Integrated Sliding-Mode Algorithms in Robot Tracking Applications

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

[a]*Dept. of Systems Engineering and Control, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain (e-mails: luigraca@isa.upv.es, asala@isa.upv.es).* **Corresponding author, Tel.: +34-963879770, Fax: +34-963879579*
[b]*CONICET and Universidad Nacional de La Plata, C.C.91 (1900), La Plata, Argentina (e-mail: fabricio@ing.unlp.edu.ar).*

Abstract

An integrated solution based on sliding mode ideas is proposed for robotic trajectory tracking. The proposal includes three sliding-mode algorithms for speed auto-regulation, path conditioning and redundancy resolution in order to fulfill velocity, workspace and C-space constraints, respectively. The proposed method only requires a few program lines and simplifies the robot user interface since it directly deals with the fulfillment of the constraints to find a feasible solution for the robot trajectory tracking in a short computation time. The proposed approach is evaluated in simulation on the freely accessible 6R robot model PUMA-560, for which the main features of the method are illustrated.

Keywords: Sliding mode, robot control, collision avoidance

1. Introduction

The main objective of robot control systems is the tracking of a reference trajectory, which involves the generation of a control signal to make the error between the robot position and the reference zero [1]. In this sense, this work presents an integrated solution for robotic trajectory tracking based on three sliding-mode algorithms recently proposed by the authors[1] for speed auto-regulation [2], path conditioning [3] and redundancy resolution [4] in order to fulfill velocity, workspace and C-space constraints, respectively. These constraints may be due to different reasons such as joint speed limits [5], joint angle limits [6], obstacle collision avoidance [7], etc.

The proposed approach, which only requires a few program lines, simplifies the user interface since the method directly deals with the fulfillment of the constraints specified by the robot end-user. Therefore, in case of relatively simple tasks the proposed method finds a feasible solution for the robot trajectory tracking in a short computation time.

The proposed approach can be useful for many type of robots and industrial applications, such as spray painting [8], arc welding [9], assembly [10], polishing [11], etc. For instance, in this work it is used a well-known and free-access six-revolute (6R) robot, the PUMA 560, for which the main distinctive features of the method are illustrated in a spray painting application.

The outline of the paper is as follows. Next section introduces some preliminaries, while Section 3, Section 4 and Section 5 present the three sliding-mode algorithms proposed for speed auto-regulation, path conditioning and redundancy resolution, respectively. A discussion about the method is given in Section 6. The proposed approach is applied in Section 7 to the PUMA-560 robot model in order to show the feasibility and effectiveness of the method. Finally, some concluding remarks are given.

2. Preliminaries and control scheme

2.1. Notation

Following the standard notation [12], consider a robot system with $\mathbf{q} = [q_1 \dots q_n]^{\mathrm{T}}$ being the robot *configuration* or n-dimensional joint position vector and $\mathbf{p} = [p_1 \dots p_m]^{\mathrm{T}}$ being the robot *pose* or m-dimensional workspace position vector. A robot is said to be *redundant* when the dimension m of the

[1]The three algorithms were individually developed and tested by the authors in previous works [2, 3, 4]. However, all three are adapted in this work to be used in a more general framework than that of previous works. In this sense, this research effectively integrates the three algorithms in the same robot control scheme and shows their complementarity for robot trajectory tracking. Another important contribution of this work is the given pseudo-code for the proposed approach (including the three sliding-mode algorithms) so that it can be easily implemented on many actual robot platforms.

workspace is less than the dimension n of the configuration space (hereafter, C-space), i.e., $m < n$. The degree of kinematic redundancy is computed as $n - m$. For the rest of the paper it is assumed that the robot at hand is redundant.

The relationship between the robot configuration and the robot pose is highly nonlinear, generically expressed as:

$$\mathbf{p} = \mathbf{l}(\mathbf{q}), \tag{1}$$

where the function $\mathbf{l}$ is called the kinematic function of the robot model.

The first order kinematics results in:

$$\dot{\mathbf{p}} = \frac{\partial \mathbf{l}(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}, \tag{2}$$

where $\mathbf{J}(\mathbf{q})$ is denoted as the $m \times n$ Jacobian matrix or simply *Jacobian* of the kinematic function.

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

$$\mathbf{p}_{ref} = \mathbf{v}(\lambda). \tag{3}$$

Finally, the gradient of a scalar function $f(x_1, \ldots, x_n)$ will be denoted $\nabla f = [\frac{\partial f}{\partial x_1} \cdots \frac{\partial f}{\partial x_n}]^{\mathrm{T}}$.

2.2. Control scheme

Fig. 1 shows the control scheme proposed in this work for robotic trajectory tracking, which contains three sliding-mode (SM) algorithms for speed auto-regulation, path conditioning and redundancy resolution. The SM speed auto-regulation block generates the motion rate parameter $\dot{\lambda}$ so that it is as close as possible to the desired value $\dot{\lambda}_d$ and that satisfies velocity constraints on the desired workspace velocity vector $\dot{\mathbf{p}}_d$, desired joint velocity vector $\dot{\mathbf{q}}_d$ and robot state $(\mathbf{q}, \dot{\mathbf{q}})$, see Section 3. The SM path conditioning block generates a modified workspace reference $\mathbf{p}^*_{ref}$ to be sent to the robot kinematic controller so that it is as close as possible to the original value $\mathbf{p}_{ref}$ and that belongs to the allowed workspace, see Section 4. The redundancy resolution block computes the desired joint velocity vector $\dot{\mathbf{q}}_d$ in order to track the desired workspace velocity vector $\dot{\mathbf{p}}_d$ as primary task, while a secondary

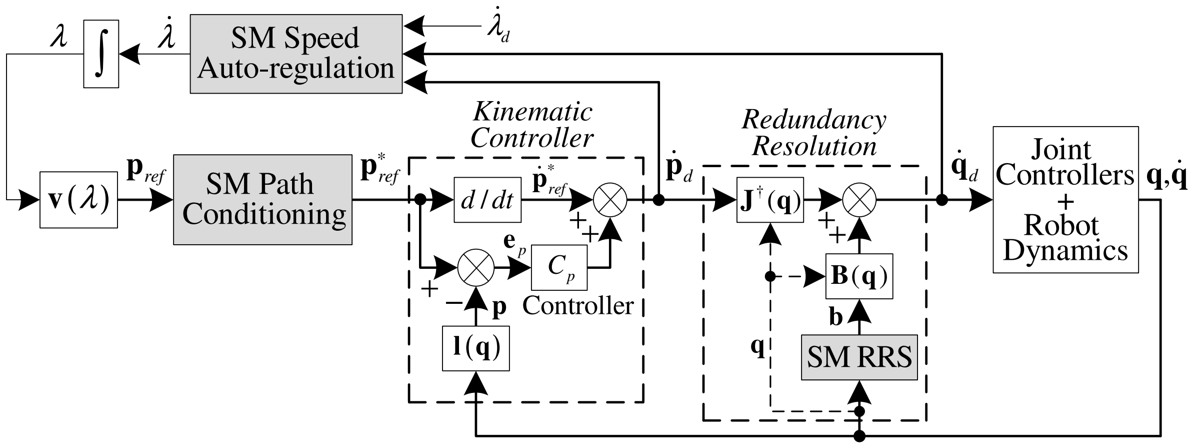


Fig. 1. Robotic trajectory tracking control scheme with SM Algorithms (shaded blocks).

goal is achieved using redundancy in order to satisfy C-space constraints on the robot state $(\mathbf{q}, \dot{\mathbf{q}})$, see Section 5. The kinematic controller generates the workspace velocity vector $\dot{\mathbf{p}}_d$ closing a loop using the robot state and the modified workspace reference $\mathbf{p}^*_{ref}$ in order to make the tracking error zero.

Kinematic controller. For this work, it is considered a classical kinematic controller utilized for robotic trajectory tracking [13], see Fig. 1, which consists of a two-degree of freedom (2-DOF) control that incorporates a correction based on the position error $\mathbf{e}_p = \mathbf{p}^*_{ref} - \mathbf{p}$ by means of the position loop controller C_p plus a feedforward term depending on the first-order time derivative of the modified workspace reference, i.e. $\dot{\mathbf{p}}^*_{ref}$. Note that, the path function $\mathbf{v}(\lambda)$ needs to be differenciable due to the feedforward term. For instance, if the reference path is given by a set of tracking points generated by the robot operator, it can be made smooth and continuous by using spline or Bézier interpolation.

Classical redundancy resolution. The desired joint velocity vector $\dot{\mathbf{q}}_d$ is computed by the redundancy block in Fig. 1 in order to satisfy the first order kinematic relation:

$$\dot{\mathbf{p}}_d = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}_d. \tag{4}$$

In general, in the case of redundant robots an infinite number of solutions for $\dot{\mathbf{q}}_d$ satisfying (4) exist[2], which are given by:

$$\dot{\mathbf{q}}_d = \mathbf{J}^{\dagger}(\mathbf{q})\, \dot{\mathbf{p}}_d + \mathbf{B}(\mathbf{q})\, \mathbf{b}, \tag{5}$$

where $\mathbf{J}^{\dagger}(\mathbf{q})$ is the so-called *right pseudo-inverse* of $\mathbf{J}(\mathbf{q})$ (i.e., $\mathbf{J}^{\dagger} \equiv \mathbf{J}^{\mathrm{T}}(\mathbf{J}\,\mathbf{J}^{\mathrm{T}})^{-1}$); $\mathbf{B}(\mathbf{q})$ is an $n \times n$ matrix whose last m column vectors are the n-dimensional null vector and whose first $n-m$ column vectors form an orthonormal basis for the null space of $\mathbf{J}(\mathbf{q})$ (e.g., this basis can be easily obtained from the singular value decomposition [15] of $\mathbf{J}(\mathbf{q})$); and $\mathbf{b}$ is the so-called *performance vector* which is an arbitrary n-dimensional column vector. The first term in (5) represents the minimum-norm solution or *base solution*, while the second term is the *homogeneous solution* that gives rise to infinite possible solutions for $\dot{\mathbf{q}}_d$ depending on the value of performance vector $\mathbf{b}$. In general, this vector can be expressed as a function of the robot state, i.e. $\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}})$. The reader is referred to literature for choices of performance vector [16, 6, 4].

2.3. Constrained control via sliding modes

Consider the following dynamical system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u + \boldsymbol{\nu} \tag{6}$$

where $\mathbf{x}$ is the state vector, u is the control input (which has been assumed scalar for simplicity), $\mathbf{f}$ and $\mathbf{g}$ are two vector fields of $\mathbf{x}$ and vector $\boldsymbol{\nu}$ accounts for the system uncertainty.

Consider also the constraint:

$$\sigma(\mathbf{x}) \leq 0, \tag{7}$$

where σ is a function of the state vector whose first-order time-derivative is obtained as:

$$\dot{\sigma} = \frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}^{\mathrm{T}} (\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})u + \boldsymbol{\nu}). \tag{8}$$

Provided $\frac{\partial \sigma(\mathbf{x})}{\partial \mathbf{x}}^{\mathrm{T}} \mathbf{g}(\mathbf{x})\|_{\sigma(\mathbf{x})=0} \neq 0$, condition $\dot{\sigma} < 0$ can be ensured on the

[2]It is implicitly assumed that $\mathbf{J}(\mathbf{q})$ is full row rank, since otherwise the robot configuration $\mathbf{q}$ is said to be *singular* [14] and the desired workspace velocity vector $\dot{\mathbf{p}}_d$ in general cannot be achieved.

border $\sigma(\mathbf{x}) = 0$ by means of a high enough input u, so as to avoid violating constraint (7). In this sense, the following switching law

$$u = \begin{cases} u_{SM} & \text{if} \quad \sigma(\mathbf{x}) \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

can be employed to enforce the system (6) to robustly fulfill a given constraint, i.e., a high enough u_{SM} will yield a switching law robust against unknown *matched* $\boldsymbol{\nu}$ [17]. Observe that as long as the system tries by itself to leave the allowed region, the above control law will give rise to a theoretical infinite switching frequency, which can be seen as an ideal SM operation with absence of open-loop phase (reaching mode). Although infinite switching frequency cannot be achieved in practice, which leads to an oscillation within a "band" around $\sigma = 0$ known as *chattering* [18], in software-based implementations this drawback becomes negligible for reasonable fast sampling rates. This is the case of the three algorithms described in the subsequent sections. Interested readers are referred to [19, 20] for further details on conventional SM control theory and to [17] for constrained control applications.

3. Sliding-mode speed auto-regulation

3.1. Problem statement

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

$$\Phi_{SA}(\dot{\mathbf{p}}_d, \dot{\mathbf{q}}_d) = \left\{ \begin{bmatrix} \dot{\mathbf{p}}_d^{\mathrm{T}} & \dot{\mathbf{q}}_d^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \mid \sigma_{SA,i}(\dot{\mathbf{p}}_d, \dot{\mathbf{q}}_d) \leq 0 \right\}, \; i = 1, ..., N_{SA}, \tag{10}$$

where $\sigma_{SA,i}$ is a function of velocity vectors $\dot{\mathbf{p}}_d$ and $\dot{\mathbf{q}}_d$ that is positive if and only if the ith-constraint is not fulfilled. For the solution later proposed in Section 3.2, it will be assumed that function $\sigma_{SA,i}$ is differentiable around the boundary given by $\sigma_{SA,i}(\mathbf{c}) = 0$

The main control goal of the speed auto-regulation algorithm (SAA) can therefore be stated as to generate a motion rate parameter $\dot{\lambda}$ so that it is as close as possible to the desired value $\dot{\lambda}_d$ and that belongs to the allowed workspace Φ_{SA} given by (10).

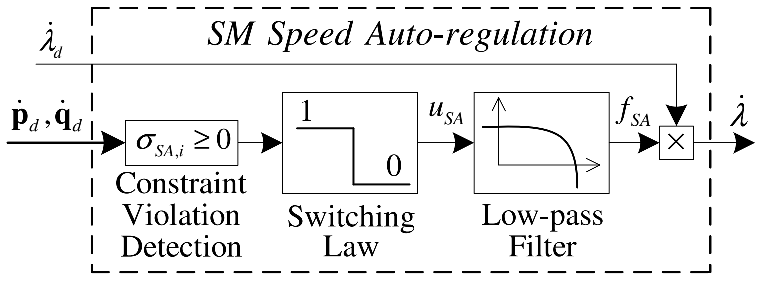


Fig. 2. Speed auto-regulation algorithm.

Constraint functions. In this work, two types of velocity constraint functions are considered for the SAA:

$$\sigma_{SA,W}(\dot{\mathbf{p}}_d) = \|\dot{\mathbf{p}}_d\|_2 - \dot{p}_{\max} \leq 0 \tag{11}$$

$$\sigma_{SA,Ji}(\dot{q}_{d,i}) = |\dot{q}_{d,i}| - \dot{q}_{\max,i} \leq 0, \tag{12}$$

where $\dot{p}_{\max}$ is the maximum speed allowed for the Euclidean-norm of the workspace velocity vector $\dot{\mathbf{p}}$ and $\dot{q}_{\max,i}$ is the maximum speed allowed for the ith-joint[3]. The first constraint is useful, for example, in spray painting applications in order to guarantee a minimum paint deposition at any point on the workspace reference path. The second type of constraint is useful to not exceed the speed limits of the joint actuators in order to avoid tracking errors, since in general they arise when the desired joint velocities are saturated. Note that, the maximum allowable speeds $\dot{p}_{\max}$ and $\dot{q}_{\max,i}$ could be computed as a function of the robot position in order to obtain lower values when the robot is working close to the operator area, or as a function of the output of proximity sensors in order to obtain lower values when a presence is detected within the robotic workcell.

3.2. Algorithm

Fig. 2 shows the diagram of the SM algorithm proposed in [2] to solve[4] the speed auto-regulation problem stated in Section 3.1. In particular, the

[3]It has been assumed for the sake of simplicity that joint velocity limits are symmetric, i.e. $\dot{q}_{\min,i} = -\dot{q}_{\max,i}$, although expression (12) can be trivially modified if that were not the case

[4]The framework of the SAA proposed in this work is more general than that presented in [2], where only joint speed limit constraints (12) were considered.

following variable structure control law is considered:

$$u_{SA} = \begin{cases} 0 & \text{if} \quad \max_i \sigma_{SA,i}(\mathbf{c}) \geq 0 \\ 1 & \text{otherwise.} \end{cases} \tag{13}$$

As shown in Fig. 2, the control signal f_{SA} is generated by passing the discontinuous signal u_{SA} through a low-pass filter. This control signal acts as a scale factor for the desired motion rate parameter $\dot{\lambda}_d$, so that $\dot{\lambda}$ is obtained as:

$$\dot{\lambda} = f_{SA}\dot{\lambda}_d. \tag{14}$$

The filter in Fig. 2 has unit gain at low frequencies and its *bandwidth* needs to be chosen sufficiently fast for quick stops to be allowed, but slow enough in order to smooth out $\dot{\lambda}$. Naturally, the best choice for the filter bandwidth is strongly related with the workspace reference trajectory to be followed.

The *order* of the filter is selected to satisfy the so-called *transversality condition* [20] for SM, which imposes that the sliding manifold must have *unitary relative degree* with respect to the discontinuous action, i.e., its first-order time derivative ($\dot{\sigma}_{SA,i}$) must explicitly depend on u_{SA}. Note that, the kinematic controller in Fig. 1 includes a first-order time derivative term, whereas the relative degree between signal $\mathbf{p}^*_{ref}$ and signal $\mathbf{p}_{ref}$ is zero, see the path conditioning algorithm proposed in Section 4.2. Hence, $\dot{\mathbf{p}}_d$ and $\dot{\mathbf{q}}_d$ (i.e., $\sigma_{SA,W}$ and $\sigma_{SA,Ji}$) explicitly depend on $\dot{\lambda}$, which in turns depends on f_{SA}, see Fig. 1. Therefore, the filter must be of first-order for $\dot{\sigma}_{SA,W}$ and $\dot{\sigma}_{SA,Ji}$ to explicitly depend onf u_{SA}. Thus, the control signal f_{SA} is generated from the discontinuous signal u_{SA} by means of a first-order low-pass filter:

$$\dot{f}_{SA} = -\alpha_{SA}\, f_{SA} + \alpha_{SA}\, u_{SA}, \tag{15}$$

where the scalar α_{SA} is the filter cutoff frequency representing the filter bandwidth. For further details see [2].

4. Sliding-mode path conditioning

4.1. Problem statement

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

$$\Phi_{PC}(\mathbf{p}) = \{\mathbf{p} \mid \sigma_{PC,i}(\mathbf{p}) \leq 0\}\,,\ i = 1, ..., N_{PC}, \tag{16}$$

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

In order to satisfy some requirements for the solution later proposed in Section 4.2, the following *assumptions* are considered to hold: the workspace reference $\mathbf{p}_{ref}$ is twice differentiable; the constraint functions $\sigma_{PC,i}$ are twice differentiable around the boundary given by $\sigma_{PC,i}(\mathbf{p}) = 0$ and their gradients $\nabla\sigma_{PC,i}$ around this boundary do not vanish. For non-differentiable constraints, there are techniques in literature [21] that may be used to enclose such non-smooth regions by smooth mathematical objects with an arbitrary degree of precision.

The main control goal of the path conditioning algorithm (PCA) can therefore be stated as to generate a modified workspace reference $\mathbf{p}^*_{ref}$ to be sent to the robot kinematic controller so that it is as close as possible to the original value $\mathbf{p}_{ref}$ and that belongs to the allowed workspace Φ_{PC} given by (16).

Improvement of the constraint space. The actual constraint space (16) will be modified to also include the speed of movement in the following way:

$$\begin{aligned}\Phi^*_{PC}(\mathbf{p},\dot{\mathbf{p}}) = \Bigg\{ [\mathbf{p}^{\mathrm{T}}\ \dot{\mathbf{p}}^{\mathrm{T}}]^{\mathrm{T}} \mid \phi_{PC,i}(\mathbf{p},\dot{\mathbf{p}}) &= \sigma_{PC,i}(\mathbf{p}) + K_{PC,i}\frac{d\sigma_{PC,i}(\mathbf{p})}{dt} \\ &= \sigma_{PC,i} + K_{PC,i}\,\nabla\sigma^{\mathrm{T}}_{PC,i}\,\dot{\mathbf{p}} \leq 0 \Bigg\},\ i = 1, ..., N_{PC},\end{aligned} \tag{17}$$

where $\phi_{PC,i}(\mathbf{p},\dot{\mathbf{p}})$ is the modified ith workspace constraint and $K_{PC,i}$ is the *constraint approaching parameter* of the PCA ith-constraint, which is a free design parameter that determines the rate of approach to the boundary of the ith-constraint. Thus, expression (17) introduces an additional degree

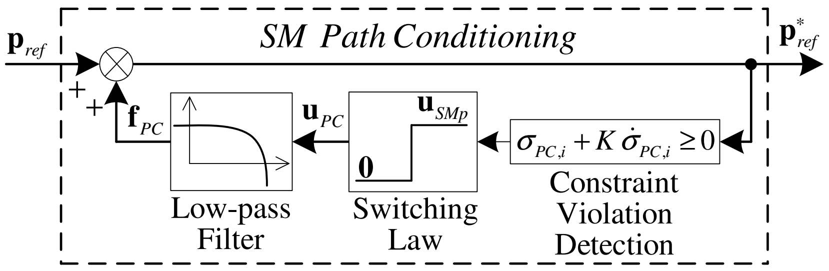


Fig. 3. Path conditioning algorithm.

of freedom necessary to reach the limit in a controlled fashion in a similar way to the classical proportional-derivative (PD) controller: the closer to the boundary of the original constraint, the lower the maximum allowed approaching speed to this boundary.

4.2. Algorithm

Fig. 3 shows the diagram of the SM algorithm proposed in [3] to solve the path conditioning problem stated in Section 4.1. The commanded workspace path is shaped by modifying the workspace reference $\mathbf{p}_{ref}$ as follows:

$$\mathbf{p}^*_{ref} = \mathbf{p}_{ref} + \mathbf{f}_{PC}, \tag{18}$$

where $\mathbf{f}_{PC}$ is the *correcting action* to the original workspace reference.

Signal $\mathbf{f}_{PC}$ is generated by passing the discontinuous signal $\mathbf{u}_{PC}$ through a low-pass filter, as shown in Fig. 3. This filter smooths out the signal added to the main control loop and it must be of second-order for $\ddot{\mathbf{p}}^*_{ref}$ to explicitly depend on $\mathbf{u}_{PC}$ (i.e., for $\dot{\phi}_{PC,i}$ to explicitly depend on $\mathbf{u}_{PC}$) in order to have unitary relative degree between the sliding manifold and the discontinuous action, as required by SM theory. Particularly, the following second-order butterworth low-pass filter could be used:

$$\ddot{\mathbf{f}}_{PC} = -\sqrt{2}\alpha_{PC}\dot{\mathbf{f}}_{PC} - \alpha_{PC}^2\mathbf{f}_{PC} + \alpha_{PC}^2\mathbf{u}_{PC}, \tag{19}$$

with the scalar α_{PC} being the filter cutoff frequency.

Moreover, the following variable structure control law is considered:

$$\mathbf{u}_{PC} = \begin{cases} \mathbf{u}_{SMp} & \text{if} \quad \max_i \phi_{PC,i}(\mathbf{p}^*_{ref}, \dot{\mathbf{p}}^*_{ref}) \geq 0 \\ \mathbf{0}_m & \text{otherwise,} \end{cases} \tag{20}$$

where $\mathbf{u}_{SMp}$ is computed as:

$$\mathbf{u}_{SMp} = -\nabla\boldsymbol{\sigma}_{PC}\mathbf{1}_h u^+_{PC}, \tag{21}$$

where $\mathbf{1}_h$ is the h-dimensional column vector with all its components equal to one, h is the number of active constraints, matrix $\nabla\boldsymbol{\sigma}_{PC}$ contains the gradient vectors $\nabla\sigma_{PC,i}$ of all active constraints and u^+_{PC} is a positive constant to be chosen high enough to establish a SM on the constraints boundary. To fulfill that, u^+_{PC} must be [3]:

$$u^+_{PC} > \sum_{i=1}^{h}(\max(D_i,0)) \Big/ \mathrm{eig}_{\min}(\nabla\boldsymbol{\sigma}^{\mathrm{T}}_{PC}\,\nabla\boldsymbol{\sigma}_{PC}), \tag{22}$$

$$\begin{aligned}\text{with}\quad D_i =& \nabla\sigma^{\mathrm{T}}_{PC,i}\left(\mathbf{p}_{ref} + \sqrt{2}\alpha^{-1}_{PC}\,\dot{\mathbf{p}}_{ref} + \alpha^{-2}_{PC}\,\ddot{\mathbf{p}}_{ref}\right)\\ &+ \alpha^{-2}_{PC}\,\dot{\mathbf{p}}^{*\,\mathrm{T}}_{ref}\,\mathbf{H}_{PC,i}\,\dot{\mathbf{p}}^{*}_{ref}\\ &- \nabla\sigma^{\mathrm{T}}_{PC,i}\left(\mathbf{p}^*_{ref} + (\sqrt{2}\alpha^{-1}_{PC} - \alpha^{-2}_{PC}\,K^{-1}_{PC,i})\dot{\mathbf{p}}^*_{ref}\right),\end{aligned} \tag{23}$$

where function $\mathrm{eig}_{\min}(\cdot)$ computes the minimum eigenvalue of a square matrix and $\mathbf{H}_{PC,i}$ denotes the Hessian matrix of second-order partial derivatives of $\sigma_{PC,i}$. In order to obtain a definite value for u^+_{PC} in (22), matrix $\nabla\boldsymbol{\sigma}^{\mathrm{T}}_{PC}$ has to be full row rank, which is the transversality condition [20] for the PCA and implies that the gradients of the active constraints must be linearly independent (obviously, $h \leq m$ must be fulfilled).

The above control law leads to a sliding regime [19] (i.e., control signal $\mathbf{u}_{PC}$ switches between $\mathbf{0}_m$ and $\mathbf{u}_{SMp}$ with a theoretically infinite frequency) on the boundary of the ith-constraint if around this boundary the system tries by itself to leave the allowed region. For further details see [3].

5. Sliding-mode redundancy resolution

5.1. Problem statement

We consider now that the robotic system to be controlled is subjected to C-space constraints given by:

$$\Phi_{RR}(\mathbf{q}) = \{\mathbf{q}\,|\,\sigma_{RR,i}(\mathbf{q}) \leq 0\}\,,\ i = 1,...,N_{RR}, \tag{24}$$

where $\sigma_{RR,i}$ is a function of the robot configuration $\mathbf{q}$ that is positive if and only if the ith-constraint is not fulfilled.

For the solution later proposed in Section 5.2, the functions $\sigma_{RR,i}$ need to be twice differentiable around the boundary given by $\sigma_{RR,i}(\mathbf{q}) = 0$ and their gradients $\nabla\sigma_{RR,i}$ around this boundary should not vanish. Moreover, it will also be assumed the *kinematic framework*, i.e. the dynamics given by the joint controllers is negligible compared to the dynamics of the workspace reference $\mathbf{p}_{ref}$, which implies that the actual joint velocity vector $\dot{\mathbf{q}}$ is approximately equal to the desired joint velocity vector $\dot{\mathbf{q}}_d$, see Fig. 1.

The main control goal can therefore be stated as to generate a joint velocity vector $\dot{\mathbf{q}}_d$ to be sent to the robot joint controllers so that the desired workspace velocity vector $\dot{\mathbf{p}}_d$ is tracked using the non-redundant degrees of freedom of the robot, while the remaining redundant degrees of freedom are used to implement a classical redundancy resolution scheme (RRS) together with a supervisory block to guarantee that $\mathbf{q}$ belongs to the allowed C-space Φ_{RR} given by (24).

Improvement of the constraint space. As before, the actual constraint space (24) will be modified in the following way:

$$\Phi^*_{RR}(\mathbf{q},\dot{\mathbf{q}}) = \Big\{ [\mathbf{q}^{\mathrm{T}}\ \dot{\mathbf{q}}^{\mathrm{T}}]^{\mathrm{T}} \mid \phi_{RR,i}(\mathbf{q},\dot{\mathbf{q}}) = \sigma_{RR,i}(\mathbf{q}) + K_{RR,i}\frac{d\sigma_{RR,i}(\mathbf{q})}{dt}$$
$$= \sigma_{RR,i} + K_{RR,i}\,\nabla\sigma^{\mathrm{T}}_{RR,i}\,\dot{\mathbf{q}} \le 0 \Big\},\ i = 1, ..., N_{RR}, \qquad (25)$$

where $\phi_{RR,i}(\mathbf{q},\dot{\mathbf{q}})$ is the modified ith C-space constraint and $K_{RR,i}$ is the constraint approaching parameter of the RRS ith-constraint, which is again a free design parameter that determines the rate of approach to the boundary of the ith-constraint.

Cartesian position constraints. In practical applications with redundant robots one common objective is that the Cartesian position $\bar{\mathbf{p}}_j = \begin{bmatrix} x_j & y_j & z_j \end{bmatrix}^{\mathrm{T}}$ of every point j of the robot belongs to the allowed Cartesian position space $\Phi_{RR,P}(\bar{\mathbf{p}}_j) = \{\bar{\mathbf{p}}_j \mid \sigma_{RR,i}(\bar{\mathbf{p}}_j) \le 0\ \forall\, i\}$. Thus, the allowed C-space results in $\Phi_{RR,C}(\mathbf{q}) = \big\{\mathbf{q} \mid \sigma_{RR,i}(\bar{\mathbf{l}}_j(\mathbf{q})) \le 0\ \forall\, i,j\big\}$, where $\bar{\mathbf{l}}_j$ is the kinematic function of the Cartesian position of point j. The infinite number of points of the robot to be considered in the above expression can reduced to a set of *characteristic points* such that the distance from every point on the boundary surface of the robot links to the closest characteristic point is less than a predetermined value which is used to enlarge the constrained

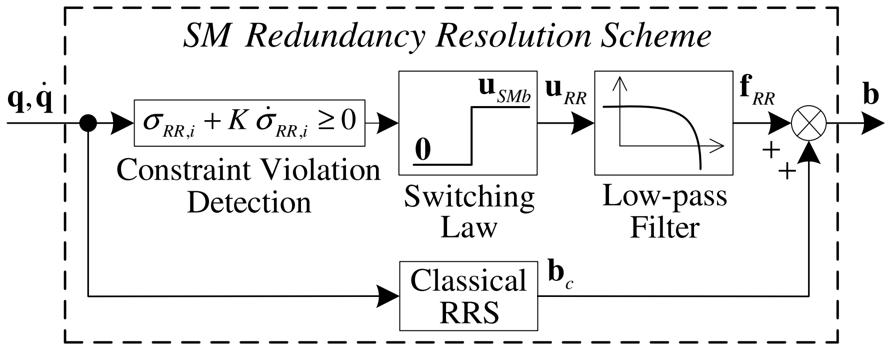

SM Redundancy Resolution Scheme
$\mathbf{q},\dot{\mathbf{q}}$
$\sigma_{RR,i} + K\,\dot{\sigma}_{RR,i} \geq 0$
Constraint Violation Detection
$\mathbf{u}_{SMb}$
$\mathbf{0}$
Switching Law
$\mathbf{u}_{RR}$
Low-pass Filter
$\mathbf{f}_{RR}$
+
+
$\mathbf{b}$
Classical RRS
$\mathbf{b}_c$

Fig. 4. Proposed redundancy resolution scheme.

region of the Cartesian position space. Some simplifications can be made in case the allowed Cartesian position space is *convex*. In such circumstances, the links could be enclosed with polyhedrons and the characteristic points to be considered are those on their vertices, whereas the original constrained region does not have to be enlarged. Moreover, if the width of the robot links is negligible, the characteristic points to be considered are reduced to the end-points of the links.

5.2. Algorithm

The RRS proposed in this research, see Fig. 4, consists of the combination of two signals:

$$\mathbf{b} = \mathbf{b}_c + \mathbf{f}_{RR}, \tag{26}$$

where $\mathbf{b}_c$ is the performance vector of a classical RRS and $\mathbf{f}_{RR}$ is a discontinuous signal generated by a supervisor block in order to fulfill C-space constraints. Thus, the supervisor block becomes active when there is a risk of violating a given constraint. Otherwise, a secondary goal given by a classical RRS is achieved using redundancy.

Signal $\mathbf{f}_{RR}$ is obtained by passing the discontinuous signal $\mathbf{u}_{RR}$ through a low-pass filter, as shown in Fig. 4. In order to have unitary relative degree between the sliding manifold and the discontinuous action, this filter must be of first-order for $\ddot{\mathbf{q}}_d$, see (5) and (26), to explicitly depend on $\mathbf{u}_{RR}$ (i.e., for $\dot{\phi}_{RR,i}$ to explicitly depend on $\mathbf{u}_{RR}$):

$$\dot{\mathbf{f}}_{RR} = -\alpha_{RR}\,\mathbf{f}_{RR} + \alpha_{RR}\,\mathbf{u}_{RR}, \tag{27}$$

with the scalar α_{RR} being the filter cutoff frequency.

Moreover, the following variable structure control law is considered:

$$\mathbf{u}_{RR} = \begin{cases} \mathbf{u}_{SMb} & \text{if} \quad \max\limits_{i} \phi_{RR,i}(\mathbf{q}, \dot{\mathbf{q}}) \geq 0 \\ \mathbf{0}_n & \text{otherwise,} \end{cases} \tag{28}$$

where $\mathbf{u}_{SMb}$ is computed as:

$$\mathbf{u}_{SMb} = -\mathbf{B}^{\mathrm{T}} \nabla \boldsymbol{\sigma}_{RR} \mathbf{1}_h u_{RR}^{+}, \tag{29}$$

where h is the number of active constraints, matrix $\nabla \boldsymbol{\sigma}_{RR}$ contains the gradient vectors $\nabla \sigma_{RR,i}$ of all active constraints and u_{RR}^{+} is a positive constant to be chosen high enough to establish a SM on the constraints boundary. To fulfill that, u_{RR}^{+} must be [4]:

$$u_{RR}^{+} > \sum_{i=1}^{h} (\max(F_i, 0)) \Big/ \operatorname{eig}_{\min}(\nabla \boldsymbol{\sigma}_{RR}^{\mathrm{T}} \, \mathbf{B} \, \mathbf{B}^{\mathrm{T}} \, \nabla \boldsymbol{\sigma}_{RR}), \tag{30}$$

$$\text{with} \quad F_i = (\alpha_{RR}^{-1} \, K_{RR,i}^{-1} - 1) \, \nabla \sigma_{RR,i}^{\mathrm{T}} \, \dot{\mathbf{q}} + \alpha_{RR}^{-1} \, \dot{\mathbf{q}}^{\mathrm{T}} \, \mathbf{H}_{RR,i} \, \dot{\mathbf{q}} + \nabla \sigma_{RR,i}^{\mathrm{T}} \, (\mathbf{J}^{\dagger} \, \dot{\mathbf{p}}_d + \mathbf{B} \, \mathbf{b}_c), \tag{31}$$

where $\mathbf{H}_{RR,i}$ denotes the Hessian matrix of second-order partial derivatives of $\sigma_{RR,i}$. In order to obtain a definite value for u_{RR}^{+} in (30), matrix $\nabla \boldsymbol{\sigma}_{RR}^{\mathrm{T}} \, \mathbf{B}$ has to be full row rank, which is the transversality condition [20] for the RRS and implies that the vectors obtained by projection of the gradients of the active constraints onto the null space of $\mathbf{J}$ must be linearly independent.

The above control law leads to a sliding regime [19] (i.e., control signal $\mathbf{u}_{RR}$ switches between $\mathbf{0}_n$ and $\mathbf{u}_{SMb}$ with a theoretically infinite frequency) on the boundary of the ith-constraint if around this boundary the system tries by itself to leave the allowed region. For further details see [4].

6. Discussion

Constraints definition. As we could use as constraint function, for example, either σ_i or $5\sigma_i$, all the PCA and RRS constraints should defined to be interpretable in a homogeneous way. Moreover, as mentioned in Section 2.3, all SM algorithms suffer from chattering and, hence, the constraints will be violated by a small amount. Therefore, the constraint functions should be defined adding a safety margin depending on the estimated chattering amplitudes, the environment modeling inaccuracies, the robot control inaccuracies, etc. The values of the chattering amplitudes for the SAA, PCA and RRS can be found in [2, 3, 4].

Main advantages of the algorithms.

- The three SM algorithms only require a few program lines and have *reduced computation time*, see the Appendix.
- The PCA and RRS nicely *complement* each other, since the former modifies the reference path to fulfill workspace constraints, while the latter uses redundancy to fulfill C-space constraints as secondary task (i.e., the RRS cannot modify the main task given by the workspace reference).
- The SAA generates, without explicit knowledge of it, the maximal tracking speed which is compatible with the velocity constraints, so that the *cycle time* of the robot task is minimized.
- The robot workspace is *fully exploited* by the PCA in order to maintain the faithfulness to the workspace reference path.
- The limit surface of the workspace and C-space constraints is reached *smoothly*, depending on a free design parameter.

Limitations of the algorithms. The SM algorithms used in this work use linear extrapolation to predict the value of the constraint functions at the next step, i.e., $\sigma(t+T) = \sigma(t) + T\dot{\sigma}(t)$. This implies that only local data of first-order derivatives are used. Therefore, since higher-order derivatives are ignored, the PCA and RRS algorithms may be blocked in trap situations. In some cases, these situations could be avoided using a planner with the complete geometric data of the problem in order to "simulate" for a large prediction horizon. However, the complexity of this planner and its computational cost are substantially greater than those of the algorithms proposed in this work, see the Appendix.

Use of the algorithms. The proposed method with three SM algorithms can be executed either *online* or *offline*. The latter requires the robot model (e.g., it is typically approximated to an integrator if the low-level control of the robot is fast enough) and, as part of the planning stage, allows to anticipate the trap situations mentioned above. The result is a sequence of joint velocities $\{\dot{\mathbf{q}}_{d0}, \dot{\mathbf{q}}_{d1}, \ldots\}$ to be sent to the robot joint controllers. If the proposed method runs online, there is a risk of having trap situations, but it has the advantage of correcting the position errors (e.g., the initial position error) and of being robust against robot modeling errors. The ideal situation is a *hybrid* execution, i.e., the method is firstly executed offline to anticipate and solve abnormal behaviors and, secondly, it is executed on the real robot system, where abnormal behaviors may arise only if the robot modeling error in the offline execution is large.

Guidelines for designing the algorithm parameters. Next, some guidelines for the conceptual design of the four groups of parameters of the SAA, PCA and RRS are given.

The value of constraint approaching parameters $K_{PC,i}$ and $K_{RR,i}$ can be interpreted as the *time constant* of the "braking" process when approaching the boundary of the original constraints $\sigma_{PC,i}$ and $\sigma_{RR,i}$. That is, when approaching a PCA or RRS constraint at high speed, it will be reached in approximately $3K_{PC,i}$ or $3K_{RR,i}$ seconds, respectively, and transversal speed will be also lowered to zero after that time has elapsed.

The value of the cutoff frequencies α_{SA}, α_{PC} and α_{RR} of the SAA, PCA and RRS filters, respectively, must be high enough to obtain a good approximation of the theoretical SM behavior, but not too high to avoid significant chattering amplitude.

The values $\|\mathbf{u}_{SMp}\|_2$ and $\|\mathbf{u}_{SMb}\|_2$ of the PCA and RRS control action amplitudes (which are directly related to u^+_{PC} and u^+_{RR}, respectively) have to be as close as possible to their lower bounds given by (22) and (30) (with, perhaps, some safety margin) in order to have reduced chattering amplitude and high chattering frequency.

The sampling periods T_{SA}, T_{PC} and T_{RR} of the SAA, PCA and RRS, respectively, have to be small enough in order for the discrete implementations of the SAA, PCA and RRS filters to work properly (i.e., $T_{SA} \ll \pi/\alpha_{SA}$, $T_{PC} \ll \pi/\alpha_{PC}$ and $T_{RR} \ll \pi/\alpha_{RR}$) and have small chattering amplitude.

Initial transient. If the tracking error is large (e.g., during the initial phase), $\dot{\mathbf{p}}_d$ and $\dot{\mathbf{q}}_d$ may *exceed* the speed limits even with $\dot{\lambda} = 0$ due to the error correction performed by the kinematic controller. This is shown intentionally in the simulation example of next section. However, this could be trivially overcome if $\dot{\mathbf{p}}_d$ and $\dot{\mathbf{q}}_d$ are saturated previously to the redundancy resolution and joint controller blocks in Fig. 1, respectively. Moreover, depending on the robot task, the tool could be switched off while the tracking error is above a prescribed threshold to avoid degrading the robot performance.

7. Simulation

In this section the main features of the proposed tracking control scheme (Fig. 1) are illustrated for the well-known 6DOF robotic arm PUMA-560 through simulation results obtained using the freely accessible *Robotics Toolbox* [22] (Release 7.1) for MATLAB®. This Toolbox includes the kinematic and dynamic model of the PUMA-560 robot, which have been used to generate the results (the reader is referred to the toolbox documentation for geometry, mass and inertia parameters). The PUMA-560 robot is a classical 6R serial manipulator with spherical wrist, which is widely used in industrial applications such as spray painting, arc welding, assembly, polishing, etc.

In particular, in this section the PUMA robot is considered to be used for a spray painting application where the tool (spray gun) does not need to be completely perpendicular to the painted surface[5], i.e., there is no reference for the tool orientation although it has to be kept within a limit (Section 7.1.3). Therefore, three elements are considered for the robot workspace vector $\mathbf{p}$: the cartesian coordinates[6] $[x\ y\ z]^{\mathrm{T}}$ of a point located along the spray gun axis at a distance d_S from its nozzle. This point tracks the reference path on the painted surface, i.e., the spray gun stand-off distance is d_S. Assuming that the spray gun axis matches the axis of the last joint of the robot, the angle of the last joint has no influence on the workspace position and, hence, the last joint will not be further considered here. Therefore, the robot has two $(5-3)$ degrees of redundancy.

7.1. Constraints

7.1.1. Speed auto-regulation

The velocity constraints for the SAA are (11) and (12). If the flow rate of the spray gun is variable, it could be adjusted proportionally to the actual workspace speed $\|\dot{\mathbf{p}}\|_2$ to obtain the same paint deposition at any point on

[5]The same example shown in this work can be extended to other types of robotic applications, e.g. laser engraving, where the reference for the tool orientation can be relaxed within a cone of allowable values.

[6]The Z-axis of the reference frame is aligned with the first joint of the robot and its origin is located at the same height of the second joint, i.e., the shoulder joint.

the reference path, and the maximum workspace speed $\dot{p}_{\max}$ in (11) would be designed for the flow rate to not exceed its maximum value. Moreover, the spray gun should be switch off when the tracking error is above a prescribed threshold, since it only makes sense to paint the surface when the actual workspace position is close to the desired reference.

For the simulations in this section, the signals u_{SA} and f_{SA} of the SAA are splitted into two subsignals $\{u_{SA,W}, u_{SA,J}\}$ and $\{f_{SA,W}, f_{SA,J}\}$ corresponding to each type of SAA constraint.

7.1.2. Path conditioning

It will be considered that the boundary of the allowed workspace is given by two vertical planes $\{a, b\}$ and two horizontal planes $\{c, d\}$ which represent the limits of the "canvas" or painted surface:

$$\sigma_{PC,a} = y^*_{ref} - y_a \leq 0 \tag{32}$$

$$\sigma_{PC,b} = -(y^*_{ref} - y_b) \leq 0 \tag{33}$$

$$\sigma_{PC,c} = z^*_{ref} - z_c \leq 0 \tag{34}$$

$$\sigma_{PC,d} = -(z^*_{ref} - z_d) \leq 0, \tag{35}$$

where y_a, y_b, z_c and z_d are the parameters of each plane.

7.1.3. Redundancy resolution

It will be considered that the boundary of the allowed Cartesian position space for every point of the robot is given by two parallel vertical planes e and f placed on both sides of the robot in order to prevent collisions with other industrial machines located close to the robot. Since this space is convex and assuming for simplicity that the width of the robot links is negligible, the following constraints must be fulfilled to ensure that every part of the PUMA robot remains within the allowed Cartesian position space (see Section 5.1):

$$\sigma_{RR,pie} = y_i - y_e \leq 0, \qquad i = 1, \ldots, 6, \tag{36}$$

$$\sigma_{RR,pif} = -(y_i - y_f) \leq 0, \qquad i = 1, \ldots, 6, \tag{37}$$

where the subindex i is associated with the end-point of the ith moving link (i.e., $\mathbf{p}_i \equiv \bar{\mathbf{p}}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^{\mathrm{T}}$ is the position of the end-point of the ith moving link) and y_e and y_f are the parameters of the vertical planes e and f, respectively.

The following constraints are also considered for the joint limits:

$$\sigma_{RR,qi} = -1 + |q_{norm,i}| \leq 0, \quad i = 1, \ldots, 5, \tag{38}$$

where $q_{norm,i} = (q_i - q_{mid,i})/(\Delta q_{\max,i}/2)$ is the normalized joint position obtained using the mid joint position $q_{mid,i}$ and the joint maximum range of motion $\Delta q_{\max,i}$. Note that the joint limits are exceeded when the absolute value of the normalized joint position is greater than one.

Finally, another constraint is considered for the tool orientation:

$$\sigma_{RR,or} = \beta - \beta_{\max} \leq 0, \tag{39}$$

where β is the angle between the tool axis (i.e., the spray gun axis) and the line perpendicular to the painted surface and $\beta_{\max}$ is the maximum allowable value for this angle.

7.2. Reference

The reference path lies in a vertical plane and is given by the following expression which resembles a "flower" with eight petals:

$$\mathbf{p}_{ref}(\lambda) = \begin{bmatrix} x_{ref}(\lambda) \\ y_{ref}(\lambda) \\ z_{ref}(\lambda) \end{bmatrix} = \begin{bmatrix} 0.7818 \\ -0.1501 + 0.255 \sin(4\lambda) \cos(\lambda - \pi/8) \\ 0.0266 + 0.255 \sin(4\lambda) \sin(\lambda - \pi/8) \end{bmatrix}, \tag{40}$$

with $\lambda = 0 ... 2\pi$, where the units for linear and angular dimensions are meters and radians, respectively.

7.3. Simulation conditions and parameter values

Simulation was run under the following conditions:

i) For the sake of simplicity, the joint controllers of the robot have been implemented as a proportional correction of the joint speed error, i.e., $\ddot{\mathbf{q}}_{dj} = K_{dj}\,(\dot{\mathbf{q}}_d - \dot{\mathbf{q}})$, where $\ddot{\mathbf{q}}_{dj}$ is the desired joint acceleration vector and K_{dj} is the gain correction which has been set to $100\,\mathrm{s}^{-1}$. The inverse dynamics of the robot is used to compute the joint torques τ_i required to achieve the desired joint acceleration vector. The main sampling period T_R of the robot (i.e., the sampling period of both the input signals to the joint controllers and the readings obtained from the robot's sensors to be used by the kinematic controller) has been set to 5 milliseconds.

ii) No classical RRS was simulated (i.e., $\mathbf{b}_c = \mathbf{0}_n$) in order to focus on the behavior of the proposed SM algorithms.

iii) For the sake of simplicity, a proportional controller has been used for the correction of the position error, i.e., $C_p = K_p$. This gain correction K_p was set to 20 s^{-1} in all three coordinates.

iv) The constraint functions of the PCA and RRS were computed using the constraint approaching parameters $K_{PC,i} = K_{RR,or} = 0.03$ s and $K_{RR,pie} = K_{RR,pif} = K_{RR,qi} = 0.1$ s.

v) The SAA was implemented using a cutoff frequency α_{SA} of 20 rad/s for the filter, the maximum speeds $\dot{p}_{\max} = 0.6$ m/s and $\dot{q}_{\max,i} = 1$ rad/s and a desired motion rate parameter $\dot{\lambda}_d$ of 1 rad/s.

vi) The PCA was implemented using a cutoff frequency α_{PC} of 20 rad/s for the filter and an amplitude $\|\mathbf{u}_{SMp}\|_2 = 0.15$ for the switching law.

vii) The RRS was implemented using a cutoff frequency α_{RR} of 20 rad/s for the filter and an amplitude $\|\mathbf{u}_{SMb}\|_2 = 5$ for the switching law.

viii) All the algorithms were implemented with a sampling period of half millisecond.

ix) The tool length was set to 144 mm, i.e., the distance from the spray gun nozzle to the wrist center is equal to 200 mm. Moreover, the spray gun stand-off distance d_S was set to the typical value 250 mm.

x) The workspace constraints were computed with $y_a = 0.07$ m, $y_b = -0.37$ m, $z_c = 0.177$ m and $z_d = -0.123$ m.

xi) The Cartesian position constraints of the RRS were computed with $y_e = 0.01$ m and $y_f = -0.26$ m and the tool orientation constraint was computed with the maximum allowable angle $\beta_{\max} = 0.5$ rad.

xii) The joint limit constraints were computed using a mid joint position vector $\mathbf{q}_{mid} = \begin{bmatrix} 0 & \pi/2 & -\pi/2 & \pi/6 & 0 \end{bmatrix}^{\mathrm{T}}$ rad and a joint maximum range of motion $\Delta\mathbf{q}_{\max} = \begin{bmatrix} 5.55 & 4.643 & 4.521 & 4.887 & 3.491 \end{bmatrix}^{\mathrm{T}}$ rad [23].

xiii) We considered an initial workspace position error $\mathbf{e}_p(0) = \begin{bmatrix} 0.05 & 0.1 & -0.05 \end{bmatrix}^{\mathrm{T}}$ m and the initial robot configuration $\mathbf{q}(0) = \begin{bmatrix} -0.316 & 1.368 & -3.821 & -0.401 & 0.923 \end{bmatrix}^{\mathrm{T}}$ rad.

7.4. Simulation results

Fig. 5 to Fig. 12 show the simulated behavior of the global system. A schematic representation of the robot and the reference paths are depicted in Fig. 5 whereas Fig. 6 shows the torques generated by the joint controllers. The behavior of the PCA is shown in Fig. 7, where it can be seen that all four workspace constraints become active at some point; the control signal $\mathbf{f}_{PC}$ is non-zero whenever some constraints $\phi_{PC,i}$ is active to guarantee the constraint fulfillment, i.e., $\min(\phi_{PC,i}) <= 0$. Fig. 8 illustrates the fulfillment of the workspace constraints with the front view of the original and modified reference paths. The behavior of the RRS is shown in Fig. 9, where it can be seen that four different constraints become active at least once and some of them are simultaneously active on some phases; as before, the control signal $\mathbf{f}_{RR}$ is non-zero whenever some constraint $\phi_{RR,i}$ is active to guarantee the constraint fulfillment, i.e., $\min(\phi_{RR,i}) <= 0$. Fig. 10 illustrates the fulfillment of the three types of RRS constraint with the top view of the paths followed by the end-points of the robot links[7] and with the variation with time of the normalized joint positions and tool orientation. Fig. 11 shows the speed profile $\dot{\lambda}$ generated by the proposed SAA, where the desired motion rate $\dot{\lambda}_d$ is achieved when the workspace speed and joint velocity constraints are not active. It can bee seen in Fig. 12 that the initial position error is made zero. Note also in this figure that the normalized workspace speed and the joint velocities are within the limits except at the beginning of the reference tracking due to the initial position error, i.e., the motion rate is zero $\dot{\lambda}$ (Fig. 11) but velocity limits are exceeded due to the position error correction performed by the kinematic controller. Therefore, at the beginning of the reference tracking, the spray gun should be switched off to avoid painting at some point far from the target point on the painted surface.

[7]The path followed by point $\mathbf{p}_5$ is not shown in Fig. 10(a) because this point lies on the straight line connecting the points $\mathbf{p}_4$ and $\mathbf{p}_6$ (i.e., $\mathbf{p}_5$ fulfills the constraints of the Cartesian position space if both $\mathbf{p}_4$ and $\mathbf{p}_6$ fulfill them, see Section 5.1). Note also that point $\mathbf{p}_1$ remains static.

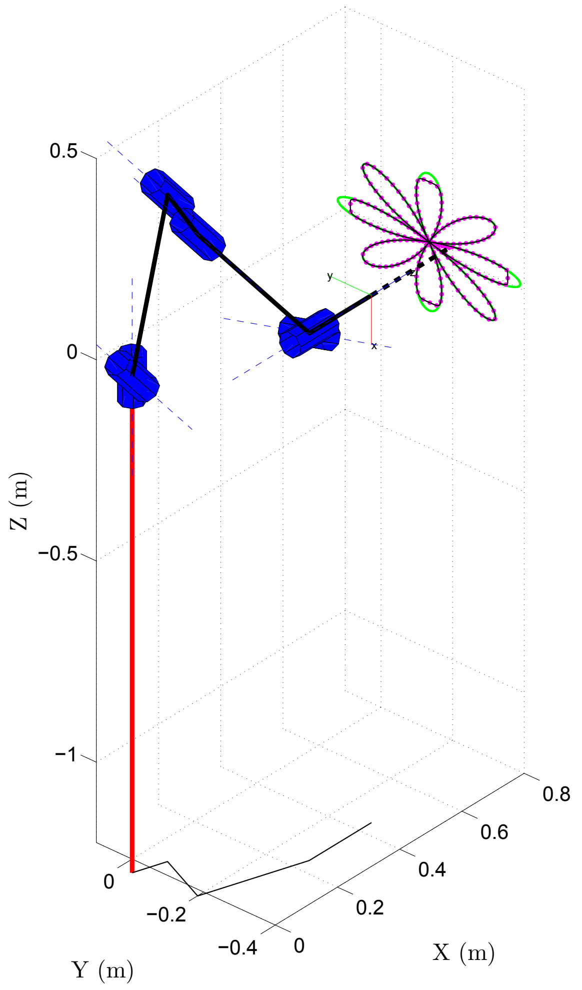


Fig. 5. 3D view of the original and modified reference paths and schematic representation of the PUMA robot in its initial configuration. The thick dashed line connects the spray gun nozzle with the point that tracks the target on the painted surface.

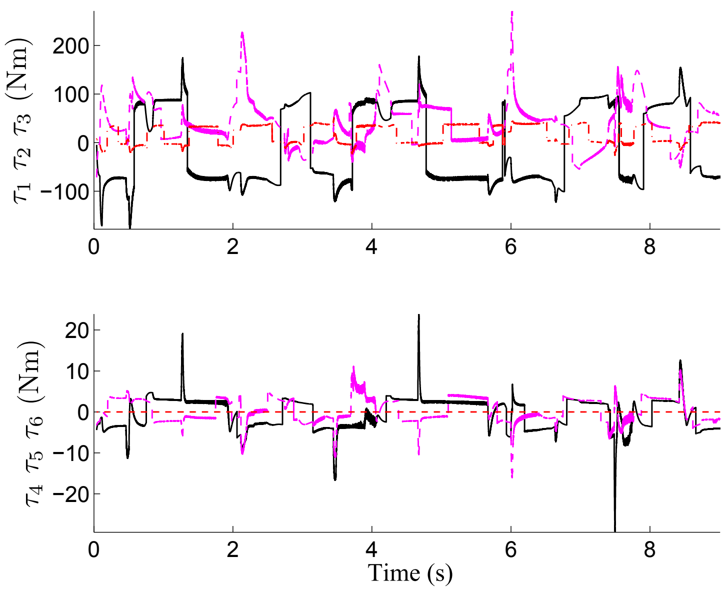


Fig. 6. Joint torques: $\{\tau_1,\tau_4\}$ (solid), $\{\tau_2,\tau_5\}$ (dashed) and $\{\tau_3,\tau_6\}$ (dashed-dotted).

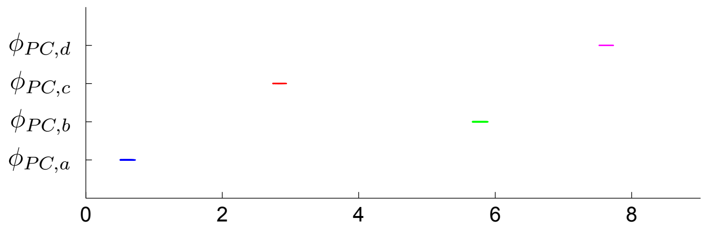
8. Conclusions

An integrated solution for robotic trajectory tracking was developed using sliding mode related concepts. In particular, the proposal includes three sliding-mode algorithms for speed auto-regulation, path conditioning and redundancy resolution. The proposed approach only requires a few program lines (see the Appendix) and simplifies the robot user interface since the method directly deals with the fulfillment of velocity, workspace and C-space constraints. Therefore, in case of relatively simple tasks the proposed method finds a feasible solution for the robot trajectory tracking in a short computation time.

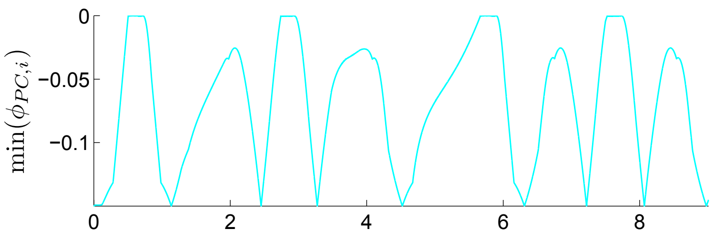
Although the method was illustrated for a particular 6R robot (PUMA-560 robot) and a particular industrial application (spray painting), the conclusions drawn for the proposed approach also apply to other robots and applications.

Appendix. Computer Implementation

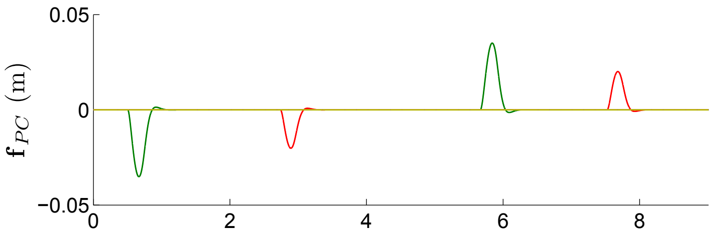
The pseudo-code of the proposed SAA (13)–(15), PCA (18)–(21) and RRS (26)–(29) is shown below. These three functions use the following auxiliary functions:



(a) Active workspace constraints.



(b) Minimum value of the PCA constraint functions.



(c) Correction values generated by the proposed PCA.

Fig. 7. Behavior of the path conditioning (abscissa axes: time in seconds).

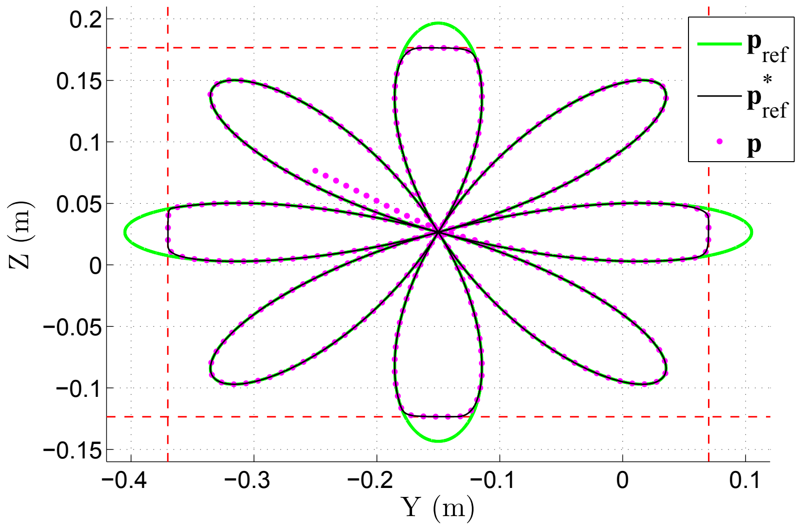


Fig. 8. Front view of the paths followed by the reference signals (original $\mathbf{p}_{ref}$ and modified $\mathbf{p}^*_{ref}$) and the tracking point ($\mathbf{p}$). The limit planes of the workspace constraints are shown as dashed lines.

- Filters: $\mathit{FiltFirstOrderSA}(\alpha_{SA}, u_{SA})$, $\mathit{FiltSecondOrderPC}(\alpha_{PC}, \mathbf{u}_{PC})$ and $\mathit{FiltFirstOrderRR}(\alpha_{RR}, \mathbf{u}_{RR})$ which are discrete time implementations of the low-pass filters (15), (19) and (27), respectively. Obviously, the filter implementations must take care of preserving their internal states between calls.

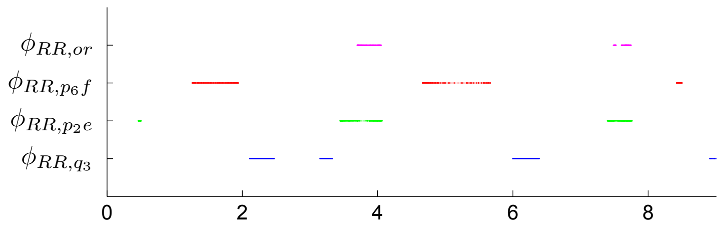
- Constraint functions and gradient vectors: $\sigma_{SA,i}(\dot{\mathbf{p}}_d, \dot{\mathbf{q}}_d)$, $\phi_{PC,i}(\mathbf{p}^*_{ref}, \dot{\mathbf{p}}^*_{ref})$, $\phi_{RR,i}(\mathbf{q}, \dot{\mathbf{q}})$, $\nabla\sigma_{PC,i}(\mathbf{p}^*_{ref})$ and $\nabla\sigma_{RR,i}(\mathbf{q})$.

- Path and kinematic functions: $\mathbf{v}(\lambda)$ and $\mathbf{l}(\mathbf{q})$.

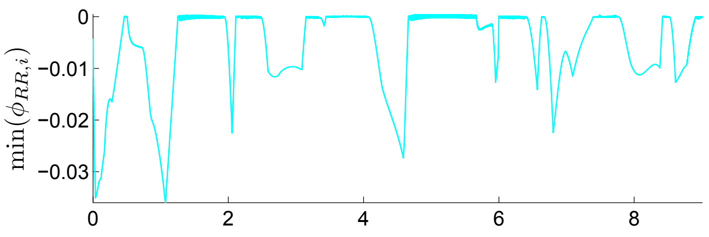
- Matrices and vectors for the redundancy resolution: pseudo-inverse Jacobian $\mathbf{J}^{\dagger}(\mathbf{q})$, basis $\mathbf{B}(\mathbf{q})$ for the null space of the robot Jacobian, and performance vector $\mathbf{b}_c(\mathbf{q}, \dot{\mathbf{q}})$ of a classical RRS.

- Sensors: $\mathit{GetSensorReadings}()$, which returns the current sensor readings $\mathbf{q}$ and $\dot{\mathbf{q}}$.

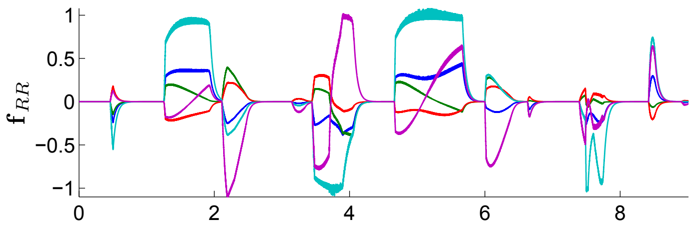
- Actuators: $\mathit{SendToJointControllers}(\dot{\mathbf{q}}_d)$, which sends the current desired joint velocity vector to the joint controllers.



(a) Active constraints.

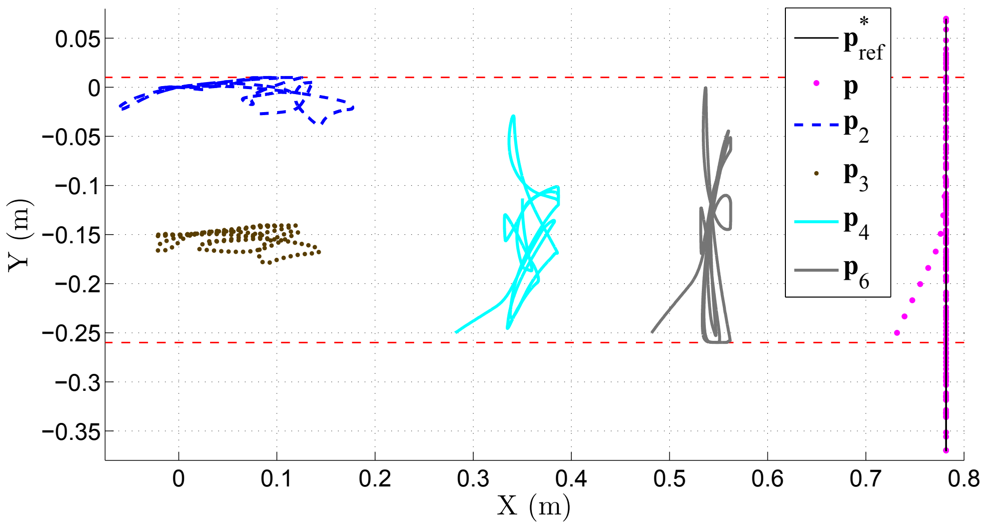


(b) Minimum value of the RRS constraint functions.

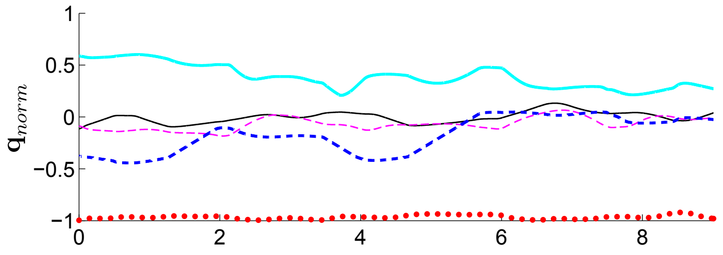


(c) Correction values generated by the proposed RRS.

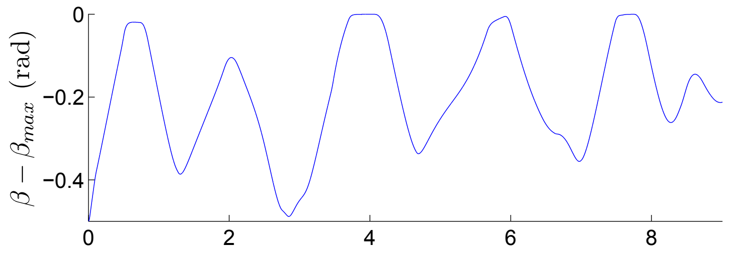
Fig. 9. Behavior of the redundancy resolution scheme (abscissa axes: time in seconds).



(a) Top view of the paths followed by several signals ($\mathbf{p}_1$ and $\mathbf{p}_5$ not shown because not needed, see footnote 7). The limit planes of the Cartesian position constraints are shown as thin dashed lines.

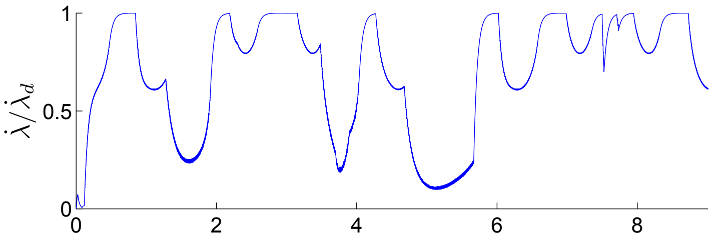


(b) Resulting normalized joint positions.

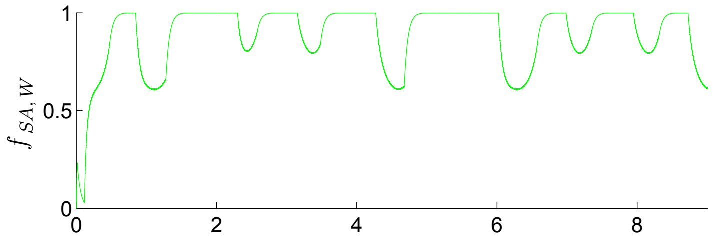


(c) Constraint function of the tool orientation.

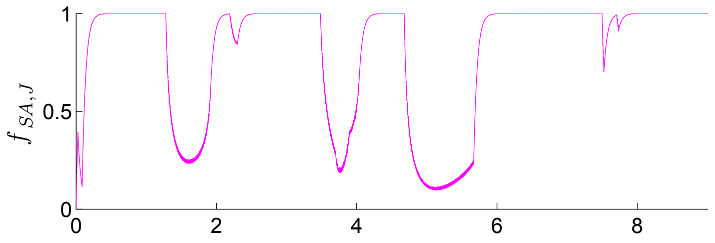
Fig. 10. Fulfillment of the RRS constraints (abscissa axes in (b) and (c): time in seconds).



(a) Scale factor generated by the proposed SAA.

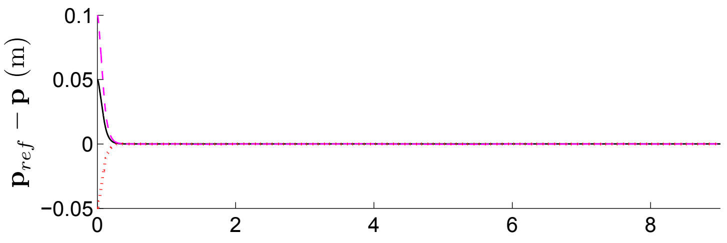


(b) Subsignal to fulfill the workspace speed constraint.

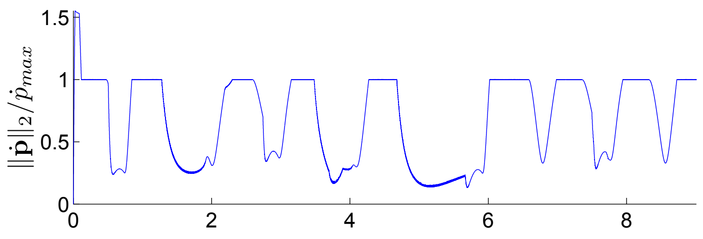


(c) Subsignal to fulfill the joint velocity constraints.

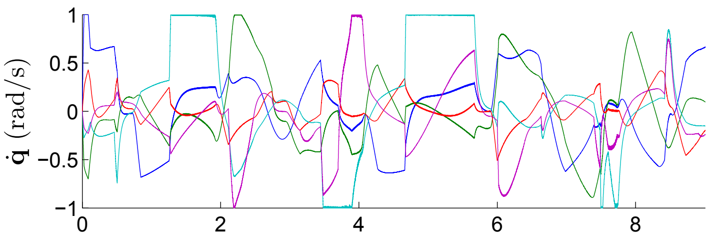
Fig. 11. Behavior of the speed auto-regulation (abscissa axes: time in seconds). Plot (a) is the product of signals in plots (b) and (c). Signal $\dot{\lambda}$ is zero while initial error is corrected (see Fig. 12)



(a) Workspace position errors.



(b) Normalized workspace speed.



(c) Joint velocities.

Fig. 12. Fulfillment of the SAA constraints after initial error correction phase (spray off). Abscissa axes: time in seconds.

The *SAA*, *PCA* and *RRS* functions are used by the *Main Loop* shown below, which calculates the robot tracking control shown in Fig. 1 at a sampling period of T_s seconds. Note that this implementation supports the claim made in the paper that the proposed approach only requires a few program lines. For the case study in Section 7, the computation time per iteration in a modern computer using MATLAB® R2009a (compiled C-MEX-file) was around 0.03 μs (microseconds) for the SAA, 0.6 μs for the PCA and 0.95 μs for the RRS.

```
Function SAA($\dot{\mathbf{p}}_d, \dot{\mathbf{q}}_d, \dot{\lambda}_d$)
1 if $\max(\sigma_{SA,1}(\dot{\mathbf{p}}_d, \dot{\mathbf{q}}_d), \ldots \sigma_{SA,N_{SA}}(\dot{\mathbf{p}}_d, \dot{\mathbf{q}}_d)) \geq 0$ then $u_{SA} = 0$;
2 else $u_{SA} = 1$ ;                                                    // Eq. (13)
3 $f_{SA}$ =FiltFirstOrderSA($\alpha_{SA}, u_{SA}$) ;                      // Eq. (15)
4 $\dot{\lambda} = f_{SA}\dot{\lambda}_d$ ;                                // Eq. (14)
5 return $\dot{\lambda}$;
```

```
Function PCA($\mathbf{p}_{ref}, \mathbf{p}^*_{ref}, \dot{\mathbf{p}}^*_{ref}$)
1 $\mathbf{k} = \mathbf{0}_m$;
2 for $i \leftarrow 1$ to $N_{PC}$ do
3     if $\phi_{PC,i}(\mathbf{p}^*_{ref}, \dot{\mathbf{p}}^*_{ref}) \geq 0$ then $\mathbf{k} = \mathbf{k} + \nabla\sigma_{PC,i}(\mathbf{p}^*_{ref})$ ;   // Eq. (21)
4 end
5 if $\|\mathbf{k}\|_2 \leq 10^{-6}$ then $\mathbf{u}_{PC} = \mathbf{0}_m$;
6 else $\mathbf{u}_{PC} = -\mathbf{k}\|\mathbf{u}_{SMp}\|_2/\|\mathbf{k}\|_2$ ;       // Eq. (20)
7 $\mathbf{f}_{PC}$ =FiltSecondOrderPC($\alpha_{PC}, \mathbf{u}_{PC}$) ;            // Eq. (19)
8 $\mathbf{p}^*_{ref} = \mathbf{p}_{ref} + \mathbf{f}_{PC}$ ;                       // Eq. (18)
9 return $\mathbf{p}^*_{ref}$;
```

```
Function RRS($\mathbf{q}, \dot{\mathbf{q}}$)
1  $\mathbf{k} = \mathbf{0}_n$;
2  for $i \leftarrow 1$ to $N_{RR}$ do
3      if $\phi_{RR,i}(\mathbf{q}, \dot{\mathbf{q}}) \geq 0$ then $\mathbf{k} = \mathbf{k} + \nabla\sigma_{RR,i}(\mathbf{q})$;
4  end
5  $\mathbf{k} = \mathbf{B}(\mathbf{q})^{\mathrm{T}}\mathbf{k}$ ;                                   // Eq. (29)
6  if $\|\mathbf{k}\|_2 \leq 10^{-6}$ then $\mathbf{u}_{RR} = \mathbf{0}_n$;
7  else $\mathbf{u}_{RR} = -\mathbf{k}\|\mathbf{u}_{SMb}\|_2/\|\mathbf{k}\|_2$ ;                 // Eq. (28)
8  $\mathbf{f}_{RR}$ =FiltFirstOrderRR($\alpha_{RR}, \mathbf{u}_{RR}$) ;                 // Eq. (27)
9  $\mathbf{b} = \mathbf{b}_c(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{f}_{RR}$ ;                          // Eq. (26)
10 return $\mathbf{b}$;
```

```
Main Loop
1  while $\lambda < \lambda_{end}$ do
2      $[\mathbf{q}, \dot{\mathbf{q}}]$ =GetSensorReadings();
3      $\lambda = \lambda + T_s\dot{\lambda}$ ;    // Discrete-time integration (see Fig. 1)
4      $\mathbf{p}_{ref} = \mathbf{v}(\lambda)$ ;                                    // Eq. (3)
5      $\mathbf{p}^*_{ref,prev} = \mathbf{p}^*_{ref}$;
6      $\mathbf{p}^*_{ref}$ =PCA($\mathbf{p}_{ref}, \mathbf{p}^*_{ref}, \dot{\mathbf{p}}^*_{ref}$) ;             // Path conditioning
7      $\dot{\mathbf{p}}^*_{ref} = (\mathbf{p}^*_{ref} - \mathbf{p}^*_{ref,prev})/T_s$ ;      // Discrete-time derivative
8      $\dot{\mathbf{p}}_d = \dot{\mathbf{p}}^*_{ref} + K_p(\mathbf{p}^*_{ref} - \mathbf{l}(\mathbf{q}))$ ;          // Kinematic controller
9      $\mathbf{b}$=RRS($\mathbf{q}, \dot{\mathbf{q}}$) ;                // Redundancy resolution scheme
10     $\dot{\mathbf{q}}_d = \mathbf{J}^{\dagger}(\mathbf{q})\dot{\mathbf{p}}_d + \mathbf{B}(\mathbf{q})\mathbf{b}$ ;                               // Eq. (5)
11     $\dot{\lambda}$= SAA($\dot{\mathbf{p}}_d, \dot{\mathbf{q}}_d, \dot{\lambda}_d$) ;    // Speed auto-regulation, $\dot{\lambda}_d$ set by
       user elsewhere
12     SendToJointControllers($\dot{\mathbf{q}}_d$);
13 end
```

References

[1] J.-C. Latombe, Robot Motion Planning, Kluwer, Boston, 1991.

[2] 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.

[3] L. Gracia, F. Garelli, A. Sala, A path conditioning method with trap avoidance, Robotics and Autonomous Systems 60 (2012) 862–873.

[4] 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.

[5] G. Antonelli, S. Chiaverini, G. Fusco, A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits, IEEE Transactions on Robotics and Automation 19 (2003) 162–167.

[6] L. Huo, L. Baron, The joint-limits and singularity avoidance in robotic welding, Industrial Robot: An International Journal 35 (2008) 456–464.

[7] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots, IEEE Transactions on Systems, Man, and Cybernetics 19 (1989) 1179–1187.

[8] M. Li, Z. Lu, C. Sha, L. Huang, Trajectory generation of spray painting robot using point cloud slicing, Applied Mechanics and Materials 44-47 (2011) 1290–1294.

[9] Z. Liu, W. Bu, J. Tan, Motion navigation for arc welding robots based on feature mapping in a simulation environment, Robotics and Computer-Integrated Manufacturing 26 (2010) 137–144.

[10] J. Gomez Ortega, J. Gamez Garcia, S. Satorres Martinez, A. Sanchez Garcia, Industrial assembly of parts with dimensional variations. Case study: Assembling vehicle headlamps, Robotics and Computer-Integrated Manufacturing 27 (2011) 1001–1010.

[11] E. Pitschke, M. Schinhaerl, R. Rascher, P. Sperber, L. Smith, R. Stamp, M. Smith, Simulation of a complex optical polishing process using a neural network, Robotics and Computer-Integrated Manufacturing 24 (2008) 32–37.

[12] J. Angeles, Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms, Springer-Verlag, New York, NJ, 3rd edition, 2007.

[13] W. Khalil, E. Dombre, Modeling, Identification and Control of Robots, Taylor & Francis Inc., Bristol, PA, 2002.

[14] 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.

[15] G. Golub, C. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[16] A. Liégeois, Automatic supervisory control of the configuration and behavior of multibody mechanisms, IEEE Transactions on Systems, Man and Cybernetics 7 (1977) 868–871.

[17] F. Garelli, R. Mantz, H. De Battista, Advanced Control for Constrained Processes and Systems, IET, Control Engineering Series, London, UK, 2011.

[18] W. Perruquetti, J. Barbot (Eds.), Sliding Mode Control in Engineering, Control Engineering Series, Marcel Dekker, 2002.

[19] C. Edwards, S. Spurgeon, Sliding Mode Control: Theory and Applications, Taylor & Francis, UK, 1st edition, 1998.

[20] V. Utkin, J. Guldner, J. Shi, Sliding Mode Control in Electro-Mechanical Systems, Taylor & Francis, London, 2nd edition, 2009.

[21] E. Bernabeu, J. Tornero, Optimal geometric modeler for robot motion planning, Journal of Robotic Systems 17 (2000) 593–608.

[22] P. Corke, A robotics toolbox for matlab, IEEE Robotics and Automation Magazine 3 (1996) 24–32.

[23] S. Elgazzar, Efficient kinematic transformations for the PUMA 560 robot, IEEE Journal of Robotics and Automation 1 (1985) 142–151.