Robotics and Autonomous Systems 60 (2012) 862–873

Contents lists available at SciVerse ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A path conditioning method with trap avoidance

Luis Gracia [a,*], Antonio Sala [a], Fabricio Garelli [b]

[a] Department of Systems Engineering and Control, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain

[b] CONICET and Universidad Nacional de La Plata, C.C.91 (1900), La Plata, Argentina

ARTICLE INFO

Article history:
Received 28 July 2011
Received in revised form
18 January 2012
Accepted 23 January 2012
Available online 30 January 2012

Keywords:
Path planning
Sliding mode
Collision avoidance

ABSTRACT

This work presents a sliding-mode method for robotic path conditioning. The proposal includes a trap avoidance algorithm in order to escape from trap situations, which are analogous to local minima in potential field-based approaches. The sliding-mode algorithm activates when the desired path is about to violate the robot workspace constraints, modifying it as much as necessary in order to fulfill all the constraints and reaching their limit surface at low speed. The proposed path conditioning algorithm can be used on-line, since it does not require a priori knowledge of the desired path, and improves the conventional conservative potential field-based approach in the sense that it fully exploits the robot workspace. The proposed approach can be easily added as an auxiliary supervisory loop to conventional robotic planning algorithms and its implementation is very easy in a few program lines of a microprocessor. The proposed path conditioning is compared through simulation with the conventional potential field-based approach in order to show the benefits of the method. Moreover, the effectiveness of the proposed trap avoidance algorithm is evaluated by simulation for various trap situations.

1. Introduction

One of the most relevant issues in robotics to perform tasks without human supervision is *path planning*, which involves finding a continuous path from the initial position to the goal position that avoids obstacles in the environment and which is feasible for the robot. This planning problem has been typically approached in two different ways, broadly classified as global and local methods [1]. Global planning is based on global knowledge of a structured environment, while local planning uses only local information gathered by proximity sensors in an unstructured environment to implement a reactive behavior for collision avoidance [2]. Most navigation architectures combine both types of approaches.

The framework of this work is a robot operating in a *structured environment* [1], where the location of the robot, its operation region and obstacles to avoid are known, while the user reference is *a priori* unknown and may not respect the workspace constraints. The use of *artificial potential fields* may be a way to address the problem. The basic concept of the potential field method [3] is to fill the robot's workspace with an artificial potential field in which the robot is attracted to its goal position and is repulsed away from the obstacles. However, unless the field is designed to abruptly decay at a short distance of the obstacle, some regions of the workspace close to the boundaries will not be reached because of repulsion.

This paper proposes an alternative solution to the above problem. The basic idea is to define a *discontinuous control law* inspired by the fact that, in the limit case, as the repulsion region decreases, a potential field could be characterized as a discontinuous force: zero away from the obstacle, a big value when touching its boundary. Discontinuous control laws, as a particular case of variable-structure control strategies, have been deeply studied in the context of sliding-mode control [4,5]. Indeed, many types of sliding-mode controllers have been developed in the past years for robotic systems [6,7]. Hence, the objective of this paper is to design a supervisory sliding-mode algorithm to modify the reference path so that the linear Cartesian coordinates of the robot's end-effector fulfill the desired workspace constraints. Sliding-mode-based supervisory blocks may also be used to handle joint speed limits [8] or to take advantage of kinematic redundancy [9].

The structure of the paper is as follows. Section 2 states the problem to be addressed, while Section 3 presents an overall description of the proposed approach. Afterwards, Section 4 develops the sliding mode path conditioning technique proposed to fulfill workspace constraints, while some remarks about it are given in Section 5. Next, Section 6 describes the trap avoidance algorithm proposed to escape from trap situations. Furthermore, Section 7 gives some guidelines to design the algorithm parameters, while a discussion about the method is given in Section 8. Section 9 presents simulation results to illustrate the performance

* Corresponding author.
E-mail addresses: luigraca@isa.upv.es (L. Gracia), asala@isa.upv.es (A. Sala), fabricio@ing.unlp.edu.ar (F. Garelli).

0921-8890/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2012.01.009

of the proposed approach. Finally, some concluding remarks are given.

2. Problem statement

Let us denote as $\mathbf{p}_{\text{ref}}(t)$ the workspace position reference of the robotic system, which can be usually expressed in terms of a desired path[1] function $\mathbf{v}(\lambda)$ whose argument is the so-called motion parameter $\lambda(t)$ as:

$$\mathbf{p}_{\text{ref}} = \mathbf{v}(\lambda). \tag{1}$$

Assumption 1. It will be assumed that $\mathbf{p}_{\text{ref}}$ is twice differentiable and that $\ddot{\mathbf{p}}_{\text{ref}}$ is reasonably bounded.

We consider that the robotic system to be controlled is subjected to workspace constraints given by:

$$\Phi_{WS}(\mathbf{p}) = \{\mathbf{p} \mid \sigma_i(\mathbf{p}) \leq 0\}, \quad i = 1, \ldots, N, \tag{2}$$

where σ_i is a function of the workspace position coordinate[2] $\mathbf{p}$ that is positive if and only if the ith-constraint is not fulfilled. Note that, $\sigma_i(\mathbf{p}) = 0$ represents the boundary of the ith-constraint. For instance, a constraint $\sigma_{\text{sphere}} = 1 - \|\mathbf{p}\|_2 \leq 0$ would indicate that the allowed workspace Φ_{WS} is included outside a sphere of radius 1, centered at the origin.

Assumption 2. It will be assumed that the functions σ_i are twice differentiable around the boundary given by $\sigma_i(\mathbf{p}) = 0$ and that their gradients[3] $\nabla\sigma_i$ around this boundary do not vanish.

For non-differentiable constraints, there are techniques in literature [10] that may be used to enclose such non-smooth regions by smooth mathematical objects with an arbitrary degree of precision.

The control goal can therefore be stated as to generate a modified position reference $\mathbf{p}^*_{\text{ref}}$ to be sent to the robotic control system so that it is as close as possible to the user-input value $\mathbf{p}_{\text{ref}}$ and that belongs to the allowed workspace Φ_{WS} given by (2).

3. Overall description of the proposed approach

3.1. Improvement of the constraint space

At this point it is important to consider the following rationale. Approaching the constraints at high speed is impractical because collisions might occur if the joint accelerations required to slow down the motion of the robot towards the constraint boundary cannot be achieved by the robot actuators due to power limitations. This is of particular significance in mechanical or robotics systems in which the inertia is large. Hence, the actual constraint space (2) will be modified to also include the speed of movement in the following way:

$$\Phi^*_{WS}(\mathbf{p}, \dot{\mathbf{p}}) = \left\{ [\mathbf{p}^{\mathrm{T}} \dot{\mathbf{p}}^{\mathrm{T}}]^{\mathrm{T}} \mid \phi_i(\mathbf{p}, \dot{\mathbf{p}}) = \sigma_i(\mathbf{p}) + K \frac{d\sigma_i(\mathbf{p})}{dt} = \sigma_i + K \nabla\sigma_i^{\mathrm{T}} \dot{\mathbf{p}} \leq 0 \right\}, \quad i = 1, \ldots, N, \tag{3}$$

where $\sigma_i(\mathbf{p})$ is the original ith workspace constraint and K is the *constraint approaching parameter*, which is a free design parameter that determines the rate of approach to the boundary of the constraints. Thus, expression (3) introduces an additional degree of freedom necessary to reach the limit in a controlled fashion. That is, the term $K\dot{\sigma}_i$ is used to anticipate when the robot is about to violate the ith-constraint in order to initiate an early corrective action, which will increase the numerical stability of the proposed algorithm. Note that $\dot{\sigma}_i = \nabla\sigma_i^{\mathrm{T}} \dot{\mathbf{p}}$. Thus, for low speeds or small K values $\Phi^*_{WS} \approx \Phi_{WS}$ given by (2). Note also that K may take different values for different constraints, if so wished.

3.2. Diagram of the method

Fig. 1 shows the diagram of the proposed method to perform an on-line robotic path conditioning so that environment limits given by Eq. (3) are fulfilled. The objective of this path conditioning algorithm (PCA) is to instantaneously modify the given position reference of the desired path $\mathbf{p}_{\text{ref}}$ to a possibly different value $\mathbf{p}^*_{\text{ref}}$ which is sent to the robotic control system when there is a risk of violating a given constraint. Moreover, a trap avoidance algorithm (TAA) is also included in Fig. 1 in order to guarantee that if no constraints are violated the value $\mathbf{p}^*_{\text{ref}}$ tends to $\mathbf{p}_{\text{ref}}$.

The commanded path is shaped by modifying the position reference $\mathbf{p}_{\text{ref}}$ as follows:

$$\mathbf{p}^*_{\text{ref}} = \mathbf{p}_{\text{ref}} + \mathbf{f}_{\text{SM}} + \mathbf{f}_{\text{RW}}, \tag{4}$$

where $\mathbf{f}_{\text{SM}}$ is the *correcting action* to the original path and $\mathbf{f}_{\text{RW}}$ is a *random walk* signal to escape from trap situations, as discussed in Section 6.

Signal $\mathbf{f}_{\text{SM}}$ is generated by passing the discontinuous signal $\mathbf{u}_1$ through a low-pass filter, as shown in Fig. 1. This filter smooths out the signal added to the main control loop and it must be of second-order for $\ddot{\mathbf{p}}^*_{\text{ref}}$ to explicitly depend on $\mathbf{u}_1$, as required in Section 4. Particularly, the following second-order Butterworth low-pass filter could be used:

$$\ddot{\mathbf{f}}_{\text{SM}} = -\sqrt{2}\alpha_1 \dot{\mathbf{f}}_{\text{SM}} - \alpha_1^2 \mathbf{f}_{\text{SM}} + \alpha_1^2 \mathbf{u}_1, \tag{5}$$

with the scalar α_1 being the filter cutoff frequency.

Signal $\mathbf{f}_{\text{RW}}$ is generated by passing the discontinuous signal $\mathbf{u}_2$ through a low-pass filter and an integrator, as shown in Fig. 1. The integrator gives rise to a random walk signal whose variance increases monotonically with time, while the filter smooths out the signal added to the main control loop. This filter must be of first-order for velocity signal $\dot{\mathbf{p}}^*_{\text{ref}}$ to be continuous:

$$\ddot{\mathbf{f}}_{\text{RW}} = -\alpha_2 \dot{\mathbf{f}}_{\text{SM}} + \alpha_2 \mathbf{u}_2, \tag{6}$$

with the scalar α_2 being the filter cutoff frequency.

Fig. 1 also shows a third loop whose objective is to stop the motion of the original reference $\mathbf{p}_{\text{ref}}$ when the TAA is active in order to avoid tracking errors. For this purpose, signal f_{SP} acts as a scale factor for the commanded motion rate $\dot{\lambda}_c$. The control signal f_{SP} is generated by passing the discontinuous signal u_{SP} through a first order low-pass filter in order to obtain a continuous velocity signal $\dot{\lambda}$.

3.3. State equation and Lie derivatives

The dynamical system given by (4)–(6) has the following state equation:

$$\begin{bmatrix} \dot{\mathbf{p}}^*_{\text{ref}} \\ \ddot{\mathbf{p}}^*_{\text{ref}} \\ \dot{\mathbf{f}}_{\text{RW}} \\ \ddot{\mathbf{f}}_{\text{RW}} \end{bmatrix} = \begin{bmatrix} \mathbf{O}_n & \mathbf{I}_n & \mathbf{O}_n & \mathbf{O}_n \\ -\alpha_1^2 \mathbf{I}_n & -\sqrt{2}\alpha_1 \mathbf{I}_n & \alpha_1^2 \mathbf{I}_n & (\sqrt{2}\alpha_1 - \alpha_2)\mathbf{I}_n \\ \mathbf{O}_n & \mathbf{O}_n & \mathbf{O}_n & \mathbf{I}_n \\ \mathbf{O}_n & \mathbf{O}_n & \mathbf{O}_n & -\alpha_2 \mathbf{I}_n \end{bmatrix} \times \begin{bmatrix} \mathbf{p}^*_{\text{ref}} \\ \dot{\mathbf{p}}^*_{\text{ref}} \\ \mathbf{f}_{\text{RW}} \\ \dot{\mathbf{f}}_{\text{RW}} \end{bmatrix} + \begin{bmatrix} \mathbf{0}_n \\ \alpha_1^2 \mathbf{p}_{\text{ref}} + \sqrt{2}\alpha_1 \dot{\mathbf{p}}_{\text{ref}} + \ddot{\mathbf{p}}_{\text{ref}} \\ \mathbf{0}_n \\ \mathbf{0}_n \end{bmatrix} + \begin{bmatrix} \mathbf{O}_n & \mathbf{O}_n \\ \alpha_1^2 \mathbf{I}_n & \alpha_2 \mathbf{I}_n \\ \mathbf{O}_n & \mathbf{O}_n \\ \mathbf{O}_n & \alpha_2 \mathbf{I}_n \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix}$$

[1] The term *path* is used in this work to refer to the spatial positions followed by three linear coordinates, e.g. the Cartesian position of the robot end-effector.

[2] The constraints could also be defined in terms of both the position and orientation of the robot end-effector, i.e. $\sigma_i(\mathbf{p}, \mathbf{o})$, or the robot configuration $\mathbf{q}$, see Section 5.3.

[3] The gradient of a scalar function $f(x_1, \ldots, x_n)$ is denoted $\nabla f = [\frac{\partial f}{\partial x_1} \ldots \frac{\partial f}{\partial x_n}]^{\mathrm{T}}$.

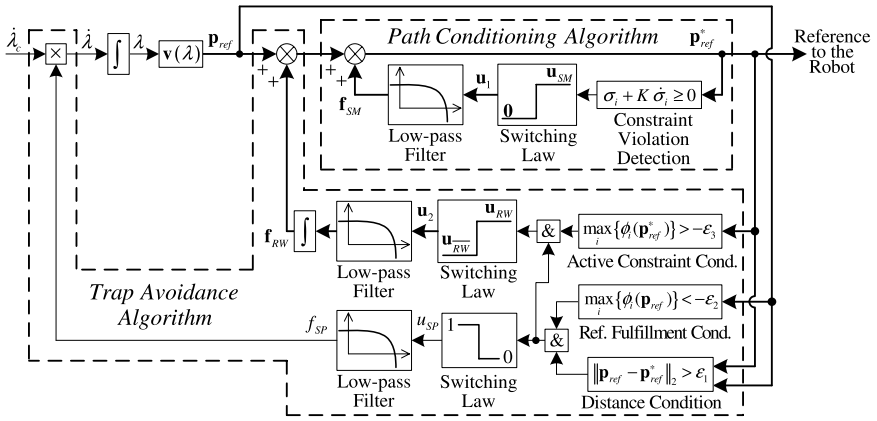

Fig. 1. Proposed method to fulfill workspace constraints.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{d}) + \mathbf{g}(\mathbf{x})\,\mathbf{u}, \tag{7}$$

where n is the dimension of position vector $\mathbf{p}$, matrices $\mathbf{O}_n$ and $\mathbf{I}_n$ denote the null and identity matrices of dimensions $n \times n$, respectively, vector $\mathbf{0}_n$ denotes the n-dimensional null column vector, vector $\mathbf{x}$ is the system *state*, vector $\mathbf{d}$ is the system *disturbance* (which is composed of the original position reference $\mathbf{p}_{\text{ref}}$ and its derivatives), vector $\mathbf{u}$ is the system control *input* (possibly discontinuous) and $\mathbf{f}$ and $\mathbf{g}$ are a vector field and a set of $2n$ vector fields, respectively.

Using the above state equation, the Lie derivatives of ϕ_i in the direction of vector field $\mathbf{f}$ and in the direction of the set of vector fields $\mathbf{g}$ are given by:

$$\partial\phi_i/\partial\mathbf{x} = \begin{bmatrix}\nabla\sigma_i^{\mathrm{T}} + K\dot{\mathbf{p}}_{\text{ref}}^{*\mathrm{T}}\mathbf{H}_i & K\nabla\sigma_i^{\mathrm{T}} & \mathbf{O}_n & \mathbf{O}_n\end{bmatrix}^{\mathrm{T}}, \tag{8}$$

$$\begin{aligned}\mathbf{L}_{\mathbf{g}}\phi_i &= (\partial\phi_i/\partial\mathbf{x})^{\mathrm{T}}\,\mathbf{g} = \begin{bmatrix}\alpha_1^2 K\nabla\sigma_i^{\mathrm{T}} & \alpha_2 K\nabla\sigma_i^{\mathrm{T}}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{L}_{\mathbf{g}}\phi_{i_n} & \mathbf{L}_{\mathbf{g}}\phi_{i_{2n}}\end{bmatrix},\end{aligned} \tag{9}$$

$$\begin{aligned}L_f\phi_i &= (\partial\phi_i/\partial\mathbf{x})^{\mathrm{T}}\,\mathbf{f} = K\nabla\sigma_i^{\mathrm{T}}\left(\alpha_1^2\mathbf{p}_{\text{ref}} + \sqrt{2}\alpha_1\dot{\mathbf{p}}_{\text{ref}} + \ddot{\mathbf{p}}_{\text{ref}}\right) \\ &\quad + K\dot{\mathbf{p}}_{\text{ref}}^{*\mathrm{T}}\mathbf{H}_i\dot{\mathbf{p}}_{\text{ref}}^{*} + K\nabla\sigma_i^{\mathrm{T}}\Bigg(\alpha_1^2(\mathbf{f}_{\text{RW}} - \mathbf{p}_{\text{ref}}^{*}) \\ &\quad + (\sqrt{2}\alpha_1 - \alpha_2)\dot{\mathbf{f}}_{\text{RW}} - (\sqrt{2}\alpha_1 - 1/K)\dot{\mathbf{p}}_{\text{ref}}^{*}\Bigg),\end{aligned} \tag{10}$$

where $\mathbf{H}_i$ denotes the Hessian matrix of second-order partial derivatives of σ_i. Note that $\mathbf{L}_{\mathbf{g}}\phi_i$ is a $2n$-dimensional row vector and $L_f\phi_i$ is a scalar value whose first term depends on the disturbances, i.e. the original position reference $\mathbf{p}_{\text{ref}}$ and its derivatives.

It is important to remark that $L_f\phi_i$ has a definite value in (10) only if the first part of both assumptions made in Section 2 are fulfilled. Note also that the first n elements of the row vector $\mathbf{L}_{\mathbf{g}}\phi_i$ (namely $\mathbf{L}_{\mathbf{g}}\phi_{i_n}$) are not the null vector since the values of α_1, K and $\nabla\sigma_i$ are not zero[4] from the above definitions and the second part of Assumption 2.

4. Path conditioning algorithm

4.1. Invariance via sliding regimes

To ensure the fulfillment of the constraints, the following invariance condition must be satisfied:

$$\dot{\phi}_i(\mathbf{x}, \mathbf{d}, \mathbf{u}) = \nabla\phi_i^{\mathrm{T}}\dot{\mathbf{x}} = \nabla\phi_i^{\mathrm{T}}\mathbf{f} + \nabla\phi_i^{\mathrm{T}}\mathbf{g}\,\mathbf{u}$$

$$= L_f\phi_i + \mathbf{L}_{\mathbf{g}}\phi_i\begin{bmatrix}\mathbf{u}_1 \\ \mathbf{u}_2\end{bmatrix} \leq 0, \quad \forall i \mid \phi_i(\mathbf{x}) = 0. \tag{11}$$

In this work, the above condition will be achieved by means of control signal $\mathbf{u}_1$, whereas control signal $\mathbf{u}_2$ will be designed to not affect this condition, see Section 6.3. Therefore, expression (11) is rewritten as:

$$\dot{\phi}_i = L_f\phi_i + \mathbf{L}_{\mathbf{g}}\phi_{i_n}\mathbf{u}_1 \leq 0, \quad \forall i \mid \phi_i(\mathbf{x}) = 0. \tag{12}$$

The following variable structure control law is considered:

$$\mathbf{u}_1 = \begin{cases}\mathbf{0}_n & \text{if } \max_i \phi_i < 0 \\ \mathbf{u}_{\text{SM}} & \text{otherwise,}\end{cases} \tag{13}$$

where $\mathbf{u}_{\text{SM}}$ is chosen to fulfill:

$$L_f\phi_i + \mathbf{L}_{\mathbf{g}}\phi_{i_n}\mathbf{u}_{\text{SM}} < 0, \quad \forall i \mid \phi_i(\mathbf{x}) \geq 0, \tag{14}$$

in order to satisfy (12).

The above control law leads to a sliding regime [4] (i.e., control signal $\mathbf{u}_1$ switches between $\mathbf{0}_n$ and $\mathbf{u}_{\text{SM}}$ with a theoretically infinite frequency) on the boundary of the ith-constraint if around this boundary $L_f\phi_i > 0$, i.e., whenever the system tries by itself to leave the allowed region. Thus, the switching law (13) does not seek for sliding mode (SM), but it becomes active if the process is at the boundary of the allowed region and about to leave it.

4.1.1. Single active constraint

For only one active constraint ϕ_i, there exists a value of $\mathbf{u}_{\text{SM}}$ fulfilling (14) only if:

$$\mathbf{L}_{\mathbf{g}}\phi_{i_n} \neq \mathbf{0}_n^{\mathrm{T}}, \tag{15}$$

which is known as *transversality condition* [11] and imposes that the sliding manifold must have *unitary relative degree* with respect to the discontinuous action, i.e., its first-order time derivative ($\dot{\phi}_i$) must explicitly depend on $\mathbf{u}_1$. This condition is always fulfilled by the proposed method, see (9) and Section 3.3.

For this case, the minimum-Euclidean-norm control action $\mathbf{u}_{\text{SM}}$ fulfilling (14) is given by a vector parallel to $\mathbf{L}_{\mathbf{g}}\phi_{i_n}^{\mathrm{T}}$:

$$\mathbf{u}_{\text{SM mini}} = -\mathbf{L}_{\mathbf{g}}\phi_{i_n}^{\mathrm{T}}\frac{L_f\phi_i}{\mathbf{L}_{\mathbf{g}}\phi_{i_n}\mathbf{L}_{\mathbf{g}}\phi_{i_n}^{\mathrm{T}}}. \tag{16}$$

Therefore, for only one active constraint it is proposed to use the following expression for the control action $\mathbf{u}_{\text{SM}}$:

$$\mathbf{u}_{\text{SM}} = -\mathbf{L}_{\mathbf{g}}\phi_{i_n}^{\mathrm{T}}u^{+} = -\alpha_1^2 K\nabla\sigma_i u^{+}, \tag{17}$$

where u^{+} is a positive constant to be chosen high enough to establish a SM on the constraint boundary. To fulfill that, the scalar factor u^{+} must be:

$$u^{+} > \frac{\max(L_f\phi_i, 0)}{\mathbf{L}_{\mathbf{g}}\phi_{i_n}\mathbf{L}_{\mathbf{g}}\phi_{i_n}^{\mathrm{T}}} = u^{\phi_i}. \tag{18}$$

[4] If the chosen value of K had been zero, i.e., no speed limitations were desired, then the low-pass filter of the PCA should have been first order, and the state $\mathbf{x}$ would not include $\dot{\mathbf{p}}$, in order to obtain a non-zero value for $\mathbf{L}_{\mathbf{g}}\phi_{i_n}$. Details omitted for brevity.

Once the switching SM is established on the constraint boundary by the control action $\mathbf{u}_{SM}$, the continuous equivalent control [4] is obtained, which is the control required to keep the system just on the constraint boundary. Consequently, the SM generated by switching law (13) produces the minimal value u^{ϕ_i} (without explicit knowledge of it) for the continuous equivalent of u^+ in order to achieve the invariance condition $\dot{\phi}_i \leq 0$.

4.1.2. Multiple active constraints

In case several constraints (say h constraints) are active, a function vector $\boldsymbol{\phi}$ and its time derivative $\dot{\boldsymbol{\phi}}$ composed of all active constraints should be considered in (12), so the constraints can be kept holding if the linear system of inequalities:

$$L_f\boldsymbol{\phi} + \mathbf{L_g}\boldsymbol{\phi}_n\mathbf{u}_1 \leq \mathbf{0}_h, \tag{19}$$

admits a solution. Indeed, the non-active constraints admit a free $\mathbf{u}_1$, hence only the active ones should be considered in the control law computation. Of course, such set of active constraints changes with time.

In the spirit of the discussion for the above one-constraint case, one of such control laws consists in generalizing (17) to:

$$\mathbf{u}_{SM} = -\mathbf{L_g}\boldsymbol{\phi}_n^{\mathrm{T}}\mathbf{1}_h u^+ = -\alpha_1^2 K \nabla\boldsymbol{\sigma}\mathbf{1}_h u^+, \tag{20}$$

where $\mathbf{1}_h$ is the h-dimensional column vector with all its components equal to one, matrix $\nabla\boldsymbol{\sigma}$ contains the gradient vectors $\nabla\sigma_i$ of all active constraints and u^+ is again a positive constant to be chosen high enough to establish a SM on the constraint boundary. To fulfill that, u^+ must be [9]:

$$u^+ > \sum_{i=1}^{h} \left(\max(L_f\phi_i, 0)\right) / \mathrm{eig}_{\min}(\mathbf{L_g}\boldsymbol{\phi}_n\mathbf{L_g}\boldsymbol{\phi}_n^{\mathrm{T}}) = u^{\boldsymbol{\phi}}, \tag{21}$$

i.e., its value depends on the inverse of the minimum singular value of $\mathbf{L_g}\boldsymbol{\phi}_n$. Therefore, in this case the condition (15) must be now changed to the "multiple-constraint" transversality condition of matrix $\mathbf{L_g}\boldsymbol{\phi}_n$ (i.e., matrix $\nabla\boldsymbol{\sigma}^{\mathrm{T}}$) having *full row rank*, which indeed covers the previous single-constraint case (a one-row matrix is full rank if there exists a non-zero element).

As in the previous single-constraint case, once the switching SM is established on the constraint boundary by the control action $\mathbf{u}_{SM}$, a continuous equivalent control [4] is obtained which is the control required to keep the system just on the boundary manifold. Consequently, the sliding regime generated by switching law (13) produces such equivalent control without explicit knowledge of it, with a reasonably low computational cost, as discussed below.

4.2. Reduced computation time

Although Eqs. (21) and (9)–(10) propose a lower bound for u^+ to be used in (20), the selection of this scalar factor can be made in a simple manner by choosing a high-enough constant. In this way, fast computation can be achieved. This is a distinctive advantage of SM algorithms [5].

Indeed, from the potential field approach point of view, the value u^+ is interpreted as the "repulsion" of the discontinuous field. This SM approach has the advantage that the magnitude $\|\mathbf{f}_{SM}\|_2$ of the correcting action required to avoid the constrained space is robustly *auto-regulated* to the continuous equivalent control [4]. This magnitude could also be computed at each iteration using (10), but it would require much more computational power than, plainly, setting u^+ to a big number which, due to the equivalent-control principle, computes the required quantity by a high-frequency switching law without explicit knowledge of the Hessian matrices $\mathbf{H}_i$, the reference $\mathbf{p}_{ref}$ and its derivatives, etc.

4.3. Strong invariance

Another distinctive feature of sliding regimes is that they are not affected by disturbances/uncertainties $\mathbf{d}$ that are collinear with the discontinuous action in the state equation, i.e., that satisfy the so-called *matching condition*. In that case, it is said that the variable structure control given by (13) presents *strong invariance* to disturbance $\mathbf{d}$ [4].

The matching condition is verified for the disturbances $\mathbf{p}_{ref}$, $\dot{\mathbf{p}}_{ref}$ and $\ddot{\mathbf{p}}_{ref}$ in the PCA, since they are collinear with the discontinuous action $\mathbf{u}_1$ in the state equation (7). Thus, the SM path conditioning presents strong invariance to the uncertainties and/or inaccuracies of the reference trajectory (i.e., $\mathbf{p}_{ref}$ and its derivatives) generated by the planner.

Note also that, the PCA is *robust* against environment modeling errors, i.e., the PCA is not affected by the inaccuracies and uncertainties of the Hessian matrices $\mathbf{H}_i$ of second-order partial derivatives of the constraint functions σ_i.

4.4. Switching frequency and chattering

As in all SM controls, the theoretically infinite switching frequency cannot be achieved in practice because all physical systems have finite bandwidth. In computer implementations, the switching frequency is directly the inverse of the sampling period. Finite-frequency commutation makes the system leave the theoretical SM and, instead, its states oscillates with finite frequency and amplitude inside a "band" around $\boldsymbol{\phi} = \mathbf{0}$, which is known as 'chattering' [5].

For active constraints, the chattering band $\Delta\phi_i$ due to the PCA is given, using the Euler-integration, by:

$$\begin{aligned}\Delta\phi_i &= T_{PC}|\mathbf{L_g}\phi_{i_n}\mathbf{u}_{SM}| = T_{PC}\alpha_1^2 K|\nabla\sigma_i^{\mathrm{T}}\mathbf{u}_{SM}| \\ &\leq T_{PC}\alpha_1^2 K\|\mathbf{u}_{SM}\|_2\|\nabla\sigma_i\|_2,\end{aligned} \tag{22}$$

where T_{PC} is the sampling period of the PCA, $\|\mathbf{u}_{SM}\|_2$ is the amplitude of the control action and $\|\nabla\sigma_i\|_2$ is the amplitude of the gradient vector, i.e., the maximum directional derivative of the constraint function σ_i.

The signal σ_i is obtained by passing signal ϕ_i through a first-order low-pass filter whose cutoff frequency is equal to $1/K$, see (3). This filter smooths out the chattering band of ϕ_i. In the worst case the chattering band $\Delta\sigma_i$ is equal to $\Delta\phi_i$ and it is reduced as the chattering frequency ω_ϕ and/or the constraint approaching parameter K increase. Thus the upper bound $\overline{\sigma}_{max}$ for signal σ_i results in:

$$\overline{\sigma}_{max} = T_{PC}\alpha_1^2 K\|\mathbf{u}_{SM}\|_2\|\nabla\sigma_i\|_2. \tag{23}$$

5. Additional remarks about the path conditioning

5.1. Constraint definition

It is advisable to properly define all σ_i functions so that their orders of magnitude are comparable, for instance, being related to the minimum distance from $\mathbf{p}^*_{ref}$ to the boundary of the ith-constraint. Say, we may consider the constraint $\sigma_{sphere} = 1 - \|\mathbf{p}^*_{ref}\|_2 \leq 0$ to indicate that the allowed workspace is included outside a sphere of radius 1, centered at the origin. However, instead of using σ_{sphere} we could have used $5\sigma_{sphere}$, or σ^3_{sphere}, etc. Therefore, in this case the first option might be advised if similar criteria are used with the rest of constraints.

5.2. Security margin

In case it is needed for security reasons, the original constraint functions σ_i may be designed conservatively, taking into account the estimated chattering amplitude $\overline{\sigma}_{max}$ and any other additional extra margin to cater for possible inaccuracies of the robot controller while tracking the reference signal or inaccuracies in the environment description.

5.3. Configuration space constraints

The proposed approach can also be used if the trajectory reference is expressed in terms of joint coordinates, e.g. the robot configuration $\mathbf{q}$, instead of workspace coordinates. In this case, the constraints must be given in the joint space or *configuration space* and all developments keep unchanged except for changing $\mathbf{p}$ to $\mathbf{q}$. For example, this method can be used to guarantee that all parts of the robot – not only the end-effector position – fulfill 3D-spatial constraints, which is useful in some applications [12]. The method can also be applied, for instance, to avoid configurations where the robot kinematics is close to being *singular*, e.g. elbow, shoulder and wrist singularities [13].

5.4. Non-static environment

The proposed approach can also be used if there are moving constraints, e.g. moving obstacles with *known* trajectories. In this case σ_i also depends explicitly on time and, hence, the derivative of ϕ_i in Eq. (12) must be replaced by $\dot{\phi}_i = \widetilde{L_f\phi_i} + \mathbf{L_g}\phi_{in}\mathbf{u}_1$, where $\widetilde{L_f\phi_i}$ is equal to $L_f\phi_i + \partial\sigma_i/\partial t + K(\partial^2\sigma_i/\partial t^2 + 2\partial\nabla\sigma_i^{\mathrm{T}}/\partial t\dot{\mathbf{p}}^*_{\mathrm{ref}})$, and $\mathbf{L_g}\phi_{in}$ and $L_f\phi_i$ are given again by (9) and (10), respectively (the explicit derivation of the above expression is omitted for brevity). Therefore, all developments keep unchanged except for changing $L_f\phi_i$ to $\widetilde{L_f\phi_i}$. Thus, only the value of the lower bound for u^+ is changed when moving constraints are considered and, hence, the iterative computation of the PCA remains the same.

6. Trap avoidance algorithm

6.1. Introduction

A trap situation is defined as the case where the user-supplied $\mathbf{p}_{\mathrm{ref}}$ temporally enters an invalid region and, after $\mathbf{p}_{\mathrm{ref}}$ reentering the valid region, the PCA output $\mathbf{p}^*_{\mathrm{ref}}$ does not converge again to it. An illustrative 2D example of trap or blocking situation is depicted in Fig. 2, where the modified position reference $\mathbf{p}^*_{\mathrm{ref}}$ will be blocked on the surface region $\partial\Phi_{1A}$ of boundary $\partial\Phi_1$ and will "lose" the original position reference $\mathbf{p}_{\mathrm{ref}}$. This phenomenon is analogous to local minima and limit cycles in potential field-based approaches and, similarly, it can be avoided if the proposed PCA is combined with a *random walk* (RW) [1] which is activated when the trap situation is detected in order to escape from it, as discussed below.

6.2. Proposed method

In order to avoid the trap situations described above, we propose in this work the *heuristic* Trap Avoidance Algorithm (TAA) shown in Fig. 1, which is based on a purely *reactive* behavior (i.e., no planning required) and consists of two supervisory loops, as explained below.

The *outer loop* stops the motion ($u_{\mathrm{SP}} = 0$) of the original reference $\mathbf{p}_{\mathrm{ref}}$ when, firstly, the distance from the modified reference $\mathbf{p}^*_{\mathrm{ref}}$ to $\mathbf{p}_{\mathrm{ref}}$ is above a given threshold ε_1 and, secondly, $\mathbf{p}_{\mathrm{ref}}$ is clearly within the allowed region, i.e. $\max_i \phi_i(\mathbf{p}_{\mathrm{ref}}) < -\varepsilon_2$, where ε_2 is a given positive threshold. Thus, path error is avoided while the modified reference $\mathbf{p}^*_{\mathrm{ref}}$ converges to $\mathbf{p}_{\mathrm{ref}}$. A first-order low-pass filter smooths out the control signal u_{SP} generated by this supervisory loop in order to obtain the continuous velocity signal $\dot{\lambda}$. The filtered control signal f_{SP} acts as a scale factor for the commanded motion rate $\dot{\lambda}_c$.

The *inner loop* activates a RW ($\mathbf{u}_2 = \mathbf{u}_{\mathrm{RW}}$) when the above stop conditions are satisfied and some constraint is active, i.e. $\max_i \phi_i(\mathbf{p}^*_{\mathrm{ref}}) > -\varepsilon_3$, where ε_3 is a small positive threshold depending on the chattering given by the SM PCA. Otherwise, the RW returns to zero, i.e., $\mathbf{u}_2 = \mathbf{u}_{\overline{RW}}$. A first-order low-pass filter smooths out the control signal $\mathbf{u}_2$ to obtain the continuous signal $\dot{\mathbf{f}}_{\mathrm{RW}}$, which represents the RW speed vector. This signal is integrated by an integrator to obtain the RW position $\mathbf{f}_{\mathrm{RW}}$, which is added to the original position reference $\mathbf{p}_{\mathrm{ref}}$. Note that this integration gives rise to a RW position whose variance increases monotonically with time whenever the RW is not returning to zero, i.e., whenever $\mathbf{u}_2 = \mathbf{u}_{\mathrm{RW}}$. This increasing-variance property allows to explore all the workspace in order to escape from the trap situation. The computation of both $\mathbf{u}_{\mathrm{RW}}$ and $\mathbf{u}_{\overline{RW}}$ is detailed below.

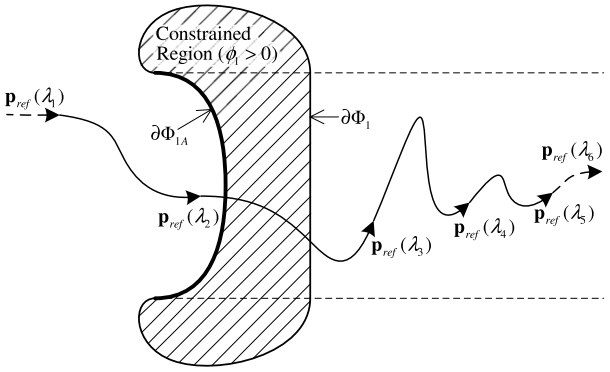

Fig. 2. 2D example of a trap situation.

6.3. Computation of the random walk

In order for the RW to return to zero, the vector $\mathbf{u}_{\overline{RW}}$ is computed as follows:

$$\mathbf{u}_{\overline{RW}} = -K_e\mathbf{f}_{\mathrm{RW}}, \tag{24}$$

where K_e is a positive constant giving rise to the specific dynamics of the RW when returning to zero, which is given by the following two poles:

$$\mathrm{poles}_{\overline{RW}} = -\alpha_2/2 \pm \sqrt{\alpha_2(\alpha_2/4 - K_e)}. \tag{25}$$

It is important to remark that the value of $\mathbf{u}_{\overline{RW}}$ does not interfere the invariance condition (14) of the SM PCA in Section 4 since $\mathbf{u}_2 = \mathbf{u}_{\overline{RW}}$ only if no constraint is active ($\max_i \phi_i(\mathbf{p}^*_{\mathrm{ref}}) \leq -\varepsilon_3$), i.e., only if the PCA is not active.

The vector $\mathbf{u}_{\mathrm{RW}}$ is computed as a random non-zero vector orthogonal to the gradients $\nabla\sigma_i$ of all active constraints for independence from the SM PCA. That is, the first-order time derivative of the sliding manifold ($\dot{\phi}_i$) does *not* explicitly depend on $\mathbf{u}_{\mathrm{RW}}$:

$$\mathbf{L_g}\boldsymbol{\phi}_{2n}\,\mathbf{u}_2 = \mathbf{L_g}\boldsymbol{\phi}_{2n}\,\mathbf{u}_{\mathrm{RW}} = \alpha_2 K\,\nabla\boldsymbol{\sigma}^{\mathrm{T}}\mathbf{u}_{\mathrm{RW}} = \mathbf{0}_h, \tag{26}$$

since vector $\mathbf{u}_{\mathrm{RW}}$ is orthogonal to each column vector of matrix $\nabla\boldsymbol{\sigma}$.

It is worth mentioning that, the value assigned to $\mathbf{u}_2$ (either $\mathbf{u}_{\mathrm{RW}}$ or $\mathbf{u}_{\overline{RW}}$) *indirectly* affects the SM PCA since this value is filtered and integrated to obtain $\dot{\mathbf{f}}_{\mathrm{RW}}$ and $\mathbf{f}_{\mathrm{RW}}$. Therefore, signal $\mathbf{u}_2$ indirectly modifies the minimum value of u^+ (21) (i.e., $L_f\phi_i$ (10)) required to establish the SM (and to fulfill the constraints) when the PCA is active.

It will be assumed that the number of active constraints is always less than n. Otherwise, such a non-zero vector $\mathbf{u}_{\mathrm{RW}}$ orthogonal to the gradients of all active constraints does not exist and, in general, some planning is required to escape from the trap situation.

The computation of vector $\mathbf{u}_{\mathrm{RW}}$ is made every sampling period T_{PC} as follows:

1. The gradient vectors $\nabla\sigma_i$ of all active constraints (i.e., those constraints that fulfill $\phi_i(\mathbf{p}^*_{\mathrm{ref}}) > -\varepsilon_3$) are made orthogonal to each other, e.g. using the Gram-Schmidt process, to obtain vectors $\nabla\sigma_i^o$.

2. A new n-dimensional random vector (namely $\mathbf{F}$) is generated every T_F seconds (i.e., every round(T_F/T_{PC}) samples) with all its components ranging from $-B$ to B, where constant B can be any real number.
3. Vector $\mathbf{F}$ is orthogonalized against all vectors $\nabla\sigma_i^o$ to obtain $\bar{\mathbf{F}}$.
4. Vector $\mathbf{u}_{RW}$ is computed as follows:

$$\mathbf{u}_{RW} = (K_c + K_v\, t_{trap})\, \bar{\mathbf{F}}/\|\bar{\mathbf{F}}\|_2, \tag{27}$$

where K_c and K_v are user-defined positive constants and t_{trap} is the time elapsed since the TAA became active, i.e., since u_{SP} was switched off for the last time.

Note that the Euclidean-norm of $\mathbf{u}_{RW}$ is a linear function of t_{trap} and, thus, the amplitude of the RW is gradually increased in order to overcome highly "attractive" trap situations, see Fig. 2.

The generation of a new random vector $\mathbf{F}$ every T_F seconds (instead of every sampling period T_{PC}) allows to avoid permanent changes in the RW speed $\dot{\mathbf{f}}_{RW}$ and, thus, the RW position $\mathbf{f}_{RW}$ is smoother. Note that, in contrast to the SM PCA, the RW speed does not need to change at a fast rate.

7. Guidelines for designing the algorithm parameters

This section will give some guidelines for the conceptual design of both PCA and TAA.

7.1. Parameters of the PCA

7.1.1. Constraint approaching parameter

The value of K can be interpreted as the *time constant* of the "braking" process when approaching the boundary of the original constraints σ_i, i.e., when approaching a constraint at high speed, the constraint will be reached in approximately $3K$ seconds and transversal speed will be also lowered to zero after that time has elapsed.

7.1.2. Cutoff frequency

The value of α_1 must be higher than the frequency content of signals $\mathbf{p}_{ref}$ and $\mathbf{f}_{RW}$ (the latter is given by α_2 and T_F when $\mathbf{u}_2 = \mathbf{u}_{RW}$ and by (25) when $\mathbf{u}_2 = \mathbf{u}_{\overline{RW}}$) in order to obtain a good approximation of the theoretical SM behavior, but not too high to avoid significant chattering (22).

7.1.3. Amplitude of the control action

The value of $\|\mathbf{u}_{SM}\|_2$ (which is directly related to u^+) has to be as close as possible to its lower bound given by (21) (with, perhaps, some safety margin) in order to have reduced chattering amplitude and high chattering frequency, see Section 4.4.

7.1.4. Sampling period

The sampling period T_{PC} of the PCA has to be small enough in order for the discrete implementation of the filter to work properly, i.e. $T_{PC} \ll \pi/\alpha_1$, and have small chattering amplitude (22). The minimum value for the sampling period T_{PC} is determined by the computation times of one iteration of the PCA, which is below one microsecond (see the Appendix). For instance, in Section 9 the PCA will operate at 1 or 0.2 ms to reduce the chattering phenomenon introduced by the SM based algorithm.

7.2. Parameters of the TAA

7.2.1. Thresholds for the conditions

The value of threshold ε_1 should be chosen small enough depending on the desired tracking error when restarting the motion of the original reference $\mathbf{p}_{ref}$ once the trap situation has been overcome. The value of threshold ε_2 should be chosen: large enough to guarantee that the original reference $\mathbf{p}_{ref}$ is clearly within the allowed region in order to activate the TAA; and small enough to avoid tracking errors once the trap situation has been overcome. The value of threshold ε_3 should be chosen: larger than the chattering band $\overline{\sigma}_{max}$ (23) to avoid disturbing the SM of the PCA; and small enough to avoid degrading the performance of the TAA, i.e., to fully exploit the allowed workspace.

7.2.2. Gain for returning to zero

In this work it will be used $K_e = \alpha_2/4$ in order to obtain a critically damped response, i.e., the RW returns to zero as quickly as possible without oscillating. Therefore, from (25) it is obtained $\text{poles}_{\overline{RW}} = -\alpha_2/2$.

7.2.3. Amplitude of the speed vector

The value of K_c should be chosen: large enough to escape from trap situations in a reasonable amount of time; and small enough to avoid fast RW speeds that cannot be attained by the robot. The value of K_v should be chosen: large enough to properly increase the amplitude of the RW speed with time in order to overcome highly attractive trap situations; and small enough to guarantee RW numerical stability.

7.2.4. Period of the random vector

The period T_F for regenerating the random vector $\mathbf{F}$ should be chosen: large enough to obtain a smooth RW position $\mathbf{f}_{RW}$; and small enough to obtain a significant variability in the RW speed $\dot{\mathbf{f}}_{RW}$.

7.2.5. Cutoff frequency

The value of α_2 should be chosen: large enough to properly react to the random vector $\mathbf{F}$ generated every T_F seconds; and small enough to avoid abrupt changes in the RW speed $\dot{\mathbf{f}}_{RW}$ when control signal $\mathbf{u}_2$ switches to another value. For example, typical values could be selected between $0.5/T_F$ and $2/T_F$.

8. Discussion

The main advantages of the proposed path conditioning are summarized below:

- It can be interpreted as a *limit case* of the conventional potential field-based approach for collision avoidance.
- The robot workspace is *fully exploited* in order to maintain the faithfulness to the original path.
- The limit surface of the constraints is reached *smoothly*, depending on a free design parameter.
- *Reduced computation time*: only linear algebra is used, no Hessian matrices are required (only gradients are used), etc.
- The SM PCA is *robust* against the environment model and against the reference trajectory generated by the planner.
- *No a priori knowledge* of the desired path is required.
- *Moving constraints* in a non-static environment are also valid as long as they are known.
- The proposed approach also includes a *trap avoidance* algorithm to escape from trap situations.

Let us now comment on the possible limitations.

Although the framework of this research is a robot operating in a *structured environment*, unexpected or a priori unknown obstacles could also be considered if they are detected on-line by the robot's proximity sensors [14]. However, this issue is out of the scope of this research and incorporating it into the here proposed control structure remains as further work.

Differing from conventional SM control where the switching is in the main control action of the controlled process, the current application presents a reference conditioning setup where SM is confined to the low-power side of the system and, hence, the so called *chattering* issues are greatly alleviated.

As stated in Assumptions 1 and 2, the original reference $\mathbf{p}_{\text{ref}}$ and the constraint functions σ_i must be twice *differentiable* and their second-order derivatives must be reasonably bounded in order to fulfill (21). If these assumptions are not satisfied at a certain time, the SM behavior of the path conditioning is temporarily lost and the constraints may be unfulfilled.

Note that, the proposed TAA is based on a RW and, therefore, it can take a considerable amount of time to escape from complex trap situations where the way out is given by an intricate maze. In some cases, then, combining the proposed TAA with a higher-level planner might be advisable in order to escape from *intricate trap situations* in a reasonable amount of time.

9. Simulation

In this section the main features of the proposed PCA are illustrated through simulation results. The first example shows the behavior of the PCA in Fig. 1 for different rates of approach to the constraints and its comparison to the conventional potential field-based approach. The second and third examples show the behavior of the PCA in combination with the TAA in Fig. 1 for escaping from a trap situation with one and multiple active constraints, respectively. Finally, a fourth example is developed to show the effectiveness of the proposed TAA to escape from a highly attractive trap situation.

9.1. First example: comparison with the conventional method

Consider a reference path given by the following helicoidal expression:

$$\begin{aligned}\mathbf{p}_{\text{ref}}(\lambda) &= [x_{\text{ref}}(\lambda)\ y_{\text{ref}}(\lambda)\ z_{\text{ref}}(\lambda)]^{\text{T}} \\ &= 0.1\begin{bmatrix}\sin(\lambda)\\ -0.75-\cos(\lambda)\\ 3.44-\lambda\end{bmatrix},\end{aligned} \tag{28}$$

with $\lambda = 0\ldots 2\pi$ and $\dot{\lambda} = 2\pi/5$, where the units for linear and angular dimensions are meters and radians, respectively.

Two workspace constraints are considered, the first one is characterized by a plane and the second one by a sphere. These constraints are given by the following expressions:

$$\sigma_1(\mathbf{p}^*_{\text{ref}}) = [0\ 1\ 0]\cdot\mathbf{p}^*_{\text{ref}} = y^*_{\text{ref}} \le 0, \tag{29}$$

$$\sigma_2(\mathbf{p}^*_{\text{ref}}) = 0.05 - \left\|\mathbf{p}^*_{\text{ref}}\right\|_2 \le 0. \tag{30}$$

The PCA shown in Fig. 1 has been implemented with a constraint approaching parameter K of 0.1 s and 0.2 s (i.e., the simulation is run twice, once for each value of K), a cutoff frequency α of 20 rad/s, a control action amplitude $\|\mathbf{u}_{\text{SM}}\|_2$ of 0.1 m and a sampling period T_{PC} of one millisecond.

In the same spirit of Eqs, (4) and (5), the conventional potential field-based[5] path conditioning is given by [1]:

$$\mathbf{p}^*_{\text{ref}} = \mathbf{p}_{\text{ref}} + \mathbf{f}_{\text{PF}}, \tag{31}$$

$$\dot{\mathbf{f}}_{\text{PF}} = F_{\text{att}} + \sum_i \left(F_{\text{rep}\,i}\right), \tag{32}$$

[5] In the conventional potential field-based method, it is usually assumed that the derivative of the position reference $\mathbf{p}_{\text{ref}}$ is zero, which is not the case of this work [15].

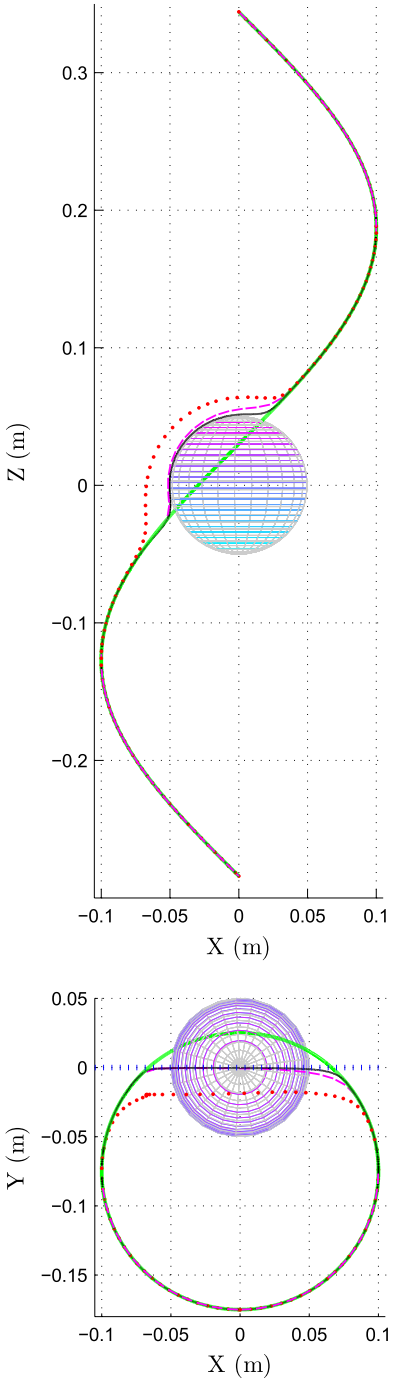

Fig. 3. First-angle projections (front view and top view with respect to the base coordinate frame) of the reference path (thick solid line) and the modified paths for the proposed PCA with $K = 0.1$ (solid line) and $K = 0.2$ (dashed line) and the conventional potential field-based approach (points). The limit plane of the first constraint is shown as a thick dotted line in the top view.

where $\mathbf{f}_{\text{PF}}$ is the correcting action, F_{att} is the *attractive force* to the current position reference $\mathbf{p}_{\text{ref}}$ and $F_{\text{rep}\,i}$ is the *repulsive force* from the ith-constraint. Thus, the sum of all "forces" determines the magnitude and direction of the derivative of the correcting action. The commonly used attractive and repulsive forces have the following form [16]:

$$F_{\text{att}} = \xi_1(\mathbf{p}_{\text{ref}} - \mathbf{p}^*_{\text{ref}}), \tag{33}$$

$$F_{\text{rep}\,i} = \begin{cases}\xi_2(\rho_i^{-1} - \rho_0^{-1})\rho_i^{-2}\nabla\rho_i & \text{if } \rho_i < \rho_0\\ 0 & \text{otherwise,}\end{cases} \tag{34}$$

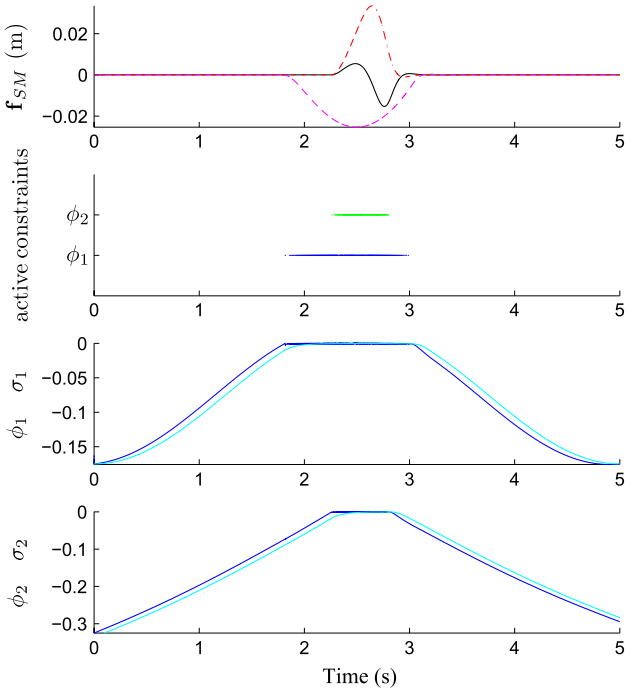

Fig. 4. Correction values $\mathbf{f}_{SM}$ of the path conditioning for $K = 0.1$, horizontal lines indicating when each constraint is active and constraint functions $\{\phi_1, \phi_2\}$ (dark line) and $\{\sigma_1, \sigma_2\}$ (light line).

where ξ_i is a positive constant, $\rho_i(\mathbf{p}^*_{\text{ref}})$ is the minimal distance from $\mathbf{p}^*_{\text{ref}}$ to the boundary of the ith-constraint and ρ_0 is a positive constant denoting the distance of influence of the constraints. The potential field-based path conditioning just described has been implemented with the following parameter values: $\xi_1 = 20\ \text{s}^{-1}$, $\xi_2 = 5 \cdot 10^{-6}\ \text{m}^4/\text{s}$ and $\rho_0 = 0.1$ m.

Fig. 3 shows the simulation results of the proposed PCA and the conventional potential field-based method. For the proposed PCA, the approach to the limit surface of the constraints is smoother for the higher value of K, which agrees with Eq. (3). For the conventional potential field-based method, the constraints are fulfilled but the workspace is not fully exploited to maintain the faithfulness to the original path. In the limit case of the potential field-based method, i.e. $\xi_2 \to 0$, the workspace is fully exploited, but at the expense of discontinuous repulsive forces and abrupt changes in the derivative of the modified reference, which requires using variable structure control concepts like those used in this work. Fig. 4 shows the correction values of the proposed PCA for $K = 0.1$ and the active constraints. Note that both constraints are simultaneously active for a period of about half second, during which the performance of the SM guarantees workspace constraint fulfillment.

9.2. Second example: trap avoidance with one active constraint

For this case, the reference path is given by the following helicoidal expression:

$$\mathbf{p}_{\text{ref}}(\lambda) = 0.1 \begin{bmatrix} \cos(\lambda) \\ \sin(\lambda) \\ \pi - \lambda \end{bmatrix}, \tag{35}$$

with $\lambda = 0 \ldots 2\pi$ and the value of the commanded motion rate $\dot{\lambda}_c$ is $2\pi/5$, where the units for linear and angular dimensions are meters and radians, respectively.

One constraint, characterized by an ellipsoid, is considered:

$$\sigma_3(\mathbf{p}^*_{\text{ref}}) = 0.1\left(1 - \left\| \begin{bmatrix} \frac{x^*_{\text{ref}}}{0.8} & \frac{y^*_{\text{ref}}}{0.8} & \frac{z^*_{\text{ref}}}{0.1} \end{bmatrix}^{\text{T}} \right\|_2\right) \leq 0. \tag{36}$$

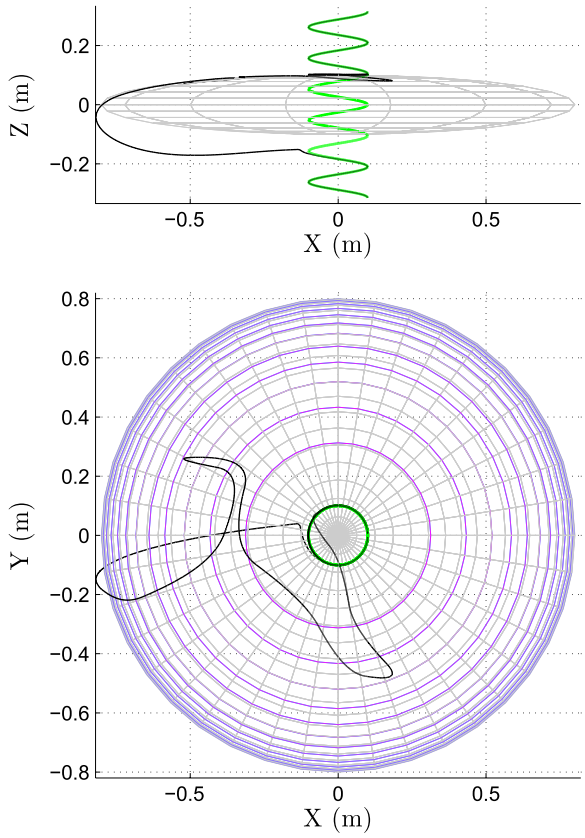

Fig. 5. First-angle projections (front view and top view) of the reference path (thick solid line) and the modified path (thin solid line) for the second example. The limit surface of the constraint is also shown.

For the simulations of this case, the PCA was implemented with a constraint approaching parameter K of 0.05 s, a cutoff frequency α of 20 rad/s and a control action amplitude $\|\mathbf{u}_{\text{SM}}\|_2$ of 1.6 m. The TAA shown in Fig. 1 was implemented under the following conditions: a cutoff frequency of 20 rad/s was used for both first-order low-pass filters that smooth out the control signals u_{SP} and $\mathbf{u}_2$, respectively; the switching law $\mathbf{u}_2$ was computed with $K_c = 2$ m/s, $K_v = 2\ \text{m/s}^2$ and $K_e = 5\ \text{s}^{-1}$; the random vector F is regenerated every $T_F = 0.1$ s; and the values of the thresholds ε_1, ε_2 and ε_3 were set to 0.05 m, 0.05 m and 0.01 m, respectively. Moreover, a sampling period of 0.2 milliseconds was used for both PCA and TAA.

Figs. 5 and 6 show the simulation results of the proposed PCA and TAA for this case. The behavior of the RW to avoid the trap situation, which is due to the small curvature of the upper pole of the ellipsoid, is shown in Fig. 5. The correction values $\mathbf{f}_{\text{SM}}$ and $\mathbf{f}_{\text{RW}}$ generated by the PCA and TAA, respectively, are shown in Fig. 6. Note that the scale factor f_{SP} for the commanded motion rate $\dot{\lambda}_c$ stops the original reference $\mathbf{p}_{\text{ref}}$ within the allowed region while the RW is active in order for the modified reference $\mathbf{p}^*_{\text{ref}}$ to converge to $\mathbf{p}_{\text{ref}}$.

9.3. Third example: trap avoidance with multiple active constraints

For this case, the reference path is given by the following sinusoidal expression:

$$\mathbf{p}_{\text{ref}}(\lambda) = 0.1 \begin{bmatrix} 0 \\ \sin(\lambda) \\ \pi - \lambda \end{bmatrix}, \tag{37}$$

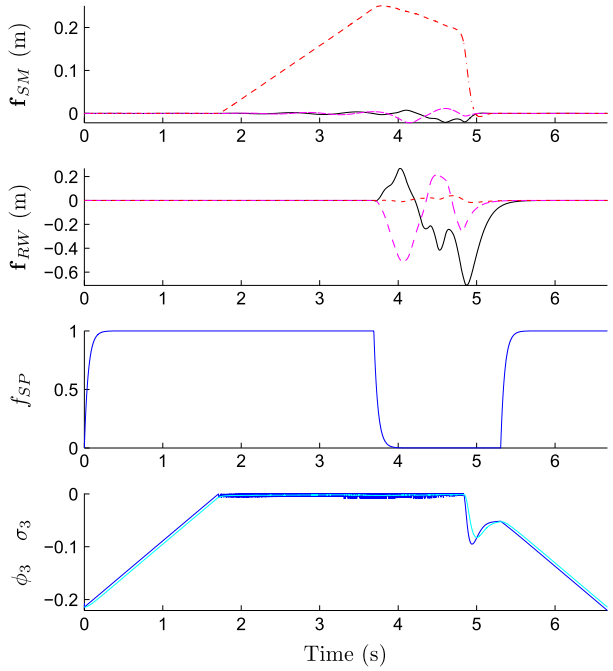

Fig. 6. Correction values $\mathbf{f}_{SM}$ and $\mathbf{f}_{RW}$ of the path conditioning and random walk, respectively, scale factor f_{SP} for the commanded motion rate $\dot{\lambda}_c$ and constraint functions ϕ_3 (dark line) and σ_3 (light line).

with $\lambda = 0 \ldots 2\pi$ and the value of the commanded motion rate $\dot{\lambda}_c$ is $2\pi/5$, where the units for linear and angular dimensions are meters and radians, respectively.

Two symmetric constraints, both of them characterized by an ellipsoid, are considered:

$$\sigma_4(\mathbf{p}^*_{\text{ref}}) = 0.1\left(1 - \left\| \begin{bmatrix} \frac{x^*_{\text{ref}} + 0.55}{0.8} & \frac{y^*_{\text{ref}}}{0.8} & \frac{z^*_{\text{ref}}}{0.1} \end{bmatrix}^{\text{T}} \right\|_2\right) \leq 0, \tag{38}$$

$$\sigma_5(\mathbf{p}^*_{\text{ref}}) = 0.1\left(1 - \left\| \begin{bmatrix} \frac{x^*_{\text{ref}} - 0.55}{0.8} & \frac{y^*_{\text{ref}}}{0.8} & \frac{z^*_{\text{ref}}}{0.1} \end{bmatrix}^{\text{T}} \right\|_2\right) \leq 0. \tag{39}$$

The PCA and the TAA were implemented under the same conditions of previous section. Figs. 7 and 8 show the simulation results for this case: the behavior of the RW to avoid the trap situation, which is due to the small curvature of the upper pole of both ellipsoids, is shown Fig. 7; whereas the correction values $\mathbf{f}_{\text{SM}}$ and $\mathbf{f}_{\text{RW}}$ generated by the PCA and TAA, respectively, are shown in Fig. 8. As before, the scale factor f_{SP} for the commanded motion rate $\dot{\lambda}_c$ stops the original reference $\mathbf{p}_{\text{ref}}$ within the allowed region while the RW is active in order for the modified reference $\mathbf{p}^*_{\text{ref}}$ to converge to $\mathbf{p}_{\text{ref}}$. Note that while the TAA is looking for a way out of the trap (i.e., $\mathbf{u}_2 = \mathbf{u}_{\text{RW}}$), the RW is performed across the intersection of both ellipsoids since both constraints are active, which is due to the fact that the mentioned intersection and the reference path lie in the same plane $x = 0$.

9.4. Fourth example: escape from a highly "attractive" trap situation

For this case, the reference path is given by the helicoidal expression (35) and the following constraint, which is characterized by a three-dimensional generalization of the oval of *Booth*, is considered:

$$\sigma_6(\mathbf{p}^*_{\text{ref}}) = 0.5 - \frac{\left\|\mathbf{p}^*_{\text{ref}}\right\|_2^2}{\left\|\begin{bmatrix} x^*_{\text{ref}} & y^*_{\text{ref}} & 0.3z^*_{\text{ref}} \end{bmatrix}^{\text{T}}\right\|_2} \leq 0. \tag{40}$$

The PCA and the TAA were implemented under the same conditions of previous section except for the parameters Kc and Kv of the TAA which were increased to 5 m/s and 5 m/s^2, respectively, in order to overcome the highly attractive trap situation. Figs. 9 and 10 show the simulation results for this case: the trap situation given by the constraint (40) is highly attractive because the modified reference generated by the PCA tends to go deeper into the upper "hole" of the oval (Fig. 9) and, therefore, the RW (Fig. 10) has to redirect this natural behavior. As before, signal f_{SP} stops the original reference $\mathbf{p}_{\text{ref}}$ within the allowed region while the RW is active in order for the modified reference $\mathbf{p}^*_{\text{ref}}$ to converge to $\mathbf{p}_{\text{ref}}$.

10. Conclusions

A sliding-mode algorithm for robotic path conditioning was proposed in this work. The algorithm modifies the desired

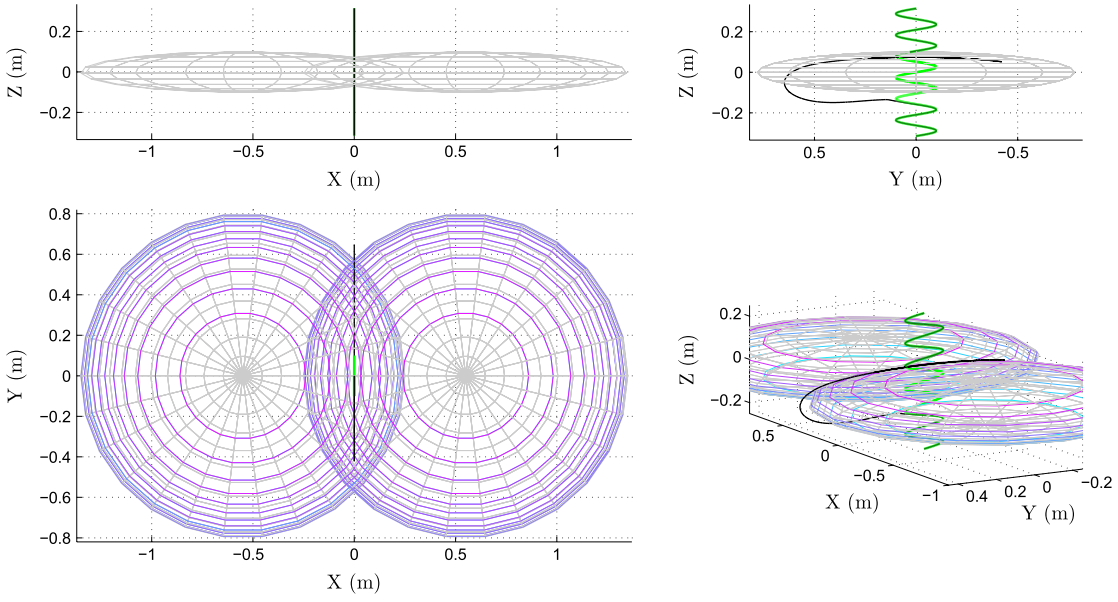

Fig. 7. First-angle projections (front view, side view and top view) and 3D view of the reference path (thick solid line) and the modified path (thin solid line) for the third example.

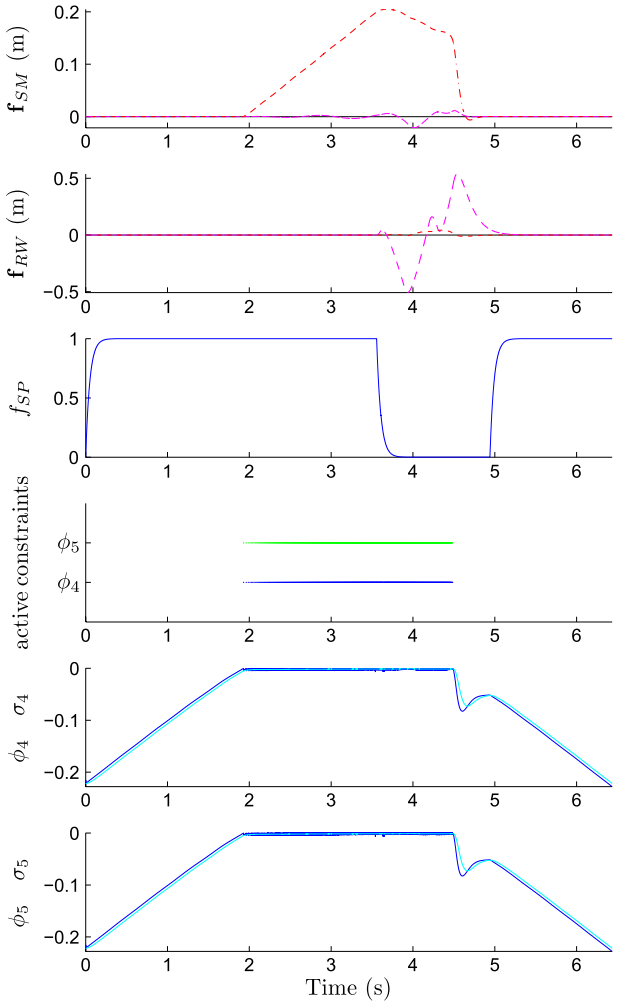

Fig. 8. Correction values $\mathbf{f}_{SM}$ and $\mathbf{f}_{RW}$ of the path conditioning and random walk, respectively, scale factor f_{SP} for the commanded motion rate $\dot{\lambda}_c$, horizontal lines indicating when each constraint is active and constraint functions $\{\phi_4, \phi_5\}$ (dark line) and $\{\sigma_4, \sigma_5\}$ (light line).

reference path in order to fulfill the robot workspace constraints. The main advantage of the proposed method over the conventional potential field-based approach for collision avoidance is that the robot workspace is fully exploited to maintain the faithfulness to the original path. The main limitation of the method is that trap situations, which are analogous to local minima in potential field-based approaches, can arise during the path conditioning. To overcome this drawback, a trap avoidance algorithm has also been developed in order to escape from trap situations once they are detected.

The proposed path conditioning algorithm can be used on-line, since it does not require a priori knowledge of the desired path, its implementation is very easy in a few program lines of a microprocessor and its computation time is short (see the Appendix). The performance and effectiveness of the proposed algorithms has been demonstrated by simulation results.

Acknowledgments

This research is partially supported by DISICOM project PROMETEO 2008/088 of Generalitat Valenciana (Spain), research project DPI2011-27845-C02-01 of the Spanish Government (Spain), research project PAID-05-11-2640 of the Technical University of Valencia (Spain), and the Argentinian Government (UNLP 11I127, CONICET PIP 112-200801-0, ANPCyT PICT 2007 00535).

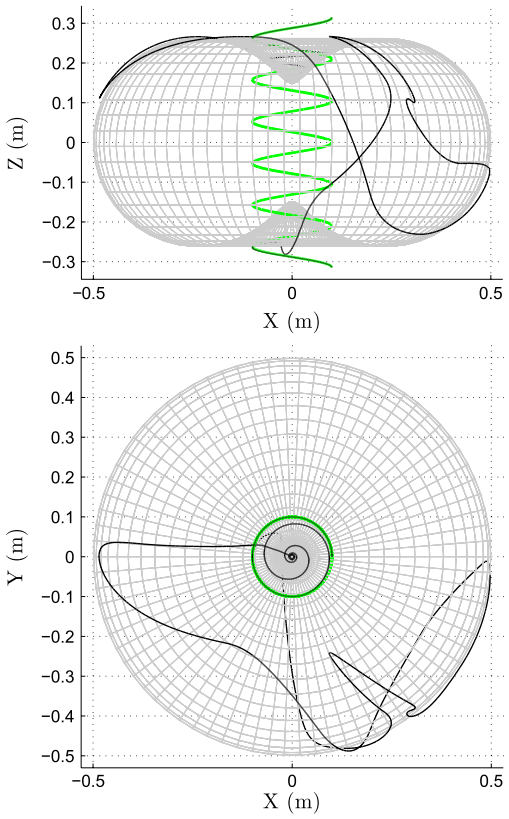

Fig. 9. First-angle projections (front view and top view) of the reference path (thick solid line) and the modified path (thin solid line) for the fourth example.

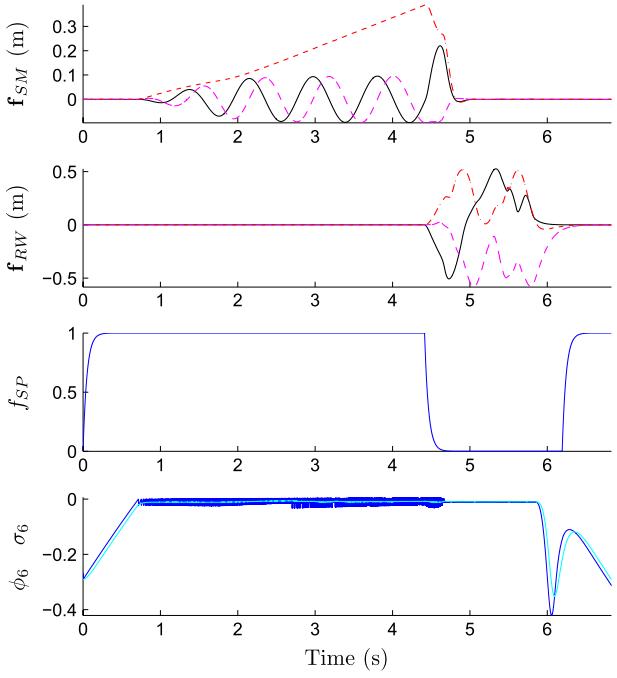

Fig. 10. Correction values $\mathbf{f}_{SM}$ and $\mathbf{f}_{RW}$ of the path conditioning and random walk, respectively, scale factor f_{SP} for the commanded motion rate $\dot{\lambda}_c$ and constraint functions ϕ_6 (dark line) and σ_6 (light line).

Appendix A. Computer Implementation

The MATLAB® code of the proposed PCA and TAA for the examples presented in Section 9 is shown below. The PCA and TAA are implemented in the *loop_pathcond* and *loop_trapavoid*

functions, respectively, which call five auxiliary functions to compute $\{\nabla\sigma_3, \nabla\sigma_4, \nabla\sigma_5, \nabla\sigma_6, \phi_3, \phi_4, \phi_5, \phi_6, \bar{\mathbf{F}}\}$. Note that this implementation supports the claim made in the paper that the proposed approach only requires a few program lines. The main script file shown below uses this implementation to obtain the modified reference generated by the proposed method for the second (Case = 1), third (Case = 2) and fourth (Case = 3) examples. The computation time per iteration in a modern computer using MATLAB® R2009a (compiled C-MEX-file) was around 0.6 μs (microseconds) for the PCA and 0.9 μs for the TAA.

```
function grad=grad3(p)
grad=-(p./[64;64;1])/max(norm((p)./[8;8;1]),1e-9);
```

```
function grad=grad4(p)
grad=-((p-[-0.55;0;0])./[64;64;1])/...
      max(norm((p-[-0.55;0;0])./[8;8;1]),1e-9);
```

```
function grad=grad5(p)
grad=-((p-[0.55;0;0])./[64;64;1])/...
      max(norm((p-[0.55;0;0])./[8;8;1]),1e-9);
```

```
function grad=grad6(p)
global Nov
grad=p.*(Nov.^2*(p'*p/max(norm(p.*Nov),1e-9)^3)-...
    2/max(norm(p.*Nov),1e-9));
```

```
function phi=phi3(p,pd)
global K;
phi=(0.1-norm(p./[8;8;1]))+K*grad3(p)'*pd;
```

```
function phi=phi4(p,pd)
global K;
phi=(0.1-norm((p-[-0.55;0;0])./[8;8;1]))+K*grad4(p)'*pd;
```

```
function phi=phi5(p,pd)
global K;
phi=(0.1-norm((p-[0.55;0;0])./[8;8;1]))+K*grad5(p)'*pd;
```

```
function phi=phi6(p,pd)
global K Nov Rov;
phi=(Rov-p'*p/max(norm(p.*Nov),1e-9))+K*grad6(p)'*pd;
```

```
function Fbar=ComputeFbar(gradsF)
for i=2:size(gradsF,2),
 gradsF(:,i-1)=gradsF(:,i-1)/max(norm(gradsF(:,i-1)),1e-9);
 for j=1:i-1,
   gradsF(:,i)=gradsF(:,i)-gradsF(:,i)'*...
               gradsF(:,i-j)*gradsF(:,i-j);
 end;
end;
Fbar=gradsF(:,end);
```

```
function [p_next,pd_next]=loop_pathcond(pref,p,pd,Case)
global Tpc Max_usm NFilt1 DFilt1 usm1 usm2 fsm1 fsm2;
if Case==1, usm=-(phi3(p,pd)>=0)*grad3(p);
elseif Case==2, usm=-(phi4(p,pd)>=0)*grad4(p)-...
              (phi5(p,pd)>=0)*grad5(p);
else usm=-(phi6(p,pd)>=0)*grad6(p); end;
usm=Max_usm*usm/max(norm(usm),1e-9);
fsm=usm*NFilt1(1)+usm1*NFilt1(2)+usm2*NFilt1(3)-...
     fsm1*DFilt1(2)-fsm2*DFilt1(3);
p_next=pref+fsm;  pd_next=(p_next-p)/Tpc;
usm2=usm1; usm1=usm; fsm2=fsm1; fsm1=fsm;
```

```
function [fsp,frw]=loop_trapavoid(pref,pdref,p,pd,Case)
global Tpc F Kc Kv Ke NFilt2 DFilt2 NFilt3 DFilt3;
global tTrap E1 E2 E3 fsp1 usp1 frw1 frw2 u21;
if Case==1, Cond1=phi3(p,pd)>-E3;
elseif Case==2, Cond1=max(phi4(p,pd),phi5(p,pd))>-E3;
else Cond1=phi6(p,pd)>-E3; end;
if Case==1, Cond2=(phi3(pref,pdref)<-E2);
elseif Case==2,
   Cond2=(max(phi4(pref,pdref),phi5(pref,pdref))<-E2);
else Cond2=(phi6(pref,pdref)<-E2); end;
Cond2=Cond2&&(norm(pref-p)>E1);
if Cond2, usp=0; tTrap=tTrap+Tpc; else usp=1; tTrap=0; end;
if Cond1&&Cond2,
  if Case==1, gradsF=grad3(p)*(phi3(p,pd)>-E3);
  elseif Case==2, gradsF=[grad4(p)*(phi4(p,pd)>-E3),...
                  grad5(p)*(phi5(p,pd)>-E3)];
  else gradsF=grad6(p)*(phi6(p,pd)>-E3); end;
  gradsF=[gradsF,F];
  Fbar=ComputeFbar(gradsF);
  u2=(Kc+Kv*tTrap)*Fbar/max(norm(Fbar),1e-9);
else u2=-Ke*frw1; end;
fsp=usp*NFilt3(1)+usp1*NFilt3(2)-fsp1*DFilt3(2);
frw=Tpc*(u2*NFilt2(1)+u21*NFilt2(2))+...
     frw1*(1-DFilt2(2))+frw2*DFilt2(2);
% for the next iteration
fsp1=fsp; usp1=usp; frw2=frw1; frw1=frw; u21=u2;
```

```
clear all;
global Tpc Max_usm K Kc Kv Ke usm1 usm2 fsm1 fsm2;
global tTrap NFilt1 DFilt1 NFilt2 DFilt2 NFilt3 DFilt3;
global E1 E2 E3 fsp1 usp1 frw1 frw2 u21 F Nov Rov;

Case=3; %{1,2,3}->{1_Ellipsoid,2_Ellipsoids,Oval}

% Initialization
Tpc=2e-4; K=0.05; wSM=20; wRW=20; wSP=20; Max_usm=1.6;
E1=0.05; E2=0.05; E3=0.01; Ke=wRW/4; tTrap=0;
if Case==3, Kc=5; Kv=5; else Kc=2; Kv=2; end;
frw=zeros(3,1); frw1=frw; frw2=frw; u21=frw;
usm1=frw; usm2=frw; fsm1=frw; fsm2=frw;
fsp1=0; usp1=0; fsp=0; lambda=0; derlambda=2*pi/5;
pref=0.1*[not(Case==2);0;pi]; p=pref;
pdref=[0;0.1;-0.1]*derlambda; pd=pdref;
Nov=[1;1;.3]; Rov=.5;
TimeF=0.1; SamplesF=round(TimeF/Tpc); F=rand(3,1)-0.5;

% Filters
[NFilt1,DFilt1]=butter(2,wSM*Tpc/pi,'low');
[NFilt2,DFilt2]=butter(1,wRW*Tpc/pi,'low');
[NFilt3,DFilt3]=butter(1,wSP*Tpc/pi,'low');

tic; % Path conditioning loop
for i=1:1e8,
  [p_next,pd_next]=loop_pathcond(pref(:,i)+...
                   frw(:,i),p(:,i),pd(:,i),Case);
  [fsp_next,frw_next]=loop_trapavoid(pref(:,i),...
                   pdref(:,i),p(:,i),pd(:,i),Case);

  p=[p,p_next]; pd=[pd,pd_next];
  frw=[frw,frw_next]; fsp=[fsp,fsp_next];
  lambda=[lambda,lambda(end)+derlambda(end)*Tpc];
  derlambda=[derlambda,(2*pi/5)*fsp_next];
  if rem(i,SamplesF)==0, F=rand(3,1)-0.5; end;

  pref=[pref,.1*[cos(6*lambda(end))*not(Case==2);
             sin(6*lambda(end));(pi-lambda(end))]];
  pdref=[pdref,(pref(:,end)-pref(:,end-1))/Tpc];

  if rem(i,.1/Tpc)==0, fprintf('Time=%.2f\n',i*Tpc); end;
  if lambda(end)>=2*pi, break; end;
end;
toc/size(pref,2) % Computation time per iteration

%***************** PLOTS *********************
% Paths followed by the original and modified references
figure; hold on; grid on; axis equal;
col=[.8,.8,.8]; colormap cool;
if Case==1, %%% 1 Ellipsoid
```

```
  [X,Y,Z]=ellipsoid(0,0,0,.8,.8,.1,13);
  contour3(X,Y,Z,24);
  surface(X,Y,Z,'EdgeColor',col,'FaceColor','none');
  axis([-.85,.85,-.85,.85,-.35,.35]); view(0,0);
elseif Case==2, %%% 2 Ellipsoids
  [X,Y,Z]=ellipsoid(-.55,0,0,.8,.8,.1,13);
  contour3(X,Y,Z,24);
  surface(X,Y,Z,'EdgeColor',col,'FaceColor','none');
  [X,Y,Z]=ellipsoid(.55,0,0,.8,.8,.1,13);
  contour3(X,Y,Z,24);
  surface(X,Y,Z,'EdgeColor',col,'FaceColor','none');
  axis([-1.35,1.35,-.8,.8,-.34,.34]); view(-90,0);
else %%% Oval
  th=linspace(0,2*pi,40); al=linspace(0,2*pi,80);
  [th,al]=meshgrid(th,al);
  aux1=Rov*sqrt(Nov(1)^2*cos(th).^2.*sin(al).^2+...
   Nov(2)^2*sin(th).^2.*sin(al).^2+Nov(3)^2*cos(al).^2);
  X=aux1.*cos(th).*sin(al); Y=aux1.*sin(th).*sin(al);
  Z=aux1.*cos(al);
  surface(X,Y,Z,'EdgeColor',col,'FaceColor','none');
  axis([-.53,.53,-.53,.53,-.34,.34]); view(0,0);
end;
plot3(pref(1,:),pref(2,:),pref(3,:),'g','LineWidth',3);
plot3(p(1,:),p(2,:),p(3,:),'r','LineWidth',2);
xlabel('X (m)'); ylabel('Y (m)'); zlabel('Z (m)');
title('{\bfp}_{ref} (g)   {\bfp}*_{ref} (r)');

% Correction values: fSM, fRW, fSP, phi_i, sigma_i
Ns=size(pref,2); time=0:Tpc:(Ns-1)*Tpc; fsm=p-pref-frw;
phi_i=zeros(1,Ns); sigma_i=zeros(1,Ns);
for i=1:Ns,
  if Case==1, phi_i(i)=phi3(p(:,i),pd(:,i));
      sigma_i(i)=phi3(p(:,i),zeros(3,1)); end;
  if Case==2, phi_i(i)=phi4(p(:,i),pd(:,i));
      sigma_i(i)=phi4(p(:,i),zeros(3,1)); end;
  if Case==3, phi_i(i)=phi6(p(:,i),pd(:,i));
      sigma_i(i)=phi6(p(:,i),zeros(3,1)); end;
end;
figure; subplot(4,1,1); hold on;
ylabel('{\bf{f}}_{SM} (m)');
plot(time,fsm(1,:),'k-',time,fsm(2,:),'m--',...
                        time,fsm(3,:),'r-.');
axis([0,time(end),min(min(fsm)),max(max(fsm))]);
subplot(4,1,2); hold on; ylabel('{\bf{f}}_{RW} (m)');
plot(time,frw(1,:),'k-',time,frw(2,:),'m--',...
                        time,frw(3,:),'r-.');
axis([0,time(end),min(min(frw)),max(max(frw))]);
subplot(4,1,3); hold on; plot(time,fsp,'b');
axis([0,time(end),0,1.1]); ylabel('f_{SP}');
subplot(4,1,4); hold on; plot(time,phi_i,'b');
plot(time,sigma_i,'c'); pos0=get(gcf,'Position');
axis([0,time(end),min(phi_i),max([0.05,phi_i])]);
ylabel('\phi_i     \sigma_i'); xlabel('Time (s)');
set(gcf,'Position',[pos0(1:3)-[0,200,0],pos0(4)*4/3]);
```

References

[1] J.-C. Latombe, Robot Motion Planning, Kluwer, Boston, 1991.
[2] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots, IEEE Transactions on Systems, Man, and Cybernetics 19 (1989) 1179–1187.
[3] E. Rimon, D. Koditschek, Exact robot navigation using artificial potential functions, IEEE Transactions on Robotics and Automation 8 (1992) 501–518.
[4] C. Edwards, S. Spurgeon, Sliding Mode Control: Theory and Applications, first ed., Taylor & Francis, UK, 1998.
[5] W. Perruquetti, J. Barbot (Eds.), Sliding Mode Control in Engineering, in: Control Engineering Series, Marcel Dekker, 2002.
[6] L.-G. Garcia-Valdovinos, V. Parra-Vega, M. Arteaga, Observer-based sliding mode impedance control of bilateral teleoperation under constant unknown time delay, Robotics and Autonomous Systems 55 (2007) 609–617.
[7] W. Bessa, M. Dutra, E. Kreuzer, An adaptive fuzzy sliding mode controller for remotely operated underwater vehicles, Robotics and Autonomous Systems 58 (2010) 16–26.
[8] F. Garelli, L. Gracia, A. Sala, P. Albertos, Sliding mode speed auto-regulation technique for robotic tracking, Robotics and Autonomous Systems 59 (2011) 519–529.
[9] L. Gracia, A. Sala, F. Garelli, A supervisory loop approach to fulfill workspace constraints in redundant robots, Robotics and Autonomous Systems 60 (2012) 1–15.
[10] E. Bernabeu, J. Tornero, Optimal geometric modeler for robot motion planning, Journal of Robotic Systems 17 (2000) 593–608.
[11] V. Utkin, J. Guldner, J. Shi, Sliding Mode Control in Electro-Mechanical Systems, second ed., Taylor & Francis, London, 2009.
[12] P. Queiros, R. Cortesao, C. Sousa, Haptic tele-manipulation for robotic-assisted minimally invasive surgery with explicit posture control, in: Proceedings of 18th Mediterranean Conference on Control and Automation, Marrakech, Morocco, 2010.
[13] L. Gracia, J. Andres, J. Tornero, Trajectory tracking with a 6R serial industrial robot with ordinary and non-ordinary singularities, International Journal of Control, Automation, and Systems 7 (2009) 85–96.
[14] A. Nuchter, 3D Robotic Mapping: The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom, Springer-Verlag, Berlin, 2009.
[15] L. Huang, Velocity planning for a mobile robot to track a moving target—a potential field approach, Robotics and Autonomous Systems 57 (2009) 55–63.
[16] S. Ge, Y. Cui, New potential functions for mobile robot path planning, IEEE Transactions on Robotics and Automation 16 (2000) 615–620.

Luis Gracia received the B.Sc. degree in electronic engineering, the M.Sc. degree in control systems engineering, and the Ph.D. in automation and industrial computer science from the Technical University of Valencia (UPV), Spain, in 1998, 2000, and 2006, respectively. He is currently an Associate Professor at the Department of Systems Engineering and Control (DISA) of the UPV. His research interests include mobile robots, robotic manipulators, and system modeling and control.

Antonio Sala received his B.Eng. degree (Hon. Deg.) from Coventry University, Coventry, UK, in 1990, the 2nd Spanish National Graduation prize (M.Sc. degree in electrical engineering) in 1993, and the Ph.D. degree in Control Engineering from the UPV 1998. He has been teaching in the UPV since 1993 at the DISA, where he is currently a full Professor, and has supervised four Ph.D. theses. He has coauthored 24 papers in middle or top impact journals, and the book "Multivariable Control Systems" (Springer, 2004), and he is the co-editor of "Iterative Identification and Control" (Springer, 2002). He is a member of IEEE, IFAC (spanish section), has served 3 years on IFAC publications executive committee and is an associate Editor of IEEE Trans. On Fuzzy Systems. His current research interests include fuzzy control, system identification, networked control systems and process control applications.

Fabricio Garelli received the B.S. and Dr. Eng. degrees in electronic engineering from National University of La Plata (UNLP), Argentina. He is currently a full Professor and a research member of the National Research Council of Argentina (CONICET) at LEICI, EE Dept., Faculty of Engineering, UNLP. He is the author of an awarded Ph.D. Thesis, more than 30 journal or conference papers and the book "Advanced Control for Constrained Processes and Systems" (IET, 2011). His research interests include multivariable nonlinear control, variable structure systems and constrained control applications.