Journal of Intelligent & Robotic Systems (2021) 101: 79
https://doi.org/10.1007/s10846-021-01344-y

REGULAR PAPER

Dynamic Movement Primitives: Volumetric Obstacle Avoidance Using Dynamic Potential Functions

Michele Ginesi[1] · Daniele Meli[1] · Andrea Roberti[1] · Nicola Sansonetto[1] · Paolo Fiorini[1]

Received: 1 July 2020 / Accepted: 8 February 2021 / Published online: 30 March 2021

Abstract

Obstacle avoidance for Dynamic Movement Primitives (DMPs) is still a challenging problem. In our previous work, we proposed a framework for obstacle avoidance based on superquadric potential functions to represent volumes. In this work, we extend our previous work to include the velocity of the system in the definition of the potential. Our formulations guarantee smoother behavior with respect to state-of-the-art point-like methods. Moreover, our new formulation allows obtaining a smoother behavior in proximity of the obstacle than when using a static (i.e. velocity independent) potential. We validate our framework for obstacle avoidance in a simulated multi-robot scenario and with different real robots: a pick-and-place task for an industrial manipulator and a surgical robot to show scalability; and navigation with a mobile robot in a dynamic environment.

Keywords Obstacle avoidance · Dynamic movement primitives · Learning from demonstration

1 Introduction

Robots are now used in complex scenarios, ranging from industrial and manufacturing processes to aerospace and health care. As their involvement in common human tasks increases, adaptability and reliability at the motion planning level are often required, and imitation of human behavior often helps in this direction.

Standard motion planning techniques, such as splines, potentials, and others [20, 21, 30, 32], work well when an objective function has to be optimized (e.g. minimize the time of execution of the trajectory, or the energy consumption). A *Learning from Demonstration* (LfD) approach, is usually preferable if one needs to learn human gestures. In LfD, a human operator shows an example of trajectory or task execution, and parameters are learned for replication in different situations and environments. In last fifteen years, various LfD approaches (such as *Gaussian Mixture Models* [17], *Extreme Learning Machines* [3, 11], and others [1]) have been developed in order to replicate human gestures. These LfD techniques may require a huge amount of demonstrations to be properly trained, which can represent a bottleneck when many different motion primitives have to be learned (e.g., for productive and cost reasons in the industry).

A crucial problem for all motion planning strategies is obstacle avoidance. Methods for obstacle avoidance may either be position-dependent [4, 31] or velocity-dependent [5, 44], with the latter guaranteeing smoother trajectories, especially in the presence of moving agents and obstacles. In this paper, we focus on the obstacle avoidance problem within the Dynamic Movement Primitives (DMPs) framework [10, 14, 23, 36]. DMPs permit to learn a trajectory from just one demonstration. They encode the trajectory in a system of second-order linear Ordinary Differential Equation (ODE), where a forcing term is learned as a linear combination of predefined time-dependent functions. They are successfully used in many robotic scenarios, such as cloth manufacturing [16], reproduction of human walk for exoskeletons [12], and collaborative bimanual tasks [6]. Obstacle avoidance for DMPs has been successfully treated for point-like obstacles (e.g. [23] and [10]). On the other hand, volumetric obstacle avoidance has been treated in our previous work [8] using potential functions. Other approaches (e.g. [22, 28, 29,

✉ Michele Ginesi
michele.ginesi@univr.it

[1] Department of Computer Science, University of Verona, Strada le Grazie 15, 37134, Verona, Italy

37]) require multiple demonstrations with different types and sizes of the obstacles to learn the obstacle avoidance behavior.

In this work we improve our previous framework [8]. In particular, we introduce a new potential function. This new potential is velocity-dependent, and this allows to achieve smoother obstacle avoidance behaviors compared to static (i.e. dependent only on position) potentials. Moreover, we will show that our approach results in trajectories that deviate less from the desired behavior than other frameworks. Both these properties are desirable in any obstacle avoidance method. In particular, a smooth trajectory is preferable since reduces both the robot's energy consumption and the damage done to the actuators [7]. Moreover, remaining closer to the desired trajectory avoid strong reduction of the available workspace [8].

We validate our approach in a simulated multi-robot coordination scenario, where three mobile robots have to reach pre-defined targets while avoiding each other and obstacles in the scene. We also show the generality of our frameworks as applied to different real robotic scenarios. In detail, we test a pick-and-place task with an industrial manipulator, combining DMP-level obstacle avoidance with collision-free inverse kinematic computation. We then show that the scalability of DMPs is preserved with our framework, replicating the pick-and-place task on a smaller setup with a bi-manual surgical robot. Finally, we show the reactivity of our approach with a mobile robot in a dynamic scene with moving obstacles to be detected by an RGB-D camera.

In Section 2 we recall the theory of DMPs, focusing, in Section 2.1, on the existing methods to treat obstacle avoidance. Then, in Section 3 we present our new dynamic potential function. In Section 4 we show our results: in Section 4.1 we analyze the computational time of the presented methods, in Section 4.2 we compare our new method to the state of the art, showing that our novel method results in a trajectory that is both smoother and it remains to the learned one; in Section 4.3 we compare our previous static potential for volumes with the new dynamic one, in a scenario with multiple mobile robots and prior scene awareness; in Section 4.4 we compare our frameworks (static and dynamic) with the aforementioned robots.

Our code, freely available at https://github.com/mginesi/dmp_vol_obst includes a Python 3.6 implementation of DMPs and our proposed approach to volumetric obstacle avoidance.

2 Dynamic Movement Primitives

Dynamic Movement Primitives (DMPs) is a framework for trajectory learning. It is based on an Ordinary Differential Equation (ODE) of spring-mass-damper type with a forcing term. This framework has numerous advantages that make it well suited for robotic applications. First, any trajectory can be learned and subsequently executed while changing starting and goal positions. Second, the executed trajectory will always converge to the goal, maintaining a similar shape to the learned trajectory. Third, the learned trajectory can be executed at different speeds simply by changing a single parameter. Finally, DMPs have been proven to be flexible enough to be extended in multiple ways: for instance, the formulation can be modified to deal with periodic movements [15, 38], to learn sensory experience [25, 26], and to work in unit quaternion space (in order to model orientations) [35, 39]. Another extension, that is the topic treated in this paper, is the inclusion of obstacle avoidance in the DMP framework [8, 10, 23].

In this Section, we recall the DMP formulation given in [10, 23, 24] upon which our work is based. Such formulation is an improvement of the original formulation by [13–15, 36]. Subsequently, in Section 2.1 we will present the state of the art of obstacle avoidance methods for DMPs, highlighting their strengths and weaknesses.

Dynamic Movement Primitives consist of the following system of Ordinary Differential Equations:

$$\begin{cases} \tau \dot{\mathbf{v}} = \mathbf{K}(\mathbf{g} - \mathbf{x}) - \mathbf{D}\mathbf{v} - \mathbf{K}(\mathbf{g} - \mathbf{x}_0)s + \mathbf{K}\mathbf{f}(s) & \text{(1a)} \\ \tau \dot{\mathbf{x}} = \mathbf{v} & \text{(1b)} \end{cases}$$

Vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^d$ are, respectively, the *position* and *velocity* of the system; and $\mathbf{x}_0, \mathbf{g} \in \mathbb{R}^d$ are, respectively, the *starting* and *goal positions*. Matrices $\mathbf{K}, \mathbf{D} \in \mathbb{R}_+^{d \times d}$ are, respectively, the *elastic* and *damping terms* of the system. Both are diagonal matrices, $\mathbf{K} = \text{diag}(K_1, K_2, \ldots, K_d)$, $\mathbf{D} = \text{diag}(D_1, D_2, \ldots, D_d)$, and satisfy the critical damping relation $D_i = 2\sqrt{K_i}$, so that the un-perturbed system, i.e. when $\mathbf{f} \equiv \mathbf{0}$, converges as fast as possible to $(\mathbf{x}, \mathbf{v}) = (\mathbf{g}, \mathbf{0})$, since $s \to 0$ and term $-\mathbf{K}(\mathbf{g} - \mathbf{x}_0)s$ vanishes as $t \to \infty$ (for details, see [10]). Scalar $\tau \in \mathbb{R}_+$ is a *temporal scaling factor* which can be used to make the execution of the trajectory faster or slower. Function $\mathbf{f} : \mathbb{R} \to \mathbb{R}^d$ is the *forcing* (also called *perturbation*) *term*. Scalar $s \in (0, 1]$ is a re-parametrization of time $t \in [0, T]$ governed by the so called *canonical system*

$$\tau \dot{s} = -\alpha s, \tag{2}$$

where $\alpha \in \mathbb{R}_+$ and the initial state is $s(0) = 1$.
The forcing term $\mathbf{f}(s) = [f_1(s), f_2(s), \ldots, f_d(s)]^\intercal$ is written in term of basis functions. Each component $f_p(s)$, $p = 1, 2, \ldots, d$ has the form

$$f_p(s) = \frac{\sum_{i=0}^{N} {}^p\omega_i \, \psi_i(s)}{\sum_{i=0}^{N} \psi_i(s)} \, s, \tag{3}$$

where ${}^p\omega_i \in \mathbb{R}$ is called *weigth*, and $\psi_i(s)$ is a basis function. In the literature, *Gaussian Radial Basis* (GRB)

function are mostly used (see e.g. [10, 14, 15, 23, 36]), even if other set of basis functions have been presented [9, 42]. These functions are defined as

$$\psi_i(s) = \exp\left(-h_i(s - c_i)^2\right), \tag{4}$$

with *centers* c_i defined as

$$c_i = \exp\left(-\alpha\, i\, \frac{T}{N}\right), \quad i = 0, 1, \ldots, N, \tag{5}$$

and *widths* defined as

$$\begin{aligned} h_i &= \frac{1}{(c_{i+1} - c_i)^2}, \quad i = 0, 1, \ldots, N-1, \\ h_N &= h_{N-1}. \end{aligned} \tag{6}$$

During the *learning phase*, a desired trajectory $\widetilde{\mathbf{x}}(t)$ and its velocity $\widetilde{\mathbf{v}}(t)$ are recorded. Then, from Eq. 1a, the desired forcing term $\widetilde{\mathbf{f}}(s(t))$ is computed (after fixing matrices $\mathbf{K}$ and $\mathbf{D}$). Finally, the weights ${}^p\omega_i$, $i = 0, 1, \ldots, N$, $p = 1, 2, \ldots, d$ that best approximate the desired forcing term $\widetilde{\mathbf{f}}$ using formulation (3) are computed using linear regression.

During the *execution phase*, starting and goal positions $\mathbf{x}_0$, $\mathbf{g}$ are set, and the forcing term $\mathbf{f}$ is computed using Eq. 3 with the weights computed before. Solving the dynamical system Eq. 1 will give a trajectory of similar shape to the learned one, that starts from $\mathbf{x}_0$ and converges to $\mathbf{g}$. In Fig. 1 an example of the spatial generalization property of the DMP framework is shown.

2.1 Methods for Obstacle Avoidance

In the literature, there exist two main ways to implement obstacle avoidance in the DMP framework. The first approach is the so-called *Stylistic DMPs* [22] in which a probability distribution $q({}^p\omega_i|\zeta)$ of the weights, conditioned to a *style parameter* ζ is learned, instead of the set of weights $\{{}^p\omega_i\}$. The style parameter can be, for instance, the size of an obstacle. The second approach, instead, consists in adding in Eq. 1a a *repulsive term* $\varphi(\mathbf{x}, \mathbf{v}) \in \mathbb{R}^d$ [8, 10, 23], that 'pushes' the trajectory away from the obstacle that then reads:

$$\tau\dot{\mathbf{v}} = \mathbf{K}(\mathbf{g} - \mathbf{x}) - \mathbf{D}\mathbf{v} - \mathbf{K}(\mathbf{g} - \mathbf{x}_0)s + \mathbf{K}\mathbf{f}(s) + \varphi(\mathbf{x}, \mathbf{v}). \tag{7}$$

In full generality, the repulsive term φ depends on both position $\mathbf{x}$ and velocity $\mathbf{v}$ of the system, but we will see that for some methods, it depends only on the position. The 'repulsive term' approach (7) can be further subdivided into two sub-categories. The first includes all those approaches that require an additional learning phase, in which executions both with and without obstacle are recorded, to model φ. For instance, in [29] and [37] a Neural Network is used to model the perturbation term. In [28] an analytical formulation is presented, but the number of free parameters that has to be tuned requires an additional learning process.

The second sub-category, in which our approach fits, comprehends all approaches in which there is no need for any additional learning phase. This is a great advantage, since the DMP can be used in virtually any situation, while the learning approaches may fail in situations too dissimilar to the ones shown during the learning phase.

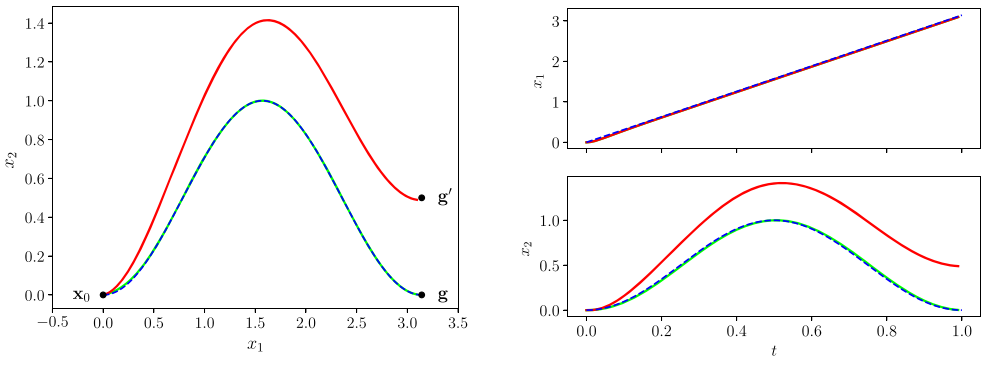
Since the proposed method enters the 'designed by hand' approaches, we will recall here, and compare our approach to later, only the methods that do not require any additional learning phase.

A potential field approach for point obstacles is proposed in [18] where an obstacle creates a potential field $U(\mathbf{x})$ at the system position $\mathbf{x}$. The perturbation term $\varphi(\mathbf{x}, \mathbf{v})$ in this case depends only on the position (and not on the velocity) and is the negative gradient of the potential:

$$\varphi(\mathbf{x}, \mathbf{v}) \equiv \varphi(\mathbf{x}) = -\nabla_{\mathbf{x}} U(\mathbf{x}), \tag{8}$$

with the potential defined as

$$U_s(\mathbf{x}) = \begin{cases} \frac{\eta}{2}\left(\frac{1}{p(\mathbf{x})} - \frac{1}{p_0}\right)^2 & \text{if } p(\mathbf{x}) \le p_0 \\ 0 & \text{if } p(\mathbf{x}) > p_0 \end{cases}, \tag{9}$$



a Trajectory. **b** Solution.

Fig. 1 Example of execution of a DMP in $\mathbb{R}^2$. On the left, the trajectory is shown. On the right, we plot the time evolution of each component. In both plots, the blue dashed line shows the desired behavior, which start at $\mathbf{x}_0 = [0, 0]^\intercal$ and ends at $\mathbf{g} = [\pi, 0]^\intercal$. The green solid line shows the execution of the learned DMP. The solid red line shows the execution of the learned DMP when changing goal position to $\mathbf{g}' = [\pi, 0.5]^\intercal$

where $\eta \in \mathbb{R}_+$ is a constant gain, $p_0 \in \mathbb{R}_+$ is the influence radius of the obstacle, and $p(\mathbf{x}) \in \mathbb{R}_+$ is the distance between the obstacle and the system's position.

It was pointed out in [23] that the perturbation term Eq. 8 obtained using Eq. 9 as potential may result in non-smooth obstacle behaviors since it does not depend on the velocity $\mathbf{v}$ of the system. Thus, the following 'dynamic' (i.e. velocity dependent) potential is proposed

$$U_d(\mathbf{x}, \mathbf{v}) = \begin{cases} \lambda(-\cos\theta)^\beta \frac{\|\mathbf{v}\|}{p(\mathbf{x})} & \text{if } \theta \in \left(\frac{\pi}{2}, \pi\right] \\ 0 & \text{if } \theta \in \left[0, \frac{\pi}{2}\right] \end{cases}, \tag{10}$$

where $\lambda, \beta \in \mathbb{R}_+$ are constant gains, and θ, depicted in Fig. 2a, is the angle taken between the current velocity $\mathbf{v}$ and the system's position $\mathbf{x}$ relative to the position $\mathbf{o}$ of the obstacle:

$$\cos\theta = \frac{\langle \mathbf{v}, \mathbf{x} - \mathbf{o} \rangle}{\|\mathbf{v}\| \, p(\mathbf{x})}, \tag{11}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard scalar product in $\mathbb{R}^d$, and $p(\mathbf{x})$ still denotes the distance between $\mathbf{x}$ and the obstacle.
For potentials depending on both position $\mathbf{x}$ and velocity $\mathbf{v}$ of the system, the perturbation term is defined as the negative gradient with respect to the position:

$$\varphi(\mathbf{x}, \mathbf{v}) = -\nabla_{\mathbf{x}} U(\mathbf{x}, \mathbf{v}). \tag{12}$$

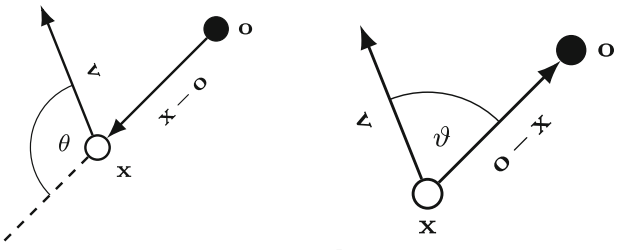
The following perturbation term was proposed in [10]:

$$\varphi(\mathbf{x}, \mathbf{v}) = \gamma \, \mathbf{R}\, \mathbf{v}\, \vartheta \, \exp(-\beta\vartheta), \tag{13}$$

where $\gamma, \beta \in \mathbb{R}_+$ are constant gains. The *steering angle* ϑ (depicted in Fig. 2b) is defined as

$$\vartheta = \arccos\left(\frac{\langle \mathbf{o} - \mathbf{x}, \mathbf{v} \rangle}{\|\mathbf{o} - \mathbf{x}\| \, \|\mathbf{v}\|}\right), \tag{14}$$

where $\mathbf{o}$ is the position of the point obstacle. Matrix $\mathbf{R}$ is defined as the rotation matrix of angle $\pi/2$ with respect to the axis generated by $(\mathbf{o} - \mathbf{x}) \times \mathbf{v}$, where $\times$ denotes the cross product in $\mathbb{R}^3$. This formulation presents an important advantage and two important shortcomings with respect to the previous two approaches. The advantage is that this formulation guarantees convergence to the goal position if the obstacles are still. On the other hand, using potential functions Eqs. 9 and 10, there may be cases in which the system remains 'trapped' in a local minima. However, as defined in [10], the matrix $\mathbf{R}$ makes sense only in $\mathbb{R}^3$ (and $\mathbb{R}^2$). Thus this approach can be used only when DMPs are used in ambient space, and not joint space. Moreover, formulation Eq. 13 does not depend on the distance from the obstacle, and the same 'importance' is given to close and far obstacles: this may result in oscillatory behaviors, as pointed out in [8].



a Angle θ defined in (11). **b** Steering angle ϑ defined in (14).

Fig. 2 Depiction on the definition of angle θ and ϑ in Eq. 11 and Eq. 14 respectively. We remark that the two angles are complementary, assuming same $\mathbf{x} - \mathbf{o}$, $\mathbf{v}$, and $\dot{\mathbf{o}} = \mathbf{0}$, since in this case the cosines are opposite: $\cos\theta = -\cos\vartheta$

The presented methods work only on point obstacles. Volumetric obstacles can be modeled using point clouds or by choosing a 'critical point' on the surface of the obstacle itself. However, both these strategies may generate odd behaviors: using a point cloud may result in high computational time, and it is in general hard to decide a priori how dense the point cloud should be, and the use of a critical point (e.g. the closer one) can result in non-smooth behaviors since this point is constantly changing.
For this reason, we proposed, in [8], a novel method to implement volumetric obstacle avoidance, based on the theory of *superquadric potential functions* [40]. In this approach, the following static potential function is defined

$$U_S(\mathbf{x}) = \frac{A \exp(-\eta\, C(\mathbf{x}))}{C(\mathbf{x})}, \tag{15}$$

where $A, \eta \in \mathbb{R}_+$ are gain parameters. Function $C : \mathbb{R}^d \to \mathbb{R}$ is an *isopotential*, that is a function satisfying

I1. The boundary of the obstacle is the zero-level set of the isopotential;
I2. The value of C increases when the distance from the obstacle increases.

An example of isopotential is the *superquadric potential function* [40]

$$C(\mathbf{x}) = \left(\left(\frac{x_1}{f_1(\mathbf{x})}\right)^{2n} + \left(\frac{x_2}{f_2(\mathbf{x})}\right)^{2n}\right)^{\frac{2m}{2n}} + \left(\frac{x_3}{f_3(\mathbf{x})}\right)^{2m} - 1, \tag{16}$$

that vanishes on the surface of a *generalized ellipsoid*. The main advantage of this isopotential is that, by tuning parameters m, n and functions $f_1, f_2, f_3 : \mathbb{R}^3 \to \mathbb{R}$, it is possible to model obstacles of any shape (their boundary will be the zero-level set of Eq. 16) [19, 27, 41].

In [8] we used a slightly simpler formulation of Eq. 16

$$C(\mathbf{x}) = \left(\frac{x_1 - \hat{x}_1}{\ell_1}\right)^{2n_1} + \left(\frac{x_2 - \hat{x}_2}{\ell_2}\right)^{2n_2} + \left(\frac{x_3 - \hat{x}_3}{\ell_3}\right)^{2n_3} - 1, \tag{17}$$

which, in general $j-$dimensional spaces generalize in

$$C(\mathbf{x}) = \sum_{j=1}^{d} \left(\frac{x_j - \hat{x}_j}{\ell_j} \right)^{2n_j} - 1.$$

The perturbation term in this approach is defined as in Eq. 8: $\varphi(\mathbf{x}, \mathbf{v}) = \varphi(\mathbf{x}) = -\nabla_{\mathbf{x}} U_S(\mathbf{x})$.

3 New Potential Function

In this work, we propose a dynamic potential function for volumetric (non-pointwise) obstacles, thus merging the frameworks (10) and (15).

Similarly to [23], we aim at designing a potential that satisfies the following three properties:

P1. The magnitude of the potential decreases with the distance of the system from the obstacle;
P2. The magnitude of the potential increases with the velocity of the system $\|\mathbf{v}\|$ and is zero when the system is not moving;
P3. The magnitude of the potential decreases with the angle between current velocity direction $\mathbf{v}/\|\mathbf{v}\|$, and the direction towards the obstacle; and, if the system is moving away from the obstacle, the potential should vanish.

To this end, mimiking (10), we define the dynamic potential function

$$U_D(\mathbf{x}, \mathbf{v}) = \begin{cases} \lambda(-\cos\theta)^{\beta} \dfrac{\|\mathbf{v}\|}{C^{\eta}(\mathbf{x})} & \text{if } \theta \in \left(\frac{\pi}{2}, \pi\right] \\ 0 & \text{if } \theta \in \left[0, \frac{\pi}{2}\right] \end{cases}, \qquad (18)$$

where λ, β, and $\eta \in \mathbb{R}_+$ are constant gains, and function $C(\mathbf{x})$ is any ispotential satisfying Properties I1 and I2 given in Section 2.1. The angle θ is taken between the system's velocity $\mathbf{v}$ and the direction between the system's position $\mathbf{x}$ and the closest point of the obstacle. Thanks to Property I2, we have that the gradient $\nabla_{\mathbf{x}} C(\mathbf{x})$ of the isopotential $C(\mathbf{x})$, is always perpendicular to the obstacle surface. Thus, at least for convex obstacles, the angle θ can be computed using

$$\cos\theta = \frac{\langle \nabla_{\mathbf{x}} C(\mathbf{x}), \mathbf{v} \rangle}{\|\nabla_{\mathbf{x}} C(\mathbf{x})\| \, \|\mathbf{v}\|}, \qquad (19)$$

while it is not well defined for non-convex obstacles. An intuition for these two observations are given in Fig. 3.

Remark 1 For non-convex obstacles, some workarounds can be used. First, if neither the starting position nor the goal is in the 'holes' of the obstacle, that is they are not in the convex hull of the obstacle, then the convex hull itself can be used as the obstacle. Second, one can think of relaxing the concept of gradient to allow sub-differentials. In such a case, the sub-gradient exists but it is not unique. Third, a non-convex obstacle can be split into multiple convex components, and each component would generate its own potential.

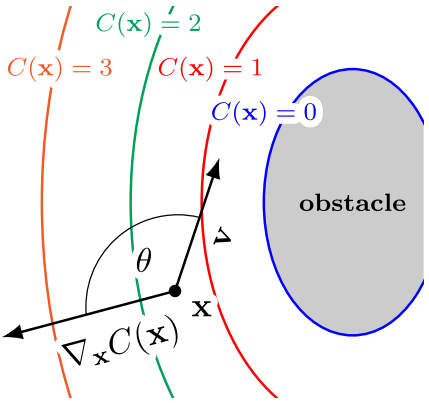
In Section 4.2 we test the first and third proposed workaround.

The potential defined in Eq. 18 clearly satisfies Properties P1., P2., and P3. Indeed, the potential is a decreasing function of $C(\mathbf{x})$ and an increasing function of θ, thus it satisfies P1 and P3. Moreover, it is an increasing function of $\|\mathbf{v}\|$, and thus it satisfies also P2.

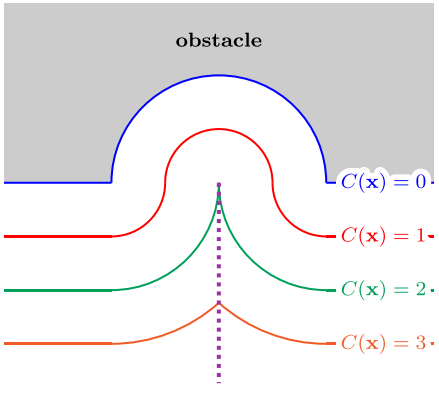
As an example, we show in Fig. 4 the potential (18) for an elliptic obstacle in $\mathbb{R}^2$, whose isopotential is

$$C(\mathbf{x}) = \left(\frac{x_1 - \widehat{x}_1}{\ell_1} \right)^2 + \left(\frac{x_2 - \widehat{x}_2}{\ell_2} \right)^2 - 1,$$

where the center of the ellipse is $\widehat{\mathbf{x}} = [\widehat{x}_1, \widehat{x}_2]^{\intercal}$ and the horizontal and vertical axes are, respectively, ℓ_1 and ℓ_2. The perturbation term is defined as in Eq. 12 and is computed

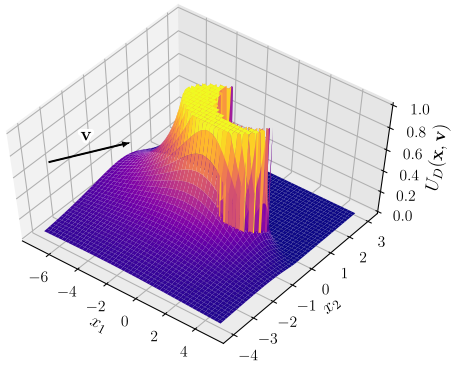


a Example of convex obstacle in which θ is well defined.

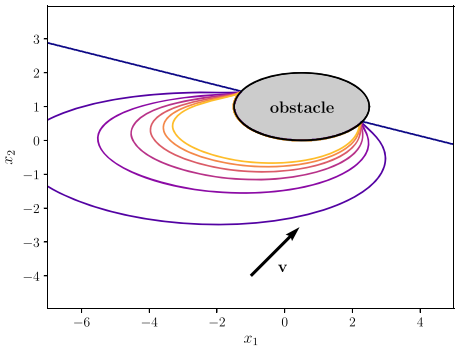


b Example of non-convex obstacle in which θ is not well defined: the purple dotted line shows the points in which $\nabla_{\mathbf{x}} C(\mathbf{x})$ does not exists.

Fig. 3 Figure 3a shows how the angle θ is defined when the gradient $\nabla_{\mathbf{x}} C(\mathbf{x})$ of the isopotential exists. Fig. 3b, instead, shows an example on how non-convex obstacles result in non differentiable isopotentials, and thus it is not possible to define the angle θ



a Mesh plot.



b Isocontour plot.

Fig. 4 Example of the dynamic potential $U_D(\mathbf{x}, \mathbf{v})$ given in Eq. 18 for an ellipse in $\mathbb{R}^2$. The velocity vector $\mathbf{v}$ is set to $\mathbf{v} = [1, 1]^\intercal$. The gains are set to $\lambda = 2$, $\beta = 2$, and $\eta = 1$. The ellipse has center in $[1/2, 1]^\intercal$, horizontal axis 2, and vertical axis 1. In both figures, the potential has been cropped at the value 1 for display purposes: it goes to infinity on half of the boundary of the obstacle (on the other half, the system goes away from the obstacle, so the potential is zero)

as follows. Clearly, for $\theta \in [0, \pi/2]$ $\varphi(\mathbf{x}, \mathbf{v}) \equiv \mathbf{0}$. When $\theta \in (\pi/2, \pi]$, instead, we compute

$$\begin{aligned}\varphi(\mathbf{x}, \mathbf{v}) &= -\nabla_{\mathbf{x}}\big(U_D(\mathbf{x}, \mathbf{v})\big) \\ &= -\nabla_{\mathbf{x}}\left(\lambda(-\cos\theta)^{\beta}\frac{\|\mathbf{v}\|}{C^{\eta}(\mathbf{x})}\right) \\ &= -\frac{\lambda\,\|\mathbf{v}\|\,(-\cos\theta)^{\beta-1}}{C^{\eta}(\mathbf{x})}\left(-\beta\,\nabla_{\mathbf{x}}(\cos\theta) + \frac{\eta\cos\theta}{C(\mathbf{x})}\nabla_{\mathbf{x}}(C(\mathbf{x}))\right).\end{aligned}$$

The term $\nabla_{\mathbf{x}}(\cos\theta)$ can be computed as

$$\begin{aligned}\nabla_{\mathbf{x}}(\cos\theta) &= \nabla_{\mathbf{x}}\left(\frac{\langle\nabla_{\mathbf{x}}C(\mathbf{x}), \mathbf{v}\rangle}{\|\nabla_{\mathbf{x}}C(\mathbf{x})\|\,\|\mathbf{v}\|}\right) \\ &= \frac{1}{\|\mathbf{v}\|\,\|C(\mathbf{x})\|^2}\Big(\|\nabla_{\mathbf{x}}C(\mathbf{x})\|\;\nabla_{\mathbf{x}}\big(\langle\nabla_{\mathbf{x}}C(\mathbf{x}), \mathbf{v}\rangle\big) \\ &\quad - \langle\nabla_{\mathbf{x}}C(\mathbf{x}), \mathbf{v}\rangle\;\nabla_{\mathbf{x}}\big(\|\nabla_{\mathbf{x}}C(\mathbf{x})\|\big)\Big).\end{aligned} \tag{20}$$

For instance, let us consider the case in which the isopotential $C(\mathbf{x})$ is an ellipsoid in $\mathbb{R}^3$ with center $\widehat{\mathbf{x}} = [\widehat{x}_1, \widehat{x}_2, \widehat{x}_3]^\intercal$, and axes (ℓ_1, ℓ_2, ℓ_3),

$$C(\mathbf{x}) = \left(\frac{x_1 - \widehat{x}_1}{\ell_1}\right)^2 + \left(\frac{x_2 - \widehat{x}_2}{\ell_2}\right)^2 + \left(\frac{x_3 - \widehat{x}_3}{\ell_3}\right)^2.$$

In this case, the gradient is

$$\nabla_{\mathbf{x}}C(\mathbf{x}) = 2\begin{bmatrix}\frac{x_1-\widehat{x}_1}{\ell_1^2}\\ \frac{x_2-\widehat{x}_2}{\ell_2^2}\\ \frac{x_3-\widehat{x}_3}{\ell_3^2}\end{bmatrix}.$$

The quantities $\nabla_{\mathbf{x}}\big(\langle\nabla_{\mathbf{x}}C(\mathbf{x}), \mathbf{v}\rangle\big)$ and $\nabla_{\mathbf{x}}\big(\|\nabla_{\mathbf{x}}C(\mathbf{x})\|\big)$ in Eq. 20 read, respectively,

$$\begin{aligned}\nabla_{\mathbf{x}}\big(\langle\nabla_{\mathbf{x}}C(\mathbf{x}), \mathbf{v}\rangle\big) &= 2\begin{bmatrix}\frac{v_1}{\ell_1^2}\\ \frac{v_2}{\ell_2^2}\\ \frac{v_3}{\ell_3^2}\end{bmatrix}, \quad \text{and} \quad \nabla_{\mathbf{x}}\big(\|\nabla_{\mathbf{x}}C(\mathbf{x})\|\big) \\ &= \frac{4}{\|\nabla_{\mathbf{x}}C(\mathbf{x})\|}\begin{bmatrix}\frac{x_1-\widehat{x}_1}{\ell_1^4}\\ \frac{x_2-\widehat{x}_2}{\ell_2^4}\\ \frac{x_3-\widehat{x}_3}{\ell_3^4}\end{bmatrix}.\end{aligned}$$

We emphasize that the proposed approach encompasses most of the desired properties of obstacle avoidance frameworks for DMPs since it is: dynamic, well defined in $\mathbb{R}^d$, volumetric, and distance-dependent. On the other hand, this approach does not guaranteed the convergence to the goal since local minima may arise. However, as we already pointed out in [8], it is unlikely to encounter a local minimum, and if it happens, a perturbation term pushing the trajectory out of it can be easily added to the DMP formulation.

In Table 1 a summary of the properties of all the approaches presented in Section 2.1 and the proposed approach is given. From this, it is possible to observe that the proposed method is the one satisfying the greatest number of desirable properties.

Remark 2 Dynamic formulations (11), (13), and (19) do not take into account the velocity of the obstacle. However, it is straightforward to extend the definition to this case by simply substituting $\mathbf{v}$ with $\mathbf{v} - \dot{\mathbf{o}}$, where $\dot{\mathbf{o}}$ denotes the velocity of the obstacle.

4 Results

4.1 Execution Time

Before showing the experiments to test our new method and compare it to the methods presented in Section 2.1, we discuss the computational time. Our DMPs' implementation and obstacle avoidance methods are implemented in Python3.6 and can be found at https://github.com/mginesi/dmp_vol_obst.

To test the execution time we generate one obstacle for each method for different dimensionalities d of the state space. Then, for each value of d, we generate 100

Table 1 Summary of the properties of various methods for obstacle avoidance

Method	Type of obstacle	Space of definition	Type of potential	Distance dependent	Guaranteed convergence
Static potential (9)	Point	$\mathbb{R}^d$, $d \in \mathbb{N}$	Static	Yes	No
Dynamic potential (10)	Point	$\mathbb{R}^d$, $d \in \mathbb{N}$	Dynamic	Yes	No
Steering angle Eq. 13	Point	$\mathbb{R}^2$, $\mathbb{R}^3$	Dynamic	No	Yes
Static potential (15)	Volumes	$\mathbb{R}^d$, $d \in \mathbb{N}$	Static	Yes	No
Dynamic potential (18)	Volumes	$\mathbb{R}^d$, $d \in \mathbb{N}$	Dynamic	Yes	No

The desired properties are underlined

random positions and velocities to compute the perturbation terms $\varphi(\mathbf{x}, \mathbf{v})$, and use these values to compute the average computational time and its standard deviation. Table 2 show the results of this test (we remark that the steering angle method makes sense only in $\mathbb{R}^2$ and $\mathbb{R}^3$ due to the cross product in the definition of $\mathbf{R}$ in Eq. 13).

The new proposed method results to be the slowest. However, it is computed in about a tenth of a millisecond which still makes it able to be computed fast enough to not influence the on-line control of most robots. For instance, our Panda industrial manipulator works at approximately 50 Hz.

Tests were performed on a machine with a quad-core Intel Core i7-7000 CPU with 16 GB of RAM.

4.2 Synthetic Experiments

In this Section, we test and compare the behaviors of the approaches recalled in Section 2.1 and our novel approach, presented in Section 3, performing the same test we performed in [8]. In the first test, we show the behaviors of all the methods in the presence of a single obstacle (an ellipse). For the point obstacle methods, the obstacle is modeled using a point cloud on the boundary of the obstacle itself. We then add a second obstacle (a circle) to the previous scenario.

Moreover, we present an additional test to show how to treat non-convex obstacles as explained in Remark 1.

Table 2 Computational time (in seconds) of the perturbation term for various methods of obstacle avoidance

		Eq. 9	Eq. 10	Eq. 13	Eq. 15	Eq. 18
$d = 2$	mean	1.31e-05	3.91e-05	6.27e-05	2.01e-05	1.14e-04
	st. dev.	6.86e-06	1.61e-05	2.57e-05	7.20e-06	3.83e-05
$d = 3$	mean	1.17e-05	2.63e-05	8.00e-05	2.11e-05	1.08e-04
	st. dev.	3.57e-06	1.03e-05	1.62e-05	5.75e-06	2.52e-05
$d = 4$	mean	1.14e-05	3.14e-05	—	2.29e-05	1.04e-04
	st. dev.	3.23e-06	1.06e-05	—	6.44e-06	1.67e-05
$d = 5$	mean	1.12e-05	2.63e-05	—	2.52e-05	1.06e-04
	st. dev.	2.69e-05	9.96e-06	—	6.44e-06	1.67e-05
$d = 6$	mean	1.20e-05	3.38e-05	—	2.73e-05	1.16e-04
	st. dev.	4.66e-06	1.20e-05	—	6.40e-06	3.31e-05
$d = 7$	mean	1.31e-05	2.43e-05	—	3.14e-05	1.20e-04
	st. dev.	4.62e-06	1.07e-05	—	8.52e-06	2.32e-05

Table 3 Hyper-parameters for obstacle avoidance methods

Method	Hyper - parameters
Static potential (9)	$p_0 = 0.1,\ \eta = 1$
Dynamic potential (10)	$\lambda = 0.2,\ \beta = 2$
Steering angle (13)	$\gamma = 20,\ \beta = 3$
Static potential (15)	$A = 10,\ \eta = 1$
Dynamic potential (18)	$\lambda = 10,\ \beta = 2,\ \eta = 1/2$

In the first test, we generate the following trajectory in the plane: $(x_1(t), x_2(t)) = (t\cos(\pi t), t\sin(\pi t)),\ t \in [0, 1]$. Then, we learn a DMP with elastic and damping constants, respectively, $\mathbf{K} = K\,\mathbf{Id}_2$ and $\mathbf{D} = D\,\mathbf{Id}_2$, where $\mathbf{Id}_2$ denotes the 2×2 identity matrix, and K and D have values $K = 1050$ and $D = 2\sqrt{K} \approx 65$. In this test, the obstacle is an ellipse centered in $(-0.5, 0.7)$ with semi-axis 0.3 and 0.2. For the tests done using point-wise obstacle avoidance methods Eqs. 9, 10, and 13, the boundary of the obstacle is discretized using fifty equally distributed points. The hyper-parameters for all the methods are given in Table 3. The resulting trajectories are shown in Fig. 5. From this first test, we compute, at each time t how much the trajectory deviate from the learned behavior in order to avoid the obstacle. This "error" is computed as

$$\varepsilon(t) = \|\mathbf{x}_{\text{true}}(t) - \mathbf{x}(t)\|,$$

where $\mathbf{x}_{\text{true}}(t)$ is the learned trajectory, and $\mathbf{x}(t)$ is the adapted behavior. In Fig. 6a it is possible to observe that the proposed method results in a trajectory that deviates less from the learned one.

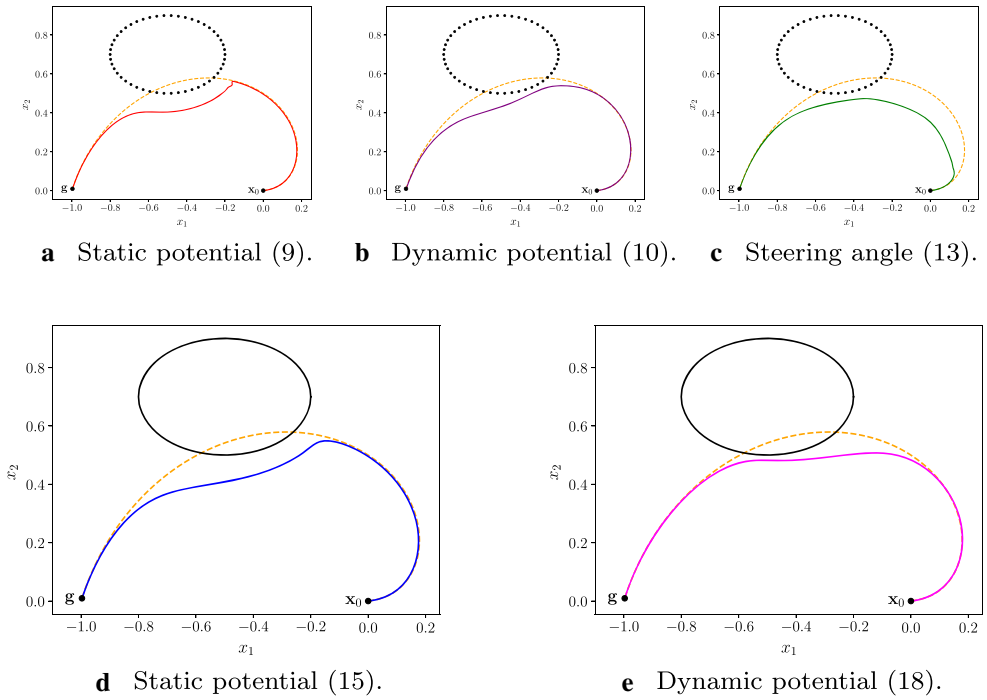
To discuss the smoothness of the different behaviors, we compute, at each time t, the norm $\|\ddot{\mathbf{x}}(t)\|$ of the acceleration $\ddot{\mathbf{x}}(t)$ of the adapted trajectory. As shown in Fig. 6b, we see that the proposed method results in the less oscillatory behavior of $\|\ddot{\mathbf{x}}(t)\|$. This last aspect makes the proposed method the most stable one when controlling the position of a robot.

Table 4 shows the maximum and average values of both the error and the acceleration norms. From it, we see that the steering angle method Eq. 13 result in smaller accelerations. However, the acceleration profile still results in more oscillatory behavior than our novel dynamic potential (18).

In summary, the proposed dynamic potential (18) gives both the smoother behavior and the trajectory that remains closer to the learned one between all the methods we presented in Section 2.1, thus making it the most suitable in real applications.

As a second synthetic experiment, we maintain the same conditions of the experiment before (desired curve, as well as DMP and obstacles' hyper-parameters), and we add a second obstacle. This new obstacle is a circle centered in $(0.15, 0.4)$ and with radius 0.1. For the point-wise obstacle avoidance methods, the circumference is discretized with fifty equally distributed nodes. Figure 7 shows the adaptation of the DMP to the presence of the obstacles, Fig. 8a shows the distance between desired trajectory and DMP, and Fig. 8b shows the 2-norm of the acceleration of the DMP as function of time.

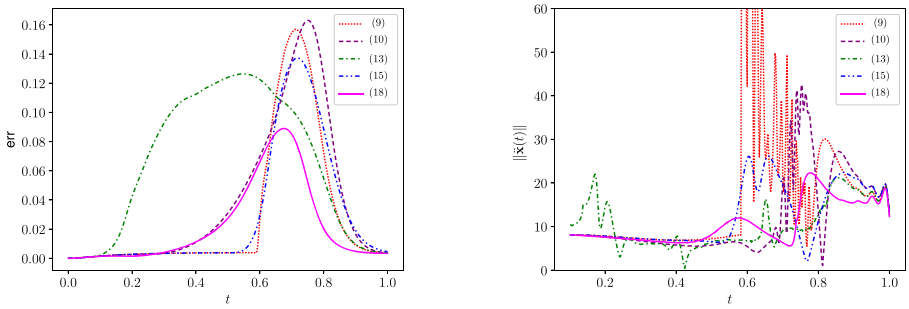
Table 4 shows the maximum and average values of both the error and the acceleration norms. As for the one obstacle test, the steering angle method Eq. 13 result in smaller



a Static potential (9). **b** Dynamic potential (10). **c** Steering angle (13).

d Static potential (15). **e** Dynamic potential (18).

Fig. 5 Obstacle avoidance behavior for the methods recalled in Section 2.1 and the proposed method from Section 3. In all plots, the dashed orange line shows the desired trajectory, while the solid line shows the adaptation of the DMP to the presence of the obstacle. In the three top figures, the black dots mark the point obstacles used as mesh. In the two bottom figures, the boundary of the obstacle is plotted using the full black line



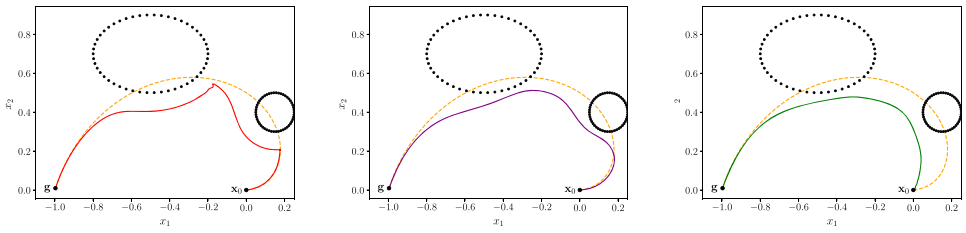
a Plot of the distance (in 2-norm) between the desired trajectory and the executed one.

b Plot of the norm of the acceleration of the executed DMP.

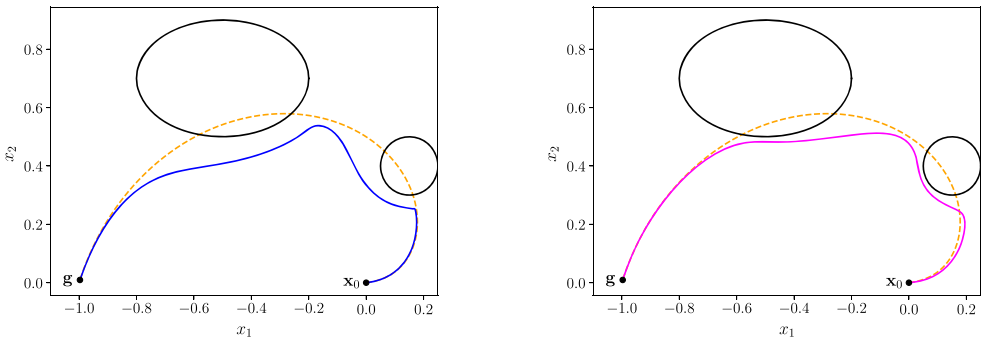
Fig. 6 For tests depicted in Fig. 5, plot of the distance between desired and executed trajectory (left), and of the norm of the acceleration (right) as functions of time

Table 4 Statistics of synthetic tests shows in Figs. 5 and 7. Minimum values for each statistic are underlined

		Eq. 9	Eq. 10	Eq. 13	Eq. 15	Eq. 18
1 obstacle	maximum error	0.157	0.163	0.126	0.137	<u>0.089</u>
	average error	0.029	0.040	0.066	0.030	<u>0.022</u>
	maximum acceleration norm	277.99	42.55	<u>22.02</u>	26.15	22.32
	average acceleration norm	19.11	12.40	<u>11.08</u>	12.77	11.20
2 obstacles	maximum error	0.210	0.205	0.149	0.150	<u>0.092</u>
	average error	0.064	0.082	0.088	0.052	<u>0.035</u>
	maximum acceleration norm	311.32	50.60	<u>21.29</u>	47.67	53.53
	average acceleration norm	30.00	15.42	<u>11.65</u>	18.11	16.13

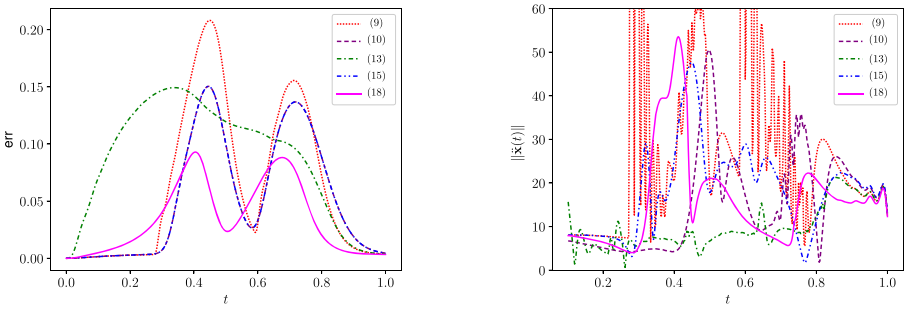


a Static potential (9). **b** Dynamic potential (10). **c** Steering angle (13).



d Static potential (15). **e** Dynamic potential (18).

Fig. 7 Obstacle avoidance behavior for the methods recalled in Section 2.1 and the proposed method from Section 3. In all plots, the dashed orange line shows the desired trajectory, while the solid line shows the adaptation of the DMP to the presence of the obstacle. In the three top figures, the black dots mark the point obstacles used as mesh. In the two bottom figures, the boundary of the obstacle is plotted using the full black line



a Plot of the distance (in 2-norm) between the desired trajectory and the executed one.

b Plot of the norm of the acceleration of the executed DMP.

Fig. 8 For tests depicted in Fig. 7, plot of the distance between desired and executed trajectory (left), and of the norm of the acceleration (right) as functions of time

accelerations, even tho the acceleration profile results more oscillatory than the dynamic potential (18).

Also in this test, it is possible to observe that the proposed method still gives the trajectory that less deviates from the learned one while maintaining the less oscillatory behavior at the acceleration level.

Finally, we present a synthetic test with a non-convex obstacle, testing two workarounds given in Remark 1. We define a 'U'-shaped non-convex obstacle and present two scenarios.

In the first scenario (Fig. 9a) the goal is inside the 'hole' of the 'U'. Thus, we subdivide the obstacle into three components (two vertical and one horizontal); then, we cover each component with a 2-dimensional pseudo-ellipsoid (i.e. we use formulation (17) with $n_1 = n_2 = 2$ and no vertical component).

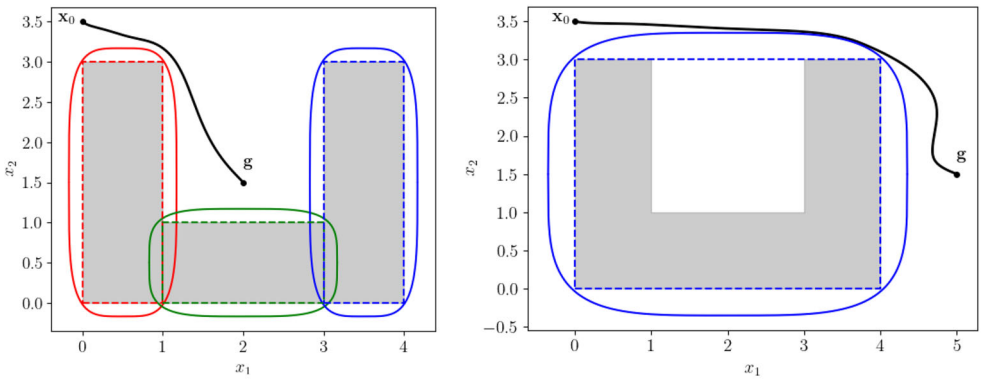
In the second scenario (Fig. 9b) neither the goal nor the initial position is in the 'hole' of the obstacle. Thus, we consider as the obstacle the convex hull of the 'U', and we cover it with the 2-dimensional pseudo-ellipsoid.

It is possible to see that both approaches result in proper obstacle avoidance behavior.

4.3 Experiments with Robots in Simulation

In this Section, we describe experiments performed with Kuka YouBot models in a simulated environment. These experiments are useful to validate the results highlighted in the previous section with an application of our framework to a more realistic use case. The simulation scene is shown in Fig. 10. It includes three YouBots that can move in a rectangular region defined by four walls (treated as obstacles), with fixed cubes as obstacles on the way. Each robot must reach a specific target position, defined by a platform with the same color as the robot. We assume that the geometry and the positions of the obstacles in the scene is known in advance. The scene is built in the popular CoppeliaSim simulation environment from Coppelia Robotics [34], which allows to simulate the dynamics of the robots and to control them through ROS topics as in real applications.

Each Youbot is controlled in position by a DMP with $\mathbf{x}, \mathbf{v} \in \mathbb{R}^2$; we do not control the orientation of the robots along their normal axis, since we are interested in the obstacle avoidance problem for Cartesian DMPs. In order to guarantee the synchronization between the robots, we construct a 6-dimensional DMP, concatenating the components $\mathbf{x}, \mathbf{v}, \dot{\mathbf{v}} \in \mathbb{R}^2$ of position, velocity, and acceleration of each YouBot in a single array. In this way, the robots share the same canonical system. The obstacle-free trajectory of each robot towards its target is a straight line. In this way, it is clear from the scene that the robots would collide during their motion. Since the objects in the scene are known a priori, one could argue



a Convex components.

b Convex hull.

Fig. 9 Different method to deal with non-convex obstacles. The obstacle is depicted with a gray shade. Dashed lines show the convex components in the plot on the left, and the convex hull in the plot on the right. Both plots show the enlarged potential (to be written as a generalized ellipsoid) in the same color as the obstacle. The black solid line shows the executed DMP

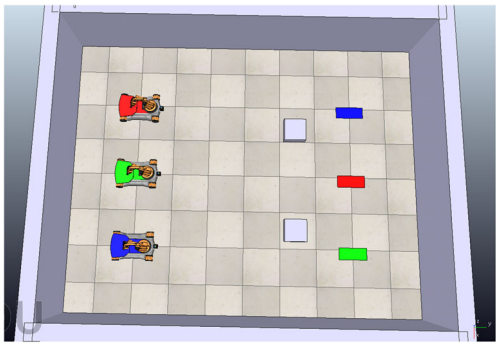


Fig. 10 The simulation scene in CoppeliaSim for the three YouBots

that the collision between the robots could be avoided by computing the trajectories in advance, and coordinating the motion of the robots (e.g. tuning the speed of each of them appropriately). Multi-robot motion coordination has been extensively studied, and it is out of the scope of this paper. We refer the reader to [43] for a recent survey. In our experiments, we have decided to simulate a more realistic multi-robot task, in which the robots do not know the trajectory of each other in advance. Hence, we model each YouBot as a dynamic potential using our formulation as in Eq. 18, so that it influences the forcing terms of the other robots. In this way, we show how our framework for obstacle avoidance is suitable for reactive motion planning. At each time step, we build an ellipsoid around each YouBot, setting $n_1 = n_2 = 1$ in Eq. 17. We control the center point of the YouBots, therefore the semi-axes of the ellipsoid are set as the full dimension of the robot (width $\times$ length) to avoid collisions. The parameters for the dynamic potential function are set as $\lambda = 60, \eta = 0.2, \beta = 2$ after empirical evaluation. When computing the forcing term for each robot, we compute the velocity term in Eq. 12 as the relative velocity between the robots. We test two different straight-line trajectories, one with null forcing term and the other with constant speed, to verify the independence of our framework with respect to the specific trajectory to be executed. The constant speed trajectory is first generated synthetically; then, the weights are learned as explained in Section 2. The DMP parameters are set as as $K = 3050, \alpha = 4, D = 2\sqrt{K} \approx 110.45$ for both sets of weights. The trajectories are computed at 1ms step of integration. We model the walls and the fixed obstacles as generalized ellipsoids (enlarged of the dimension of the YouBots), setting $n_1 = n_2 = 2$ in Eq. 17 to better approximate the sharp edges. We compare the performance of our previous static obstacle formulation (15) with our novel one, modeling the fixed obstacles with both methods. The results are shown in Fig. 11.

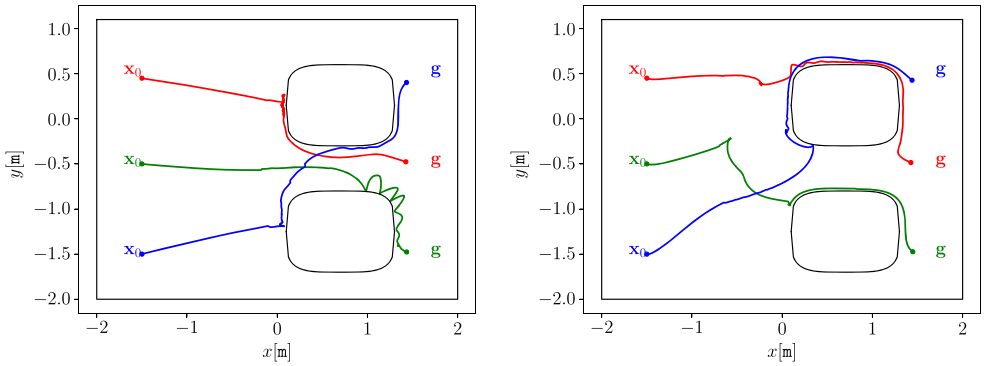
Figure 11a-b are obtained setting $A = 60, \eta = 2$ in Eq. 15. Figure 11c-d are obtained setting $\lambda = 60, \beta = 2, \eta = 2$ in Eq. 18. Parameters are set after empirical evaluation. We notice that the dynamic potential formulation results in smoother trajectories as the robots move close to the cubes in the scene. Indeed, in Fig. 11a it is possible to observe that both the red and green YouBots have oscillatory behaviors near the obstacles, as for the red and blue Youbots in Fig. 11b. On the other hand, oscillations are greatly reduced when the dynamic potential is used, as can be seen in Fig. 11c and d.

4.4 Experiments on Real Setups

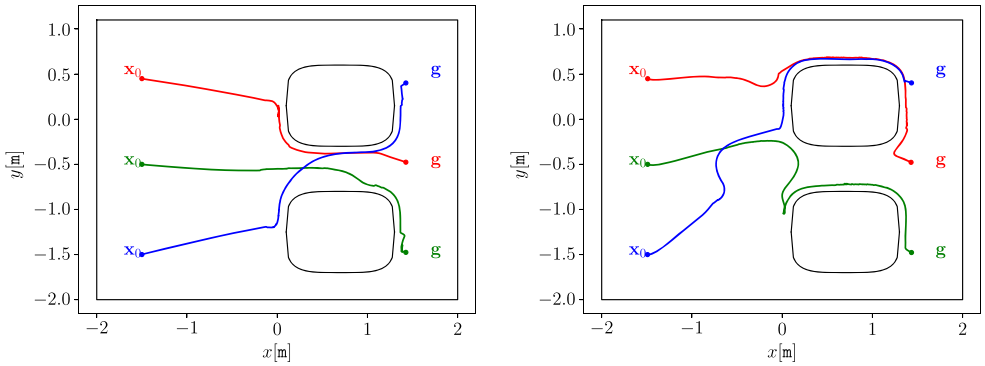
We now show the results of the tests performed on different robots. At first, we tested our obstacle avoidance framework on an industrial manipulator Panda from Franka Emika, studying a standard pick-and-place task with pegs and rings. Then, we replicated the same task on a smaller setup with a surgical robot da Vinci from Intuitive Surgical, showing that our framework is able to scale with the dimension of the setup. Finally, a scenario with a YouBot in a partially unstructured environment is tested, showing how our framework can be easily integrated with scene reconstruction techniques through vision sensors.

4.4.1 Experiments with Panda Robot

The setup for the Panda robot is shown in Fig. 12a. The robot must pick the green ring and place it on the green peg. On the way to the peg, the robot has to avoid the red peg, i.e. neither the end effector nor the grasped ring has to hit the peg. The task can be described by a simple state machine with four actions/states: *move to ring, grasp, move to peg* and *release gripper*. The moving actions are kinematically described with two DMPs in Cartesian space with null weights, i.e. straight-line trajectories, with $K = 1050, \alpha = 4, D = 2\sqrt{K}$. The trajectories describe the motion of the center of the gripper of the robot. Notice from Fig. 12a that the encumbrance of the end effector is significant, and controlling only the center of the gripper does not guarantee safe collision avoidance. As explained in our previous work [8], there are two solutions to this issue. One is to enlarge the radial dimension of the pegs according to the size of the end effector. The second solution is to exploit the kinematic redundancy of the 7-DOF manipulator and compute an obstacle-free joint configuration for each point in the DMP. We have chosen the latter approach to limit the size of the obstacles and, hence, maximize the reachable workspace for the robot. We control the robot through its standard MoveIt/ROS interface, setting TRAC-IK [2] as the inverse kinematics solver. TRAC-IK is a state-of-the-art library for this purpose, and it has been chosen because it allows defining optimal metrics to compute the joint configuration from a given pose. We set the solver to compute an inverse kinematics solution



a Null weights, static potential. **b** Constant speed, static potential.



c Null weights, dynamic potential. **d** Constant speed, dynamic potential.

Fig. 11 DMPs with constant speed and with null weights of the three YouBots in simulation. Obstacles are represented in the scene with the superquadric isopotential approximation, enlarged of the dimensions of the YouBots. The walls are represented as a rectangle containing the robots and the other obstacles for simplicity. Trajectories are referred to the center points of the robots

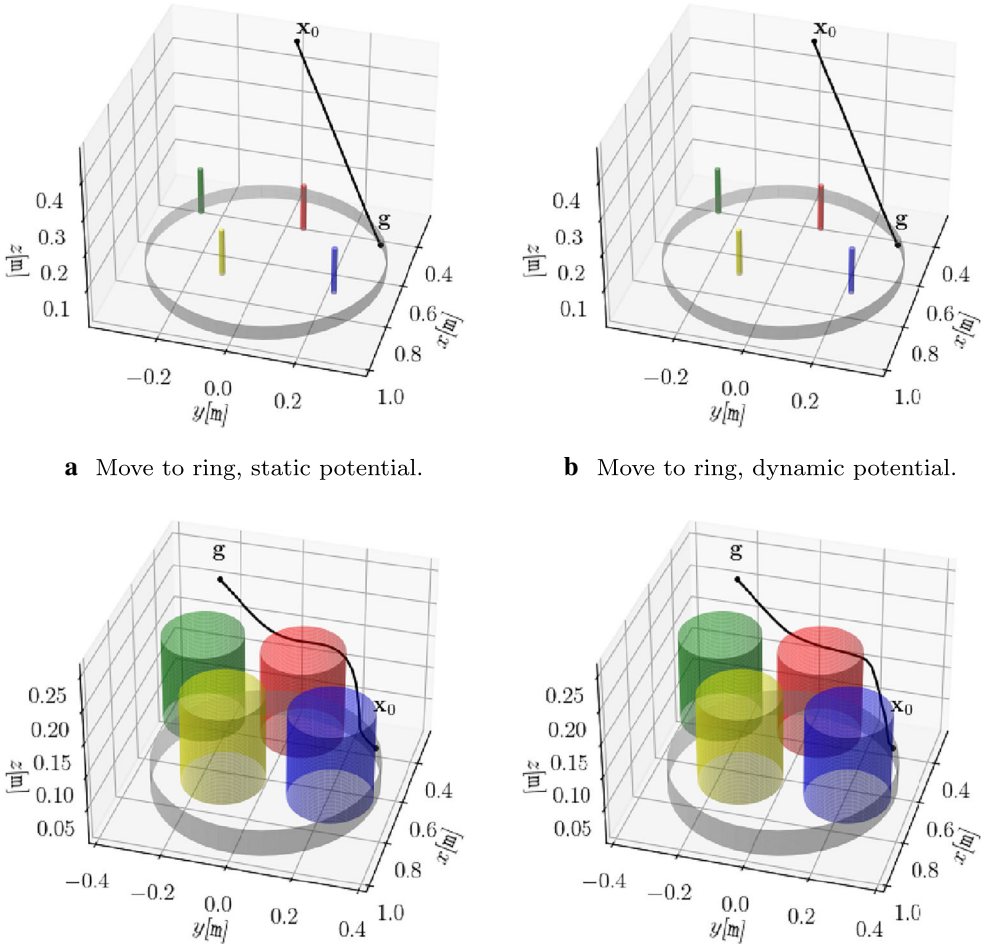
which maximizes the manipulability of the robot, defined as $\sqrt{\det(\mathbf{J}\mathbf{J}^{\intercal})}$ [8]. Though we do not control the orientation of the end effector with our DMP formulation, we constrain the orientation to be within 5° (along each axis) from the initial orientation for each Cartesian waypoint (shown in Fig. 12a). Then, we gradually relax this tolerance if no inverse kinematics solution is found. We also constrain two consecutive joint configurations to differ no more than 45° on each joint, so that we are able to avoid abrupt movements during the execution. The scene (location of the peg base, the pegs, and the ring) is assumed to be known in advance. Hence, obstacles (the base and the pegs) are represented as superquadric potential function shaped as cylinders (assuming the $z-$axis as the normal to the base, exponents in Eq. 17 are set as $n_1 = n_2 = 1, n_3 = 2$). Figure 12 shows the main steps in the task execution.

After the ring is grasped (Fig. 12b), we have that both the end-effector and the ring should avoid the pegs. Thus, when the ring is held by the robot, we 'enlarge' each obstacle by the radius of the ring. Indeed, in Fig. 13a and b we have that the robot is not holding the ring, and the pegs are modeled with their actual radius. On the other hand, in Fig. 13c and d the ring is held by the robot, and the obstacles are larger.

In Fig. 13 we show the trajectories for the two actions. Notice that the radial dimension of the pegs is enlarged when moving to the peg, i.e. in Fig. 13d and d, since we need to avoid that the grasped ring hits the obstacles. Hence, the radius of the base of the pegs is augmented by the radius of the ring. Obstacles are modelled with our dynamic potential formulation when moving to the ring, setting $\lambda = 10, \eta = 1, \beta = 1$ in Eq. 18. We compare our novel approach with the static potential formulation in Eq. 15, setting $A = 10, \eta = 1$.

a Initial setup. **b** Grasped ring. **c** Avoided obstacle. **d** Released ring.

Fig. 12 The pick-and-place task with the Panda robot



a Move to ring, static potential.

b Move to ring, dynamic potential.

c Move to peg, static potential.

d Move to peg, dynamic potential.

Fig. 13 Moving trajectories for the pick-and-place task with the Panda robot. Axes coordinates are referred to the frame of the base link of the robot

Figure 13 shows the result of these experiments. Figure 13a and b shows that, both for the static and dynamic volume potentials (15) and (18), the trajectories for the move-to-ring gesture do not result perturbed from the presence of the obstacles, since there is no risk of collisions.

On the other hand, we see that for the move-to-peg gesture, the trajectories deviate from the straight-line behavior. Both the static (15) and dynamic (18) potentials result in a proper obstacle avoidance behavior.

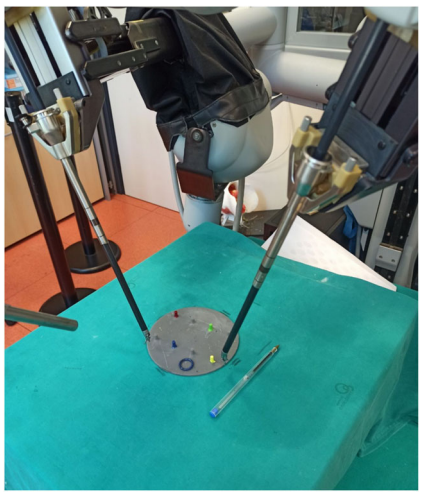
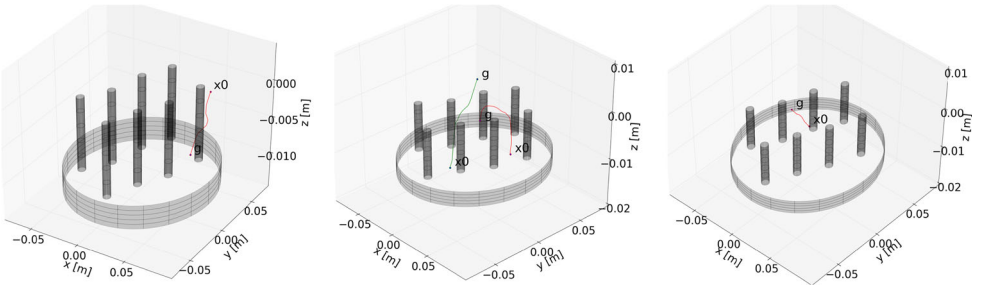


Fig. 14 The setup for the peg transfer task with the da Vinci surgical robot: PSM1 on the right and PSM2 on the left

4.4.2 Experiments with the da Vinci Surgical Robot

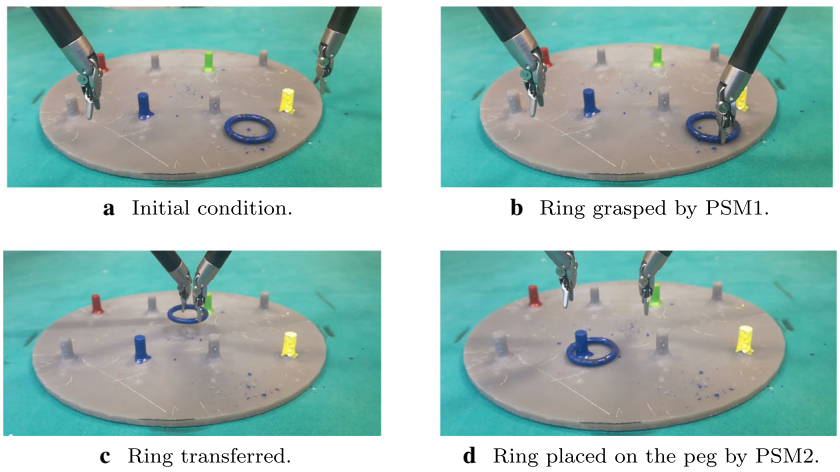
We replicate the peg transfer task using the da Vinci surgical robot from Intuitive Surgical controlled through the da Vinci Research kit [1] and ROS, with the setup shown in Fig. 14. The robot has two arms, named PSM1 and PSM2. Hence, we modify the state machine for the task. The PSM1 must move to the blue ring, grasp it, move the ring to the center of the base and exchange it with the PSM2. Then, the PSM2 carries the ring to the blue peg and the task ends. The scene description (locations of pegs, the ring, and the base) is assumed as a prior. We have designed the initiallocation of the arms and the ring in such a way that

[1] https://github.com/jhu-dvrk/dvrk-ros/tree/master/dvrk_python



a Move to ring with PSM1. **b** Transfer between PSMs (PSM1 in red, PSM2 in green). **c** Move to peg with PSM2.

Fig. 15 Trajectories executed by the da Vinci arms. Trajectories are referred to the center of the grippers, and they are expressed in a fictitious coordinate frame common to the PSMs, obtained using our calibration procedure presented in [33]



a Initial condition. **b** Ring grasped by PSM1. **c** Ring transferred. **d** Ring placed on the peg by PSM2.

Fig. 16 Main steps of the peg transfer task with the da Vinci surgical robot

Fig. 17 The YouBot with the Realsense D435 camera on its front

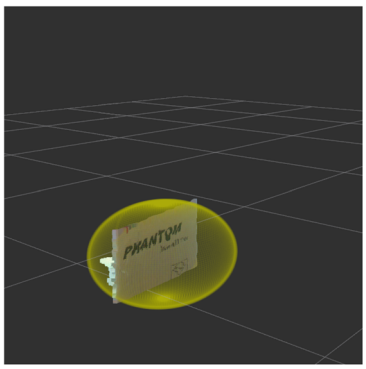


Fig. 18 Point cloud filtered with the ellipsoid created around the object

Fig. 19 Main steps of the YouBot task with the obstacle added to the scene during the execution

a **b** **c**

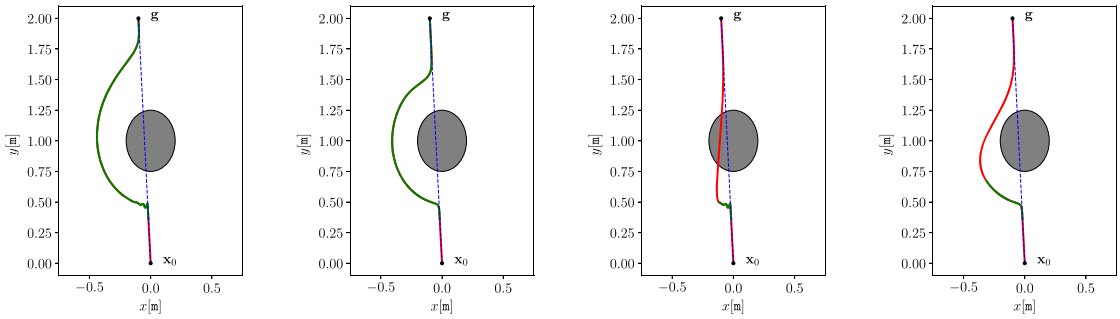
the pegs act as obstacles for the robot. In order to make a comparison with the task with the Panda robot, the *transfer* action can be seen as a combination of a *move to ring* action for the PSM2 and a *move to peg* action for the PSM1, where the goal is actually the center of the base instead of a real peg. Hence, the actions executed by the surgical robot can be interpreted as a replication of the actions of the industrial manipulator, just scaled on a smaller size of the setup. For this reason, we represent the obstacles with the same superquadric potentials (i.e. the same parameters) as in the Panda task. The trajectories of the robot are again described by Cartesian DMPs with null weights, and we build a single 6-dimensional DMP in order to share the same canonical system for the arms. We first test our novel dynamic potential formulation to model the obstacles. However, the DMPs does not converge to the goal in the transfer and when moving to the peg. On the contrary, our static potential formulation converges smoothly, as shown in Fig. 15. Figure 16 shows the main steps of the task execution. Notice that we do not need to compute inverse kinematics solutions from the DMP waypoints as with the industrial manipulator since the arms of the surgical robot have 6 degrees of freedom, and we force the orientation of the grippers to stay constant during the task. Moreover, Fig. 14 shows that the encumbrance of the grippers is minor, hence they safely avoid obstacles.

4.4.3 Experiments with Real YouBot

We test our obstacle avoidance framework with one real YouBot. The robot must move forward in a corridor for 2 meters to a pre-defined target, with an obstacle on its way. Differently from simulations presented in Section 4.3, we only assume that the walls are known in advance and modeled as superquadric potentials. On the contrary, the obstacle on the path of the robot is unknown, and it can be added to and moved away from the scene during the execution. Hence, the YouBot is equipped with a Realsense D435 RGB-depth camera from Intel as shown in Fig. 17, in order to record the point cloud of the scene in real-time. At each time-step, the point cloud is filtered along the world z-axis to remove the floor and on its own depth to remove points beyond the target. Then it is clustered into separate point clouds for each object in the scene and is registered with the previous point cloud in a common reference frame to update the scene Fig. 18. Finally an ellipsoid as in

Fig. 20 Main steps of the YouBot task with the obstacle added to and removed from the scene during the execution

a Start with no obstacle. **b** Avoiding obstacle right before it is removed. **c** Goal reached.



a Static potential, persistent obstacle. **b** Dynamic potential, persistent obstacle. **c** Static potential, temporal obstacle. **d** Dynamic potential, temporal obstacle.

Fig. 21 Results for the experiment with the YouBot with null forcing term. In each plot, the ellipse shows the modeled obstacle, and the dashed blue line shows the desired (obstacle-free) behavior. The full line shows the adapted trajectory. The red portion shows the trajectory executed without the presence of the obstacle, and the green portion the part of trajectory executed when the obstacle is present

Eq. 17 is fitted with $n_1 = n_2 = 1$, enlarging axes of the dimensions of the robot (since the motion of the robot is 2-dimensional, we consider only the planar coordinates of the ellipsoid). Fitting a pure ellipsoid rather than a pseudo-ellipsoid ($n_1 = n_2 = 2$) guarantees a smoother perturbation to the trajectory of the robot and leverages the real-time computational complexity. The camera and the YouBot controller communicate through a ROS network.

We control the robot with a 2-dimensional DMP with three different behaviors: null forcing term $\mathbf{f} \equiv \mathbf{0}$, constant velocity, and a half-circle trajectory. For each of the three DMPs' behaviors, we test two different scenarios. At first (Fig. 19) we add a box as an obstacle on the way to the goal right after the YouBot has started moving, and we keep the obstacle in position. In the second scenario (Fig. 20), we firstly add the box in the scene, and then we remove it after some time. We test both the static Eq. 15 and the dynamic Eq. 18 potential formulations.

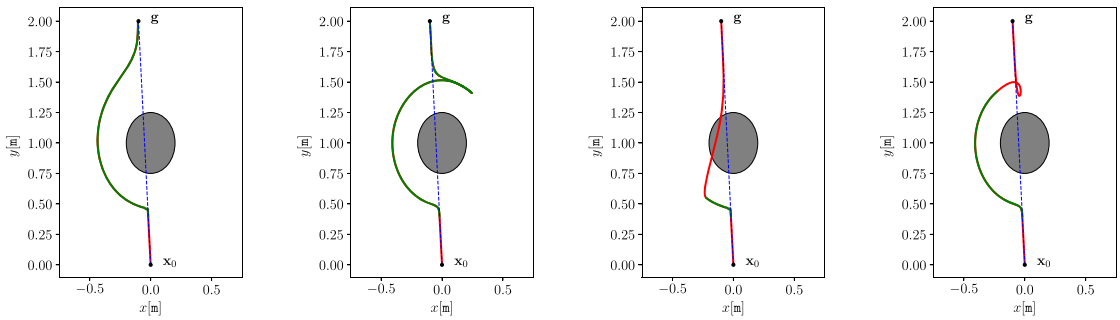
The DMP's parameters are $K = 500$, $D = 2\sqrt{K} \approx 44.72$, and $\alpha = 4$. The obstacle's parameters are $A = 1$, $\eta = 1$ for the static potential (15), and $\lambda = 1$, $\beta = 1$, and $\eta = 1$ for the dynamic potential (18).

Figure 21 shows the result for the null-forcing term DMP, Fig. 22 shows the result for the constant-velocity DMP, and Fig. 23 shows the result for the hal-circle DMP.

From these tests, we see that both static and dynamic methods result in the obstacle being successfully avoided; even if some differences emerge.

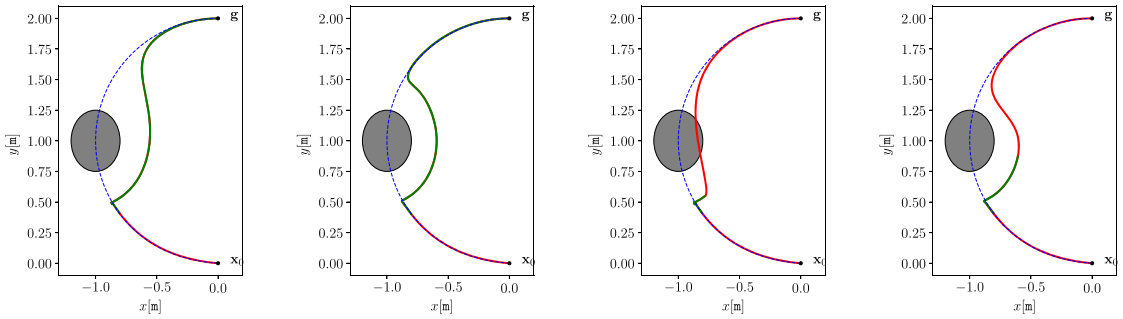
When the obstacle remains in the scene (subfigures (a) and (b)), we see that the dynamic potential (18) usually results in a trajectory that deviates less from the desired trajectory.

The tests in which the obstacle is removed (subfigures (c) and (d)) show that the static potential result in the robot remaining 'trapped' longer behind the obstacle, while the dynamic potential is able to deviate from the trajectory sooner.



a Static potential, persistent obstacle. **b** Dynamic potential, persistent obstacle. **c** Static potential, temporal obstacle. **d** Dynamic potential, temporal obstacle.

Fig. 22 Results for the experiment with the YouBot with constant velocity. In each plot, the ellipse shows the modeled obstacle, and the dashed blue line shows the desired (obstacle-free) behavior. The full line shows the adapted trajectory. The red portion shows the trajectory executed without the presence of the obstacle, and the green portion the part of trajectory executed when the obstacle is present



a Static potential, persistent obstacle **b** Dynamic potential, persistent obstacle **c** Static potential, temporal obstacle **d** Dynamic potential, temporal obstacle

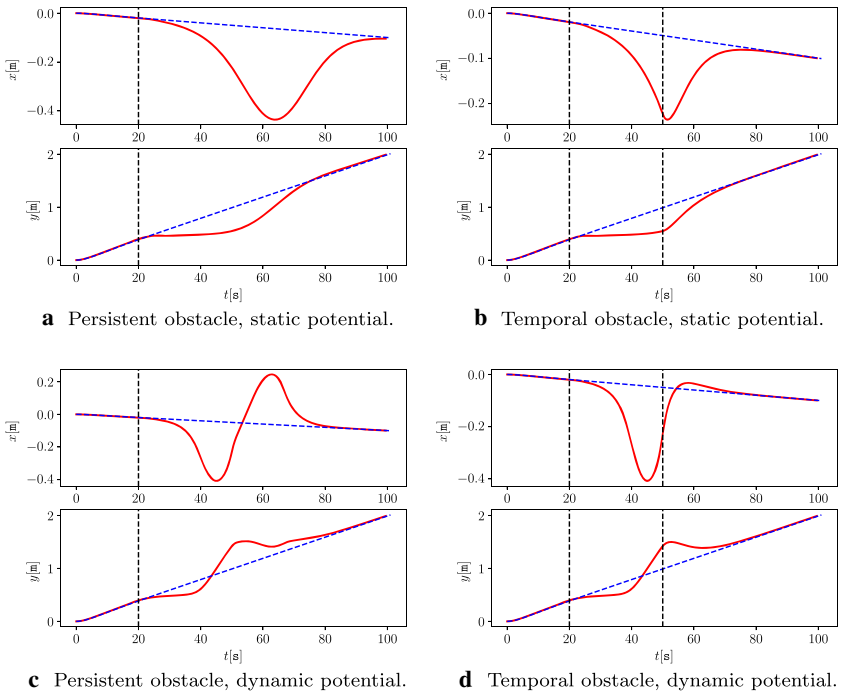
Fig. 23 Results for the experiment with the YouBot with the half-circle. In each plot, the ellipse shows the modeled obstacle, and the dashed blue line shows the desired (obstacle-free) behavior. The full line shows the adapted trajectory. The red portion shows the trajectory executed without the presence of the obstacle, and the green portion the part of trajectory executed when the obstacle is present

Figure 4.4.3 shows that the static potential result in an oscillatory behavior when the trajectory starts to deviate from the obstacle.

The only case in which the dynamic potential seems to be worse than the static one is for the trajectory with constant velocity (Fig. 22). This follows from the fact that, once the obstacle is surpassed, the dynamic potential is null and the trajectory is no longer pushed to the left, and the system is only pushed to the right by the DMP's dynamic. This reasoning is more clear to understand from Fig. 24, which plot the time evolution of the trajectories shown in Fig. 22.

5 Conclusions

In this paper we have presented a new dynamic potential formulation for obstacle avoidance with DMPs in the Cartesian space. This formulation extends our previous static potential one based on position, in that it takes into account the velocity of the system governed by the DMP and of the obstacle. We have designed synthetic experiments and tests with simulated and real robots to compare our frameworks with the state-of-the-art ones existing in the literature about DMPs. Experiments with



a Persistent obstacle, static potential. **b** Temporal obstacle, static potential. **c** Persistent obstacle, dynamic potential. **d** Temporal obstacle, dynamic potential.

Fig. 24 Solutions for the trajectories obtained with constant velocity DMP. In all plots, the blue dashed line shows the desired solution, and the red solid line shows the obstacle-avoidance behavior. In Fig. 24a and c the black dashed line shows the time at which the obstacle is inserted in the scene. In Fig. 24b and d the obstacle is present in the time interval marked by the two black dashed vertical lines

real robots are performed with an industrial manipulator, a surgical robot and a mobile robot, in order to show the generality of our framework. One advantage of our formulations is that they consider volumetric obstacles, instead of point-like obstacles as other state-of-the-art methods, guaranteeing a more stable behavior. Volumes are modeled with superquadric functions, which allow describing shapes of real objects with an arbitrary degree of approximation. The synthetic experiments show that our potential formulations guarantee smoother acceleration behavior and minimal deviation from the obstacle-free trajectory defined by the forcing term of the DMP. Moreover, the simulation experiments with three mobile robots show that our formulations can cope with multi-robot obstacle avoidance in real-time, without any predefined coordination strategy between the robots. Our new dynamic potential formulation generates fewer oscillations in the proximity of the obstacles with respect to the static potential one. In fact, the dynamic potential depends on the relative speed of the robot with respect to the obstacles, hence it deviates the trajectory earlier when the obstacle is approached. However, the experiments with real robots show that the dynamic potential can result in higher deviations from the original trajectory, depending on the forcing term of the DMP and the position of obstacles in the scene. On the contrary, the static potential performs better in all the experiments with the real robots, including the scenario with the mobile robot when an obstacle is added on its way during the execution. The experiments with the industrial manipulator and the surgical robot on a pick-and-place task show that our frameworks can scale to different dimensions of the setup. The major drawback of our formulations is that they do not guarantee convergence to the goal, which is a typical issue with potential-based formulations.

Future research will focus on the extension of our frameworks to the quaternion space. In fact, while the superquadric description of the volumes allows to approximate the shapes of real objects and to save more of the available workspace, the obstacle-aware adaptation of the orientation of the robot is not implemented at the moment. Hence, the obstacles should be enlarged of the dimension of the end effector. This is particularly evident in the scenario with the industrial manipulator, which has a huge end-effector. We have partially solved this issue in our experiments, exploiting the kinematic redundancy of the robot and an efficient inverse kinematics solver to generate obstacle-free joint configurations from the Cartesian DMP waypoints. However, this yields to higher computational time and slower execution. We believe that representing the obstacles directly in the quaternion space at the DMP level would improve the performances.

Author Contributions *Conceptualization:* Michele Ginesi. *Data curation:* Daniele Meli, Andrea Roberti. *Formal Analysis:* Michele Ginesi, Daniele Meli, Andrea Robeti. *Funding acquisition:* Paolo Fiorini. *Investigation:* Michele Ginesi, Daniele Meli, Andrea Roberti, Nicola Sansonetto. *Methodology:* Michele Ginesi, Daniele Meli, Andrea Roberti, Nicola Sansonetto. *Project administration:* Paolo Fiorini. *Resources:* Paolo Fiorini. *Software:* Michele Ginesi. *Supervision:* Nicola Sansonetto, Paolo Fiorini. *Validation:* Daniele Meli, Andrea Roberti. *Visualization:* Michele Ginesi, Daniele Meli, Andrea Roberti. *Writing – original draft:* Michele Ginesi, Daniele Meli. *Writing – review and editing:* Michele Ginesi, Daniele Meli, Nicola Sansonetto, Paolo Fiorini.

Funding Open access funding provided by Università degli Studi di Verona within the CRUI-CARE Agreement. This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, ARS (Autonomous Robotic Surgery) project, grant agreement No. 742671.

Availability of Data and Materials The presented framework is publicly available at https://github.com/mginesi/dmp_vol_obst.

Declarations

Conflict of Interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Albrecht, S., Ramirez-Amaro, K., Ruiz-Ugalde, F., Weikersdorfer, D., Leibold, M., Ulbrich, M., Beetz, M.: Imitating human reaching motions using physically inspired optimization principles. In: 2011 11th IEEE-RAS International Conference on Humanoid Robots, pp. 602–607. IEEE (2011)
2. Beeson, P., Ames, B.: Trac-Ik: An open-source library for improved solving of generic inverse kinematics. In: 2015 IEEE-RAS 15Th International Conference on Humanoid Robots (Humanoids), pp. 928–935. IEEE (2015)
3. Duan, J., Ou, Y., Hu, J., Wang, Z., Jin, S., Xu, C.: Fast and stable learning of dynamical systems based on extreme learning machine. IEEE Trans Syst Man Cybern. Syst. (99) 1–11 (2017)
4. Fahimi, F., Nataraj, C., Ashrafiuon, H.: Real-time obstacle avoidance for multiple mobile robots. Robotica **27**(2), 189 (2009)
5. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. **17**(7), 760–772 (1998)

6. Gams, A., Nemec, B., Ijspeert, A.J., Ude, A.: Coupling movement primitives: Interaction with the environment and bimanual tasks. IEEE Trans. Robot. **30**(4), 816–830 (2014)
7. Gasparetto, A., Zanotto, V.: A new method for smooth trajectory planning of robot manipulators. Mechan. Machine Theory **42**(4), 455–471 (2007)
8. Ginesi, M., Meli, D., Calanca, A., Dall'Alba, D., Sansonetto, N., Fiorini, P.: Dynamic movement primitives: Volumetric obstacle avoidance. In: 2019 19th International Conference on Advanced Robotics (ICAR), pp. 234–239 (2019). https://doi.org/10.1109/ICAR46387.2019.8981552
9. Ginesi, M., Sansonetto, N., Fiorini, P.: Overcoming some drawbacks of dynamic movement primitives. arXiv:1908.10608 (2019)
10. Hoffmann, H., Pastor, P., Park, D.H., Schaal, S.: Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference On, pp. 2587–2592. IEEE (2009)
11. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: Theory and applications. Neurocomputing **70**(1-3), 489–501 (2006)
12. Huang, R., Cheng, H., Guo, H., Chen, Q., Lin, X.: Hierarchical Interactive Learning for a Human-Powered Augmentation Lower Exoskeleton. In: Robotics and Automation (ICRA), 2016 IEEE International Conference On, pp. 257–263. IEEE (2016)
13. Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., Schaal, S.: Dynamical movement primitives: Learning attractor models for motor behaviors. Neural computation **25**(2), 328–373 (2013)
14. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference On, vol. 2, pp. 1398–1403. IEEE (2002)
15. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Learning attractor landscapes for learning motor primitives. In: Advances in Neural Information Processing Systems, pp. 1547–1554 (2003)
16. Joshi, R.P., Koganti, N., Shibata, T.: Robotic cloth manipulation for clothing assistance task using dynamic movement primitives. In: Proceedings of the Advances in Robotics, p. 14. ACM (2017)
17. Khansari-Zadeh, S.M., Billard, A.: Learning stable nonlinear dynamical systems with gaussian mixture models. IEEE Trans. Robot. **27**(5), 943–957 (2011)
18. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings 1985 IEEE International Conference on Robotics and Automation, vol. 2, pp. 500–505. IEEE (1985)
19. Khosla, P., Volpe, R.: Superquadric artificial potentials for obstacle avoidance and approach. In: Proceedings. 1988 IEEE International Conference on Robotics and Automation, pp. 1778–1784. IEEE (1988)
20. Lin, C., Chang, P., Luh, J.: Formulation and optimization of cubic polynomial joint trajectories for industrial robots. IEEE Trans. Autom. Control **28**(12), 1066–1074 (1983)
21. Magid, E., Keren, D., Rivlin, E., Yavneh, I.: Spline-based robot navigation. In: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference On, pp. 2296–2301. IEEE (2006)
22. Matsubara, T., Hyon, S.H., Morimoto, J.: Learning stylistic dynamic movement primitives from multiple demonstrations. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference On, pp. 1277–1283. Citeseer (2010)
23. Park, D.H., Hoffmann, H., Pastor, P., Schaal, S.: Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In: Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference On, pp. 91–98. IEEE (2008)
24. Pastor, P., Hoffmann, H., Asfour, T., Schaal, S.: Learning and generalization of motor skills by learning from demonstration. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference On, pp. 763–768. IEEE (2009)
25. Pastor, P., Kalakrishnan, M., Righetti, L., Schaal, S.: Towards associative skill memories. In: Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference On, pp. 309–315. IEEE (2012)
26. 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)
27. Perdereau, V., Passi, C., Drouin, M.: Real-time control of redundant robotic manipulators for mobile obstacle avoidance. Robot. Auton. Syst. **41**(1), 41–59 (2002)
28. Rai, A., Meier, F., Ijspeert, A., Schaal, S.: Learning coupling terms for obstacle avoidance. In: 2014 IEEE-RAS International Conference on Humanoid Robots, pp. 512–518. IEEE (2014)
29. Rai, A., Sutanto, G., Schaal, S., Meier, F.: Learning feedback terms for reactive planning and control. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 2184–2191. IEEE (2017)
30. Ratliff, N., Zucker, M., Bagnell, J.A., Srinivasa, S.: Chomp: Gradient optimization techniques for efficient motion planning. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference On, pp. 489–494. IEEE (2009)
31. Rezaee, H., Abdollahi, F.: Adaptive artificial potential field approach for obstacle avoidance of unmanned aircrafts. In: 2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 1–6. IEEE (2012)
32. Rimon, E., Koditschek, D.E.: Exact robot navigation using artificial potential functions. IEEE Trans. Robot. Autom. **8**(5), 501–518 (1992)
33. Roberti, A., Piccinelli, N., Meli, D., Fiorini, P.: Rigid 3d calibration in a robotic surgery scenario. Hamlyn Symposium on Medical Robotics (HSMR) in submission (2020)
34. Rohmer, E., Singh, S.P.N., Freese, M.: Coppeliasim (Formerly V-Rep): A versatile and scalable robot simulation framework. In: Proc. of The International Conference on Intelligent Robots and Systems (IROS) Www.coppeliarobotics.com (2013)
35. Saveriano, M., Franzel, F., Lee, D.: Merging position and orientation motion primitives. In: International Conference on Robotics and Automation (ICRA), 2019 (2019)
36. Schaal, S.: Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In: Adaptive Motion of Animals and Machines, pp. 261–280. Springer (2006)
37. Sutanto, G., Su, Z., Schaal, S., Meier, F.: Learning sensor feedback models from demonstrations via phase-modulated neural networks. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1142–1149. IEEE (2018)
38. 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)
39. Ude, A., Nemec, B., Petrić, T., Morimoto, J.: Orientation in cartesian space dynamic movement primitives. In: Robotics and Automation (ICRA), 2014 IEEE International Conference On, pp. 2997–3004. IEEE (2014)
40. Volpe, R.: Real and artificial forces in the control of manipulators: theory and experiments. Ph.D. thesis, PhD thesis, Carnegie Mellon University Department of Physics (1990)
41. Volpe, R., Khosla, P.: Manipulator control with superquadric artificial potential functions: Theory and experiments. IEEE Trans Syst Man Cybern **20**(6), 1423–1436 (1990)
42. 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: Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference On, pp. 3765–3771. IEEE (2016)
43. Yan, Z., Jouandeau, N., Cherif, A.A.: A survey and analysis of multi-robot coordination. Int. J. Adv. Robot. Syst. **10**(12), 399 (2013)
44. Zhang, W., Rodríguez-seda, E.J., Deka, S.A., Amrouche, M., Hou, D., Stipanović, D.M., Leitmann, G.: Avoidance control with relative velocity information for lagrangian dynamics. J. Intell. Robot. Syst. 1–16 (2019)

Michele Ginesi received his Master's Degree in Mathematics at the University of Verona in 2017. In the same year, he joined the Ph.D. program in Computer Science at the same University, working at the Altair Robotics Laboratory. His research focuses on the definition of a model to learn surgical gestures to automate simple interventions in the context of Minimally Invasive Robotic Surgery.

Daniele Meli received his Master's Degree in Automation Engineering at the Polytechnic University of Bari in 2017. In the same year, he joined the Ph.D. program in Computer Science at the University of Verona, working at the Altair Robotics Laboratory. His current research focuses on explainable AI tools for application in safe task planning and learning for Minimally Invasive Surgery.

Andrea Roberti received the M.S. degree in computer science from the University of Verona, Italy, in 2018, where he is currently working toward the Ph.D. degree in computer science. His research interests include computer vision with applications to robotics, planning and control of mobile sensors.

Nicola Sansonetto received his Laurea in Physics and Ph.D. in Mathematics at the University of Padova, Italy. He then was post-doc at the Department of Computer Science of the University of Verona and at the Department of Mathematics Levi-Civita at the University of Padova. In 2016 joined the Altair Lab at the University of Verona and in 2019 got the position of Associate researcher at the same University. His research is meanly devoted to physical and engineering applications of differential geometry, geometric control theory, and applied dynamical systems.

Paolo Fiorini received the Laurea in EE from the University of Padova, (Italy), the MSEE from the University of California at Irvine (USA), and the Ph.D. in ME from UCLA (USA). From 1977 to 1985 he worked on microprocessor-based controllers for consumer and industrial systems. From 1985 to 2000, he was with NASA Jet Propulsion Laboratory, California Institute of Technology, where he worked on autonomous and teleoperated systems for space experiments and exploration. In 2001 he joined the University of Verona (Italy) where is Full Professor of Computer Science. His research focuses on autonomous robots for medical and surgical applications.