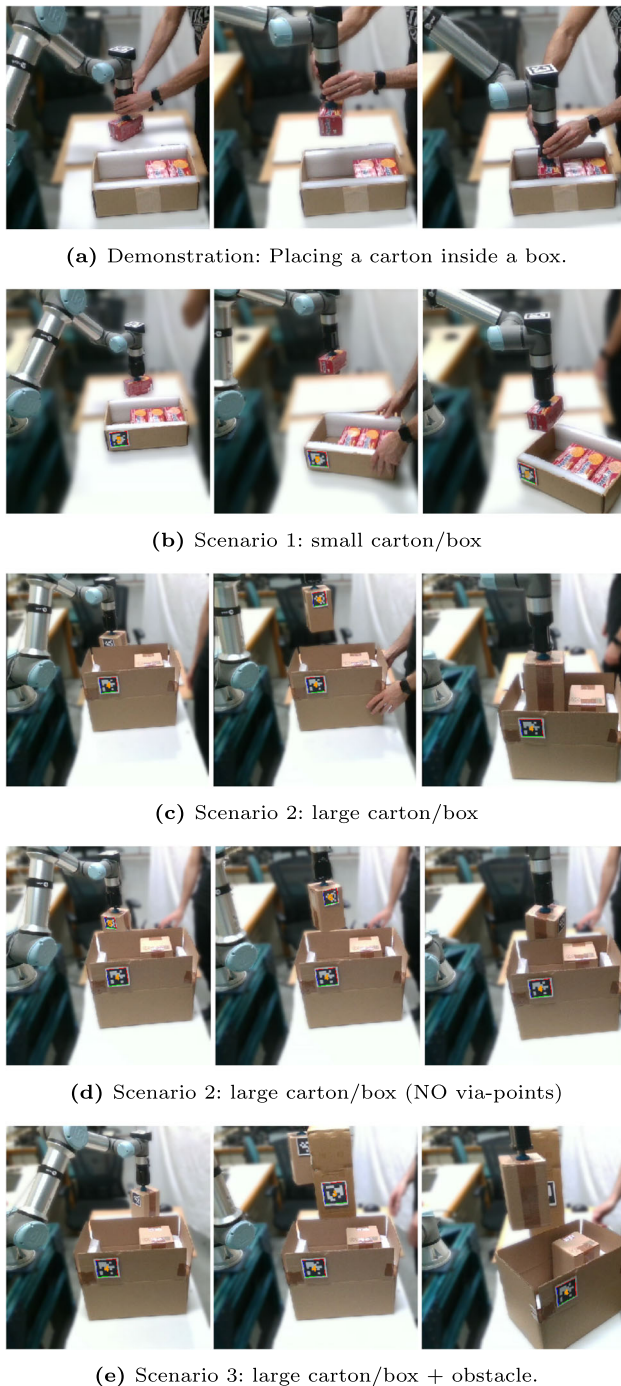**(e)** Scenario 3: large carton/box + obstacle.

Fig. 7 Demonstration and execution snapshots

code was implemented in c++ and run on a desktop pc with an Intel® Core™ i7-9700 processor. In all cases, the optimization at each control cycle was below 0.8 ms, which is well within the 2ms control cycle of the robot. For the canonical system from Eq. 5, we use $\dot{s}_d = 1/T_f$, with $T_f = 10$ sec.

The pose of all object's that are employed in the experiments are defined by apriltags which are tracked using a realsense2 camera at 30 Hz. In general the target pose where the carton should be placed in the box can be provided by a perception system that determines the packing plan given the current box packing/occupancy state. For simplicity, we predefine the target placing pose relative to the box's pose. For the short box we employ 2 via-points placed every 5 cm above the target pose and for the taller box 5 via-points placed every 6 cm above the target. These via-points have the same orientation as the target pose. For the obstacle a higher level perception system could be employed for generating via-points to circumvent the obstacle. Developing such a system is beyond the scope of this work, therefore we consider for simplicity two predefined via-points relative to the obstacle's center pose on the xy plane. The z position and orientation at each via-point is set equal to the DMP^{++} reference at the corresponding phase variable instance. To determine whether these via-points should be incorporated in the DMP, we consider an augmented bounding box around the obstacle, equal to the size of the obstacle + the half size of the carton. At each control cycle, we sample the DMP reference trajectory at 30 future points and check if any of these points are inside the augmented box in order to use the obstacle's via-points in the DMP.

5.3 Results

For the first scenario (small carton, short box), snapshots of the execution are shown in Fig. 7b, while the results are plotted in Fig. 8. In particular, in Fig. 8a, the oriented path executed by the robot is shown, where also the initial and final box, target pose and associated via-points are visualized (the initial ones with opaque colors and the final ones with more vivid colors). The target change is also indicated by a light red line and we further plot the demonstrated path (cyan dotted line) for comparison. In Fig. 8b the translational and rotational velocity norm profile for the execution (blue line) and the demo (green line) are plotted, as well as the target displacement dynamics (magenta line). The demo velocity profile is scaled temporarily to the execution time duration, by multiplying the velocity with $T_{f,d}/T_f$, where $T_{f,d} = 13.5$ sec is the demo duration. Moreover, the distance between the initial and target demo poses are $\left\|p_{g,d} - p_{0,d}\right\| = 0.39$ m for the position and $\left\|\log(Q_{g,d} * Q_{0,d})\right\| = 48$ degrees for the orientation, while during execution the corresponding initial displacements were larger with values $\left\|p_g(0) - p_0\right\| = 0.45$ m and $\left\|\log(Q_g(0) * Q_0)\right\| = 67$ degrees. These larger displacements during execution cause the velocity profile to scale up initially (for $t \in [0\ 3]$ sec) compared to the demo. As the target position is further displaced after $t = 3$ sec, the velocity during execution further scales up compared to the demo. Nevertheless, the scaled execution norm velocity profile resembles the demonstrated one as expected. Some discrepancies are expected due to the target change and the
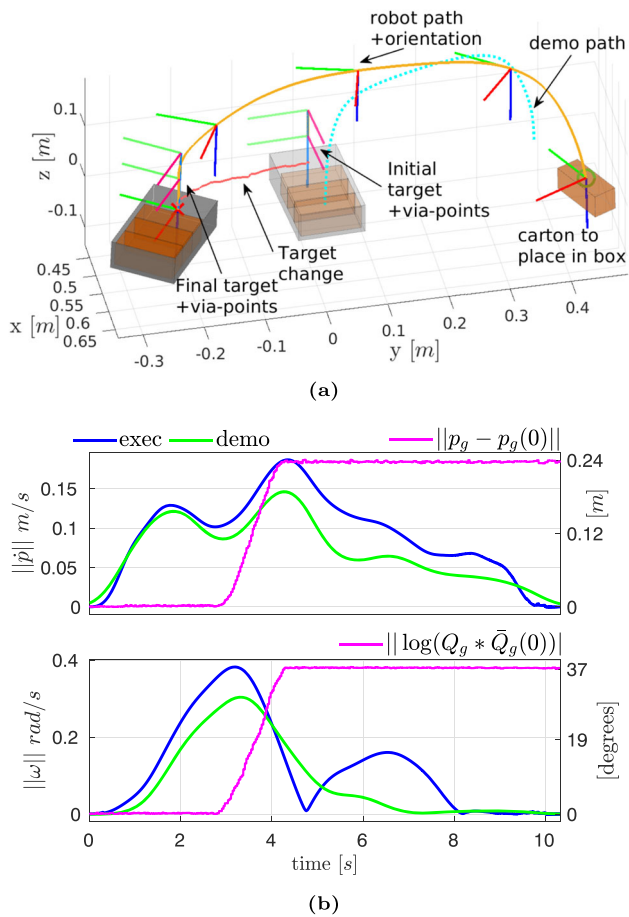
(a)

(b)

Fig. 8 Experiment 1: Place small carton in short box. The box is displaced online. (a): Robot oriented Cartesian path and visualization of the carton, box and the associated target and via-points. (b): Velocity and target change
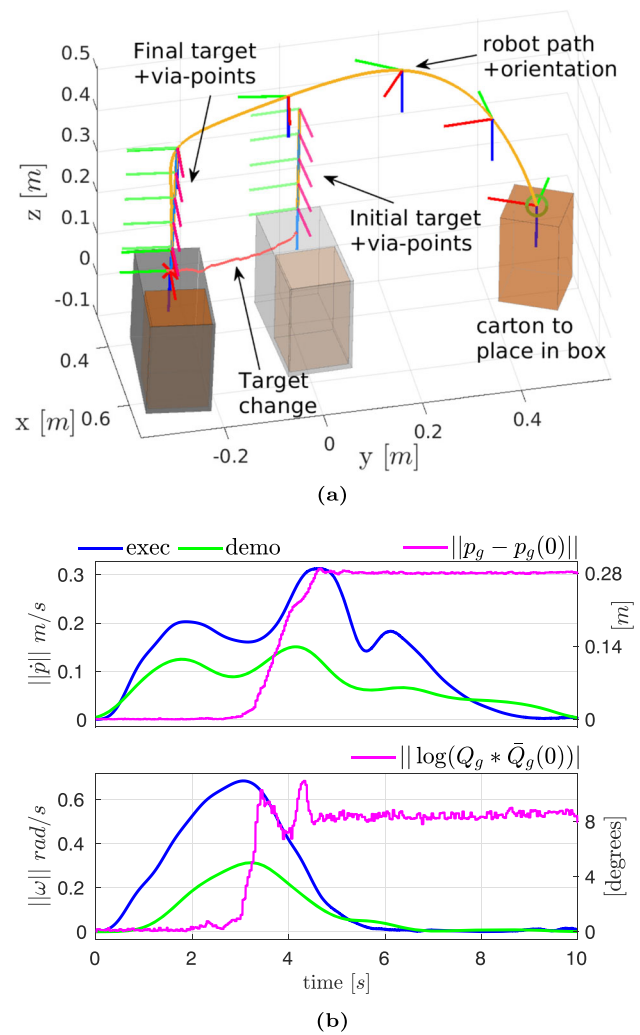

(a)

(b)

Fig. 9 Experiment 2: Place large carton in tall box. The box is displaced online. (a): Robot oriented Cartesian path and visualization of the carton, box and the associated target and via-points. (b): Velocity and target change

continuity and via-point constraints. Observe also from the demo orientation velocity profile (Fig. 8b bottom) that the orientation settles before the final target position is reached. This is also the case in the execution, where however, due to the orientation displacement by a total of 37 degrees (from $t = 3$ until $t = 4.2$ sec), the velocity profile is readjusted to reach the new target pose. Notice also that despite the target displacement which is a bit noisy and updated at a lower rate (33 ms compared to the 2 ms control cycle), the generated velocity profile is smooth owing to the current state constraint in Eq. 10e.

For the second scenario (large carton, tall box), snapshots of the execution are shown in Fig. 7c, while the results are plotted in Fig. 9, with the Cartesian oriented path and the relevant via-points in Fig. 9a and the translational and rotational velocity profile as well as the target change in Fig. 9b. Despite the larger shape of the carton and the box, the via-points contribute in the successful execution. Notice that without the via-points, the execution would fail, as shown in the execution snapshots in Fig. 7d. Training a new DMP with a new

demonstration for the larger boxes could solve this issue. However this is time-consuming and impractical, as for every different object shape, a different DMP would be required. Instead, one can employ a single DMP, which already encapsulates the general desired motion shape-pattern, and adjust-scale it according to the manipulated object's size using via-points in the DMP spatial generalization. Notice also that using two separate DMPs, one for reaching the box from the top and the second for inserting the carton would increase the system's complexity with an additional DMP. Moreover, as also discussed in the introduction, in more general and dynamic scenarios, using via-points is more flexible than keeping track of multiple DMPs to connect the via-points and also making additional modifications to ensure a smooth transition between the DMPs. In our case, passing through the via-points and the continuity and smoothness of

the motion is handled automatically through the constraints in Eq. 10, which further makes no prior assumption regarding the number of via-points, while also accommodating the case where they change dynamically.

Finally, in the last scenario shown in Fig. 7e, we see yet another case of incorporating dynamically via-points in the DMP generalization, in the context of obstacle avoidance. Via-points can prove useful in such scenarios to steer the robot along a route that is more ergonomic and clutter-free. This can also be combined with the use of artificial potential functions for volumetric obstacle avoidance [30] to ensure that no collisions occur. Since this is not the main focus in this work and to keep things simple we define the via-points at a distance from the obstacle, considering the dimensions of the obstacle and the carton. The results are plotted in Fig. 10 where the final obstacle pose is plotted with magenta, along with its augmented bounds, shown in light red. The obstacle via-points, from which the robot passes to circumvent the obstacle, are depicted with cyan asterisks. Notice that the DMP trajectory can in general pass through the augmented obstacle bounds. They are not a forbidden region. Instead, they are used here as a rough estimate to detect when to include the obstacle via-points, from which if the DMP passes, it should avoid the obstacle.

A video with all the experimental scenarios along with explanations and visualizations can be found at https://youtu.be/1Nlzv_yjPwU

6 Discussion

The conducted simulations and experiments validate the capability of the proposed method to incorporate dynamic via-points in DMP and improve its spatial generalization. In particular, the classical DMP [1] and subsequent variations to improve its spatial generalization [11, 12] have been shown to produce problematic spatial generalization. This is attributed to the fact that these methods scale the demonstrated trajectory based on the ratio $\frac{g - y_0}{g_d - y_{d,0}}$, i.e. the distance between the new initial and target position, over the demonstrated distance. This fact alone makes these methods sensitive to the initial and final values of the demo and the new ones, and therefore prone to overscaling. In contrast, the proposed method does not suffer from such over-scaling, as it tries to minimize the distance from the demonstrated acceleration, which encapsulates the shape of the demo. Therefore, the proposed method retains the shape of the demo while ensuring scaling of the generated trajectory to the new target. Additionally, the proposed novel generalization as it was shown, can be combined with our previous work [27] to impose kinematic inequality constraints in order to generate feasible trajectories in the presence of kinematic bounds related to the robot.
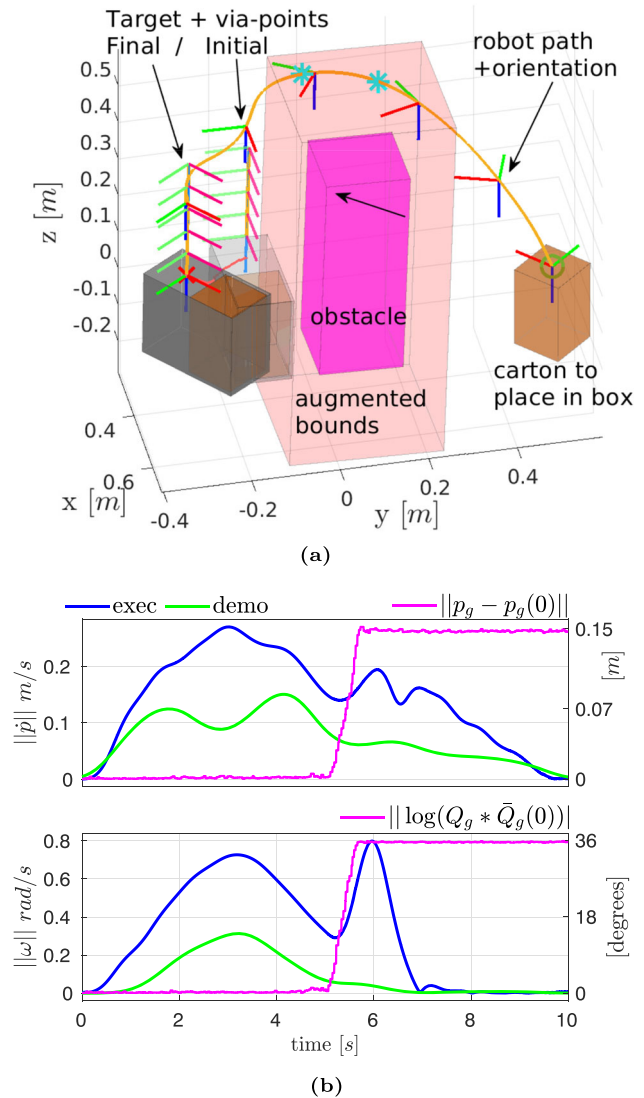

(a)

(b)

Fig. 10 Experiment 3: Place large carton in tall box. The box is displaced online and a dynamic obstacle is introduced. (a): Robot oriented Cartesian path and visualization of the carton, box, obstacle and the associated target and obstacle-avoidance via-points. (b): Velocity and target change

Moreover, previous DMP methods were unable to incorporate via-points, let alone dynamic ones. Other DMP-like methods, like VMP, have only considered static via-points. With the incorporation of dynamic via-points, the proposed method enriches the spatial generalization capabilities of DMP, allowing them to account for the geometry/shape of the involved objects, as in the packing scenario, presented in Section 5.

Another example application that was presented, was the use of via-points to dictate how the DMP should shape its trajectory in the presence of other objects, like obstacles. Via-points can prove useful in such scenarios to steer the robot along a route that is more ergonomic and clutter-free.

This can also be combined with the use of artificial potential functions for volumetric obstacle avoidance [30] to ensure that no collisions occur. Another scenario, discussed in the Introduction, where via-points could prove expedient is to use them so as to push aside other objects that obstruct reaching pathways to a desired target [15].

Finally, it should be noted that in this work, via-points relative to each object were set in advance and tracked during execution via apriltags. While this is a viable solution for scenarios where the shape and type (e.g. obstacle or not) of the objects that can be involved are known apriori, in more general scenarios, a higher level perception system would be required to classify the objects in the scene and extract appropriate key-points (via-points) relative to each one.

7 Conclusions

In this work we presented an improved spatial generalization for DMP that enables the incorporation of dynamic via-points by proposing an on-line adaptation scheme for the DMP weights that minimizes the distance from the learned demonstrated acceleration profile. Thus, the demonstrated motion pattern is retained and dynamic via-points and initial/final state constraints are satisfied. Comparative simulations with other SoA methods showcase the advantages of the proposed method. Further validation is carried out via experimental packing scenarios in a dynamic environment where the incorporation of via points enables the use of a single demonstration for executing packing even when obstacles are introduced in the scene and objects and boxes of different sizes are used, whose pose can even change dynamically.

Appendix A: Unit Quaternion Preliminaries

Given a rotation matrix $\boldsymbol{R} \in SO(3)$, an orientation can be expressed in terms of the unit quaternion $\boldsymbol{Q} \in \mathbb{S}^3$ as $\boldsymbol{Q} = [w \ \boldsymbol{v}^T]^T = [\cos(\theta_2) \ \sin(\theta_2)\boldsymbol{k}^T]^T$, where $\boldsymbol{k} \in \mathbb{S}^2$, $\theta_2 = \theta/2$ with $\theta \in [-\pi, \pi)$ are the equivalent unit axis - angle representation. The quaternion product between the unit quaternions $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$ is denoted as $\boldsymbol{Q}_1 * \boldsymbol{Q}_2$. The inverse of a unit quaternion is equal to its conjugate which is $\boldsymbol{Q}^{-1} = \bar{\boldsymbol{Q}} = [w \ -\boldsymbol{v}^T]^T$. The logarithmic $\boldsymbol{\eta} = \log(\boldsymbol{Q})$ and exponential $\boldsymbol{Q} = \exp(\boldsymbol{\eta})$ mappings $\log : \mathbb{S}^3 \to \mathbb{R}^3$, $\exp : \mathbb{R}^3 \to \mathbb{S}^3$ respect the manifold's geometry and are defined as follows:

$$\log(\boldsymbol{Q}) \triangleq \begin{cases} 2\cos^{-1}(w)\frac{\boldsymbol{v}}{||\boldsymbol{v}||}, & |w| \neq 1 \\ [0, 0, 0]^T, & \text{otherwise} \end{cases} \tag{18}$$

$$\exp(\boldsymbol{\eta}) \triangleq \begin{cases} [\cos(||\boldsymbol{\eta}/2||), \sin(||\boldsymbol{\eta}/2||)\frac{\boldsymbol{\eta}^T}{||\boldsymbol{\eta}||}]^T, & ||\boldsymbol{\eta}|| \neq 0 \\ [1, 0, 0, 0]^T, & \text{otherwise} \end{cases} \tag{19}$$

It can be found analytically that the relations between the rotational velocity and the derivative of the quaternion logarithm are:

$$\boldsymbol{\omega} = \boldsymbol{J}_\eta \dot{\boldsymbol{\eta}} \tag{20}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_\eta^\dagger \boldsymbol{\omega} \tag{21}$$

where

$$\boldsymbol{J}_\eta \triangleq \boldsymbol{k}\boldsymbol{k}^T + \frac{\sin(\theta_2)\cos(\theta_2)}{\theta_2}(\boldsymbol{I}_3 - \boldsymbol{k}\boldsymbol{k}^T) + \frac{\sin^2(\theta_2)}{\theta_2}[\boldsymbol{k}]_\times \tag{22}$$

$$\boldsymbol{J}_\eta^\dagger = \boldsymbol{k}\boldsymbol{k}^T + \frac{\theta_2\cos(\theta_2)}{\sin(\theta_2)}(\boldsymbol{I}_3 - \boldsymbol{k}\boldsymbol{k}^T) - \theta_2[\boldsymbol{k}]_\times \tag{23}$$

where $[k]_\times$ denotes the skew-symmetric matric of $\boldsymbol{k}$. For $\theta = 0$ it can be easily verified that taking the limit of Eqs. 22, 23 and using L'Hospital's rule we get $\boldsymbol{J}_\eta = \boldsymbol{J}_\eta^\dagger = \boldsymbol{I}_3$.

Differentiating Eqs. 20, 21 we can obtain the relations between the quaternion logarithm second time derivative and the rotational acceleration:

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}_\eta \ddot{\boldsymbol{\eta}} + \dot{\boldsymbol{J}}_\eta \dot{\boldsymbol{\eta}} \tag{24}$$

$$\ddot{\boldsymbol{\eta}} = \boldsymbol{J}_\eta^\dagger \dot{\boldsymbol{\omega}} + \dot{\boldsymbol{J}}_\eta^\dagger \boldsymbol{\omega} \tag{25}$$

where

$$\dot{\boldsymbol{J}}_\eta = \left(1 - \frac{\sin(\theta_2)\cos(\theta_2)}{\theta_2}\right)(\dot{\boldsymbol{k}}\boldsymbol{k}^T + \boldsymbol{k}\dot{\boldsymbol{k}}^T) + \frac{\sin^2(\theta_2)}{\theta_2}[\dot{\boldsymbol{k}}]_\times$$
$$+ \left(\frac{1 - 2\sin^2(\theta_2)}{\theta_2} - \frac{\sin(\theta_2)\cos(\theta_2)}{\theta_2^2}\right)\dot{\theta}_2(\boldsymbol{I}_3 - \boldsymbol{k}\boldsymbol{k}^T)$$
$$+ \left(\frac{2\sin(\theta_2)\cos(\theta_2)}{\theta_2} - \frac{\sin^2(\theta)}{\theta_2^2}\right)\dot{\theta}_2[\boldsymbol{k}]_\times \tag{26}$$

$$\dot{\boldsymbol{J}}_\eta^\dagger = \left(1 - \frac{\theta_2\cos(\theta_2)}{\sin(\theta_2)}\right)(\dot{\boldsymbol{k}}\boldsymbol{k}^T + \boldsymbol{k}\dot{\boldsymbol{k}}^T) - \dot{\theta}_2[\boldsymbol{k}]_\times - \theta_2[\dot{\boldsymbol{k}}]_\times$$
$$+ \left(\frac{\sin(\theta_2)\cos(\theta_2) - \theta_2}{\sin^2(\theta_2)}\right)\dot{\theta}_2(\boldsymbol{I}_3 - \boldsymbol{k}\boldsymbol{k}^T) \tag{27}$$

with $\dot{\theta}_2 = \frac{1}{2}\boldsymbol{k}^T\dot{\boldsymbol{\eta}}$ and $\dot{\boldsymbol{k}} = \frac{1}{2}\left(\frac{1 - \boldsymbol{k}\boldsymbol{k}^T}{\theta_2}\dot{\boldsymbol{\eta}}\right)$. Taking the limit of Eqs. 26, 27 for $\theta = 0$ we have that $\dot{\boldsymbol{J}}_\eta = \dot{\boldsymbol{J}}_\eta^\dagger = \boldsymbol{0}$.

Finally, the relations between the Cartesian torque $\boldsymbol{\tau}$ and its transformation in the quaternion logarithm space $\boldsymbol{\tau}_\eta$ is given by:

$$\boldsymbol{\tau}_\eta = \boldsymbol{J}_\eta^T \boldsymbol{\tau} \tag{28}$$

$$\boldsymbol{\tau} = (\boldsymbol{J}_\eta^\dagger)^T \boldsymbol{\tau}_\eta \tag{29}$$

These mappings follow readily from the preservation of power, i.e. it should hold that $\boldsymbol{\omega}^T \boldsymbol{\tau} = \dot{\boldsymbol{\eta}}^T \boldsymbol{\tau}_\eta$.

Appendix B: Recursive Least Squares derivations

Here we provide some useful results in the form of theorems which facilitate the proof of the update formulas Eqs. 11, 13 that solve Eq. 10. Results in Section 2 from [31] for vectors are here extended for matrices in Theorems 1, 2, and 5. Theorems 1, 2 are used in the proof of Theorems 3 - 5, and Theorems 2 - 5 are employed in the proof for solving Eq. 10.

Theorem 1 *The solution to the problem:*

$$min_W tr\{(Y - W^T \Phi)^T R^{-1}(Y - W^T \Phi)\} \tag{30}$$

with $Y \in \mathbb{R}^{n \times m}$, $W \in \mathbb{R}^{k \times n}$, $\Phi \in \mathbb{R}^{k \times m}$, $rank(\Phi) = k$, $R \in \mathcal{S}_{++}^n$ is given by:

$$W_0 = P_0 \Phi R^{-1} Y^T \tag{31}$$

$$P_0 = (\Phi R^{-1} \Phi^T)^{-1} \tag{32}$$

where $P_0 > 0$.

Proof Taking the derivative of f_0 w.r.t. W and solving for W, it is straightforward to verify that the solution is indeed given by W_0, where $P > 0$ since $R > 0$ and $rank(\Phi) = k$. □

Theorem 2 *The solution to the problem:*

$$min_W f_0(W) + tr\{(Z - W^T H)^T R_1^{-1}(Z - W^T H)\} \tag{33}$$

with $f_0(W) = tr\{(Y - W^T \Phi)^T R^{-1}(Y - W^T \Phi)\}$, $Y \in \mathbb{R}^{n \times m}$, $W \in \mathbb{R}^{k \times n}$, $\Phi \in \mathbb{R}^{k \times m}$, $rank(\Phi) = k$, $R, R_1 \in \mathcal{S}_{++}^n$, $Z \in \mathbb{R}^{n \times l}$, $H \in \mathbb{R}^{k \times l}$, can be obtained from the solution W_0, P_0 of $min_W f_0(W)$ as follows:

$$W_1 = W_0 + P_0 H(R_1 + H^T P_0 H)^{-1}(Z - W_0^T H)^T \tag{34}$$

$$P_1 = P_0 - P_0 H(R_1 + H^T P_0 H)^{-1} H^T P_0 \tag{35}$$

$$P_1^{-1} = P_0 + H R_1^{-1} H^T \tag{36}$$

where $P_1 > 0$.

Proof We can rewritte the cost function in Eq. 33 as

$$tr\{(\bar{Y} - W^T \bar{\Phi})^T \bar{R}^{-1}(\bar{Y} - W^T \bar{\Phi})\}$$

with

$$\bar{Y} = [Y^T \ Z^T]^T, \ \bar{\Phi} = [\Phi^T \ H^T]^T, \ \bar{R} = \text{blkdiag}(R, R_1)$$

and apply the result of Theorem 1 to get:

$$W = P_1^{-1} \bar{\Phi} \bar{R}^{-1} \bar{Y}^T = P_1^{-1}(\Phi R^{-1} Y^T + H R_1^{-1} Z^T) \tag{37}$$

where $P_1^{-1} = \bar{\Phi} \bar{R}^{-1} \bar{\Phi}^T = (P_0^{-1} + H R_1^{-1} H^T)$ with $P_0^{-1} = \Phi R^{-1} \Phi^T$. It follows that $P_1^{-1} > 0$, since $P_0^{-1} > 0$, due to $R > 0$ and $rank(\Phi) = k$, and $H R_1^{-1} H^T \geq 0$. Applying the matrix inversion lemma it follows that P_1 is indeed given by Eq. 35 and substituting it in Eq. 37 we get:

$$W = W_0 - P_0 H(R_1 + H^T P_0 H)^{-1}(W_0^T H)^T + (P_0 - P_0 H(R_1 + H^T P_0 H)^{-1} H^T P_0) H R_1^{-1} Z^T \tag{38}$$

where $W_0 = P_0 \Phi R^{-1} Y^T$, $P_0 = (\Phi R^{-1} \Phi^T)^{-1}$ is indeed the solution of $min_W f_0(W)$ based on Theorem 1. We can further process the last term in Eq. 38, i.e.:

$$(P_0 - P_0 H(R_1 + H^T P_0 H)^{-1} H^T P_0) H R_1^{-1} Z^T$$
$$= (P_0 H - P_0 H(R_1 + H^T P_0 H)^{-1} H^T P_0 H) R_1^{-1} Z^T$$
$$= P_0 H(I_k - (R_1 + H^T P_0 H)^{-1}(\pm R_1 + H^T P_0 H)) R_1^{-1} Z^T$$
$$= P_0 H(R_1 + H^T P_0 H)^{-1} Z^T \tag{39}$$

and substituting it back to Eq. 38 we arrive at the solution given by Eq. 34. □

Theorem 3 *Problem Eq. 33 is equivalent to the problem*

$$min_W tr\{(W - W_0) P_0^{-1}(W - W_0)^T\} + \tag{40}$$
$$tr\{(Z - W^T H)^T R_1^{-1}(Z - W^T H)\} \tag{41}$$

where W_0, P_0 are the solution to $min_W f_0(W)$.

Proof Taking the gradient and solving w.r.t. W we can readily obtain Eq. 37 after which the same analysis as in the proof of Theorem 2 follows. □

Theorem 4 *The optimization problem*

$$min_W f_0(W) \tag{42}$$
$$s.t. \ W^T H = Z$$

with $f_0(W) = tr\{(Y - W^T \Phi)^T R^{-1}(Y - W^T \Phi)\}$, $Y \in \mathbb{R}^{n \times m}$, $W \in \mathbb{R}^{k \times n}$, $\Phi \in \mathbb{R}^{k \times m}$, $rank(\Phi) = k$, $R \in \mathcal{S}_{++}^n$, $Z \in \mathbb{R}^{n \times l}$, $H \in \mathbb{R}^{k \times l}$, $rank(H) = l \leq k$, is equivalent to the problem:

$$min_W f_0(W) + tr\{(Z - W^T H)^T R_\epsilon^{-1}(Z - W^T H)\} \tag{43}$$

for $R_\epsilon \to 0^+$.

Proof To find the solution of Eq. 42 we introduce the Language multipliers $V \in \mathbb{R}^{n \times l}$ and form the Lagrangian

$L(W, V) = f_0(W) + 2\text{tr}\{V^T(W^T H - Z)\}$. The KKT conditions are:

$$\frac{\partial L}{\partial W} = -\Phi(Y - W^T \Phi)^T + HV = 0 \tag{44}$$

$$\frac{\partial L}{\partial V} = H^T W - Z^T = 0 \tag{45}$$

Solving Eq. 44 for W we get:

$$W = W_0 - P_0 HV \tag{46}$$

where $W_0 = P_0 \Phi Y^T$, $P_0 = (\Phi R^{-1} \Phi^T)^{-1}$ which is indeed invertible since $\text{rank}(\Phi) = k$. Since P_0 is invertible and $\text{rank}(H) = l$, i.e. has full column rank, we can substitute Eq. 46 in Eq. 45 to solve for V:

$$V = -(H^T P_0 H)^{-1}(Z - W_0^T H)^T$$

Substituting the last equation in Eq. 46, we obtain the solution:

$$W_1 = W_0 + P_0 H(H^T P_0 H)^{-1}(Z - W_0^T H)^T \tag{47}$$

Turning our attention to problem Eq. 43 we can apply the result of Theorem 2 and notice that for $R_1 = R_\epsilon \to 0^+$ we get indeed the same solution as in Eq. 47. □

Theorem 5 *Given the solution* W_1, P_1 *of the problem*

$$min_W f_0(W) + tr\{(Z - W^T H)^T R_1^{-1}(Z - W^T H)\} \tag{48}$$

with $f_0(W) = tr\{(Y - W^T \Phi)^T R^{-1}(Y - W^T \Phi)\}$, $Y \in \mathbb{R}^{n \times m}$, $W \in \mathbb{R}^{k \times n}$, $\Phi \in \mathbb{R}^{k \times m}$, $rank(\Phi) = k$, R, $R_1 \in \mathcal{S}_{++}^n$, $Z \in \mathbb{R}^{n \times l}$, $H \in \mathbb{R}^{k \times l}$, *the solution to the problem:*

$$min_W f_0(W) \tag{49}$$

can be calculated as:

$$\hat{W}_0 = W_1 + P_1 H(-R_1 + H^T P_1 H)^{-1}(Z - W_1^T H)^T \tag{50}$$

$$\hat{P}_0 = P_1 - P_1 H(-R_1 + H^T P_1 H)^{-1} H^T P_1 \tag{51}$$

$$\hat{P}_0^{-1} = P_1^{-1} - H R_1^{-1} H^T \tag{52}$$

where $\hat{P}_0 > 0$.

Proof Denoting

$$f_1(W) = f_0(W) + \text{tr}\{(Z - W^T H)^T R_1^{-1}(Z - W^T H)\}$$

it follows that

$$f_0(W) = f_1(W) - \text{tr}\{(Z - W^T H)^T R_1^{-1}(Z - W^T H)\}$$

We can rewrite $f_1(W)$ as:

$$f_1(W) = \text{tr}\{(\bar{Y} - W^T \bar{\Phi})^T \bar{R}^{-1}(\bar{Y} - W^T \bar{\Phi})\}$$

with

$$\bar{Y} = [Y^T \ Z^T]^T, \ \bar{\Phi} = [\Phi^T \ H^T]^T, \ \bar{R} = \text{blkdiag}(R, R_1)$$

Minimizing $f_0(W)$ we have:

$$\frac{\partial f_0}{\partial W} = \frac{\partial \left(f_1(W) - \text{tr}\{(Z - W^T H)^T R_1^{-1}(Z - W^T H)\}\right)}{\partial W} = 0$$
$$(P_1^{-1} - H R_1^{-1} H^T)W = (\bar{\Phi} \bar{R}^{-1} \bar{Y}^T - H R_1^{-1} Z^T) \tag{53}$$

where $P_1^{-1} = \bar{\Phi} \bar{R}^{-1} \bar{\Phi}^T = P_0^{-1} + H R_1^{-1} H^T$ and $P_0^{-1} = \Phi R^{-1} \Phi^T$. Since $R > 0$ and $\text{rank}(\Phi) = k$ it follows that $P_0^{-1} > 0$ which also entails that $P_1^{-1} > 0$ as $R_1 > 0$ and $H R_1^{-1} H^T \geq 0$. Notice also that $(P_1^{-1} - H R_1^{-1} H^T) = P_0^{-1}$ hence we can invert it in Eq. 53 to get:

$$W = (P_1^{-1} - H R_1^{-1} H^T)^{-1}(\bar{\Phi} \bar{R}^{-1} \bar{Y}^T + H(-R_1^{-1})Z^T)$$

Applying the matrix inversion lemma we have:

$$W = W_1 - P_1 H(-R_1 + H^T P_1 H)^{-1}(W_1^T H)^T + $$
$$(P_1 - P_1 H(-R_1 + H^T P_1 H)^{-1} H^T P_1)H(-R_1^{-1})Z^T \tag{54}$$

where $W_1 = P_1 \bar{\Phi} \bar{R}^{-1} \bar{Y}^T$, $P_1 = (\bar{\Phi} \bar{R}^{-1} \bar{\Phi}^T)^{-1}$ is indeed the solution of $min_W f_1(W)$ based on Theorem 1. Notice that $-R_1 + H^T P_1 H$ is in fact negative definite, hence it is invertible. To prove so, consider the following matrix:

$$M = \begin{bmatrix} P_1^{-1} & H \\ H^T & R_1 \end{bmatrix}$$

Since $P_1^{-1} > 0$ and the Schur complement of M w.r.t. the upper left block matrix is $P_1^{-1} - H R_1^{-1} H^T = P_0^{-1} > 0$ it follows that $M > 0$ [32]. Therefore, as $R_1 > 0$ and $M > 0$ it should also hold that the Schur complement of M w.r.t. the lower right block matrix should be positive definite, i.e.:

$$R_1^{-1} - H^T P_1 H > 0 \Rightarrow -R_1^{-1} + H^T P_1 H < 0$$

We can further process the last term in Eq. 54 similar to the analysis in Eq. 39 to find that:

$$(P_1 - P_1 H(-R_1 + H^T P_1 H)^{-1} H^T P_1)H(-R_1^{-1})Z^T = $$
$$P_1 H(-R_1 + H^T P_1 H)^{-1} Z^T$$

and substituting the last in Eq. 54 we obtain the result given by Eq. 50. Finally, having already established that $\boldsymbol{P}_1 > \boldsymbol{0}$ and $-\boldsymbol{R}_1^{-1} + \boldsymbol{H}^T \boldsymbol{P}_1 \boldsymbol{H} < \boldsymbol{0}$ it follows that $-\boldsymbol{P}_1 \boldsymbol{H}(-\boldsymbol{R}_1 + \boldsymbol{H}^T \boldsymbol{P}_1 \boldsymbol{H})^{-1} \boldsymbol{H}^T \boldsymbol{P}_1 \geq \boldsymbol{0}$ hence $\hat{\boldsymbol{P}}_0 > \boldsymbol{0}$. $\qquad\square$

Declarations

Conflict of Interests   The authors declare that there isn't any kind of conflict of interest.

Ethics Approval   Not applicable.

Consent to Participate   Not applicable.

Consent for Publication   Not applicable.

References

1. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: Learning attractor models for motor behaviors. Neural Comput. **25**(2), 328–373 (2013)
2. Pastor, P., Kalakrishnan, M., Meier, F., Stulp, F., Buchli, J., Theodorou, E., Schaal, S.: From dynamic movement primitives to associative skill memories. Rob. Auton. Syst. **61**(4), 351–361 (2013). https://doi.org/10.1016/j.robot.2012.09.017. Models and Technologies for Multi-modal Skill Training
3. Pastor, P., Righetti, L., Kalakrishnan, M., Schaal, S.: Online movement adaptation based on previous sensor experiences. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 365–371 (2011). https://doi.org/10.1109/IROS.2011.6095059
4. Ude, A., Gams, A., Asfour, T., Morimoto, J.: Task-specific generalization of discrete and periodic dynamic movement primitives. IEEE Trans. Robot. **26**(5), 800–815 (2010). https://doi.org/10.1109/TRO.2010.2065430
5. Umlauft, J., Sieber, D., Hirche, S.: Dynamic movement primitives for cooperative manipulation and synchronized motions. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 766–771 (2014). https://doi.org/10.1109/ICRA.2014.6906941
6. Mülling, K., Kober, J., Kroemer, O., Peters, J.: Learning to select and generalize striking movements in robot table tennis. Int. J. Rob. Res. **32**, 263–279 (2013). https://doi.org/10.1177/0278364912472380
7. Gams, A., Petric, T., Nemec, B., Ude, A.: Learning and adaptation of periodic motion primitives based on force feedback and human coaching interaction. In: 2014 IEEE–RAS International Conference on Humanoid Robots, pp. 166–171 (2014). https://doi.org/10.1109/HUMANOIDS.2014.7041354
8. Luo, L., Foo, M.J., Ramanathan, M., Er, J.K., Chiam, C.H., Li, L., Yau, W.Y., Ang, W.T.: Trajectory generation and control of a lower limb exoskeleton for gait assistance. J. Intell. Robot. Syst. **106**(3), 64 (2022). https://doi.org/10.1007/s10846-022-01763-5
9. Billard, A., Calinon, S., Dillmann, R., Schaal, S.: In: Siciliano, B., Khatib, O. (eds.) Robot Programming by Demonstration, pp. 1371–1394. Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-30301-5_60
10. Hoffmann, H., Pastor, P., Park, D., Schaal, S.: Biologically-inspired dynamical systems for movement generation: Automatic real–time goal adaptation and obstacle avoidance. In: 2009 IEEE International Conference on Robotics and Automation, pp. 2587–2592 (2009)
11. Ginesi, M., Sansonetto, N., Fiorini, P.: Overcoming some drawbacks of dynamic movement primitives. Rob. Auton. Syst. **144**, 103844 (2021). https://doi.org/10.1016/j.robot.2021.103844
12. Koutras, L., Doulgeri, Z.: A novel dmp formulation for global and frame independent spatial scaling in the task space. In: 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pp. 727–732 (2020). https://doi.org/10.1109/RO-MAN47096.2020.9223500
13. Dragan, A.D., Muelling, K., Andrew Bagnell, J., Srinivasa, S.S.: Movement primitives via optimization. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2339–2346 (2015). https://doi.org/10.1109/ICRA.2015.7139510
14. Zhou, Y., Gao, J., Asfour, T.: Learning via-point movement primitives with inter- and extrapolation capabilities. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4301–4308 (2019). https://doi.org/10.1109/IROS40897.2019.8968586
15. Mghames, S., Hanheide, M., Ghalamzan E., A.: Interactive movement primitives: Planning to push occluding pieces for fruit picking. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2616–2623 (2020). https://doi.org/10.1109/IROS45743.2020.9341728
16. Nemec, B., Ude, A.: Action sequencing using dynamic movement primitives. Robotica **30**(5), 837–846 (2012). https://doi.org/10.1017/S0263574711001056
17. Kulvicius, T., Ning, K., Tamosiunaite, M., Worgötter, F.: Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. IEEE Trans. Robot. **28**(1), 145–157 (2012). https://doi.org/10.1109/TRO.2011.2163863
18. Cardoso, C., Jamone, L., Bernardino, A.: A novel approach to dynamic movement imitation based on quadratic programming. In: 2015 IEEE International Conference on Robotics and Automa-

tion (ICRA), pp. 906– 911 (2015). https://doi.org/10.1109/ICRA.2015.7139285

19. Kim, J.-J., Park, S.-Y., Lee, J.-J.: Adaptability improvement of learning from demonstration with sequential quadratic programming for motion planning. In: 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1032–1037 (2015). https://doi.org/10.1109/AIM.2015.7222675

20. Maeda, G., Ewerton, M., Lioutikov, R., Amor, H.B., Peters, J., Neumann, G.: Learning interaction for collaborative tasks with probabilistic movement primitives. In: 2014 IEEE-RAS International Conference on Humanoid Robots, pp. 527–534 (2014)

21. Maeda, G.J., Neumann, G., Ewerton, M., Lioutikov, R., Kroemer, O., Peters, J.: Probabilistic movement primitives for coordination of multiple human-robot collaborative tasks. Auton. Robots **41**(3), 593–612 (2017). https://doi.org/10.1007/s10514-016-9556-2

22. Weitschat, R., Aschemann, H.: Safe and efficient human-robot collaboration part ii: Optimal generalized human-in-the-loop real-time motion generation. IEEE Robot. Autom. Lett. **3**(4), 3781–3788 (2018). https://doi.org/10.1109/LRA.2018.2856531

23. Paraschos, A., Daniel, C., Peters, J., Neumann, G.: Probabilistic movement primitives. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. NIPS'13, pp. 2616–2624. Curran Associates Inc., Red Hook, NY, USA (2013)

24. Paraschos, A., Daniel, C., Peters, J., Neumann, G.: Using probabilistic movement primitives in robotics. Auton. Robots **42**(3), 529–551 (2018). https://doi.org/10.1007/s10514-017-9648-7

25. Huang, Y., Rozo, L., Silvério, J., Caldwell, D.G.: Kernelized movement primitives. Int. J. Rob. Res. **38**(7), 833–852 (2019). https://doi.org/10.1177/0278364919846363

26. Sidiropoulos, A., Doulgeri, Z.: A reversible dynamic movement primitive formulation. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 3147–3153 (2021). https://doi.org/10.1109/ICRA48506.2021.9562059

27. Sidiropoulos, A., Papageorgiou, D., Doulgeri, Z.: A novel framework for generalizing dynamic movement primitives under kinematic constraints. Auton. Robots (2022). https://doi.org/10.1007/s10514-022-10067-4

28. Koutras, L., Doulgeri, Z.: A correct formulation for the orientation dynamic movement primitives for robot control in the cartesian space. In: Proceedings of The 3rd Conference on Robot Learning (2019)

29. Wang, R., Wu, Y., Chan, W.L., Tee, K.P.: Dynamic movement primitives plus: For enhanced reproduction quality and efficient

trajectory modification using truncated kernels and local biases. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3765– 3771 (2016). https://doi.org/10.1109/IROS.2016.7759554

30. Ginesi, M., Meli, D., Roberti, A., Sansonetto, N., Fiorini, P.: Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions. J. Intell. Robot. Syst. **101**(4), 79 (2021). https://doi.org/10.1007/s10846-021-01344-y

31. Kailath, T., Sayed, A.H., Hassibi, B.: Linear Estimation. Prentice-Hall information and system sciences series. Prentice Hall, Upper Saddle River, NJ (2000)

32. Strang, G.: Linear Algebra and Its Applications. Thomson, Brooks/Cole, Belmont, CA (2006)

Antonis Sidiropoulos obtained his diploma (MSc equivalent) from the Department of Electrical and Computer Engineering (ECE) at Aristotle University of Thessaloniki (AUTH) in 2016. He earned his Ph.D. from the Automation & Robotics Lab at AUTH in 2023. His research interests encompass physical human-robot interaction, robot learning from demonstrations, online motion planning with dynamical systems, and optimal estimation and control.

Zoe Doulgeri (IEEE senior member) is a Professor of Robotics and Control of Manufacturing Systems and the director of the Automation and Robotics Lab of the Department of Electrical and Computer Engineering of the Aristotle University of Thessaloniki (AUTH). She received the diploma of Electrical Engineering from AUTH and an M.Sc. (DIC) in Control Systems, an M.Sc.(DIC) in Social and Economic Aspects of Science and Technology in Industry and a Ph.D in Mechanical Engineering, from the Imperial College, London, UK. Her current research interests include the topics of physical human robot interaction, robot teaching and learning by demonstration, bimanual mobile robots, object grasping and manipulation with analytical and data based learning methods and the control of uncertain robotic systems. She is currently an evaluator of Horizon Europe research and innovation projects.