Autonomous Robots
https://doi.org/10.1007/s10514-022-10067-4

A novel framework for generalizing dynamic movement primitives under kinematic constraints

Antonis Sidiropoulos[1] · Dimitrios Papageorgiou[1] · Zoe Doulgeri[1]

Received: 14 December 2021 / Accepted: 23 September 2022

Abstract
In this work, we propose a novel framework for generalizing a desired trajectory pattern, encoded using Dynamic Movement Primitives (DMP), subject to kinematic constraints. DMP have been extensively used in robotics for encoding and reproducing kinematic behaviours, thanks to their generalization, stability and robustness properties. However, incorporating kinematic constraints has not yet been fully addressed. To this end, we design an optimization framework, based on the DMP formulation from our previous work, for generalizing trajectory patterns, encoded with DMP subject to kinematic constraints, considering also time-varying target and time duration, via-point and obstacle constraints. Simulations highlight these properties and comparisons are drawn with other approaches for enforcing constraints on DMP. The usefulness and applicability of the proposed framework is showcased in experimental scenarios, including a handover, where the target and time duration vary, and placing scenarios, where obstacles are dynamically introduced in the scene.

Keywords Dynamic movement primitives · Online trajectory adaptation · Constrained optimization · Constrained motion generation

1 Introduction

The remarkable advancements in robotics over the past few decades have triggered an increasing interest in research oriented towards deploying robots in industrial and household environments. Designing autonomous robotic systems that can co-exist with humans and efficiently replicate their capabilities poses a major challenge. To this end PbD (Programming by Demonstration) has been proposed as an effective means for endowing robots with human capabilities (Billard et al., 2008). This is tightly coupled with the mathematical model that is adopted for encoding the demonstrated trajectory pattern and its generalization properties.

Movement primitives have been extensively used for encoding and generalizing trajectory patterns. In this category, Dynamic Movement Primitives is a celebrated and widely employed method, due to it spatio-temporal generalization capabilities, their stability properties and robustness to perturbations (Ijspeert et al., 2013). However, one cannot guarantee a-priori that the new generalized trajectory, produced by the DMP, will not exceed the robot's or task-related kinematic constraints, i.e. constraints relating to position, velocity and/or acceleration. There are even cases where this generalization can be problematic causing overshoots and in turn large velocities and accelerations. Moreover, changing the target position online (during execution) or the motion duration can result in undesired discontinuities and/or high accelerations.

1.1 Related works

Many works in the literature have attempted to tackle these issues. For the case of large overshoots, modified DMP formulations have been proposed (Hoffmann et al., 2009; Koutras and Doulgeri, 2020). These works however try to mitigate the negative effects of "over-scaling", but without enforcing specific bounds on the position, velocity and

✉ Antonis Sidiropoulos
antosidi@ece.auth.gr

Dimitrios Papageorgiou
dimpapag@ece.auth.gr

Zoe Doulgeri
doulgeri@ece.auth.gr

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

Published online: 10 October 2022

acceleration of the generated trajectory. Thus, Koutras and Doulgeri (2020) can still produce large spatial scalings in some cases, while (Hoffmann et al., 2009) fails to reproduce the demonstrated trajectory pattern to new targets that are relatively far away from the demonstrated one. In (Papageorgiou and Doulgeri, 2020), a parametric DMP formulation is proposed for the problem of spatial generalization under spatial constraints, given a known task geometry. However, velocity or acceleration constraints are not considered. For the case of discontinuous changes in the target position, an exponential goal filtering was proposed in Ijspeert et al. (2013), which however requires additional tuning or even on-line adjustment of the exponential filter parameter to ensure the desired speed of convergence to the target. Moreover, all these works cannot guarantee the satisfaction of position, velocity and acceleration constraints.

To this end, repelling forces generated from artificial potential fields have been proposed to enforce position limits in Gams et al. (2009) or for velocity limits in Dahlin and Karayiannidis (2020). Nonetheless, one major drawback of these methods is that they can incur large accelerations. Tuning of the repelling force gain can mitigate this effect in some cases, at the cost however of larger deviations from the nominal trajectory pattern. What is more, tuning may have to be repeated for a different trajectory or task. Consequently, with repelling forces, one cannot anticipate how large will be the accelerations and the deviations from the nominal trajectory. Transforming the DMP trajectory through the hyperbolic tangent to enforce position limits was proposed in Duan et al. (2018), which however does not account for velocity, acceleration or via-point constraints.

Adjusting a movement primitive to pass from via-points, while being a useful property, is yet another source of prospective constraints violations. The ability of ProMP (Probabilistic Movement Primitives) to adjust to via-points (Paraschos et al., 2018) was exploited in Mghames et al. (2020) for pushing pieces that occlude strawberries to be harvested. In Ben Amor et al. (2014), a probabilistic extension of the original DMP was proposed that could include via-points, but it requires specifying also the velocity and acceleration for a specific via-point that may be considered a disadvantage (Maeda et al., 2014). Despite the via point inclusion, (Mghames et al., 2020; Ben Amor et al., 2014) do not account for velocities and/or acceleration constraints along the generated trajectory.

Another way to handle constraints is by means of optimization. To the best of the authors knowledge, there are only a few works in the literature that have adopted this approach with DMP. The authors in Cardoso et al. (2015) use a weighted sum of Gaussian functions that generate the acceleration. The weights of these Gaussians are optimized by solving offline a quadratic program (QP) that minimizes the error between the generated acceleration and the temporarily scaled demonstrated acceleration, subject to kinematic constraints. They then integrate this model and provide it as reference to an impedance controller, realizing a dynamical system analogous to that of a DMP. However, the generated optimized trajectory cannot scale temporarily or spatially if the target or time duration change on-line and consequently this optimization cannot be performed on-line. In Krug and Dimitrov (2015), Model Predictive Control (MPC) is used with DMP, for merging multiple DMP patterns. The aim is to find the percentage by which each DMP should contribute to the generated motion which is a different problem to the one considered in this work. The proposed optimization in Krug and Dimitrov (2015) introduces also affine constraints for obstacles approximated by their convex hull. However, the latter is applied only in 2D simulations, without considering velocity/acceleration constraints or the computational load of the optimization. In Liang et al. (2021), the authors propose to tackle the motion retargeting problem for generating dual-arm sign language motions by applying an offline constrained optimization that minimizes the difference from trajectories generated by DMP, which encode the human demonstrations.

There exist also other approaches that explore trajectory generation under constraints, but do not employ DMP. For intance, Explicit Reference Governors are utilized in Merckaert et al. (2022) to enforce input (actuator limits) and state constraints (joint position, velocity limits) in real-time, for robots operating close to humans. However, reaching of a desired pose is considered, and not generalization of a trajectory pattern under constraints. Generating constrained trajectories with ProMP was proposed in Frank et al. (2021), by formulating the problem as an optimization of the ProMP's distribution, where the constraints are realized by imposing low probability of the ProMP's distribution close to the constraints. However, its real-time execution capabilities are not addressed. Finally, while there are a lot works in literature on trajectory planning that consider constraints (Wen & Pagilla, 2021; Buizza Avanzini et al. 2018), the optimization that is carried out does not consider a reference trajectory pattern, in contrast to the case examined here.

1.2 Contribution

In this work, we propose an optimization framework that achieves optimal generalization of a learned trajectory pattern to a new fixed target and time duration under position, velocity and acceleration kinematic constraints including via points and obstacles for static scenes (off-line) and an MPC-like extension for handling constraints introduced in real-time and varying target and/or time duration (on-line). Both approaches leverages the DMP formulation introduced in our previous work (Sidiropoulos & Doulgeri, 2021). The off-line approach produces the global optimal solution which is the optimal choice if no dynamic changes occur. The on-line approach considers only a confined local time horizon, hence it may produce locally optimal (w.r.t. the entire time duration) solutions, which is nevertheless an inevitable compromise to achieve real-time performance while handling on-line dynamic changes and constraints. The distinctive feature of the proposed framework is that it can generate optimal DMP trajectories, accounting for all the aforementioned type of constraints. In contrast, previous works are designed to address only a subset of them. Specifically, (Gams et al., 2009; Duan et al., 2018), focus on position and Dahlin and Karayiannidis (2020) on velocity constraints too. Via points are only considered in Mghames et al. (2020), Ben Amor et al. (2014) and only obstacles in Krug and Dimitrov (2015). Finally, (Cardoso et al., 2015; Liang et al., 2021) were applied and tested only off-line. Moreover, those that do not employ optimization produce sub-optimal trajectories, in the sense that the generated trajectory may deviate from the unconstrained DMP trajectory even when the latter satisfies the constraints. The advantages and efficiency of our method is highlighted through simulations, which include comparisons with other methods, and practical experimental scenarios.

The rest of this paper is organized as follows: In Sect. 2 we provide preliminaries on the new DMP formulation. Section 3 presents the proposed off-line optimization framework, compared against other methods. This framework is extended in Sect. 4 to also accommodate online trajectory modifications subject to kinematic constraints, providing also several comparative simulations. Experimental validation is performed in Sect. 5 and the conclusions are drawn in Sect. 6. The code for the simulations and experiments is available at https://github.com/Slifer64/novel-DMP-constraints.git.

2 DMP-preliminaries

In this section we briefly introduce the basics of the DMP formulation from Sidiropoulos and Doulgeri (2021) for 1-DoF. The DMP encodes a desired trajectory $y_r(t)$ and can generalize this trajectory starting from a new initial position y_0 towards a new target/final position g with time duration t_f. The DMP's evolution is driven by the canonical system, which provides the phase variable σ (time substitute), to avoid direct time dependency. The DMP and the canonical system are given by:

$$\ddot{y} = \ddot{y}_\sigma - D(\dot{y} - \dot{y}_\sigma) - K(y - y_\sigma) \tag{1}$$

$$\dot{\sigma} = h(\sigma)/\tau \ , \ \sigma(0) = 0 \ , \ \sigma(t_f) = 1 \tag{2}$$

where y is the position, K, D are positive scalars and y_σ provides the generalized trajectory:

$$y_\sigma \triangleq k_s(\boldsymbol{\phi}(\sigma)^T \boldsymbol{w} - \hat{y}_0) + y_0 \tag{3}$$

$$\dot{y}_\sigma = k_s \boldsymbol{\phi}_1(\sigma)^T \boldsymbol{w} \tag{4}$$

$$\ddot{y}_\sigma = k_s \boldsymbol{\phi}_2(\sigma)^T \boldsymbol{w} \tag{5}$$

where the desired trajectory y_r is encoded as a weighted sum of K Gaussians through $\boldsymbol{\phi}(\sigma)^T\boldsymbol{w}$, $\boldsymbol{\phi}(\sigma)^T = [\phi_1(\sigma) \cdots \phi_K(\sigma)]/\sum_{k=1}^K \phi_k(\sigma)$, with $\phi_k(\sigma) = e^{-h_k(\sigma-c_k)^2}$. The spatial scaling term k_s is given by:

$$k_s \triangleq \frac{g - y_0}{\hat{g} - \hat{y}_0} \tag{6}$$

where $\hat{g} = \boldsymbol{\phi}(1)^T\boldsymbol{w}$, $\hat{y}_0 = \boldsymbol{\phi}(0)^T\boldsymbol{w}$. The velocity and acceleration can be obtained analytically from Eq. (4), (5) where $\boldsymbol{\phi}_1(\sigma) \triangleq \frac{\partial\boldsymbol{\phi}}{\partial\sigma}\dot{\sigma}$ and $\boldsymbol{\phi}_2(\sigma) \triangleq \frac{\partial^2\boldsymbol{\phi}}{\partial^2\sigma}\dot{\sigma}^2 + \frac{\partial\boldsymbol{\phi}}{\partial\sigma}\ddot{\sigma}$. The weights $\boldsymbol{w}$ are optimized using Least Squares (LS) or Locally Weighted Regression (LWR) (Ijspeert et al., 2013) so that $\boldsymbol{\phi}^T\boldsymbol{w} \approx y_r$. To achieve good approximation, a general heuristic is to place the centers c_i of the Gaussian kernels equally spaced in time and then define the inverse widths of the Gaussians as $h_i = \frac{a_h}{(c_{i+1}-c_i)^2}$, $h_N = h_{N-1}$, $i = 1, \cdots, N$, where $a_h > 0$ is a scaling factor controlling the overlapping between the kernels. Regarding the canonical system Eq. (2), $\tau > 0$ controls the speed of the phase variable's evolution and $h(\sigma)$ can be any function such that its integral $\frac{1}{\tau}\int_0^t h(\sigma(u))du$ is a continuous monotonically evolving function that satisfies the boundary conditions $\sigma_0 = 0$, $\sigma_f = 1$ and is constant outside the interval $[0, \ 1]$. A valid choice could be for instance $\tau = t_f$ and $h(\sigma) = 1$ for $\sigma \in [0, \ 1]$ and zero outside.

This specific DMP formulation has been shown in Sidiropoulos and Doulgeri (2021) to be mathematically equivalent to the classical DMP formulation from Ijspeert et al. (2013). Additional it has the advantage, over the classical DMP, that the DMP's reference position, velocity and acceleration are affine w.r.t. the DMP weights and that we can obtain the DMP trajectory values for an arbitrary time instant σ from Eq. (3–5) without needing to perform explicitly any integration. This key property lends itself to the proposed optimization in this work.

3 Off-line optimal DMP

Generalization of the learned kinematic behaviour to different targets and with different time durations is a very expedient property of DMP. However, one cannot guarantee a-priori that the new generalized trajectory, which will be produced by the DMP, will not exceed the robot's or user-specified kinematic constraints, i.e. constraints relating to position, velocity and/or acceleration. There are even cases where this generalization, especially the spatial generalization, can be problematic causing overshoots and in turn large velocities and accelerations.

3.1 Proposed solution

We propose to tackle these issues by means of optimization. The high level objective is to find the trajectory that satisfies the kinematic constraints and it is the closest to the nominal trajectory, i.e. the unconstrained trajectory that would be produced by the DMP. This objective is formulated as a convex optimization problem which provides the optimal DMP weights that generalize the trajectory learned by the DMP, under the enforced constraints. We begin by presenting our approach for a single DoF denoted here by y. Given the desired time duration T of the executed trajectory and the kinematic constraints $\underline{y}(\sigma) \leq y(\sigma) \leq \bar{y}(\sigma)$, $\underline{\dot{y}}(\sigma) \leq \dot{y}(\sigma) \leq \bar{\dot{y}}(\sigma)$, $\underline{\ddot{y}}(\sigma) \leq \ddot{y}(\sigma) \leq \bar{\ddot{y}}(\sigma)$ we wish to find the trajectory that satisfies these constraints and is the closest to the unconstrained trajectory $y_d(\sigma)$, $\dot{y}_d(\sigma)$, $\ddot{y}_d(\sigma)$ produced by Eq. (3–5) using the initial trained DMP weights. The kinematic constraints could be for instance the robot's joint limits, if y represents the position of a joint, or reflect the workspace and speed limits if y refers to the task space coordinates.

We first discretize the kinematic constraints as $\underline{y}_i \leq y_i \leq \bar{y}_i$, $\underline{\dot{y}}_i \leq \dot{y}_i \leq \bar{\dot{y}}_i$, $\underline{\ddot{y}}_i \leq \ddot{y}_i \leq \bar{\ddot{y}}_i$, defined at M discrete points[1] $i = 1...M$ corresponding to the normalized timestamps $\sigma_i = t_i/T$ equally spaced in [0 1]. For simplicity, we assume a linear canonical system $\dot{\sigma} = 1/T$ for $s \in [0\ 1)$ and $\dot{\sigma} = \ddot{\sigma} = 0$ otherwise. Moreover, to simplify the problem formulation and make it independent of the spatial scaling, we consider the scaled weights $\boldsymbol{w}_s \triangleq k_s \boldsymbol{w}$ and transform any position y according to:

$$h_0(y) \triangleq (y - y_0) + k_s \hat{y}_0 \tag{7}$$

We can now form the following QP:

$$\begin{aligned}
\min_{\mathbf{w}_s} & \sum_{i=1}^{L} (1-\lambda)||h_0(y_{d,i}) - \boldsymbol{\phi}^T(\sigma_i)\boldsymbol{w}_s||_2^2 \\
& + \lambda||\dot{y}_{d,i} - \boldsymbol{\phi}_1^T(\sigma_i)\boldsymbol{w}_s||_2^2 \\
\text{st. } & h_0(\underline{y}_i) \leq \boldsymbol{\phi}^T(\sigma_i)\boldsymbol{w}_s \leq h_0(\bar{y}_i) \\
& \underline{\dot{y}}_i \leq \boldsymbol{\phi}_1^T(\sigma_i)\boldsymbol{w}_s \leq \bar{\dot{y}}_i \quad \forall\, i = 1...M \\
& \underline{\ddot{y}}_i \leq \boldsymbol{\phi}_2^T(\sigma_i)\boldsymbol{w}_s \leq \bar{\ddot{y}}_i \\
& \boldsymbol{\phi}^T(0)\boldsymbol{w}_s = h_0(\boldsymbol{y}_0), \quad \boldsymbol{\phi}^T(1)\boldsymbol{w}_s = h_0(\boldsymbol{g})
\end{aligned} \tag{8}$$

where the objective function has L data points, with normalized timestamps chosen uniformly in [0 1] and λ determines whether we want to optimize w.r.t. the position trajectory ($\lambda = 0$) or velocity profile ($\lambda = 1$). The equality constraints in Eq. (8) are the initial and final value constraints. The optimal solution can be retrieved through $\boldsymbol{w}^* = k_s^{-1}\boldsymbol{w}_s^*$. We can also incorporate additional equality constraints, related for instance to via-points, by transforming them through Eq. (7). We can even set the velocity at the final position to be non-zero, e.g. for a striking motion. In this way, hitting primitives can be readily realized, in contrast to other approaches that modify the DMP model solely to achieve this purpose (Kober et al., 2010; Mülling et al., 2013). The extension of Eq. (8) to multiple DoFs is straightforward.

Remark 1 Notice that the DMP formulation from Sidiropoulos and Doulgeri (2021), in contrast to the classical DMP (Ijspeert et al., 2013), has the property that the DMP position, velocity and acceleration is affine w.r.t. the DMP weights $\boldsymbol{w}$ (see Eq. 3–5). This expedient property allows us to express the problem as a QP and solve it efficiently, which cannot be attained with the classical DMP. To the best of the authors' knowledge, this is the first realization of a DMP formulation that can achieve generalization under kinematic constraints, while being mathematically equivalent with the classical DMP (Sidiropoulos & Doulgeri, 2021) and sharing all of its favourable properties.

3.2 Simulations

To highlight the efficacy of the proposed optimization procedure we carried out several simulations, using as training data a demonstration obtained via kinesthetic guidance on a ur5e robot. We train a DMP with 30 kernels applying LS. For reproduction, we offset the target $\boldsymbol{g}$ by $[0.7\ -0.7\ 0.05]$ from the demonstrated target $\boldsymbol{g}_d$. The kinematic constraints are $[-1.2\ -1.2\ 0.22]^T \leq \boldsymbol{p} \leq [1.2\ 1.2\ 0.5]^T$, $-0.3 \leq \dot{p}_j \leq 0.3$ and $-0.4 \leq \ddot{p}_j \leq 0.4$ for $j \in \{x, y, z\}$. Equality constraints are used for setting the initial and final position with zero velocity and acceleration. For the optimization we use $L = M = 200$ points uniformly distributed in [0 1] and for solving this QP problem the OSQP library was used (Stellato et al., 2020). The value of λ is set to 0 when optimizing the position and 1 when optimizing the velocity.

We demonstrate first the results of the proposed approach for optimizing w.r.t. to the position (path), referred as DMP$_p^*$, and for optimizing w.r.t. the velocity (shape), referred as DMP$_v^*$. The Cartesian paths are plotted at the left of Fig. 1a along with the demo (grey dashed line) and the unconstrained DMP path (blue dashed line). The demonstrated target is plotted with magenta 'x' mark and the new target with red. The position bounds in the z-axis are also visualized with light red planes. Optimizing the position (purple line) results in saturation at the position limits, while optimizing the velocity (green line) scales down the trajectory, preserving the initial shape. Which of the two is better is application dependent.

[1] Due to the continuity of the Gaussian functions, it is sufficient to specify a finite number of datapoints M to ensure that the constraints will also be satisfied for intermediate data points as well.

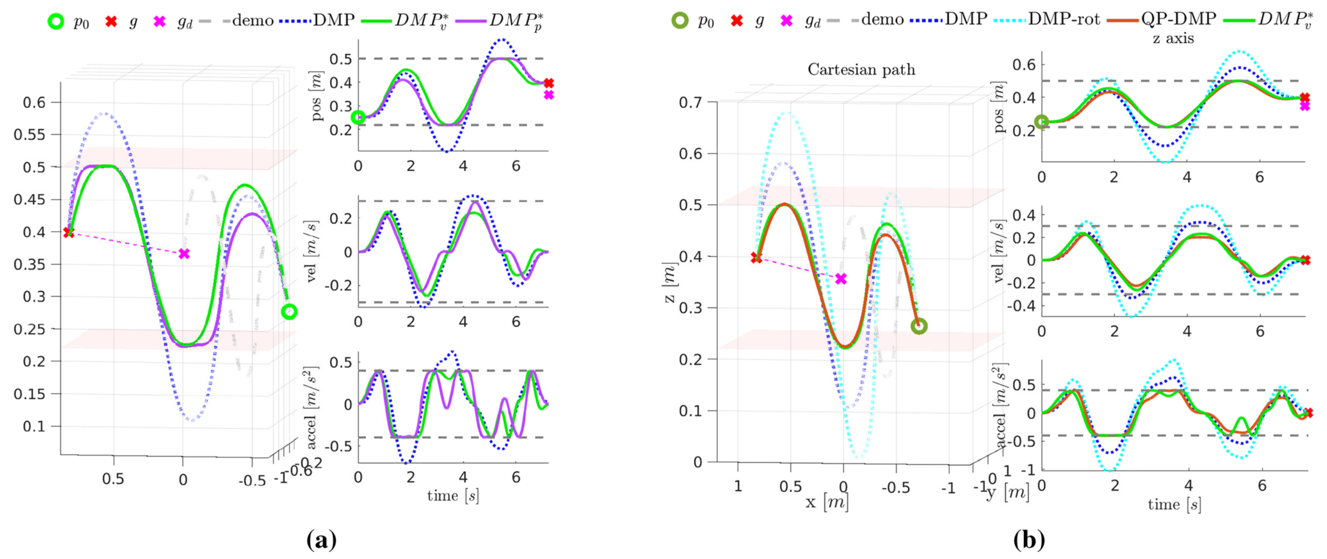


Fig. 1 **a** DMP* with position and velocity optimization. Left: Cartesian path. Right: Trajectory along z axis. **b** Comparison of DMP*with other approaches. Left: Cartesian path. Right: Trajectory along z axis

The position, velocity and acceleration trajectories along the z-axis plotted at the right of Fig. 1a, with dashed gray lines denoting the kinematic limits, show that the kinematic constraints were satisfied.

We also compare our method against the original DMP, the modified DMP proposed in Koutras and Doulgeri (2020) aimed at mitigating the effects of over-scaling by applying a rotation (referred hereafter as DMP-rot) and the approach proposed in Cardoso et al. (2015), which also applies optimization w.r.t. the acceleration (referred hereafter as QP-DMP). We consider the same trajectory and constraints as before. The results are depicted in Fig. 1b. We observe that both the DMP and DMP-rot violate the constraints. In particular, DMP-rot produces a spatial scaling larger than the DMP, because the final position $\boldsymbol{g}$ is afar from the initial $\boldsymbol{y}_0$ in the xy-plane, so the distance $||\boldsymbol{g} - \boldsymbol{y}_0||$ which scales all DoFs in DMP-rot, over-scales the trajectory along the z-axis. The large scaling of the DMP is expected, as the demonstrated initial and final positions were relatively close. In contrast, the proposed approach, depicted with green solid line, produces a predictable and reliable behaviour that satisfies the kinematic constraints. Regarding the QP-DMP (light brown line), the results are relatively close to the ones obtained with DMP*. Both approaches satisfy the constraints and the shape of the trajectory they produce resembles the shape of the unconstrained nominal trajectory (blue dashed line). However, the QP-DMP cannot be used if the target changes, whereas DMP* can still be used, even though it may not be optimal due to the target being altered. Moreover, the QP-DMP cannot optimize w.r.t. the position, while with DMP* this option is available. Also notice that the QP-DMP requires storing all demonstrated acceleration data to perform the optimization and the use of the acceleration as training can introduce a lot of noise in practice.

4 On-line optimal DMP

Given an $n - DoF$ trajectory pattern encoded by a DMP as in Eq. (3–5) using K Gaussian kernels, which produces the reference trajectory $\boldsymbol{y}_d \in \mathbb{R}^n$, we tackle here the problem of on-line generalization of this motion pattern for a new target and motion duration, that can be time-varying in general, under kinematic, obstacle or via-point constraints, which can be introduced on-line. In such cases, the optimallity of DMP* is compromised. Merely confining the time horizon in Eq. (8) to a smaller local time window could result in the optimizer generating "locally" optimal solutions, which can even result in infeasibility in the next optimization horizon due to conflicting constraints .[2] Increasing the prediction horizon could alleviate this issue, but a larger horizon can make the computational cost prohibitive for real-time use. In the following, we detail how to extend DMP* so as to address these issues.

4.1 Proposed solution

The proposed solution is presented schematically in Fig. 2. The DMP model takes as input at the current time step j

[2] Such is the case for instance if in order to achieve better tracking (i.e. lower cost for the objective function) within a horizon, the position comes very close to the upper limit, and the velocity and acceleration are positive. The optimizer may be unable to find a feasible solution at the start of the next horizon, as due to the acceleration limits, it may not be possible to make the velocity negative in time, before the position breaches the upper limit.

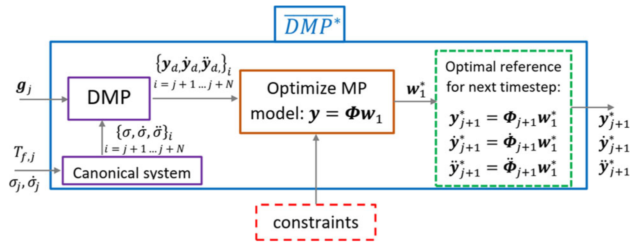

$\overline{DMP}^*$
g_j
DMP
$\{y_d, \dot{y}_d, \ddot{y}_d\}_i$ $i = j+1 \dots j+N$
Optimize MP model: $y = \Phi w_1$
w_1^*
Optimal reference for next timestep:
$y^*_{j+1} = \Phi_{j+1} w_1^*$
$\dot{y}^*_{j+1} = \dot{\Phi}_{j+1} w_1^*$
$\ddot{y}^*_{j+1} = \ddot{\Phi}_{j+1} w_1^*$
y^*_{j+1} $\dot{y}^*_{j+1}$ $\ddot{y}^*_{j+1}$
$\{\sigma, \dot{\sigma}, \ddot{\sigma}\}_i$ $i = j+1 \dots j+N$
$T_{f,j}$
$\sigma_j, \dot{\sigma}_j$
Canonical system
constraints

Fig. 2 On-line optimal DMP block diagram

the current target $\boldsymbol{g}_j$ and time duration $T_{f,j}$ and generates the reference trajectory $\boldsymbol{y}_d$, $\dot{\boldsymbol{y}}_d$, $\ddot{\boldsymbol{y}}_d$ over a horizon of N datapoints, where the first one, $j+1$, corresponds to the next control cycle T_s and the rest are spaced equally in time by $\Delta t \gg T_s$, to make the optimizer more far-sighted in order to produce better solutions and anticipate imminent infeasibilities. We optimize an MP (Movement Primitive) model, that has to produce a trajectory as close as possible to these N points while also satisfying the constraints. The MP is defined as $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w}_1$, where $\boldsymbol{w}_1 \in \mathbb{R}^{Kn}$ are the weights to be optimized and $\boldsymbol{\Phi} \triangleq \boldsymbol{I}_n \otimes \bar{\boldsymbol{\phi}}^T$, where $\otimes$ denotes the Kronecker product and $\bar{\boldsymbol{\phi}} \in \mathbb{R}^K$ are normalized Gaussian kernels as in the classical DMP, which are however truncated (the k-th component of $\bar{\boldsymbol{\phi}}$ is set to zero if it is less than 10^{-6}). The latter plays a decisive role in making the optimization problem sparser, enabling the solution to be derived in real-time. The optimal trajectory for the next control cycle at $j+1$ is generated by the optimized MP through $\boldsymbol{y}^*_{j+1} = \boldsymbol{\Phi}_{j+1}\boldsymbol{w}_1^*$ and its derivatives (similar to Eqs. (4), (5)[3]), where $\boldsymbol{w}_1^*$ are determined by solving in real-time at the current time instant j the following optimization problem:

$$\begin{aligned}
\min_{\boldsymbol{w}_1, \boldsymbol{s}} \; & \sum_{i=j+1}^{j+N} (\Psi_i \boldsymbol{w}_1 - z_{d,i})^T \boldsymbol{Q}_i (\Psi_i \boldsymbol{w}_1 - z_{d,i}) + \boldsymbol{s}^T \boldsymbol{Q}_s \boldsymbol{s} \\
\text{s.t. } & \boldsymbol{l}_i \le \boldsymbol{A}_i \boldsymbol{w}_1 + \boldsymbol{I}_{3n} \boldsymbol{s} \le \boldsymbol{u}_i \; , \; i = j+1, ..., j+N \\
& ||\boldsymbol{s}||_1 \le \bar{\boldsymbol{s}} \\
& \boldsymbol{A}_j \boldsymbol{w}_1 = \boldsymbol{b}_j \\
& \boldsymbol{A}_L \boldsymbol{w}_1 = \boldsymbol{b}_L
\end{aligned} \tag{9}$$

The cost function comprises two terms. The first one penalizes the error between the MP generated position and velocity $\Psi_i \boldsymbol{w}_1$, where $\boldsymbol{\Psi}_i = [\boldsymbol{\Phi}_i^T \; \dot{\boldsymbol{\Phi}}_i^T]^T$, and the desired one $z_{d,i} = [\boldsymbol{y}_{d,i}^T \; \dot{\boldsymbol{y}}_{d,i}^T]^T$, which is produced on-line by the DMP using Eqs. (3), (4). Matrix $\boldsymbol{Q}_i = \text{blkdiag}((1-\lambda)\boldsymbol{I}_n, \lambda \boldsymbol{I}_n)$, discriminates between optimizing the path ($\lambda = 0$) or the velocity profile ($\lambda = 1$). To endow the optimizer with greater flexibility in finding feasible solutions we consider the kinematic limits as hard bounds and introduce the lower and upper soft limits $\boldsymbol{l}_i$, $\boldsymbol{u}_i \in \mathbb{R}^{3n}$ for position, velocity and acceleration within the hard limits. We want to preferably operate within the soft limits and only exceed them if infeasibility would be inevitable otherwise. To this end, we introduce the relaxation variables $\boldsymbol{s} = [\boldsymbol{s}_p^T \; \boldsymbol{s}_v^T \; \boldsymbol{s}_a^T]^T \in \mathbb{R}^{3n}$, which are the maximum position, velocity and acceleration soft limit violations in the current horizon. To forbid the violation of the hard limits, the relaxation variables are bounded by the margin between the soft and hard limits $\bar{\boldsymbol{s}} = [\bar{s}_p \boldsymbol{1}_{1\times n} \; , \; \bar{s}_v \boldsymbol{1}_{1\times n} \; , \; \bar{s}_a \boldsymbol{1}_{1\times n}]^T$, through $||\boldsymbol{s}||_1 \le \bar{\boldsymbol{s}}$. If the soft limits are exceeded, we want the solution to ultimately return back within them. To this end we penalize the relaxation variables through the second term in the cost function $\boldsymbol{s}^T \boldsymbol{Q}_s \boldsymbol{s}$. Constraints regarding position, velocity and acceleration soft limits are introduced via $\boldsymbol{l}_i \le \boldsymbol{A}_i \boldsymbol{w}_1 + \boldsymbol{I}_{3n}\boldsymbol{s} \le \boldsymbol{u}_i$, with $\boldsymbol{A}_i \triangleq \boldsymbol{A}(\sigma_i) = [\boldsymbol{\Phi}_i^T \; \dot{\boldsymbol{\Phi}}_i^T \; \ddot{\boldsymbol{\Phi}}_i^T]^T$, where we also insert the relaxation variables to allow violation of the soft limits when needed to avoid infeasibility. Finally, the initial state constraint, which ensures smooth generated positions and velocities and continuous accelerations, is enforced through $\boldsymbol{A}_j \boldsymbol{w}_1 = \boldsymbol{b}_j$, with $\boldsymbol{A}_j \triangleq \boldsymbol{A}(\sigma_j)$ and $\boldsymbol{b}_j = [\boldsymbol{y}_j^T \; \dot{\boldsymbol{y}}_j^T \; \ddot{\boldsymbol{y}}_j^T]^T$ being the current state, while the final state constraint is enforced by $\boldsymbol{A}_L \boldsymbol{w}_1 = \boldsymbol{b}_L$, with $\boldsymbol{A}_L \triangleq \boldsymbol{A}(\sigma_L)$ and $\boldsymbol{b}_L = [\boldsymbol{y}_{d,L}^T \; \dot{\boldsymbol{y}}_{d,L}^T \; \ddot{\boldsymbol{y}}_{d,L}^T]$ being the desired final state. Similarly we can add additional constraints, such as via-points or obstacle constraints as we show in the simulations of Sect. 4.2. Henceforth, we will refer to the online approach as $\overline{DMP}^*$.

Problem (9) resembles MPC (Model Predictive Control), with the major difference being that in our case we have an analytic expression of the system's state, instead of using state equations in the form of a dynamical system. The latter is viable thanks to our DMP formulation (Sidiropoulos & Doulgeri, 2021), which is crucial for allowing us to employ large time steps Δt within the points of the optimization horizon to make the optimizer more far-sighted. This is in contrast to the classical DMP formulation, which would require numerical integration, increasing considerably the computational burden. The use of truncated kernels $\bar{\boldsymbol{\phi}}$ is also important as it increases the sparsity of the problem, allowing us to use larger horizons N while still obtaining the solution in real-time. Finally, in addition to having a large N, the relaxation variables further reinforce the optimizer's flexibility in finding feasible solutions. Details are deferred in Appendix A, where simulations studies make more palpable the above claims.

Remark 2 In order for the soft limits exceedance cost to be comparable or even greater than the tracking cost, so that solutions within the soft limits are favoured, a reasonable heuristic is to choose the gains in $\boldsymbol{Q}_s = \text{blkdiag}(q_p \boldsymbol{I}_n, \; q_v \boldsymbol{I}_n, \; q_a \boldsymbol{I}_n)$ as $q_k \ge N||[\boldsymbol{l}_{k,i}^T, \boldsymbol{u}_{k,i}^T]||_\infty / \bar{s}_k$

[3] The k-th truncated Gaussian is $\bar{\phi}_k(\sigma) = \begin{cases} \phi_k(\sigma) & , \phi_k(\sigma) > 10^{-6} \\ 0 & , \text{otherwise} \end{cases}$. The derivative is approximated by $\dot{\bar{\phi}}_k(\sigma) = -2h_k(\sigma - c_k)\bar{\phi}_k(\sigma)\dot{\sigma} \approx \dot{\phi}_k(\sigma)$ for small values of the truncation threshold.

for $k \in \{p, v\}$ for position, velocity and $q_a \geq 0.1N||[\boldsymbol{l}_{a,i}^T, \boldsymbol{u}_{a,i}^T]||_\infty/\bar{s}_a$ for acceleration (the 0.1 multiplier is to account for the larger magnitudes that accelerations usually have compared to position/velocity).

4.2 Simulations

In this section we carry out several simulations, demonstrating the results of the proposed optimization for generalizing a DMP trajectory, while changing the target, time duration, introducing via-points on-line, or avoiding obstacles, under kinematic constraints. We also compare our approach with two alternatives methods for imposing kinematic constraints on a DMP, i.e. incorporation of repelling forces in the DMP model, generated from artificial potential fields, for avoiding the kinematic limits and apply MPC directly on the DMP model's state equations as detailed in Appendix B. In all cases, we set $N = 10$, $\Delta t = 100$ ms, i.e. the optimizer sees ahead around 1 sec. The margin between soft and hard limits is 0.02 m for position, 0.1 m/s for velocity and 0.5 m/s^2 for acceleration. The gains in $\boldsymbol{Q}_s$ were chosen $[10^5\ 100\ 1]$ in the spirit of Remark 2 (despite small variations in the kinematic bounds, we have more or less that $q_p \geq 10^3$, $q_v \geq 10$ and $q_a \geq 1$)

To solve problem given in Eq. (9), we collect the optimization variables in $\boldsymbol{x} = [\boldsymbol{w}_1^T \boldsymbol{s}^T]^T$, and rewrite it as a QP:

$$\begin{aligned} \min_x\ & x^T \mathbf{H} x + 2\mathbf{q}^T x \\ \text{s.t.}\ & l \leq \mathbf{A}x \leq u \\ & \mathbf{A}_{eq} x = \mathbf{b}_{eq} \end{aligned} \tag{10}$$

where $\boldsymbol{H} = \text{blkdiag}(\sum_{i=j+1}^{j+N} \Psi_i^T \boldsymbol{Q}_i \Psi_i\ ,\ \boldsymbol{Q}_s)$, $\boldsymbol{q} = [-\sum_{j+1}^{j+N} \boldsymbol{z}_{d,i}^T \boldsymbol{Q}_i \Psi_i\ ,\ \boldsymbol{0}_{1\times 3n}]^T$, $\boldsymbol{A}_e q = \begin{bmatrix} \boldsymbol{A}_j & \boldsymbol{0}_{3n\times 3n} \\ \boldsymbol{A}_L & \boldsymbol{0}_{3n\times 3n} \end{bmatrix}$, $\boldsymbol{b}_{eq} = \begin{bmatrix} \boldsymbol{b}_j \\ \boldsymbol{b}_L \end{bmatrix}$, $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{j+1} & \boldsymbol{I}_{3n} \\ \vdots & \vdots \\ \boldsymbol{A}_{j+N} & \boldsymbol{I}_{3n} \\ \boldsymbol{0}_{3n\times nK} & \boldsymbol{I}_{3n} \end{bmatrix}$, $\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_{j+1} \\ \vdots \\ \boldsymbol{u}_{j+N} \\ \bar{\boldsymbol{s}} \end{bmatrix}$ and similarly for $\boldsymbol{l}$. We employ sparse matrices and use the OSQP library (Stellato et al., 2020) to solve the above problem on-line at each control cycle.

4.2.1 Variable target and variable time duration

To test the $\overline{DMP}^*$ for variable target (final) position, we assume that the target $\boldsymbol{g}$ changes from the initial demonstrated target $\boldsymbol{g}_d$ to a new target $\boldsymbol{g}'$ based on the dynamics:

$$\dot{\boldsymbol{g}} = 5(\boldsymbol{g}' - \boldsymbol{g}) \tag{11}$$

and is updated in the simulation at a rate of 30 Hz, to emulate the delay from a vision sensor that tracks the target. For testing the variable time duration, we assume the following dynamics:

$$\dot{T}_f = 100(T_f' - T_f) \tag{12}$$

where T_f is the current and T_f' the new desired motion duration. For enforcing the desired time duration we employ the following canonical system:

$$\ddot{\sigma} = \begin{cases} 50(\dot{\sigma}_d - \dot{\sigma}), & \sigma < 1 \\ -400\dot{\sigma} - 800(\sigma - 1), & \text{otherwise} \end{cases} \tag{13}$$

where the desired speed of evolution of the phase variable is $\dot{\sigma}_d = (1 - \sigma)/(T_f - t)$, so that the total time duration is T_f.

The results are depicted in Fig. 3, where we also plot the response of the DMP without constraints. In this and all subsequent figures the soft limits are plotted with dashed magenta and the hard limits with dashed gray line. On the left, we see the results for changing the motion duration T_f from 7 to 5 sec at time instant $t = 2$ sec. On the right we view the results for updating online the target $\boldsymbol{g}$, with an update rate of 30 Hz. We can see that $\overline{DMP}^*$ manages to produce a solution within the soft limits, whereas the DMP not only exceeds even the hard limits, but also produces large accelerations. This is typical in DMP for changing the time duration or the target. These issues can be alleviated by filtering the new time duration or target, but the gain of such filters would require tuning. More importantly though, one cannot provide guarantees that the magnitude of the produced acceleration will be within specific bounds and that the DMP will both reach the target and have the desired time duration (the latter two can be greatly affected by the gains used in filtering). In this scenario, the merit of $\overline{DMP}^*$ is that such issues are automatically handled by the optimizer.

4.2.2 Via-points

To illustrate the adaptation to via-points on-line, we set 3 via-points, each one introduced in the $\overline{DMP}^*$ 1.75 sec before its time instant. For comparison, we also simulate the DMP*, providing it with all via-points offline. The results are presented in Fig. 4, with the Cartesian path on the left and the trajectory along the z-axis on the right. The trajectory produced by $\overline{DMP}^*$ stays within the soft limits and after it has passed from all via-points, its shape resembles that of the DMP. It is also interesting to note that the solution produced by the $\overline{DMP}^*$ is quite close to that of DMP^*.

4.2.3 Obstacle avoidance

For avoiding obstacles, we assume that the obstacles are expressed w.r.t. to a bounding ellipsoid, although capsules or polytopes can also be handled similarly. We consider

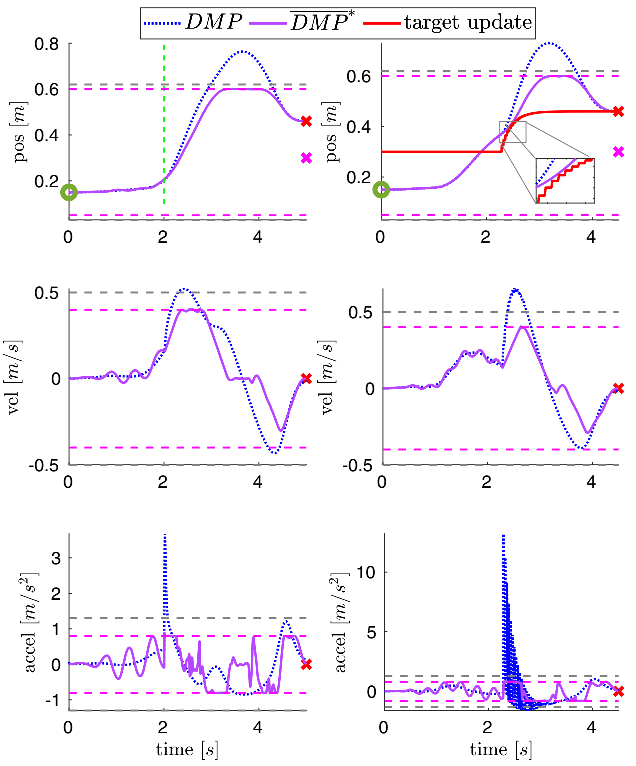


Fig. 3 Left: The time duration of the motion is changed from 7 sec to 5 sec at $t = 2$ sec. Right: The target changes online, with an update rate of 30 Hz

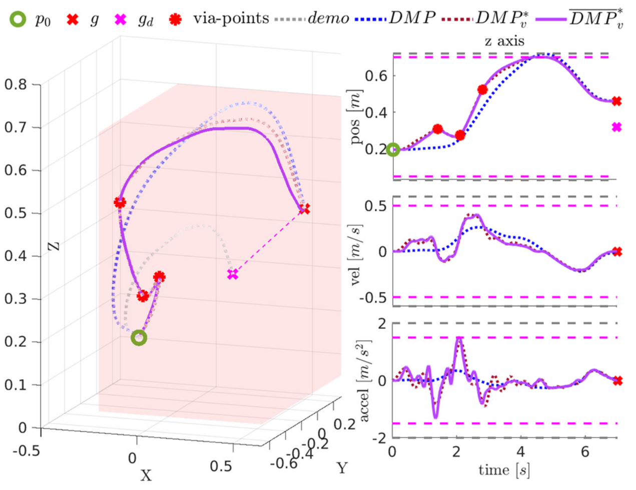


Fig. 4 Via-points. Left: Cartesian path. Right: trajectory along z-axis

that obstacle j is expressed as $\mathcal{E}_j = \{x \in \mathbb{R}^n : (\boldsymbol{x} - \boldsymbol{c}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x} - \boldsymbol{c}_j) = 1\}$ and assume that obstacles don't overlap. We take each predicted point $\boldsymbol{y}_i$ of $\overline{DMP}^*$ along the prediction horizon and include a plane constraint for time-step σ_i if $(\boldsymbol{y}_i - \boldsymbol{c}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{y}_i - \boldsymbol{c}_j) < 1.1$ (we want the constraint to be activated even when a point is outside but close to the obstacle, since the marginal value 1 can cause chattering of the solution between two consecutive optimization horizons). The plane constraint is defined by calculating $y_{\mathcal{E}} = \boldsymbol{c}_j + (\boldsymbol{y}_i - \boldsymbol{c}_j)/\sqrt{(\boldsymbol{y}_i - \boldsymbol{c}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{y}_i - \boldsymbol{c}_j)}$, which is the point on the surface of $\mathcal{E}_j$ along the ray from $\boldsymbol{c}_j$ to $\boldsymbol{y}_i$ and taking the tangent plane of $\mathcal{E}_j$ on $y_{\mathcal{E}}$, with normal $\boldsymbol{n}_{\mathcal{E}} = \nabla_{\boldsymbol{x}}((\boldsymbol{x} - \boldsymbol{c}_j)^T \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x} - \boldsymbol{c}_j))_{\boldsymbol{x}=y_{\mathcal{E}}}$. Hence, the constraint is $\boldsymbol{n}_{\mathcal{E}}^T(\boldsymbol{\Phi}(\sigma_i)\boldsymbol{w}_1 - \boldsymbol{y}_{\mathcal{E}}) \geq 0$. This process adds at most N additional linear constraints in Eq. (9). Simulation results with 2 obstacles are given in Fig. 5, both for optimizing the path (purple) and for optimizing the velocity (green). In both cases, the solution had to momentarily exceed the soft limits, mainly to avoid the obstacles, but it always remained within the hard limits. It can also be observed that optimizing the velocity retains the shape of the trajectory, while optimizing the position produces a path that is closer to the unconstrained one, as expected.

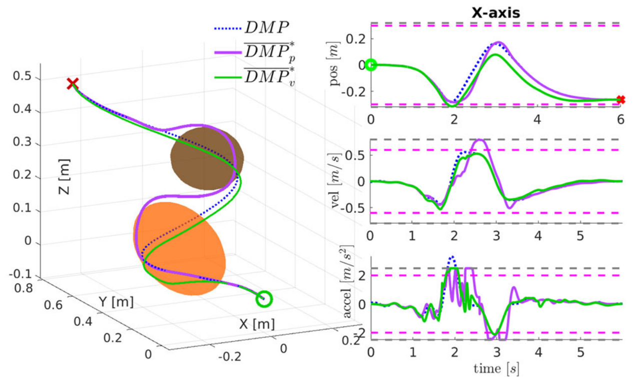


Fig. 5 Left: Optimal path for optimizing position (purple) or optimizing velocity (green). Right: Trajectory along x-axis (Color figure online)

4.2.4 Comparison with repelling forces

The results of the comparison between $\overline{DMP}^*$ (for optimizing the position) and DMP with repelling forces (DMP-RF) are plotted in Fig. 6, with the Cartesian path on the left and the trajectories along the x-axis on the right. We can see that the DMP-RF position trajectory (yellow line) is very close to that of $\overline{DMP}^*$ (purple line). However, it generates very large accelerations that considerably exceed hard limits. Further tuning of the repelling forces open parameters could help reduce the acceleration, but at the expense of larger deviations from the nominal trajectory. In contrast, $\overline{DMP}^*$ produces a feasible trajectory, without any need for tuning, which only exceeds slightly the soft velocity and acceleration limits towards the end so as to reach the target in time with zero velocity and acceleration. Also, it provides the option of optimizing either the position or the velocity profile as we have already shown, while the latter is not attainable with repelling forces.

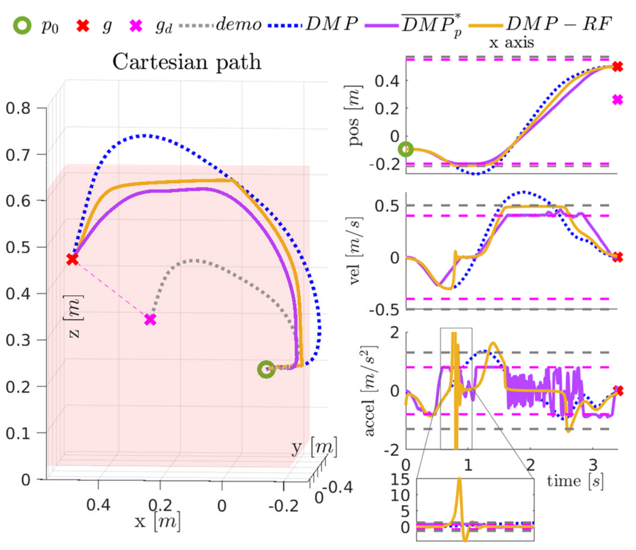


Fig. 6 $\overline{DMP}^*$ vs DMP with repelling forces. Left: Cartesian path. Right: trajectory along x-axis

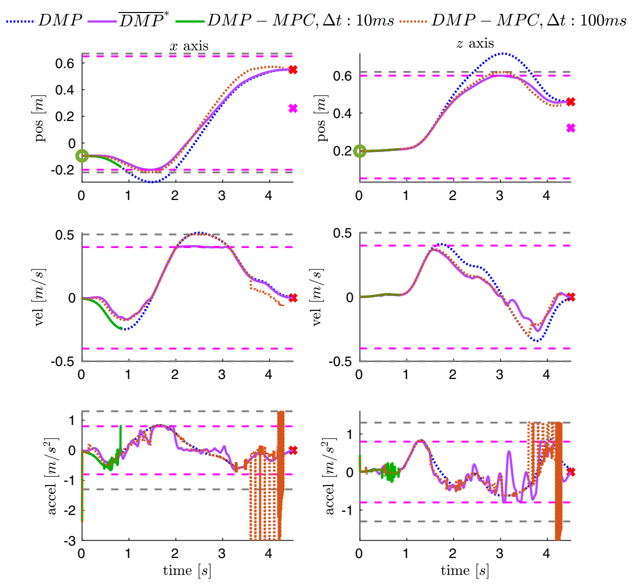


Fig. 7 $\overline{DMP}^*$ vs MPC based on the DMP model. Left: trajectory along x-axis. Right: trajectory along z-axis

4.2.5 Comparison with MPC

The comparison between $\overline{DMP}^*$ and MPC based on the DMP model (DMP-MPC) is shown in Fig. 7. In both methods, we optimize w.r.t. the velocity and use a horizon of $N = 10$ points. The trajectory along the x axis is plotted on the left, and along the z-axis on the right. We tested MPC with time-step $\Delta t = 10$ ms (green line) and $\Delta t = 100$ ms (dashed light brown line). With a small time-step, the DMP-MPC fails to find a feasible solution at $t = 0.8$ sec, as the position along the x-axis has reached the lower limit and the velocity is still negative. This is related to the fact that a small time-step would require a large horizon N so that the optimizer can forestall such infeasible situations. Using MPC with a larger time-step, $\Delta t = 100$ ms, allows the MPC to bypass this infeasibility. Nonetheless, the acceleration is much more noisier and also the acceleration limits can be easily violated. This is because, if Δt is relatively large, the acceleration constraint in Eq. (17) can be a poor estimate of the actual acceleration. Moreover, the MPC fails also in this case to find a feasible solution along the z-axis at $t = 4.3$ sec, because it cannot reach the target position with zero velocity and acceleration within the specified time duration. In contrast $\overline{DMP}^*$ manages to find a solution, since the trajectory can be analytically retrieved at each time instant thanks to the new DMP formulation (Sidiropoulos & Doulgeri, 2021), regardless of how large Δt may be. Moreover, due to the continuity of the Gaussian kernels, the optimization at the N discrete points of the current horizon, affects also their neighboring points, that will be considered in future optimization windows.

As a final remark, notice that all trajectories generated by $\overline{DMP}^*$ have smooth positions and velocities and continuous accelerations, thanks to the equality constraint $\boldsymbol{A}_j \boldsymbol{w}_1 = \boldsymbol{b}_j$ in Eq. (9).

5 Experimental results

In this section we apply the proposed online DMP optimization framework in two practical scenarios. The first one is a handover of a bottle to a human, highlighting how the proposed method can handle online variations of the target and time duration of a motion, while satisfying kinematic constraints. The second experiment involves placing an object inside a bin with obstacles being dynamically introduced on the scene. This scenario will showcase the ability of the proposed method to adjust online the DMP trajectory by introducing via-points, specified online according to the obstacles, while again respecting the kinematic constraints. In each experimental scenario, the trajectory pattern that is used is demonstrated using kinesthetic guidance and is encoded in a DMP. We use a ur5e robot, with control cycle $T_s = 2$ ms, with a RG2FT gripper at its wrist. We also use a realsense2 camera and apriltags (Wang & Olson, 2016) for tracking the position of objects relevant to each task. In all experiments, we choose for the $\overline{DMP}^*$ $K = 30$ kernels, optimization horizon $N = 10$ with $\Delta t = 100$ ms, relaxation limits $[0.03,\ 0.1,\ 1.0]$ and relaxation cost weightings $[10^5,\ 100,\ 1]$ for position, velocity and acceleration respectively. In the handover, we optimize w.r.t. the velocity profile ($\lambda = 1$) and in the placing task we optimize the position ($\lambda = 0$). In all experiments, the average running time for the optimization was around 1 ms.

The handover scenario is depicted at the top of Fig. 8. The position of the apriltag that is attached at the human's hand is used as the target position for the handover. Therefore the target varies, since the human moves his hand. Moreover, we modify the motion duration T_f as follows:

$$T_f(t) = \begin{cases} t + 4||\boldsymbol{p}(t) - \boldsymbol{g}(t)||, & ||\boldsymbol{p}(t) - \boldsymbol{g}(t)|| > 0.25 \\ \min\{t + \dfrac{0.25}{||\boldsymbol{p}(t) - \boldsymbol{g}(t)||}, 4||\boldsymbol{p}(0) - \boldsymbol{g}(0)||\}, & \text{o/w} \end{cases}$$

where $\boldsymbol{p}(t)$ is the current position of the robot's end-effector and $\boldsymbol{g}(t)$ is the apriltag's position (plus a constant offset, so that the target is on the human's palm and not its back). This adaptive law decreases the duration as the distance between robot and human gets smaller, but after a threshold (0.25 m) the duration increases again so that the robot decelerates when getting very close to the human to make him feel more comfortable. For enforcing the new time duration, the canonical system given in Eq. (13) is used. More sophisticated approaches for estimating the handover position and time duration could be used, instead of the simple heuristics we adopt here, but this is beyond the scope of our work. Here we want to showcase how the $\overline{DMP}^*$ can be employed in a scenario that involves online adjustments of the target and the time duration of a motion, while respecting the enforced kinematic limits. The experimental results are depicted in Fig. 9a. On the left, the Cartesian path of the $\overline{DMP}^*$ is plotted with purple line and the human's hand position (target) with red line. The DMP path, without constraints, is also plotted with blue line. On the right, the trajectory along the x axis is plotted, which is the antipodal axis between robot and human. The bottom right plot shows the evolution of the motion duration T_f. The target makes some small steps (see magnified subplot on the top right subplot Fig. 9a), which is due to camera's frame rate and sensor noise. This in turn introduces discontinuities in the DMP trajectory (blue line) and the acceleration constraints are violated. Still, the $\overline{DMP}^*$ produces a continuous trajectory within the soft limits.

The setup for the second experiment is depicted at the bottom of Fig. 8. In this scenario, the robot has to place a small cube inside a bin. During execution, two obstacles are introduced in the scene. The first one is present from the start of the motion and the second one is introduced dynamically while the robot is moving. For the first obstacle, the 4 orange via-points, relative to its apriltag, are introduced, shown in Fig. 8, so that the robot passes over the obstacle to avoid it. For the second obstacle, the 3 red via-points are associated with its apriltag. In this case we want to locally modify the DMP trajectory so that the robot pushes aside the obstacle and the area in front of the bin is cleared for the placing to ensue. In general, the via-points that modify locally the DMP trajectory could be supplied by a higher level perception system. This is beyond the scope of this work, so we have manually

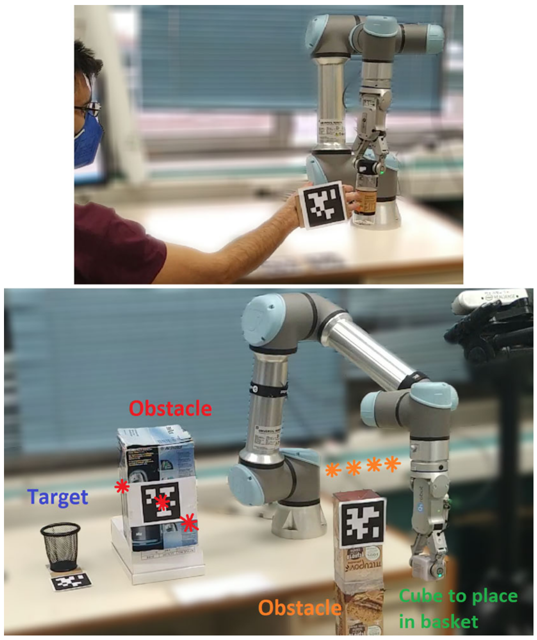
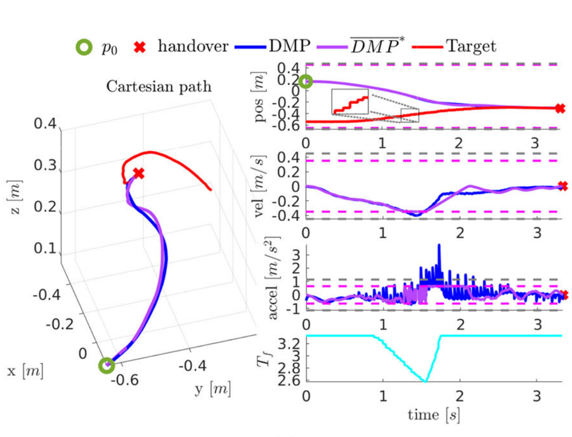


Fig. 8 Experimental scenarios. Top: handover. Bottom: placing of cube inside bin

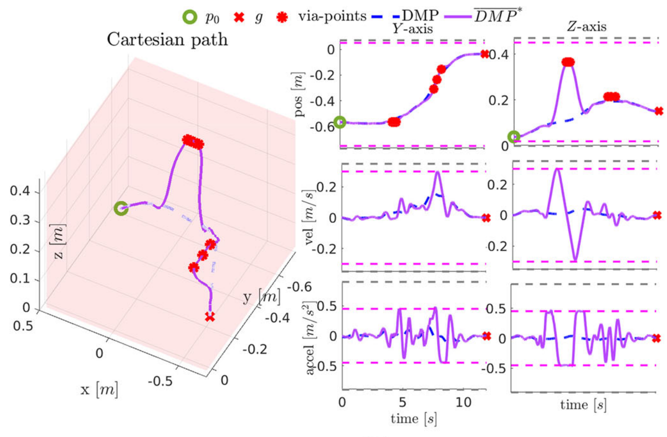
prespecified for each apriltag the number and relative positions of the via-points w.r.t. the apriltag's position. At the left of Fig. 9b the path executed by $\overline{DMP}^*$ is shown with purple, with the via-points visualized with red asterisks. The DMP path without the via-points is also depicted with blue dashed line. Without the via-points, the first obstacles would have been thrown down and the second one would have been pushed towards the bin, jeopardizing the successful placing of the cube. At the middle of Fig. 9b, we see the adjustment along the z-axis to bypass the first obstacle and at the right, the movement along the y-axis to push aside the second obstacle. We can also see that the hard limits are respected, with very small violations of the soft limits in the velocity and acceleration. These small violations occur momentarily during the time window that the trajectory adjustment takes place and are instantly minimized. A video with the experiments can be found in https://youtu.be/21HvNpjpBGU, which also includes supplementary simulations and experiments on dynamic obstacle avoidance.

6 Conclusions

In this work, we presented a novel framework for generalizing a desired trajectory pattern, encoded using DMP, subject to kinematic constraints. The proposed framework can handle dynamic adjustments of the DMP trajectory, due to target, motion duration changes, incorporation of via-points and/or obstacles, while satisfying kinematic constraints. Comparative simulations demonstarte the efficacy of our framework,



(a)



(b)

Fig. 9 **a** Handover experiment. Left: Cartesian path. Right: trajectory along the x-axis. **b** Placing experiment. Left:Cartesian path. Right: trajectories along y and z axes

compared to other relevant approaches. Experimental results were also carried out, that validate the applicability and effectiveness of our approach. As a future work we aim at incorporating also constraints for the entire robot arm, apart from the robot's end-effector.

Funding Open access funding provided by HEAL-Link Greece.

Declarations

Conflict of interest No funding was received for conducting this work and there isn't any kind of conflict of interest.

Appendix A: Impact of on-line optimization parameters

Here we present simulations that provide further insight on the impact of the parameters of the proposed on-line optimization method.

The effect of the prediction time step Δt is illustrated on the left of Fig. 10, where we perform velocity profile optimization for $N = 10$ and $\Delta t \in \{10, 100\}$ ms and also $N = 50$ and $\Delta t = 10$ ms. The optimization with small prediction time-step (green line) reaches an infeasible state at 2.5 sec and appears to optimize the position instead of the velocity profile. Both these issues relate to the small prediction time-step which renders the optimization "short-sighted". Thus it neither anticipates possible future infeasible states nor captures the velocity profile. In contrast, with the proposed larger time-step ($\Delta t = 100$ ms, purple line), the velocity profile is preserved and the optimizer does not stumble in any infeasible state. It is also interesting to note that the latter is very close to the ideal case of a large optimization horizon with small time step ($N = 50$ $\Delta t = 10$ ms, dashed brown line), which is however not practical due to the excessive computational overhead.

It is important to emphasize that using such a large time-step is made possible due to the DMP structure given in Eq. (1) which allows to obtain the position, velocity and acceleration at any time instant without needing to explicitly perform any numerical integrations in contrast to the original DMP formulation from Ijspeert et al. (2013) or the QP-DMP (Cardoso et al., 2015). It should also be noted that a very large time step Δt may not be favourable. This is because we consider a sparse (w.r.t. time) set of N points, so if the points are very far apart temporarily, the optimization will not be able to capture the shape of the trajectory, or even prevent imminent infeasibilities.

In the presence of relatively strict constraints infeasibility may occur, even if a relatively large Δt is used. This is where the relaxation variables come into play, making the optimization more flexible and effective in bypassing infeasible states. To highlight the utility of relaxation variables, we compare the use of soft limits and relaxations variables with the case where only the hard limits are used. In all cases we optimize w.r.t. the position and set $\Delta t = 100$ ms. The results are shown on the right of Fig. 10, where the constraints are more strict, in the sense that the unconstrained trajectory violates both

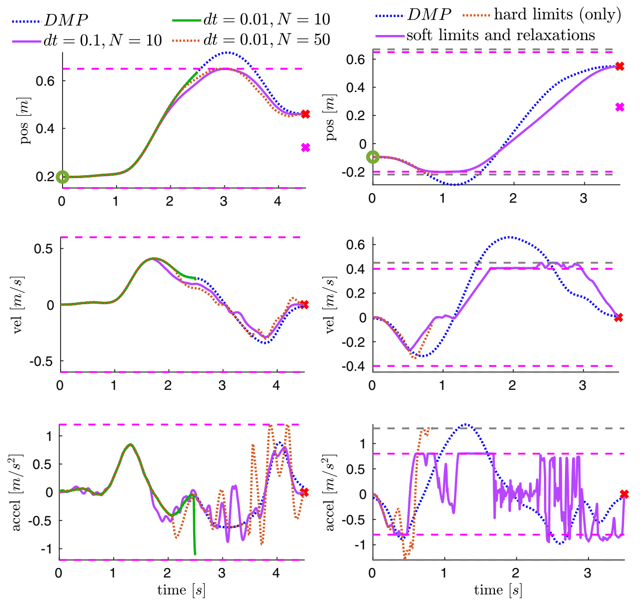


Fig. 10 Effect of proposed modifications on the on-line DMP optimization. Left: effect of prediction time-step. Right: effect of relaxation variables

position, velocity and acceleration limits. Using soft limits and relaxations we obtain a feasible solution (purple line) that exceeds the soft limits slightly in the position and velocity only in some optimization steps. Using directly the hard limits the optimizer stumbles upon an infeasible state at 0.84 sec.

Finally, we evaluate the effect that truncated kernels have on the average running time for solving problem Eq. (10) and on the sparsity of the cost function matrix H and the inequality constraints matrix A for different values of the problem size, i.e. the number of DoFs n, the kernels K and optimization horizon N. We denote the sparsity of the a matrix P as sp(P), which is the number of non-zero elements over the total number of elements. The running time is measured by executing the optimization in C++ code on a desktop pc with an Intel ®Core ™i7-9700 processor. For testing, we consider the trajectory along the z-axis and the constraints that were used in Sect. 3 and is shown in Fig. 1b. This trajectory is replicated n times (one for each DoF). We chose this DMP trajectory, since it violates both position, velocity and acceleration constraints, hence it will be more challenging to the solver. The results are imprinted on Table 1. For relative small problem size (1st case), the improvement with truncated kernels is relatively small. However, as the problem size increases, the impact of truncated kernels becomes more evident. For instance, in the 3rd case, which is a typical example of the problem size that one would encounter in practice, the running time is around 1 ms. Even in the last case, where the problem size is relatively large, the running time stays below 2 ms. In contrast, without the use of truncated kernels, the running time approaches 3 ms for the 3rd case and may even become prohibitive for larger problem sizes (case 4). The effect of truncated kernels is also reflected in the problem's sparsity, which makes the problem at least 3 times sparser for larger problem sizes (cases 3 and 4).

Appendix B: DMP with repelling forces and MPC

Incorporation of repelling forces, generated from artificial potential fields, in the DMP model for avoiding the kinematic limits can be realized as follows:

$$\ddot{y} = -Ky - D\dot{y} + f(s) + k_p f_p + k_v f_v \tag{14}$$

where $f(s) = Ky_s + D\dot{y}_s + \ddot{y}_s$, f_p, f_v are the repelling forces for position and velocity limits respectively and $k_p, k_v > 0$ are tunable gains. Notice that in this case it is not easy to enforce acceleration constraints, since we would have to increase the order of the system Eq. (14) and also consider the stability of the system's characteristic polynomial. Different types of repelling forces have been proposed in the literature (Gams et al., 2009; Dahlin & Karayiannidis, 2020). We opted for the repelling force proposed in Kastritsi et al. (2019) as it distorts less the trajectory away from the constraints, thus serves as a better competitor against our method. For one DoF denoted by y and limit y_l the repelling force is given

Table 1 Average running time (± standard deviation) for solving problem (10) and sparsity of the cost function matrix H and the inequality constraints matrix A for different optimization horizon N, kernels K and DoFs n

Prob. size	Truncated (10^{-6})			Non-truncated		
N K n	sp (H)	sp (A)	Runtime (ms)	sp (H)	sp (A)	Runtime (ms)
8 20 1	38%	45.7%	0.21 ± 0.02	76.2%	82.2%	0.31 ± 0.02
10 25 3	9.6%	12.6%	0.51 ± 0.05	26.7%	28.4%	1.05 ± 0.05
10 30 6	3.9%	5.4%	0.94 ± 0.18	13.8%	14.4%	2.71 ± 0.76
14 40 8	2.3%	3.2%	1.71 ± 0.25	10.8%	11.2%	7.70 ± 0.81

by:

$$f_{rep}(y, y_l) = \frac{-2\ln(1-\psi)}{d_0^2(1-\psi)}(d_0 - |p - p_l|)\frac{p - p_l}{|p - p_l|} \quad (15)$$

for $|p - p_l| \leq d_0$ and is zero otherwise, where $\psi = \frac{(|p-p_l|-d_0)^2}{d_0^2}$ and $d_0 > 0$ is the distance after which the repelling force is activated. Based on Eq.(15) we choose $f_p = f_{rep}(y, \underline{y}) + f_{rep}(y, \bar{y})$ and $f_v = f_{rep}(\dot{y}, \underline{\dot{y}}) + f_{rep}(\dot{y}, \bar{\dot{y}})$. The gains for the repelling forces were set to $f_p = 0.05$ and $f_v = 0.5$ and the distance below which the repelling forces are activated was $d_0 = 0.03$.

To apply MPC on the DMP model, we have to introduce the discrete state space equations of the DMP:

$$\boldsymbol{z}_{i+1} = \boldsymbol{F}_i \boldsymbol{z}_i + \boldsymbol{B}_i \boldsymbol{v}_i \quad (16)$$

where $\boldsymbol{z}_i = [\boldsymbol{y}_i^T \ \dot{\boldsymbol{y}}_i^T]^T$, $\boldsymbol{F}_i = \boldsymbol{I}_6 + \begin{bmatrix} \boldsymbol{0}_3 & \boldsymbol{I}_3 \\ -\boldsymbol{K} & -\boldsymbol{D} \end{bmatrix}\Delta t$, with Δt being the time-step and $\boldsymbol{B}_i = \begin{bmatrix} \boldsymbol{0}_3 \\ \boldsymbol{I}_3 \end{bmatrix}$. Denoting the optimization variables vector as $\boldsymbol{x} = [\boldsymbol{z}_{j+1} \dots \boldsymbol{z}_{j+N} \ \boldsymbol{v}_j \dots \boldsymbol{v}_{j+N-1}]$, the following optimization problem has to be solved at each time-step j:

$$\begin{aligned} \min_x & \sum_{i=j+1}^{j+N} (z_i - z_{d,i})^T \mathbf{Q}_i (z_i - z_{d,i}) \\ \text{s.t.} \quad & z_i = \mathbf{F}_{i-1} z_{i-1} + \mathbf{B}_{i-1} v_{i-1} \\ & \underline{z}_i \leq z_i \leq \bar{z}_i \\ & \underline{\ddot{\mathbf{y}}}_i \leq \mathbf{D}_a (z_{i+1} - z_i) \leq \bar{\ddot{\mathbf{y}}}_i \end{aligned} \quad (17)$$

where $\boldsymbol{D}_a = [\boldsymbol{0}_3 \ \boldsymbol{I}_3/\Delta t]$.

References

Ben Amor, H., Neumann, G., Kamthe, S., Kroemer, O., Peters, J. (2014). Interaction primitives for human-robot cooperation tasks. In *2014 IEEE International Conference on Robotics and Automation (ICRA),* pp. 2831–2837 https://doi.org/10.1109/ICRA.2014.6907265

Billard, A., Calinon, S., Dillmann, R., Schaal, S. (2008). In textit-Siciliano, B., Khatib, O. (eds.) Robot Programming by Demonstration, pp. 1371–1394. Springer. https://doi.org/10.1007/978-3-540-30301-5_60

Buizza Avanzini, G., Zanchettin, A. M., & Rocco, P. (2018). Constrained model predictive control for mobile robotic manipulators. *Robotica, 36*(1), 19–38. https://doi.org/10.1017/S0263574717000133.

Cardoso, C., Jamone, L., Bernardino, A. (2015) A novel approach to dynamic movement imitation based on quadratic programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA),* pp. 906–911 . https://doi.org/10.1109/ICRA.2015.7139285

Dahlin, A., & Karayiannidis, Y. (2020). Adaptive trajectory generation under velocity constraints using dynamical movement primitives. *IEEE Control Systems Letters, 4*(2), 438–443. https://doi.org/10.1109/LCSYS.2019.2946761.

Duan, A., Camoriano, R., Ferigo, D., Calandriello, D., Rosasco, L., Pucci, D. (2018). Constrained dmps for feasible skill learning on humanoid robots. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids),* pp. 1–6 https://doi.org/10.1109/HUMANOIDS.2018.8624934

Frank, F., Paraschos, A., & vander Smagt, P., Cseke, B. (2021). Constrained probabilistic movement primitives for robot trajectory adaptation. *IEEE Transactions on Robotics*. https://doi.org/10.1109/TRO.2021.3127108.

Gams, A., Ijspeert, A. J., Schaal, S., & Lenarcic, J. (2009). On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots, 27,* 3–23. https://doi.org/10.1007/s10514-009-9118-y.

Hoffmann, H., Pastor, P., Park, D., Schaal, S. (2009). 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.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., & Schaal, S. (2013). Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation, 25*(2), 328–373.

Kastritsi, T., Papageorgiou, D., Sarantopoulos, I., Doulgeri, Z., Rovithakis, G.A. (2019). Stability of active constraints enforcement in sensitive regions defined by point-clouds for robotic surgical procedures. In *2019 18th European Control Conference (ECC),* pp. 1604–1609 https://doi.org/10.23919/ECC.2019.8796278

Kober, J., Mülling, K., Krömer, O., Lampert, C.H., Schölkopf, B., Peters, J. (2010). Movement templates for learning of hitting and batting. In *2010 IEEE International Conference on Robotics and Automation* pp. 853–858 https://doi.org/10.1109/ROBOT.2010.5509672

Koutras, L., Doulgeri, Z. (2020). 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 . https://doi.org/10.1109/RO-MAN47096.2020.9223500

Krug, R., & Dimitrov, D. (2015). Model predictive motion control based on generalized dynamical movement primitives. *Journal of Intelligent & Robotic Systems, 77*(1), 17–35. https://doi.org/10.1007/s10846-014-0100-3.

Liang, Y., Li, W., Wang, Y., Xiong, R., Mao, Y., Zhang, J. (2021). Dynamic movement primitive based motion retargeting for dual-arm sign language motions. In *2021 IEEE International Conference on Robotics and Automation (ICRA),* pp. 8195–8201 https://doi.org/10.1109/ICRA48506.2021.9561120

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

Merckaert, K., Convens, B., Wu, C.-J., Roncone, A., Nicotra, M. M., & Vanderborght, B. (2022). Real-time motion control of robotic manipulators for safe human-robot coexistence. *Robotics and Computer-Integrated Manufacturing, 73,* 102223. https://doi.org/10.1016/j.rcim.2021.102223.

Mghames, S., Hanheide, M., Ghalamzan, E. A.(2020). 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 https://doi.org/10.1109/IROS45743.2020.9341728

Mülling, K., Kober, J., Kroemer, O., & Peters, J. (2013). Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research, 32*(3), 263–279. https://doi.org/10.1177/0278364912472380.

Papageorgiou, D., Doulgeri, Z. (2020). Learning by demonstration for constrained tasks. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN),* pp. 1088–1093 https://doi.org/10.1109/RO-MAN47096.2020.9223579

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

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

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation, 12*(4), 637–672. https://doi.org/10.1007/s12532-020-00179-2.

Wang, J., Olson, E. (2016). AprilTag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 4193–4198. IEEE, https://doi.org/10.1109/IROS.2016.7759617

Wen, Y., R. Pagilla, P. (2021). Path-constrained optimal trajectory planning for robot manipulators with obstacle avoidance. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Accepted).