Robust Fulfillment of Constraints in Robot Visual Servoing

Pau Muñoz-Benavent[a], Luis Gracia*[a], J. Ernesto Solanes[a], Alicia Esparza[a], Josep Tornero[a]

[a]*Instituto IDF, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain (e-mail: luigraca@isa.upv.es, fax: +34-963879579).* **Corresponding author*

Abstract

In this work, an approach based on sliding mode ideas is proposed to satisfy constraints in robot visual servoing. In particular, different types of constraints are defined in order to: fulfill the visibility constraints (camera field-of-view and occlusions) for the image features of the detected object; to avoid exceeding the joint range limits and maximum joint speeds; and to avoid forbidden areas in the robot workspace. Moreover, another task with low-priority is considered to track the target object. The main advantages of the proposed approach are low computational cost, robustness and fully utilization of the allowed space for the constraints. The applicability and effectiveness of the proposed approach is demonstrated by simulation results for a simple 2D case and a complex 3D case study. Furthermore, the feasibility and robustness of the proposed approach is substantiated by experimental results using a conventional 6R industrial manipulator.

Keywords: visual servoing, sliding mode, robot control

1. Introduction

Visual servoing (VS) refers to the motion control of a robot system using visual feedback signals from a vision device [1]. For this purpose, a computer vision algorithm must be used to obtain the visual *features* of the target object present in the scene and observed by the camera. This information is used to compute the robot control law in order to achieve the desired robot pose. Taking into consideration the workspace in which the control law is computed [1], the following classification can be made: position-based visual

servoing (PBVS), in which the control law is carried out in the operational space, the relative 3D pose of the object is reconstructed from visual features with respect to the camera-robot system and the error is defined between the computed current and desired 3D poses; and image-based visual servoing (IBVS), in which the control low is directly computed in the image space and the error is defined between current and desired visual features in the image.

Regardless of the workspace in where VS control laws are computed, the following mechanical constraints can be violated: *joint range limits*; *maximum joint speeds*; and *forbidden areas*, such as the ones defined to avoid kinematic singularity, to avoid collisions [2] between the robot manipulator and objects in the environment, etc. Furthermore, since the VS control law depends on the visual feedback, it is convenient to consider the so-called *visibility constraint* in order to keep the image features within the camera field-of-view (FOV) and to avoid occlusions with the obstacles in the environment during all the task[1].

Due to the fact that the violation of any of the aforementioned mechanical and visual constraints can lead to the VS control task failure, different approaches have been presented to address this issue. For instance, based on the idea of *combining advantages of PBVS and IBVS* while trying to avoid their shortcomings [6]: authors in [7] presented a switching method between IBVS and PBVS; authors in [8] introduced a switching approach which uses the classic PBVS control law and backward motion along the camera optical axis; authors in [9] proposed a switching approach using hybrid VS control laws and pure translation motions; authors in [10] introduced a path planning and PBVS-IBVS switching method in order to deal with image singularities and local minima; authors in [11] presented a combination approach which uses 2D and 3D information from IBVS and PBVS to ensure the *visibility constraint*; and authors in [12] proposed a combination method based on weighting IBVS and PBVS control strategies with a 5D objective function.

Other proposals rely on *path planning* algorithms: besides of the work of [10] commented above, authors in [13] presented a shortest-path method to guarantee both shortest Cartesian trajectory and object visibility; authors in [14] presented a path planning method which uses a probabilistic road

[1]Some approaches [3, 4, 5] provide solutions when loss of the image features occur based on the prediction of the feature behavior, although the main problem of these solutions is that robustness and convergence cannot be guaranteed, specially when the target is moving along an unknown or unpredictable trajectory.

map; authors in [15] introduced a global path planning method to take into account visibility, workspace and joint constraints; authors in [16] addressed the issue with a path planning approach based on the use of homogeneous forms and linear matrix inequalities; authors in [17] proposed a path planning approach using search trees and IBVS trajectory tracking; authors in [18] introduced a time-independent path tracking in the image and 3D space approach for unstructured environments; authors in [19] presented a vision-based trajectory planning approach from the point of view of a constrained optimal control problem, solved by using the Gauss pseudo-spectral method.

Furthermore, there are some proposals relying on *online corrective* terms: authors in [20] introduced a partitioned approach to IBVS control with the combination of a potential function for giving solution to the *visibility constraint* issue; authors in [21] developed a path-following IBVS controller that utilizes a potential function to incorporate *motion* constraints; and authors in [22] and [23] presented an approach that employs a specialized potential function, namely navigation function.

In addition, some authors have focused his research on proposing more complex VS controllers to address the commented constraints. For instance, authors in [24, 25, 26] introduced control laws based on model predictive control frameworks, whilst authors in [27] on control Lyapunov functions. Moreover, authors in [28, 29] developed several control laws in order to deal with joint limits and space singularities.

On the other hand, other authors have focused on providing more feasible trajectories in other to avoid visibility and mechanical constraints. Thus, in [30], authors dealt with the visibility constraint problem using a neural network approach which assists a Kalman filter, whilst in [31], circular-like trajectories are designed to ensure shorter displacements and visibility.

Finally, some authors relay their proposals on new VS control tasks. For instance, in [5], the camera invariant VS approach is redefined to take into account the changes of visibility in image features, and in [32], a global full-constraining task is divided into several subtasks that can be applied or inactivated to take into account potential constraints of the environment.

This paper addresses the problem of mechanical and visual constraints in VS with an alternative solution to all mentioned above. The proposed method can be interpreted as a limit case of *artificial potential fields* [33]. 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 constraint

limits, and a large value when touching them. One of the advantages of this approach is that the allowed space is fully utilized, although some corrective speed-related terms are needed to avoid approaching the limits at high speed.

Discontinuous control laws have been deeply studied in the context of sliding mode control (SMC) [34, 2]. Concretely, in VS field of research SMC has been used mainly to increase the robustness against errors while executing the main robot control task [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47]. However, to the best of the authors knowledge, SMC techniques have not yet been used in VS to fulfill constraints.

It may perhaps be observed that the algorithm solution proposed in this work cannot be seen as a conventional SMC, since the algorithm is only activated when the VS system is about to violate any constraint, whilst a pure SMC would always be active to keep the system on the sliding surface. Besides the SMC algorithm to fulfill the visibility constraints, another task with low-priority [48] is considered to track the target object.

The paper is organized as follows: next section introduces some preliminaries and objectives, while Section 3 presents the basic theory used in this work. The proposed method is developed in Section 4, while some important remarks about the method are given in Section 5. The main advantages and disadvantages of the proposed approach are discussed in Section 6. Subsequently, Section 7 presents the conditions considered for the simulations and experiments. The proposed approach is applied in Section 8 and Section 9 to a simple 2D case and a complex 3D case study, respectively, in order to show its applicability and effectiveness. The feasibility and robustness of the proposed approach is substantiated by experimental results in Section 10 using a conventional 6R industrial manipulator: the Kuka KR6 R900 sixx (Agilus). Finally, some concluding remarks are given.

2. Preliminaries and objective

Coordinate frames. Fig. 1 shows the coordinate frames involved in the VS problem: F robot base frame; E robot end-effector frame; C current camera frame; C^* desired camera frame; O object frame; $C2$ camera frame for eye-to-hand configuration (in this case the camera does not move with the robot).

Kinematics. The VS application is characterized by the so-called visual *feature vector* $\mathbf{s}$, which is computed from image measurements [1]. In general, this vector depends on the robot *configuration* $\mathbf{q}$ and on the time for the case

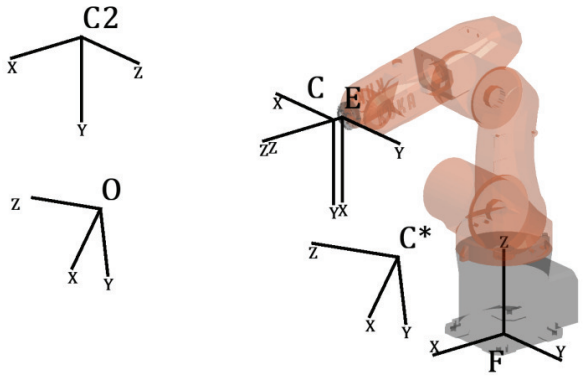

Fig. 1. Frames involved in visual servoing.

of a *moving target* object, that is:

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

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

The first-order kinematics of the feature vector $\mathbf{s}$ results in:

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{l}(\mathbf{q}, t)^{\mathrm{T}}}{\partial \mathbf{q}} \dot{\mathbf{q}} + \frac{\partial \mathbf{l}(\mathbf{q}, t)}{\partial t} = \mathbf{J}_s \dot{\mathbf{q}} + \partial \mathbf{s}/\partial t, \tag{2}$$

where $\partial \mathbf{s}/\partial t$ is due to the target motion and $\mathbf{J}_s$ is the resulting Jacobian matrix, which can be expressed as a concatenation of three different Jacobian matrices:

$$\mathbf{J}_s(\mathbf{q}, t) = \mathbf{L}_s(\mathbf{q}, t)\, {}^c\mathbf{V}_e\, {}^e\mathbf{J}_e(\mathbf{q}), \tag{3}$$

where $\mathbf{L}_s$ is the so-called *interaction matrix* related to the visual feature vector $\mathbf{s}$; ${}^c\mathbf{V}_e$ is the spatial motion transformation matrix from the camera frame C to the end-effector frame E (which is constant for eye-in-hand systems); and ${}^e\mathbf{J}_e$ is the robot Jacobian expressed in the end-effector frame (other coordinate frames could be used for eye-to-hand configuration). For more details on the computation of $\mathbf{J}_s$ see [1, 49].

The second-order kinematics of the feature vector $\mathbf{s}$ results in:

$$\ddot{\mathbf{s}} = \mathbf{J}_s\ddot{\mathbf{q}} + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{s}}/\partial t. \tag{4}$$

Reference. The robot system should carry out a task, which in VS applications consists on achieving a reference value for the visual feature vector $\mathbf{s}$. Hence, the robot task is given by the following equation:

$$\mathbf{s}(\mathbf{q}, t) = \mathbf{s}_{ref}(t), \tag{5}$$

where $\mathbf{s}_{ref}(t)$ is the reference for the visual feature vector and can be either constant or varying in time.

Computer vision algorithm. This algorithm is required for both PBVS and IBVS and is composed of three parts: the first part consists of the *image processing* for obtaining the image plane coordinates (u_i, v_i) of all the visual feature; the second part consists of the *coordinate transformation* for converting the pixel coordinates (u_i, v_i) to the corresponding value in the normalized image plane using the matrix of the camera intrinsic parameters; and the third part consists of the *pose estimation* of the camera (eye-in-hand) or robot (eye-to-hand) from the features of the second part. The output of the computer vision algorithm, both for PBVS and IBVS, is the visual feature vector $\mathbf{s}$. This work assumes existence of this computer vision algorithm.

Robot control. This work also assumes the existence of an underlying robot control in charge of achieving a particular joint acceleration from an acceleration command $\ddot{\mathbf{q}}_c$. Nevertheless, the actual joint acceleration $\ddot{\mathbf{q}} = \mathcal{C}\ddot{\mathbf{q}}_c + \mathbf{d}_c$ will not be exactly the commanded one $\ddot{\mathbf{q}}_c$, where $\mathcal{C}$ represents the dynamics of the low-level control loop and $\mathbf{d}_c$ represents inaccuracies due to disturbances. However, in this work it is assumed that the dynamics of $\mathcal{C}$ is fast enough compared to that of $\ddot{\mathbf{q}}_c$ so that the relationship:

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_c + \mathbf{d}_c, \tag{6}$$

holds approximately true, avoiding the need of including extra state variables. Note that, the *dynamic model* of the robot system should be taken into account to properly design the mentioned underlying joint controller. Obviously, for stability reasons, the bandwidth of this underlying robot control should be faster than that of the used kinematic control.

Problem definition. The goal of this work is to design a VS system that is aware of the robot configuration $\mathbf{q}$ and that generates the commanded joint acceleration vector $\ddot{\mathbf{q}}_c$ to be sent to the joint controllers of the robot, so that the actual visual feature vector $\mathbf{s}$ tracks the reference value $\mathbf{s}_{ref}$ and the system satisfies the following requirements:

- the image features of the target object fulfill the visibility constraints (camera FOV and occlusions);
- the joint range limits and the maximum joint speeds are not exceeded during the operation;
- and the robot does not collide during the motion with the objects of the environment that are located within its workspace.

3. Geometric invariance using sliding mode control

This section develops a non-conventional SMC that will be subsequently used by the proposed approach. Let us consider a dynamical system with n_x states and n_u inputs given by:

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

where $\mathbf{x}(t)$ is the state vector, $\mathbf{d}(t)$ is an unmeasured disturbance or model uncertainty, $\mathbf{u}(t)$ is the control input vector (possibly discontinuous), $\mathbf{f}$ is a vector field and $\mathbf{g}$ is a set of n_u vector fields.

Consider also that the system state vector $\mathbf{x}$ is subject to user-specified inequality constraints $\phi_i(\mathbf{x}) \leq 0$, $i = 1, \ldots, N$, where $\phi_i(\mathbf{x})$ is the ith inequality constraint function. Thus, the region Φ of the state space compatible with the constraints on state $\mathbf{x}$ is given by:

$$\Phi = \{\mathbf{x} \,|\, \phi_i(\mathbf{x}) \leq 0\}, \quad i = 1, \ldots, N. \tag{8}$$

From the invariance point of view, the objective is to find a control input $\mathbf{u}$ such that the trajectories originating in Φ remain in Φ for all times t, i.e., the control input $\mathbf{u}$ must ensure that the right hand side of (7) points to the interior of Φ at all points in the boundary of Φ. Mathematically, the

invariance of Φ is guaranteed by an input $\mathbf{u}$ such that[2]:

$$\begin{aligned}\frac{d(\phi_i(\mathbf{x}))}{dt} &= \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\dot{\mathbf{x}} = \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{f}(\mathbf{x},\mathbf{d}) + \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{g}(\mathbf{x})\,\mathbf{u} \\ &= L_f\phi_i(\mathbf{x},\mathbf{d}) + \mathbf{L_g}\phi_i(\mathbf{x})\mathbf{u} \leq 0, \quad \forall i \mid \phi_i(\mathbf{x}) \geq 0, \end{aligned} \tag{9}$$

where ∇ denotes the gradient vector and the scalar $L_f\phi_i$ and the n_u-dimensional row vector $\mathbf{L_g}\phi_i$ denote the Lie derivatives of $\phi_i(\mathbf{x})$ in the direction of vector field $\mathbf{f}$ and in the direction of the set of vector fields $\mathbf{g}$, respectively. The constraints such that $\phi_i(\mathbf{x}) \geq 0$ are denoted as *active* constraints.

In general, any vector $\mathbf{u}$ such that the scalar $\mathbf{L_g}\phi_i\mathbf{u}$ is negative (i.e., any vector pointing toward the interior of the allowed region) can be used to satisfy (9). In particular, this work considers a simple strategy involving only gradient computation and simple matrix operations. It is proposed to use the variable structure control law below to make the set Φ invariant:

$$\mathbf{u} = \begin{cases} \mathbf{0} & \text{if} \quad \max_i \{\phi_i(\mathbf{x})\} < 0 \\ \mathbf{u}_c & \text{otherwise,} \end{cases} \tag{10}$$

where vector $\mathbf{u}_c$ is chosen to satisfy:

$$\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{u}_c = -\mathbf{1}_b\,u^+, \tag{11}$$

where matrix $\mathbf{L_g}\boldsymbol{\phi}$ contains the row vectors $\mathbf{L_g}\phi_i$ of all active constraints, b is the number of active constraints, $\mathbf{1}_b$ is the b-dimensional column vector with all its components equal to one and u^+ is the so-called switching gain, which a positive constant to be chosen high enough to satisfy (9). In particular, one set of *sufficient*, but not necessary, conditions for making the set Φ invariant are that matrix $\mathbf{L_g}\boldsymbol{\phi}$ is *full row rank* and that:

$$u^+ > \sum_{i=1}^{b} (\max(L_f\phi_i, 0)). \tag{12}$$

where $L_f\boldsymbol{\phi}$ is a column vector containing the elements $L_f\phi_i$ of all constraints.

[2]Note that it is assumed that the constraint function ϕ_i is differentiable around the boundary given by $\phi_i = 0$.

Proof. See Appendix A. □

When the state trajectory tries by itself to leave the allowed region Φ, the above control law (10) will make $\mathbf{u}$ switch between $\mathbf{0}$ and $\mathbf{u}_c$ at a theoretically infinite frequency, which can be seen as an ideal sliding mode (SM) behaviour with no open-loop phase (reaching mode) [34]. In fact, this approach is a non-conventional SMC. Once SM is established on the boundary of Φ by the control action $\mathbf{u}$, a continuous equivalent control [34] can be obtained, i.e., the control required to keep the system on the boundary of Φ. Hence, the SMC generated by (10) produces such control action without explicit knowledge of it and with a low computational cost, which is a typical advantage of SMC strategies [50].

4. Proposed approach

The approach developed in this section to address the problem defined in Section 2 is based on task-priority and the SMC presented in Section 3.

4.1. System tasks

Two tasks with different priority levels are considered:

- The first level (high-priority task) includes the *constraints* that must be satisfied at all times to avoid that the image features of the target object leave the camera FOV or get occluded by the obstacles, to avoid exceeding the robot limits and to avoid invasion of forbidden space.
- The second level (low-priority task) is designed for *reference tracking* in order that the visual feature vector $\mathbf{s}$ follows the reference $\mathbf{s}_{ref}$: deviations from the reference trajectory are allowed if such deviations are required to fulfill the above constraints.

The input to these tasks is the robot state $\{\mathbf{q}, \dot{\mathbf{q}}\}$ and each task gives an acceleration equality whose square error must be minimized. The equality for each task is generically given by:

$$\mathbf{A}_1 \ddot{\mathbf{q}}_c = \mathbf{b}_1 \tag{13}$$

$$\mathbf{A}_2 \ddot{\mathbf{q}}_c = \mathbf{b}_2, \tag{14}$$

where matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ and vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ for each task are assumed known and subscript represents the priority order (1 for highest priority).

The acceleration equality for the first level is obtained below using the non-conventional SMC developed in Section 3, in order to fulfill the corresponding constraints.

The commanded joint acceleration vector $\ddot{\mathbf{q}}_c$, which serves as input to the joint controllers of the robots, is obtained by the task-priority redundancy resolution [48] as follows:

$$\ddot{\mathbf{q}}_c = \mathbf{A}_1^{\dagger}\mathbf{b}_1 + (\mathbf{A}_2(\mathbf{I} - \mathbf{A}_1^{\dagger}\mathbf{A}_1))^{\dagger}(\mathbf{b}_2 - \mathbf{A}_2\mathbf{A}_1^{\dagger}\mathbf{b}_1), \tag{15}$$

where $\mathbf{I}$ denotes the identity matrix of suitable size and superscript $\dagger$ denotes the well-known Moore-Penrose pseudoinverse[3].

4.2. Lie derivatives

In order to use the theory in Section 3, a dynamical system in the form of Eq. (7) is considered with the state vector $\mathbf{x} = \begin{bmatrix}\mathbf{q}^{\mathrm{T}} & \dot{\mathbf{q}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, the disturbance vector $\mathbf{d} = \mathbf{d}_c$ and the input vector $\mathbf{u} = \ddot{\mathbf{q}}_c$. Hence, the model is a double integrator, and from (6) the state equation results in:

$$\dot{\mathbf{x}} = \begin{bmatrix}\mathbf{O} & \mathbf{I} \\ \mathbf{O} & \mathbf{O}\end{bmatrix}\mathbf{x} + \begin{bmatrix}\mathbf{0} \\ \mathbf{d}_c\end{bmatrix} + \begin{bmatrix}\mathbf{O} \\ \mathbf{I}\end{bmatrix}\mathbf{u}, \tag{16}$$

and the Lie derivatives in (9) for the constraint function ϕ_i are given by:

$$\mathbf{L}_{\mathbf{g}}\phi_i = \nabla\phi_i^{\mathrm{T}}\,\mathbf{g} = (\partial\phi_i/\partial\dot{\mathbf{q}})^{\mathrm{T}} \tag{17}$$

$$L_f\phi_i = \nabla\phi_i^{\mathrm{T}}\,\mathbf{f} = (\partial\phi_i/\partial\mathbf{q})^{\mathrm{T}}\ \dot{\mathbf{q}} + (\partial\phi_i/\partial\dot{\mathbf{q}})^{\mathrm{T}}\ \mathbf{d}_c. \tag{18}$$

4.3. Level 1: visibility and robot constraints

To satisfy the constraints, the SMC in Section 3 is considered with the dynamical system and Lie derivatives (17) and (18).

4.3.1. Modified constraints

Approaching the constraints at high speed is not advisable because, in general, large joint accelerations $\ddot{\mathbf{q}}$ would be required to slow down the robot

[3]Pseudoinverse may be computed via the singular value decomposition (SVD) method [51] and using a tolerance to set to zero the very small singular values in order to avoid extremely large values for the commanded accelerations.

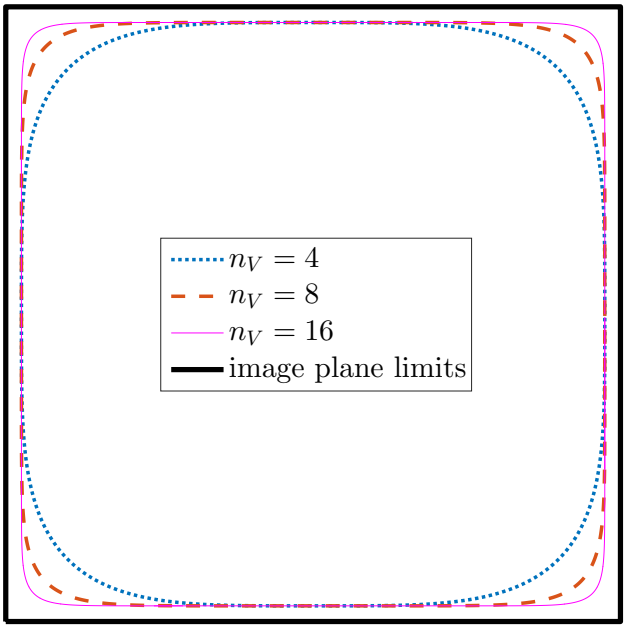

Fig. 2. Boundary of the FOV visibility constraint (i.e., $\sigma_{V,i} = 0$) for a safety margin m_V of 5% and differnt values of n_V.

motion in order to keep it on the constraint boundary. Therefore, the actual constraint $\sigma_i(\mathbf{q})$ will be modified to include the speed of movement as follows:

$$\phi_i(\mathbf{q}, \dot{\mathbf{q}}) = \sigma_i + K_i \dot{\sigma}_i = \sigma_i + K_i \left(\partial \sigma_i / \partial \mathbf{q}\right)^{\mathrm{T}} \dot{\mathbf{q}} \leq 0 \tag{19}$$

where K_i is a free design parameter that determines the rate of approach to the boundary of the constraint.

4.3.2. FOV constraints

The image features of the target object should not leave the camera FOV since they are required to compute the robot control law. In this work, to avoid that, a constraint is defined for each image feature in order to confine them within the image plane limits. For this purpose, we consider the formula of the *superellipse* [52]: $|x/a|^n + |y/b|^n = 1$, that for $n > 2$ is called hyperellipse and looks like a rectangle with rounded corners, see Fig. 2. Therefore, the visibility constraint $\sigma_{V,i}$ for the target's image feature

i is given by:

$$\sigma_{V,i}(u_i, v_i) = \left| \frac{u_i}{\frac{W_V}{2}(1 - m_V)} \right|^{n_V} + \left| \frac{v_i}{\frac{H_V}{2}(1 - m_V)} \right|^{n_V} - 1 \leq 0, \tag{20}$$

where u_i and v_i are the pixel coordinates of the image feature i in the image plane with respect to the center of the image; n_V is the hyperellipse smoothing parameter, which is a design parameter to smooth more or less the rounded corners of the constraint boundary (see Fig. 2); W_V and H_V are the width and height, respectively, of the rectangle representing the image plane limits (thick dark line in Fig. 2) in pixels; and m_V is the safety margin for the visibility constraints to cater for possible errors and inaccuracies, e.g., a safety margin of 5% is represented in Fig. 2.

Since the pixel coordinates (u_i, v_i) of the feature depend on the robot configuration $\mathbf{q}$, the above constraints will be modified as indicated in Section 4.3.1 for the sliding manifold to have relative degree one with respect to the control variable $\ddot{\mathbf{q}}_c$, that is:

$$\phi_{V,i}(\mathbf{q}, \dot{\mathbf{q}}) = \sigma_{V,i} + K_{V,i}\dot{\sigma}_{V,i} = \sigma_{V,i} + K_{V,i}\, \nabla\sigma_{V,i}^{\mathrm{T}}\, \dot{\mathbf{q}} \leq 0, \tag{21}$$

where $K_{V,i}$ is the approaching parameter to the hyperellipse.

Using the chain rule, the gradient vector $\nabla\sigma_{V,i}$ for the FOV constraints is obtained as follows:

$$\begin{aligned} \nabla\sigma_{V,i} &= \begin{bmatrix} \partial u_i/\partial \mathbf{q} & \partial v_i/\partial \mathbf{q} \end{bmatrix} \begin{bmatrix} \partial\sigma_{V,i}/\partial u_i \\ \partial\sigma_{V,i}/\partial v_i \end{bmatrix} = \mathbf{J}_{sx}^{\mathrm{T}} \begin{bmatrix} \partial\sigma_{V,i}/\partial u_i \\ \partial\sigma_{V,i}/\partial v_i \end{bmatrix} \\ &= (\mathbf{L}_{sx}\, {}^{c}\mathbf{V}_{e}\, {}^{e}\mathbf{J}_{e})^{\mathrm{T}} \begin{bmatrix} \partial\sigma_{V,i}/\partial u_i \\ \partial\sigma_{V,i}/\partial v_i \end{bmatrix}, \end{aligned} \tag{22}$$

where Jacobian matrix $\mathbf{J}_{sx}$ represents the first-order kinematics of the image plane coordinates with respect to $\mathbf{q}$, which is splitted into three matrices $\mathbf{L}_{sx}\, {}^{c}\mathbf{V}_{e}\, {}^{e}\mathbf{J}_{e}$ analogously to (3), and the partial derivatives $\partial\sigma_{V,i}/\partial u_i$ and $\partial\sigma_{V,i}/\partial v_i$ are straightforward obtained from (20) as:

$$\frac{\partial\sigma_{V,i}}{\partial u_i} = \frac{n_V \mathrm{sign}(u_i)\, |u_i|^{n_V - 1}}{\left(\frac{W_V}{2}(1 - m_V)\right)^{n_V}} \quad \frac{\partial\sigma_{V,i}}{\partial v_i} = \frac{n_V \mathrm{sign}(v_i)\, |v_i|^{n_V - 1}}{\left(\frac{H_V}{2}(1 - m_V)\right)^{n_V}}, \tag{23}$$

where $\mathrm{sign}(\cdot)$ is the *sign* function, and $\mathbf{L}_{sx}$ represents the well-known inter-

action matrix typically used in IBVS, which is given by [1]:

$$L_{sx} = \begin{bmatrix} -Z^{-1} & 0 & xZ^{-1} & xy & -(1+x^2) & y \\ 0 & -Z^{-1} & yZ^{-1} & 1+y^2 & -xy & -x \end{bmatrix}$$
$$\text{with } x = u_i/F_u \text{ and } y = v_i/F_v, \tag{24}$$

where F_u and F_v are the ratio between the focal length and the size of a pixel and Z is the distance from the camera to the target.

4.3.3. Occlusion constraints

The image features may be occluded during the servoing task by the obstacles located between the camera and the target. To avoid this situation, a constraint can be used to guarantee that all image features do not enter the area defined by these obstacles, which represents a forbidden area. The procedure is as follows: in the first place, the computer vision algorithm described in Section 2 has to detect the obstacle in the image plane; then, a specific differentiable function (e.g., circle, hyperellipse, etc.) has to be used to enclose the obstacle; and finally, the corresponding inequality constraint is obtained as $\sigma_{V,i}(u_i, v_i) \leq 0$, where, as opposed to the above FOV constraints, function $\sigma_{V,i}$ is negative for a point outside the enclosed area and positive otherwise. The gradient vector computation is given by (22), (24) and the partial derivatives $\partial\sigma_{V,i}/\partial u_i$ and $\partial\sigma_{V,i}/\partial v_i$ of the specific function $\sigma_{V,i}$ used to enclosed the object.

4.3.4. Constraints for the joint range limits

The following constraints are considered for the joint limits:

$$\begin{aligned} \sigma_{R,qi}(\mathbf{q}) &= -1 + \frac{|\, q_i - q_{\text{mid},i}|}{\Delta q_{\text{max},i}/2} + m_{R,q} \\ &= -1 + |\, \widetilde{q}_i| \; + m_{R,q} \leq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{25}$$

where $q_{\text{mid},i}$ and $\Delta q_{\text{max},i}$ are the mid position and maximum range of motion, respectively, for joint i, $\widetilde{q}_i$ represents the normalized joint position and $m_{R,q}$ is a *safety margin* for the joint limit constraints to cater for possible errors and inaccuracies.

The above constraints are modified as indicated in Section 4.3.1 as follows:

$$\phi_{R,qi}(\mathbf{q}, \dot{\mathbf{q}}) = \sigma_{R,qi} + K_{R,qi}\dot{\sigma}_{R,qi} = \sigma_{R,qi} + K_{R,qi}\, \nabla\sigma_{R,qi}^{\mathrm{T}}\, \dot{\mathbf{q}} \leq 0, \tag{26}$$

where $K_{R,qi}$ is the approaching parameter to the joint limits.

The gradient vector $\nabla\sigma_{R,qi}$ is straightforward obtained from (25) as:

$$\nabla\sigma_{R,qi} = \begin{bmatrix} 0 & \cdots & \text{sign}(\widetilde{q}_i) & \cdots & 0 \end{bmatrix}^{\mathrm{T}}. \tag{27}$$

4.3.5. Constraints for the maximum joint speeds

The following constraints are considered for the joint speeds:

$$\begin{aligned} \phi_{R,si}(\dot{\mathbf{q}}) &= -1 + \frac{|\ \dot{q}_i|}{\dot{q}_{\max,i}} + m_{R,s} \\ &= -1 + \left|\ \widetilde{\dot{q}}_i\right| + m_{R,s} \leq 0, \quad i = 1, \ldots, n, \end{aligned} \tag{28}$$

where $\dot{q}_{\max,i}$ and $-\dot{q}_{\max,i}$ are the maximum and minimum[4] speed, respectively, for joint i, $\widetilde{\dot{q}}_i$ represents the normalized joint velocity and $m_{R,s}$ is the safety margin for the joint speed constraints.

The constraint modification in Section 4.3.1 is not required for this case since the above constraints depends on the robot speed $\dot{\mathbf{q}}$ and, therefore, the sliding manifold has relative degree one with respect to the discontinuous control action, i.e., $\dot{\phi}_{R,si}$ explicitly depends on signal $\ddot{\mathbf{q}}_c$, as required by SMC theory [34].

The gradient vector $\nabla\phi_{R,si}$ is obtained from (28) as:

$$\nabla\phi_{R,si} = \begin{bmatrix} 0 & \cdots & \text{sign}(\widetilde{\dot{q}}_i) & \cdots & 0 \end{bmatrix}^{\mathrm{T}}. \tag{29}$$

4.3.6. Constraints for forbidden areas in the robot worksapce

Forbidden areas in robot workspace can be defined in order to avoid kinematic singularity, to avoid collisions between the robot manipulator and objects in the environment[5], etc. To fulfill this type of constraints, the Cartesian position $\mathbf{p}_j = [x_j \ \ y_j \ \ z_j]^{\mathrm{T}}$ of every point j of the robot must belongs to the allowed workspace $\Phi_{WS}(\mathbf{p}_j) = \{\mathbf{p}_j \mid \sigma_{R,oi}(\mathbf{p}_j) \leq 0 \ \forall\, i\}$, where $\sigma_{R,oi}$ is the constraint function of object i, e.g., this function could

[4]For simplicity, both speed limits are considered symmetric. If that would not be case, constraints in (28) can be readily split into two constraints for maximum and minimum speeds. Details omitted for brevity.

[5]Note that, in general, objects in the robot environment give rise to both occlusion constraints and forbidden areas in the workspace.

be the negative value of the distance from position $\mathbf{p}_j$ to the boundary surface of the object. Hence, the allowed C-space results in $\Phi_{CS}(\mathbf{q}) = \{\mathbf{q} \,|\, \sigma_{R,oi}(\mathbf{l}_j(\mathbf{q})) = \sigma_{R,oij}(\mathbf{q}) \leq 0 \;\forall\, i,j\}$, where $\mathbf{l}_j$ 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 *robot characteristic points* such that the distance from every point on the boundary surface of the robot links to the closest robot characteristic point is less than a predetermined value which is used to enlarge the constrained region of the workspace.

The above constraints are modified as indicated in Section 4.3.1 as follows:

$$\phi_{R,oij}(\mathbf{q}, \dot{\mathbf{q}}) = \sigma_{R,oij} + K_{R,oij}\dot{\sigma}_{R,oij} = \sigma_{R,oij} + K_{R,oij}\, \nabla\sigma_{R,oij}^{\mathrm{T}}\, \dot{\mathbf{q}} \leq 0, \qquad (30)$$

where $K_{R,oij}$ is the approaching parameter to the boundary surface of the i-th object.

The gradient vector $\nabla\sigma_{R,oij}$ is obtained as follows:

$$\nabla\sigma_{R,oij} = (\partial\mathbf{p}_j/\partial\mathbf{q})\,(\partial\sigma_{R,oi}/\partial\mathbf{p}_j) = {}^0\mathbf{J}_{pj}^{\mathrm{T}}\,(\partial\sigma_{R,oi}/\partial\mathbf{p}_j)\,, \qquad (31)$$

where ${}^0\mathbf{J}_{pj}$ is the Jacobian matrix for the robot point $\mathbf{p}_j$ expressed in the robot base frame, which is obtained from the robot kinematics.

4.3.7. Acceleration equality for Level 1

The partial derivatives of the constraint functions $\{\phi_{V,i}, \phi_{R,qi}, \phi_{R,si}, \phi_{R,oij}\}$ are needed to compute the Lie derivatives $\{\mathbf{L_g}\phi_{V,i}, \mathbf{L_g}\phi_{R,qi}, \mathbf{L_g}\phi_{R,si}, \mathbf{L_g}\phi_{R,oij}\}$ and $\{L_f\phi_{V,i}, L_f\phi_{R,qi}, L_f\phi_{R,si}, L_f\phi_{R,oij}\}$ with (17)–(18). From . (21), (26), (28) and (30), these partial derivatives result in:

$$(\partial\phi_{V,i}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{V,i}^{\mathrm{T}} + K_{V,i}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma V,i} \qquad (32)$$

$$(\partial\phi_{R,qi}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{R,qi}^{\mathrm{T}} + K_{R,qi}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma R,qi} \qquad (33)$$

$$(\partial\phi_{R,si}/\partial\mathbf{q})^{\mathrm{T}} = 0 \qquad (34)$$

$$(\partial\phi_{R,oij}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{R,oij}^{\mathrm{T}} + K_{R,oij}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma R,oij} \qquad (35)$$

$$(\partial\phi_{V,i}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = K_{V,i}\nabla\sigma_{V,i}^{\mathrm{T}} \qquad (36)$$

$$(\partial\phi_{R,qi}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = K_{R,qi}\nabla\sigma_{R,qi}^{\mathrm{T}} \qquad (37)$$

$$(\partial\phi_{R,si}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = \nabla\phi_{R,si}^{\mathrm{T}} \qquad (38)$$

$$(\partial\phi_{R,oij}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = K_{R,oij}\nabla\sigma_{R,oij}^{\mathrm{T}} \qquad (39)$$

where $\mathbf{H}_{\sigma V,i}$, $\mathbf{H}_{\sigma R,qi}$ and $\mathbf{H}_{\sigma R,oij}$ denote the Hessian matrix of second-order partial derivatives of $\sigma_{V,i}$, $\sigma_{R,qi}$ and $\sigma_{R,oij}$, respectively, and all the elements except the ith of row vectors in (37) and (38) are zero.

Thus, according to (17) and (36)–(39), equation (11) for the first priority task is given by:

$$\begin{bmatrix} \mathbf{K}_V \, \nabla\boldsymbol{\sigma}_V^{\mathrm{T}} \\ \mathbf{K}_{R,q} \, \nabla\boldsymbol{\sigma}_{R,q}^{\mathrm{T}} \\ \nabla\boldsymbol{\phi}_{R,s}^{\mathrm{T}} \\ \mathbf{K}_{R,o} \, \nabla\boldsymbol{\sigma}_{R,o}^{\mathrm{T}} \end{bmatrix} = - \begin{bmatrix} \mathbf{1}_{b,V} u_V^+, \\ \mathbf{1}_{b,q} u_{R,q}^+ \\ \mathbf{1}_{b,s} u_{R,s}^+ \\ \mathbf{1}_{b,o} u_{R,o}^+ \end{bmatrix} =$$

$$\mathbf{L_g}\boldsymbol{\phi}_1 \ddot{\mathbf{q}}_c = - \, \mathbf{u}_1^+, \tag{40}$$

where $\mathbf{K}_V$, $\mathbf{K}_{R,q}$ and $\mathbf{K}_{R,o}$ are diagonal matrices with diagonal entries $K_{V,i}$, $K_{R,qi}$ and $K_{R,oij}$, respectively; matrices $\{\nabla\boldsymbol{\sigma}_V, \nabla\boldsymbol{\sigma}_{R,q}, \nabla\boldsymbol{\phi}_{R,s}, \nabla\boldsymbol{\sigma}_{R,o}\}$ contain the vectors $\{\nabla\sigma_{V,i}, \nabla\sigma_{R,qi}, \nabla\phi_{R,si}, \nabla\sigma_{R,oij}\}$, see Eqs. (36)–(39), of all *active* constraints; $\{u_V^+, u_{R,q}^+, u_{R,s}^+, u_{R,o}^+\}$ are the chosen value of u^+ for each type of constraint; and $\{\mathbf{1}_{b,V}, \mathbf{1}_{b,q}, \mathbf{1}_{b,s}, \mathbf{1}_{b,o}\}$ are column vectors with all its components equal to one and their size is equal to the number of active constraints of each type.

Therefore, by comparing the acceleration equalitiy (40) for the first task with (13), it is obtained that $\mathbf{A}_1 = \mathbf{L_g}\boldsymbol{\phi}_1$ and $\mathbf{b}_1 = -\mathbf{u}_1^+$.

It is important to remark that, according to (40), only the gradient vectors of the active constraints (i.e., those with $\phi_i \geq 0$) are required to compute the control action of the first task.

4.4. Level 2: reference tracking

For the reference tracking, this work considers the classical operational space robot control [53], that taking into account (4) and (6), results in:

$$\mathbf{J}_s \ddot{\mathbf{q}}_c = \ddot{\mathbf{s}}_c - (\mathbf{J}_s \mathbf{d}_c + \dot{\mathbf{J}}_s \dot{\mathbf{q}} + \partial\dot{\mathbf{s}}/\partial t), \tag{41}$$

where $\ddot{\mathbf{s}}_c$ is the commanded acceleration for the visual feature vector.

Moreover, considering the classical acceleration-based kinematic controller used for trajectory tracking [54], i.e., a correction based on the position and velocity errors plus a feedforward of the second-order derivative of the reference, the commanded acceleration $\ddot{\mathbf{s}}_c$ results in:

$$\ddot{\mathbf{s}}_c = \ddot{\mathbf{s}}_{ref} - K_{T,p}\mathbf{e} - K_{T,v}\dot{\mathbf{e}}, \tag{42}$$

where $\mathbf{e}$ is the error of the visual feature vector, i.e., $\mathbf{e} = \mathbf{s} - \mathbf{s}_{ref}$, and $K_{T,p}$ and $K_{T,v}$ are the correction gains for the position and velocity errors, respectively. Note that the dynamics (i.e., the poles) of this kinematic controller is given by the roots of the polynomial with coefficients $[1\ K_{T,v}\ K_{T,p}]$. For instance, if $K_{T,v} = 2\sqrt{K_{T,p}}$ a critically damped response is obtained.

In the simulations and experiments of this work the correction gains are defined as follows. On the one hand, it is used $K_{T,v} = 3\sqrt{K_{T,p}}$ in order to obtain an overdamped response. On the other hand, the position correction gain is given by an *exponential* waveform depending on the magnitude of the position error, i.e., $K_{T,p}(\|\mathbf{e}\|_2)$ with $K_{T,p}(\infty) = 0$ and two parameters: gain for zero error $K_{T,p}(0)$ and derivative of the gain for zero error $\dot{K}_{T,p}(0)$. This approach allows to use a smaller gain at the beginning when the initial error is large to obtain a smooth behavior and a larger gain at the end when the final error is small to achieve promptly the reference value.

By comparing the acceleration equalitiy (41) for the second task with equation (14), it is obtained that $\mathbf{A}_2 = \mathbf{J}_s$ and $\mathbf{b}_2 = \ddot{\mathbf{s}}_c - (\mathbf{J}_s\mathbf{d}_c + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{s}}/\partial t)$.

Note that, the acceleration-based robot control given by (41)–(42) has already been used in VS appliactions by [55] for PBVS and by [56] for IBVS.

5. Additional remarks

5.1. Discontinuous control action

In this work, the joint accelerations are considered as the SM discontinuous control action, which yields two advantages: firstly, the joint velocities are continuous, i.e., the control is smoother; and secondly, it allows using the constraint modification in Section 4.3.1 in order to reach smoothly the boundary of the original constraints.

In practice, if the order of the actual control action for the system at hand does not match the order of the discontinuous control action, a filter[6] (or a pure integrator) with the right order can be used between both signals. For instance, if the actual control action are the joint positions, which is the case for experiments in Section 10, a double integrator can be applied to the discontinuous control in order to compute the actual control action, which is continuous.

[6]If a filter is considered, it has to be properly designed since it limits the bandwidth of the controlled system.

5.2. Task abort

As indicated in Section 3, the stability of the SMC used in the high-priority task to fulfill the constraints is guaranteed if matrix $\mathbf{L_g}\boldsymbol{\phi}$ is full row rank and the switching gain u^+ fullfills (12). However, if the former is not satisfied at a certain time, e.g., the number of active constraints is larger than the number of joints, the robot operation should be aborted since the fulfillment of the constraints cannot be guaranteed.

5.3. Chattering

Discrete-time implementations of any practical SMC makes the system leave the ideal SM and oscillate with finite frequency and amplitude inside a band around $\boldsymbol{\phi} = \mathbf{0}$, which is called *chattering* [34]. The upper bound for the chattering band $\triangle\boldsymbol{\phi}$ of the proposal can be obtained using the Euler-integration of the discontinuous control action given by (11), that is:

$$\triangle\boldsymbol{\phi} = T_s \, |\mathbf{L_g}\boldsymbol{\phi} \, \mathbf{u}_c| = T_s \, u^+ \, \mathbf{1}_b, \tag{43}$$

where T_s is the sampling time of the robotic system and the value of u^+ is $\{u_V^+, u_{R,q}^+, u_{R,s}^+, u_{R,o}^+\}$ for the visibility and robot constraints.

5.4. Non-static constraints

The proposed approach can also be used if there are non-static constraints, e.g., moving obstacles for the occlusion avoidance constraints or a *moving target object*. In that case ϕ_i also depends explicitly on time and, therefore, the derivative of ϕ_i in equation (9) must be replaced by $\dot{\phi}_i = \widetilde{L_f\phi}_i + \mathbf{L_g}\phi_i \, \mathbf{u}$, where $\widetilde{L_f\phi}_i$ is equal to $L_f\phi_i + \partial\phi_i/\partial t$, and $\mathbf{L_g}\phi_i$ and $L_f\phi_i$ are given again by (17) and (18), respectively. Thus, all developments keep unchanged except for changing $L_f\phi_i$ to $\widetilde{L_f\phi}_i$. Hence, only the value of the lower bound for the switching gain u^+ is changed when non-static constraints are considered and, therefore, the iterative computation of the algorithm remains the same.

5.5. Differentiability of the constraint functions

As indicated in Section 3, the constraint functions ϕ_i must be differentiable. If this assumption is not satisfied at a certain point of the constraint boundary, the SM behavior of the proposed method is temporarily lost and the constraints may be unfulfilled. In particular, if the non-allowed region defined by the constraint boundary at the non-differentiable point is *concave*

the constraint is unfulfilled and if it is *convex* it is fulfilled, see Section 8.2 for an illustrative example.

6. Advantages and disadvantages of the proposal

Advantages of the proposed SMC to fulfill visibility and robot constraints:

- In contrast to other similar techniques, like the *artificial potential fields* used in [21], the proposed method *fully utilizes* the available allowed space, e.g., the rectangle representing the image plane limits for the FOV constraint, see Section 10. Moreover, the boundary of the constraints is *reached smoothly* depending on a free design parameter. Thus, the velocity perpendicular to the constraint boundary is progressively reduced to zero.

- The method uses *partial information* of the system model, i.e., the Lie derivatives $L_f\phi_i$ (18) are not needed, only the Lie derivatives $\mathbf{L_g}\phi_i$ (17) are required. Therefore, only first-order derivatives (gradient vectors, Jacobian matrices, etc.) are needed, see (36)–(40), and no second-order derivatives (Hessian matrices, derivative of Jacobians, etc.) are required, see (32)–(35).

- The method is *robust* against environment modeling errors, i.e., it is not affected by the inaccuracies and uncertainties in the second-order derivatives mentioned above. Furthermore, the first-order derivatives (gradient vectors, etc.) required for the SMC law (40) could be relatively inaccurate and the performance of the algorithm would still be satisfactory (see Section 10.2) as long as the SM control action is able to switch the value of the active constraint functions from positive to negative[7].

- The algorithm only requires a few program lines and has *reduced computation time* since only linear algebra is used, see Appendix B.

Main limitations of the method:

[7]The gradient vector of a constraint function represents the direction of *maximum* increase of the function. However, other similar directions may also be useful to properly modify the value of the constraint function.

- The SMC algorithm uses linear extrapolation (i.e., local first-order derivatives) to predict the value of the constraint functions at the next time step. Hence, the algorithm may be blocked in *trap situations* [57]. However, similarly to other reactive planning techniques such as potential fields, the control remains stable at the *local minima* reached by the robot system, see Section 8.2 for an illustrative example. Some of these trap situations could be avoided using high-level planning with the complete data of the problem. However, the complexity and computational cost for this planner are significantly greater than those of the approach proposed in this work, see Appendix B.

- Like other SMC applications, the proposed method has the *chattering* drawback, see Section 5.3. Nevertheless, the chattering problem becomes negligible for reasonable fast sampling rates, see (43).

7. Conditions for the simulations and experiments

Several simulations and experiments are presented below considering both PBVS and IBVS, where the eye-in-hand configuration is used, i.e., camera rigidly attached to the robot end-effector.

For the case of PBVS simulations and experiments, the typical visual feature vector is considered:

$$\mathbf{s} = \begin{bmatrix} {}^{C^*}\mathbf{t}_C^{\mathrm{T}} & {}^{C^*}\theta\mathbf{u}_C^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{44}$$

where the first element represents a translation vector and the second element gives the angle/axis parameterization for the rotation, both between the desired camera pose and current camera pose, see [1] for further details.

For the case of IBVS experiments, the visual feature vector is given by the eight image plane coordinates of the four markers of the target object, as usual.

The simulation results presented below were obtained using MATLAB®. Details of pseudo-code and computing load for the proposed method is detailed in Appendix B.

8. Simulation: first example

As first example, a simple two-dimensional (2D) robot system is considered for better illustration of the main features of the algorithm. The robot

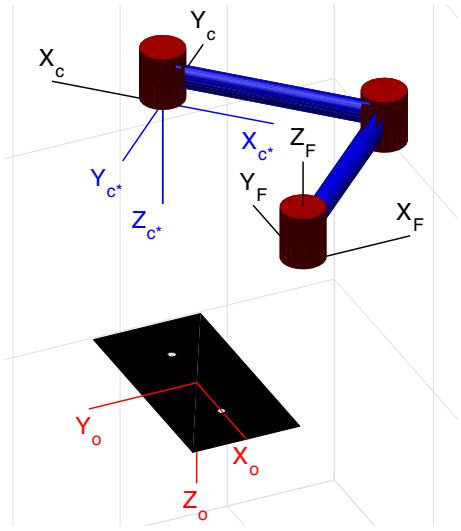

Fig. 3. System used for 2D simulation: 3R planar robot, target object with two features and coordinate frames.

considered for this case consists of a planar mechanisms composed by four links (the first of them is fixed) connected serially by three revolute joints, i.e., a 3R planar robot. This robot is used to perform a positioning task with respect to a motionless target object while fulfilling FOV and occlusion constraints. Fig. 3 depicts the VS application in consideration with the following elements: 3R robot, target object, as well as the involved frames: robot base frame F, object frame O, initial camera frame C and desired camera frame C^*. The Jacobian matrix ${}^e\mathbf{J}_e$ for this 3R robot can be readily obtained [53] taking into account the Denavit-Hartenberg (DH) parameters shown in Table 1. In order to better illustrate the behavior of the proposed SMC algorithm, only the visibility constraints described in Section 4.3.2 and Section 4.3.3 are considered for this first example.

Link i	θ_i (rad)	d_i (m)	a_i (m)	α_i (rad)
1	q_1	0	1	0
2	q_2	0	1	0
3	q_3	0	0	$-\pi$

Table 1. DH parameters for the 3R planar robot.

For analysis purposes, two occlusion constraints are defined as follows:

$$\sigma_{V,b}(u_i, v_i) = 1 - 45^{-2}\left(0.7^2(u_i + 50)^2 + ((v_i - 20) - (u_i + 50)^2)^2\right) \leq 0 \quad (45)$$

$$\sigma_{V,h}(u_i, v_i) = 1 - 35^{-2}\left(0.8^2(u_i + 102)^2 + (v_i - |u_i + 102|)^2\right) \leq 0, \quad (46)$$

where the shape of the first and second occlusion constraints are a "bowl" and a "heart", respectively, and their partial derivatives with respect to u_i and v_i, which are required for the gradient vector computation, are straightforward obtained from the above expressions.

8.1. Simulation conditions and parameter values for the first example

i) The camera is attached to the robot end-effector, i.e., the camera pose is equivalent to the end-effector pose.

ii) Parameters used for the FOV and occlusion constraints: $m_v = 5\%$, $n_V = 16$, $K_{V,i} = K_{V,b} = K_{V,h} = 0.2$ and $u_V^+ = 10$.

iii) Parameters used for the kinematic controller: $K_{T,p}(0) = 20$ and $\dot{K}_{T,p}(0) = 40$.

iv) A static target object is considered with two markers[8], see Fig. 3.

v) The initial error for the camera position and orientation is zero and π radians in the Z-axis, respectively, see Fig. 3.

vi) The algorithm was computed with a sampling time T_s of 5 milliseconds.

8.2. Simulation results for the first example

Four different simulations have been carried out for the first example in order to highlight the main features of the proposed method.

The first simulation only considers the FOV constraints. Fig. 4 shows the result for this case, where it can be seen that the second feature remains in the FOV, where the allowed space is fully utilized and the boundary of the hyperellipse constraint is reached smoothly. Note also that: the initial

[8]For clarity in the figures, only two markers are considered for the 2D simulation. Note that, regarding data redundancy in the image plane, using two markers in 2D robot systems (three variables) is analogous to using four markers in 3D robot systems (six variables), as usually considered.

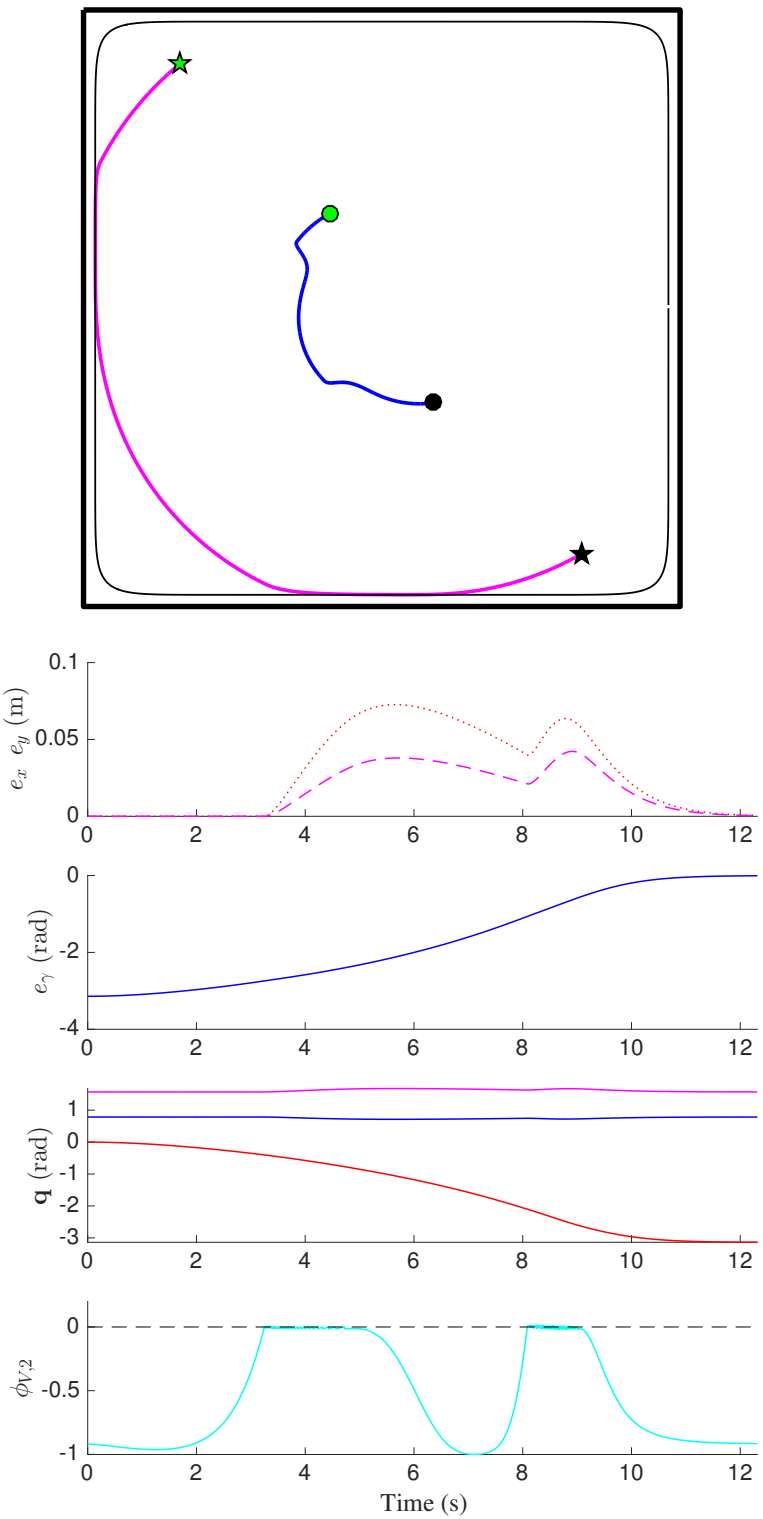

Fig. 4. First example considering only FOV constraints. Top: FOV constraint (thin line) and feature trajectories (thick lines) in the image plane, where circle and star symbols are used to represent the starting (green-light) and final (dark-black) positions for the first and second feature, respectively. Bottom: position and orientation errors; joint positions; and FOV constraint function for the second feature.

error for the camera orientation is made zero; a position error arises when the FOV constraint is activated, which subsequently is made zero; the FOV constraint becomes active at the time intervals 3-5 s and 8-9 s for the second image feature; and the FOV constraint is fulfilled, i.e., $\phi_{V,2} \leq 0$.

For the second simulation, the position correction gain for zero error has been increased to $K_{T,p}(0) = 50$, which yields a faster robot motion, i.e., the positioning task is perform around three times faster. In this case, two values have been considered for the switching gain: the one used in the first simulation and another one increased to $u_V^+ = 50$. The results for both simulations are depicted in Fig. 5, where it can bee seen that the smaller value of the switching gain is not enough to fulfill the FOV constraint. In contrast, when the large value is used the FOV constraint is fulfilled. Therefore, it can be concluded that, in general, the switching gain of the proposed SMC has to be increased when faster robot motions are considered since larger corrections are needed, i.e., the lower bound in (12) for the switching gain is larger. However, the resulting chattering amplitude is also increased, which can be appreciated by comparing the constraint function $\phi_{V,2}$ in Fig. 5 to that in Fig. 4. Therefore, the sampling time has to be reduced in accordance to keep the same chattering amplitude, see (43).

For the third simulation, it is considered the FOV constraint together with the occlusion bowl-shaped constraint function in (45). Fig. 6 shows the results for this case, where it can be seen that the first feature is blocked in a trap situation with the highly attractive bowl-shaped constraint. Therefore, the robot reaches a local minima where the orientation error is zero but the position error is not zero. Note that, although reference value has not been achieved, the SMC is stable at the mentioned local minima.

The fourth simulation considers the FOV constraint together with the occlusion heart-shaped constraint function in (46). For this simulation, two cases are analyzed: the heart-shaped constraint function with a clockwise rotation of 90 degrees and with no rotation. On the one hand, Fig. 7 shows the results for the first case, where it can be seen that the convex non-differentiable point of the heart-shaped constraint is overcome with no problem. On the other hand, Fig. 8 shows the results for the second case, where it can be seen that the constraint is unfulfilled (i.e., $\phi_{V,1occ} > 0$) at the concave non-differentiable point.

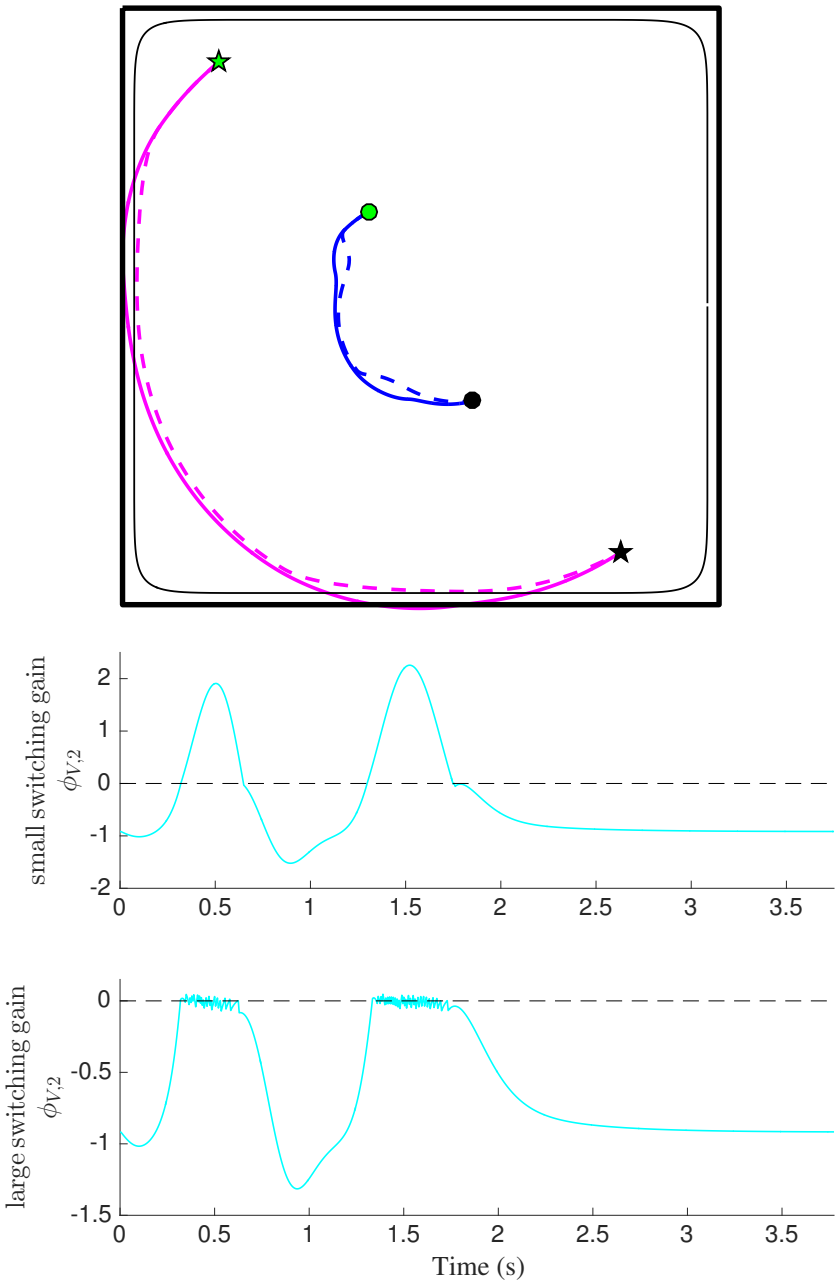

Fig. 5. First example using fast motion. Top: feature trajectories in the image plane for small (thick-solid line) and large (thick-dashed line) switching gains. Bottom: FOV constraint function for the second feature using small and large switching gains.

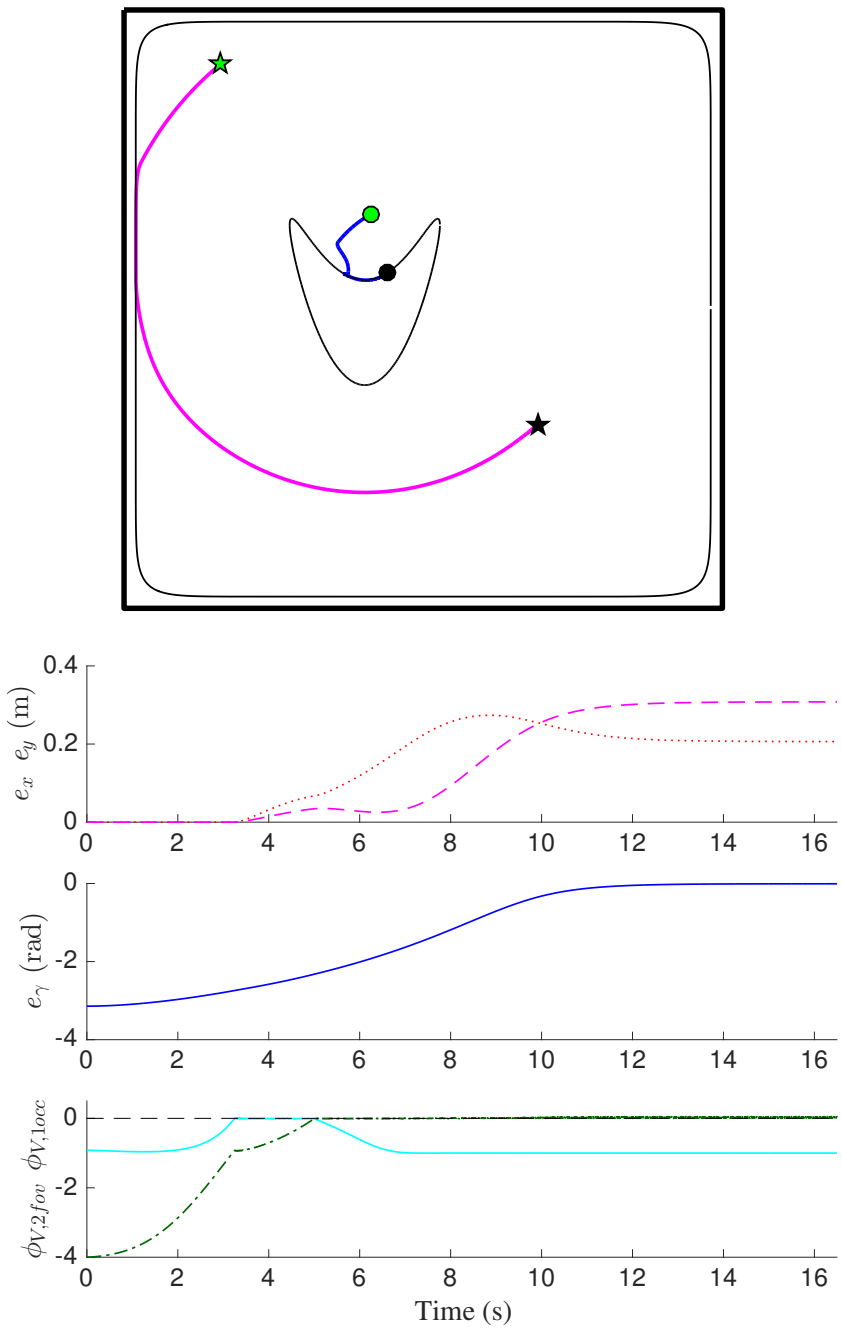

Fig. 6. First example in a trap situation. Top: feature trajectories (thick lines) and occlusion bowl-shaped constraint (thin line) in the image plane. Bottom: position and orientation errors; and FOV (cyan-solid line) and occlusion (green-dash-dotted line) constraint functions.

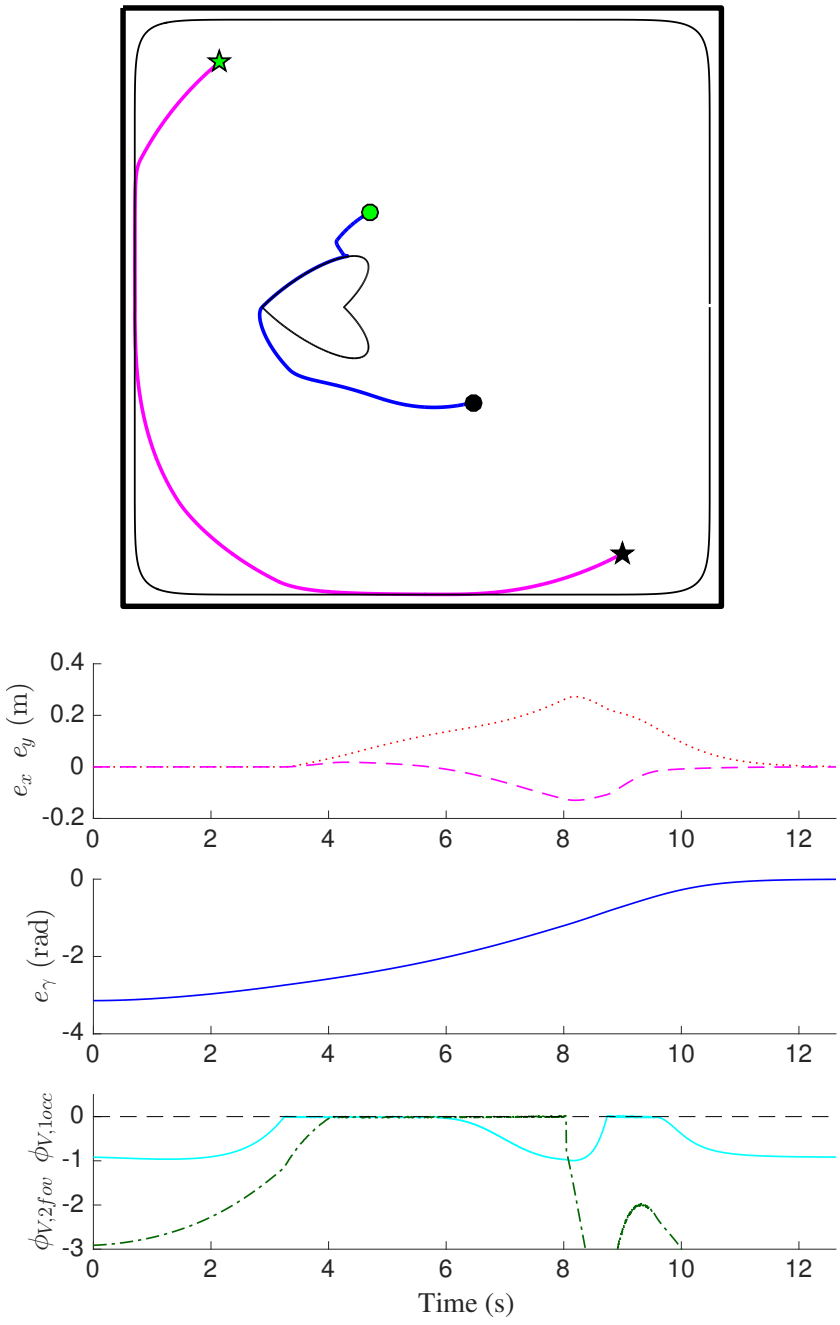

Fig. 7. First example for a convex non-differentiable point. Top: feature trajectories (thick lines) and occlusion heart-shaped constraint (thin line) in the image plane. Bottom: position and orientation errors; and FOV (cyan-solid line) and occlusion (green-dash-dotted line) constraint functions.

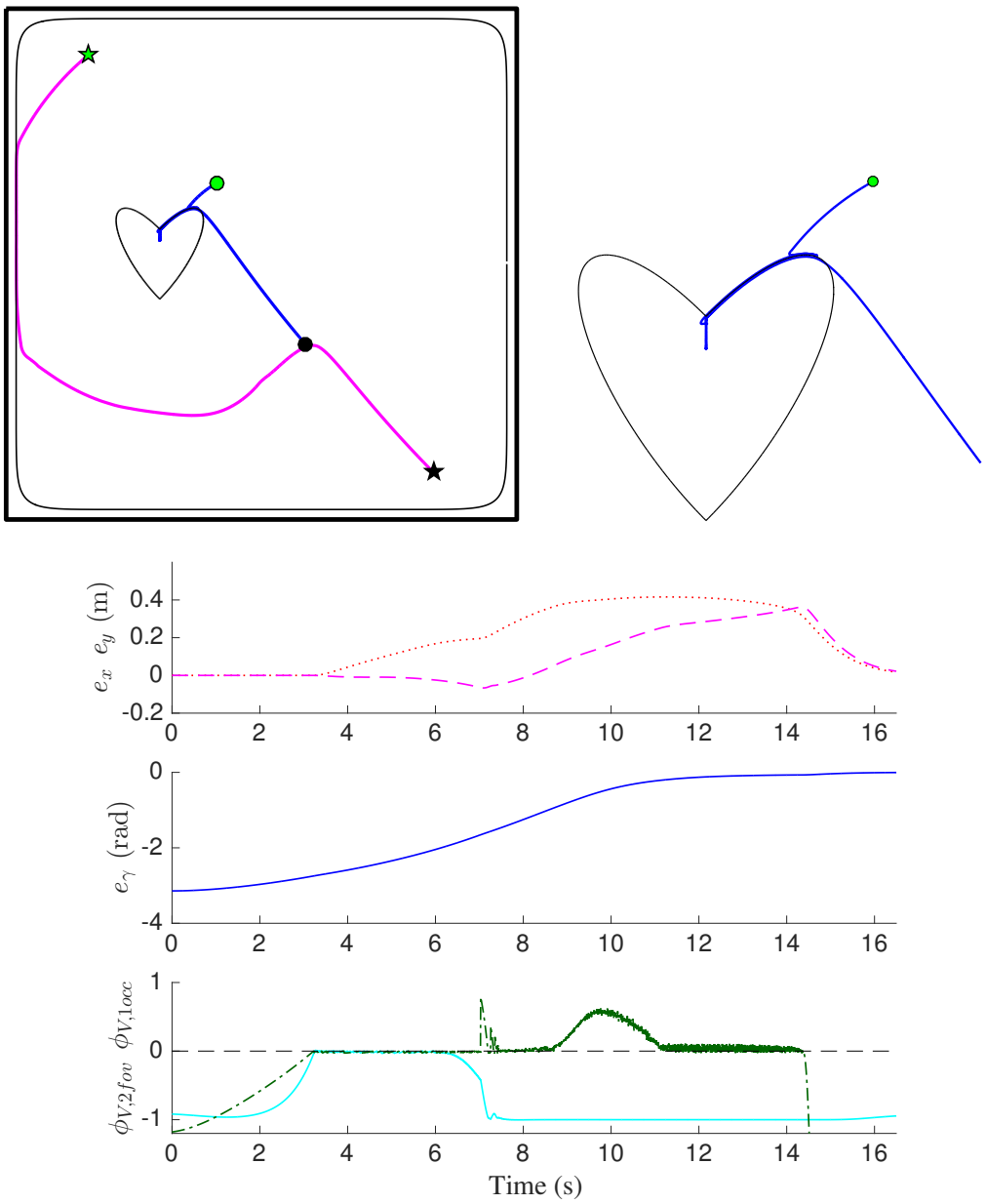

Fig. 8. First example for a concave non-differentiable point. Top-left: feature trajectories (thick lines) and occlusion heart-shaped constraint (thin line) in the image plane. Top-right: detail view of the heart-shaped constraint unfulfillment. Bottom: position and orientation errors; and FOV (cyan-solid line) and occlusion (green-dash-dotted line) constraint functions.

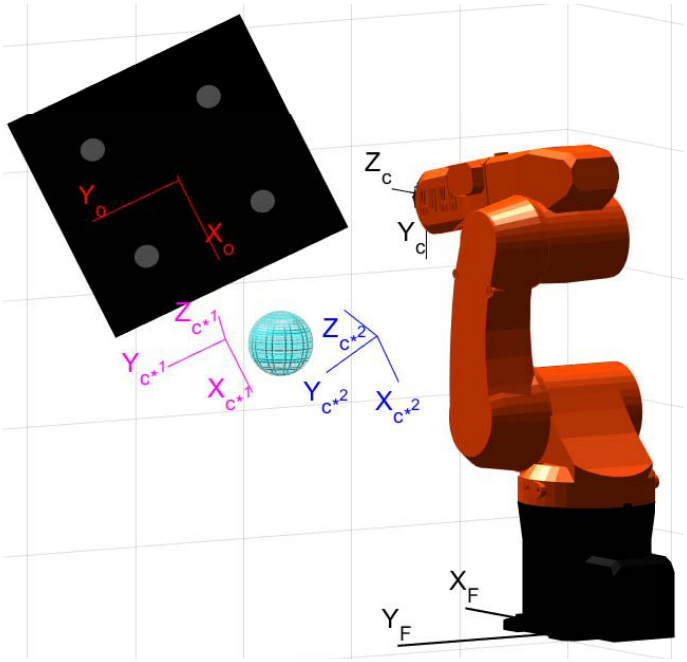

Fig. 9. System used for 3D simulation: 6R robot, target object with four markers, sphere representing a forbidden area and coordinate frames.

9. Simulation: case study

A three-dimensional case study is presented in this section to demonstrate the general effectiveness and applicability of the method.

In the proposed case study, a classical 6R serial manipulator with spherical wrist is considered to perform two consecutive positioning tasks with respect to a motionless target object while fulfilling the visibility and robot constraints described in Section 4.3. Fig. 9 depicts the VS application in consideration with the following elements: 6R robot, target object, sphere representing a forbidden area, as well as the involved frames: robot base frame F, object frame O, initial camera frame C, desired camera frame C^{*1} for the first phase and desired camera frame C^{*2} for the second phase. The Jacobian matrix ${}^{e}\mathbf{J}_{e}$ for this 6R robot can be readily obtained [53] taking into account the DH parameters shown in Table 2.

The constraint function $\sigma_{R,o1}$ for the sphere representing a forbidden area

Link i	θ_i (rad)	d_i (m)	a_i (m)	α_i (rad)
1	q_1	0.335	0.075	$-\pi/2$
2	q_2	0	0.27	0
3	q_3	0	0.09	$\pi/2$
4	q_4	-0.295	0	$-\pi/2$
5	q_5	0	0	$\pi/2$
6	$q_6 - \pi/2$	-0.08	0	π

Table 2. DH parameters for the 6R robot.

is given by:

$$\sigma_{R,o1}(\mathbf{p}_j) = r_s - \|\mathbf{p}_j - \mathbf{p}_s\|_2 + m_{R,o}, \tag{47}$$

where r_s and $\mathbf{p}_s$ are the radius and center, respectively, of the sphere, $m_{R,o}$ is the safety margin for the constraint and $\mathbf{p}_j$ is the Cartesian position of the considered point of the robot. This forbidden are could represent, for instance, a region where the robot is close to kinematic singularity. For simplicity, it is assumed that no visibility problems are caused by this forbidden area.

Therefore, the partial derivative of $\sigma_{R,o1}$ with respect to $\mathbf{p}_j$, which is needed for computing the gradient vector in (31), results in:

$$\frac{\partial \sigma_{R,oi}}{\partial \mathbf{p}_j} = -\frac{\mathbf{p}_j - \mathbf{p}_s}{\|\mathbf{p}_j - \mathbf{p}_s\|_2}. \tag{48}$$

9.1. Simulation conditions and parameter values for the case study

i) The camera is attached to the robot end-effector, i.e., the camera pose is equivalent to the end-effector pose.

ii) Parameters used for the FOV constraint: $m_v = 5\%$, $n_V = 16$, $K_{V,i} = 0.1$ and $u_V^+ = 2$.

iii) Parameters used for the joint limit constraints: $m_{R,q} = 0$, $K_{R,qi} = 0.1$, $u_{R,q}^+ = 2$, $q_{\text{mid},6} = -0.5$ and $\Delta q_{\max,6} = 1.2$ rad. The range constraints for the remaining joints are omitted for simplicity.

iv) Parameters used for the joint speed constraints: $m_{R,s} = 0$, $u_{R,s}^+ = 1$

and $\dot{q}_{\max,2} = 0.4$ rad/s. The speed constraints for the remaining joints are omitted for simplicity.

v) Parameters used for the constraint for forbidden areas in the robot workspace: $m_{R,o} = 0$, $K_{R,o1} = 0.1$, $u^{+}_{R,o} = 0.2$, $r_s = 0.05$ m and $\mathbf{p}_s = \begin{bmatrix} 0.57 & 0.2 & 0.405 \end{bmatrix}^{\mathrm{T}}$ m. For simplicity, in the simulation only the Cartesian position $\mathbf{p}_e$ of the robot end-effector will be evaluated as point $\mathbf{p}_j$ in the constraint for forbidden areas.

vi) Parameters used for the kinematic controller: $K_{T,p}(0) = 20$ and $\dot{K}_{T,p}(0) = 60$.

vii) A static target object is considered with four markers, representing the vertices of a square with a side length of 0.2 m, see Fig. 9.

viii) The algorithm was computed with a sampling time T_s of 2 milliseconds.

9.2. Simulation results for the case study

The results of the simulation are depicted at different figures. Fig. 10 shows that the position and orientation error is made zero for both phases, where the desired camera frame is changed from the first to the second phase around time instant 4 s. Fig. 10 also shows the trajectories followed by the image features in the image plane. Note that, the second and third features remain in the FOV fully utilizing the allowed space and reaching smoothly the boundary of the hyperellipse constraint.

Fig. 11 shows that: all the constraints are fulfilled, i.e., $\max(\phi_i) \leq 0$; the joint limit constraint for the sixth joint (dark-dashed line in the first plot) and the speed constraint for the second joint (dark-dashed line in the second plot) become active during the first phase; the constraint for forbidden areas becomes active for the time interval 6-8 s during the second phase; and the FOV constraint becomes active for the second and third feature during the first phase and for the second feature during the second phase. It is interesting to remark that in some phases of the simulation there are up to three active constraints at a time.

Finally, Fig. 12 depicts four snapshot frames of a 3D representation of the robot at different time instants, including a detail view of the spherical forbidden area, where it can be seen that this constraint is fulfilled.

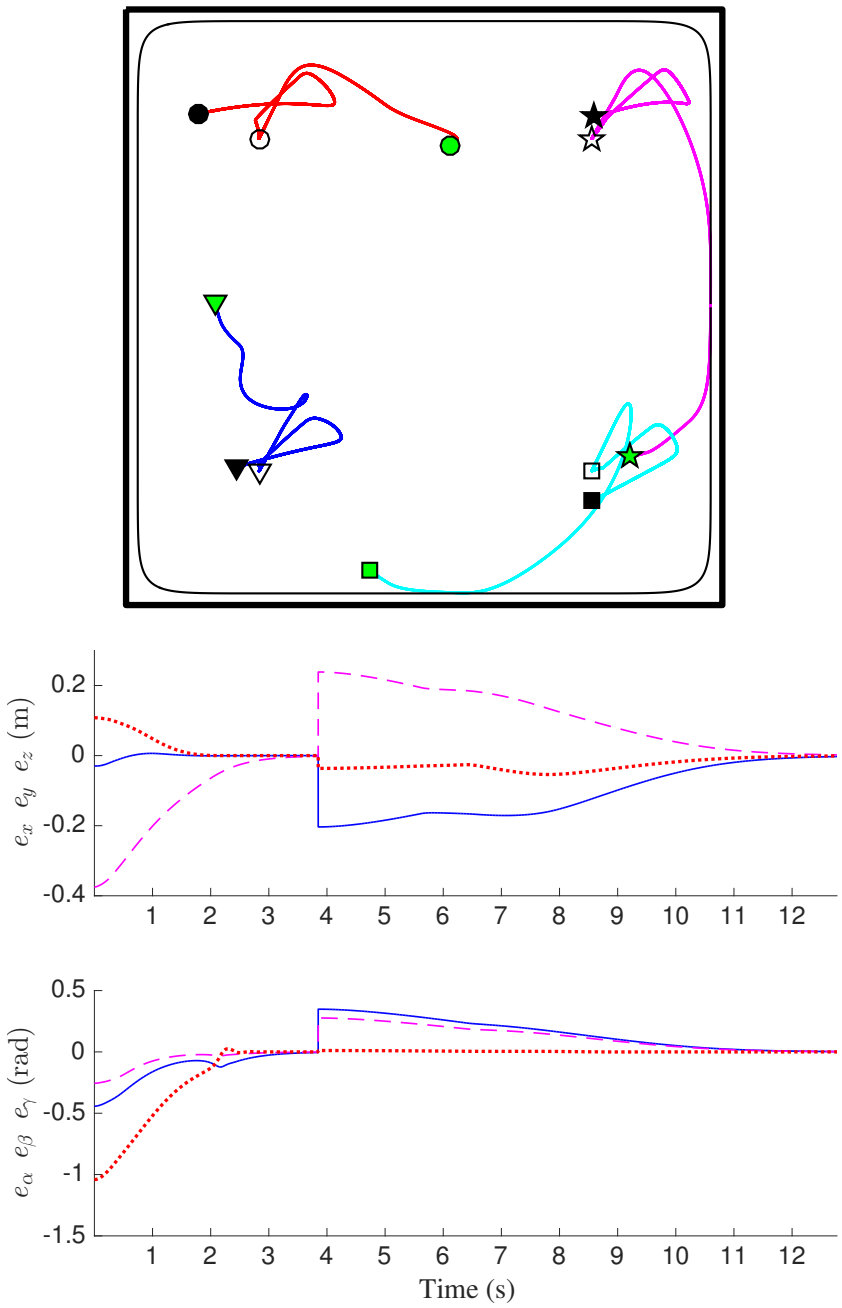

Fig. 10. Simulation for the case study. Top: image features trajectories, where a symbol (circle, triangle, square, star) is used to represent the starting (green-light) and final positions (white, dark-black) in both phases for each feature. Bottom: Position and orientation error: e_x and e_α (solid-blue), e_y and e_β (dashed-magenta) and e_z and e_γ (dotted-red).

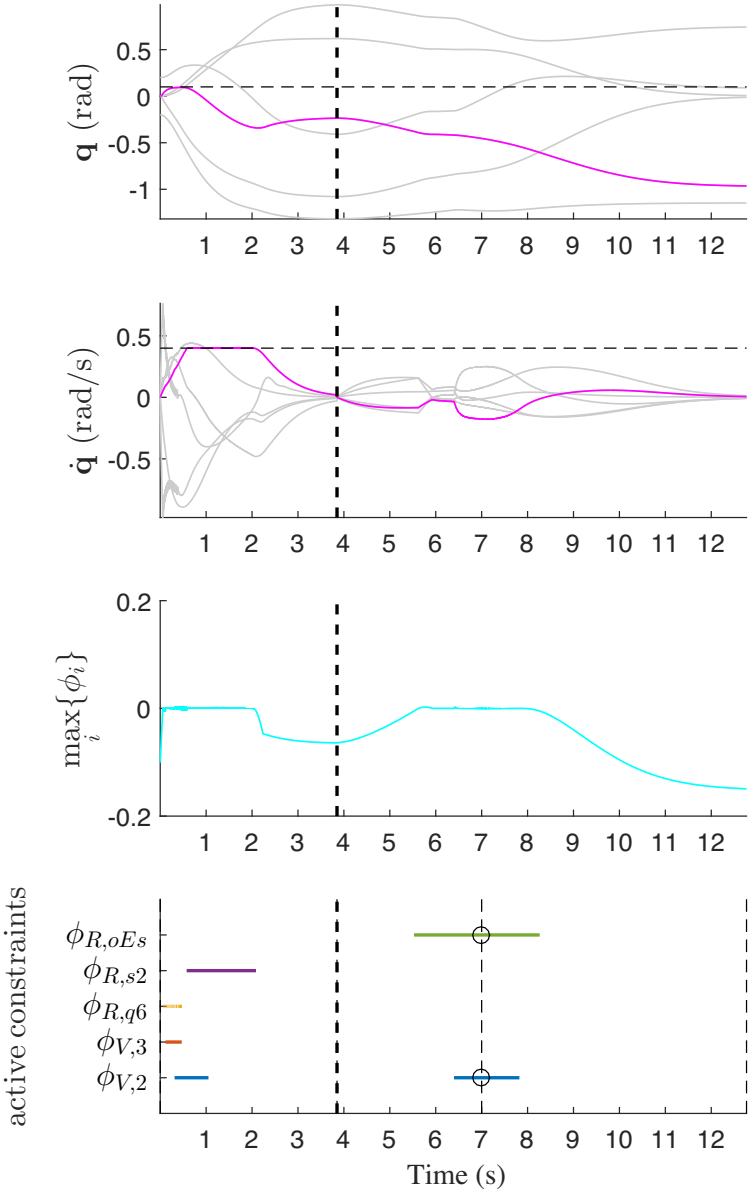

Fig. 11. Simulation for the case study. From top to bottom plots: (1) joint postions $\mathbf{q}$ (the horizontal dark-dashed line represents the joint limit for the active constraint; (2) joint speeds $\dot{\mathbf{q}}$ (the horizontal dark-dashed line represents the speed limit for the active constraint; (3) maximum value of the constraint functions ϕ_i; (4) horizontal lines indicating when a constraint is active (the dashed vertical lines correspond to the time instants of the frames in Fig. 12 and the circles indicate the active constraints at those instants). The thick dashed vertical line represents the time instant when the desired camera frame is changed from the first to the second phase.

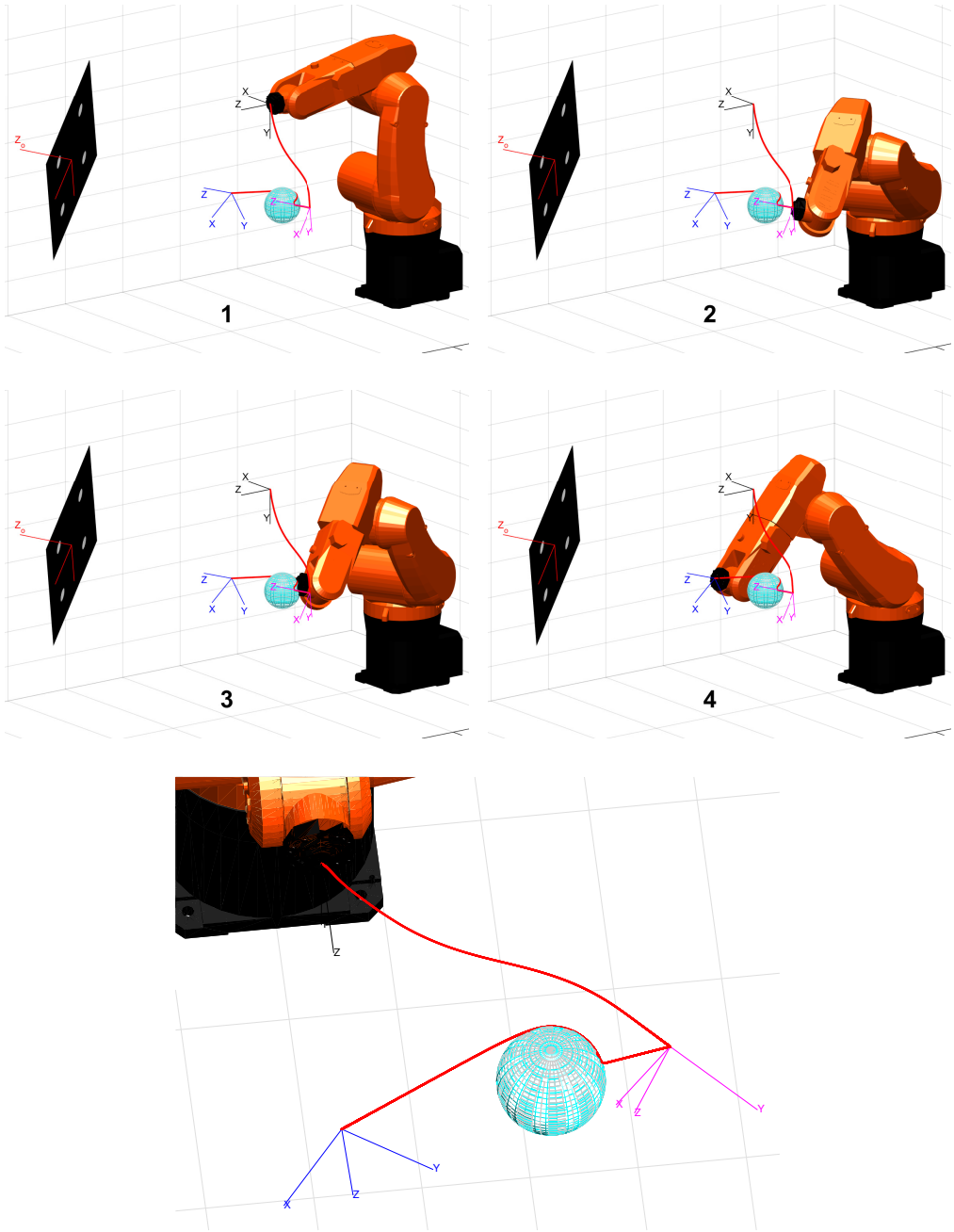

Fig. 12. Simulation for the case study. Top: sequence of frames (frames 1 to 4) showing the robot configuration during the simulation and the path follwed by the robot end-effector. The active constraints at each frame are shown in Fig. 11. Bottom: detailed view of the fulfillment of the constraint for the spherical forbidden area.

10. Experiments: visibility constraints in PBVS and IBVS

The proposed SMC has been implemented to obtain real experiments in order to demonstrate its feasibility and robustness. The following setup has been used (see Fig. 13): a Kuka KR6 R900 sixx robot manipulator, coined as Agilus, in ceiling-mounted position, is equipped with the Kuka.RobotSensorInterface (RSI) technology that allows external real-time communication using the Ethernet UDP protocol; a general purpose web cam rigidly attached to the robot end-effector, which is used for image acquisition; a screen, which is used to display the target object markers; and an external PC with Ubuntu 12.04 OS prompted with real time kernel that implements the computer vision and control algorithms proposed in this work. The position of the image features is updated using the dot tracker in ViSP (Visual Servoing Platform) [58], whilst the object pose is estimated to update the visual feature vector $\mathbf{s}$ and to compute $\mathbf{L}_s$. The Jacobian matrix ${}^e\mathbf{J}_e$ for this robot can be readily obtained taking into account the DH parameters shown in Table 3.

Fig. 13. Experimental setup: 6R serial industrial manipulator in ceiling position with the camera rigidly attached to the robot end-effector (eye-in-hand) and a screen to display the object markers.

10.1. Experimental conditions and parameter values

i) The commanded joint accelerations $\ddot{\mathbf{q}}_c$ computed by the proposed algorithm are double integrated to obtain the commanded joint positions $\mathbf{q}_c$ sent to the robot controller.

Link i	θ_i (rad)	d_i (m)	a_i (m)	α_i (rad)
1	q_1	-0.400	0.025	$\pi/2$
2	q_2	0	-0.455	0
3	q_3	0	-0.035	$-\pi/2$
4	q_4	-0.420	0	$\pi/2$
5	q_5	0	0	$-\pi/2$
6	q_6	-0.080	0	π

Table 3. DH parameters for the 6R robot used in the experiment.

ii) The perspective camera model without distortion is considered with the following parameters: $[F_u, F_v] = [710.1, 709.8]$ pixels and $[W_V, H_V] = [640, 480]$ pixels. The camera to end-effector transformation matrix is ${}^c\mathbf{M}_e = \begin{bmatrix} 0 & 0.07 & -0.05 & 0 & 0 & -\pi/2 \end{bmatrix}^{\mathrm{T}}$, where the first three elements are the Cartesian coordinates, in meters, and the last three are the roll, pitch and yaw angles in radians.

iii) Four markers define the object, representing the vertices of a square with a side length of 17 centimeters.

iv) Parameters used for the visibility constraints due to the limited camera FOV in PBVS: $m_V = 30$ pixels, $n_V = 16$, $K_{V,i} = 3$ and $u_V^+ = 1$.

v) Parameters used for the visibility constraints due to object occlusion in IBVS: $m_V = 0$ pixels, $K_{V,i} = 3$, $u_V^+ = 1$ and the forbidden area in the image plane is given by an hyperellipse with the parameters $[x, a, y, b, n_V] = [460, 100, 375, 100, 4]$.

vi) Parameters used for the kinematic controller: $K_{T,p}(0) = 0.6$ and $\dot{K}_{T,p}(0) = 10$.

vii) In order to verify the robustness of the proposed approach, a gradient vector with error $\nabla\boldsymbol{\sigma}_{Ve}$ is also considered introducing a signed variation in percentage of the computed value $\nabla\boldsymbol{\sigma}_V$ as follows:

$$\nabla\boldsymbol{\sigma}_{Ve} = \nabla\boldsymbol{\sigma}_V + c_e \begin{bmatrix} -1 & 1 & 1 & 1 & -1 & -1 \end{bmatrix}^{\mathrm{T}} \circ |\nabla\boldsymbol{\sigma}_V|, \qquad (49)$$

where symbol $\circ$ denotes the element-wise or Hadamard product and the scalar c_e is the introduced error. In particular, a 30% percentage

error is considered, i.e., $c_e = 0.3$.

ix) The control period T_s is set to 100 milliseconds due to the requirements of image acquisition and processing.

10.2. Experimental results

To illustrate the applicability of the method, PBVS and IBVS experiments have conducted for positioning the robot with respect to a motionless target object while fulfilling FOV and occlusion constraints, respectively. Note that, for stability reasons, the movement speed in the experiments has to be relatively slow to meet the limited bandwidth of the discontinuous control action, which is given by the sampling period of 0.1 seconds required for image acquisition and processing.

10.2.1. ***Experiment 1****: Visibility constraints due to the limited camera FOV in PBVS*

A video of the PBVS[9] experiment using the gradient vector with no errors can be accessed from the web link in [59]. For this experiment, Fig. 14 shows the position and orientation errors, the control action $\ddot{\mathbf{q}}_c$ and the constraint function ϕ_V, whereas Fig. 15 shows the trajectories followed by the image features. Note that, one of the markers remains in the FOV fully utilizing the allowed space and reaching smoothly the boundary of the hyperellipse constraint.

A second PBVS experiment has been conducted to verify the robustness of the proposed approach using the gradient vector $\nabla\boldsymbol{\sigma}_{Ve}$ with a 30% error. The result is very similar to the first PBVS experiment, where no errors were introduced. In particular, Fig. 16 compares the image features trajectories and the commanded joint speeds $\dot{\mathbf{q}}_c$ for both experiments, as they are the only variables where a little difference can be appreciated.

10.2.2. ***Experiment 2****: Visibility constraints due to object occlusion in IBVS*

A video of the IBVS experiment using the gradient vector with no errors can be accessed from the web link in [60]. For this experiment, Fig. 17 shows the image errors, the control action $\ddot{\mathbf{q}}_c$ and the constraint function ϕ_V,

[9]Note that it is more likely that the image features leave the camera FOV in PBVS because the control law is defined in the Cartesian space.

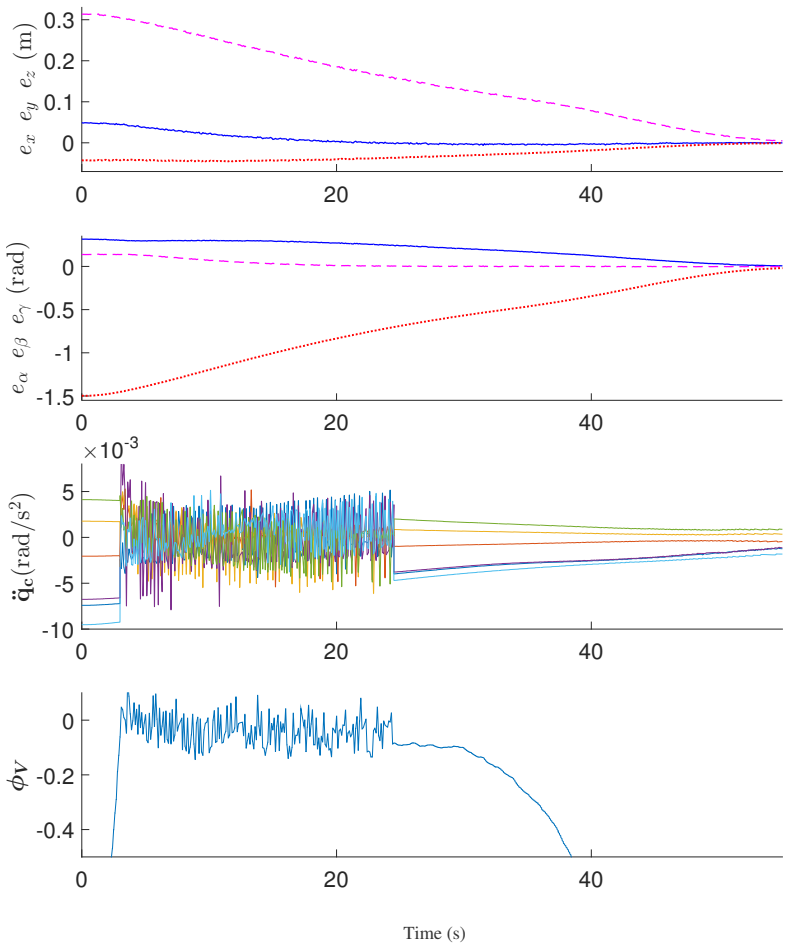

Fig. 14. PBVS experiment with no errors in the gradient vector. From top to bottom plots: (1) position errors; (2) orientation errors; (3) control action $\ddot{\mathbf{q}}_c$; (4) constraint function vector ϕ_V.

whereas Fig. 18 shows the trajectories followed by the image features. Note that, one of the markers remains visible, i.e., outside the forbidden area in the image plane, fully utilizing the allowed space and reaching smoothly the boundary of the hyperellipse constraint.

Again, a second IBVS experiment has been conducted to verify the robustness of the proposed approach using the gradient vector $\nabla\boldsymbol{\sigma}_{Ve}$ with a 30% error. The result is very similar to the first IBVS experiment, where no errors were introduced. In particular, Fig. 19 compares the image features trajectories and the commanded joint speeds $\dot{\mathbf{q}}_c$ for both experiments, as

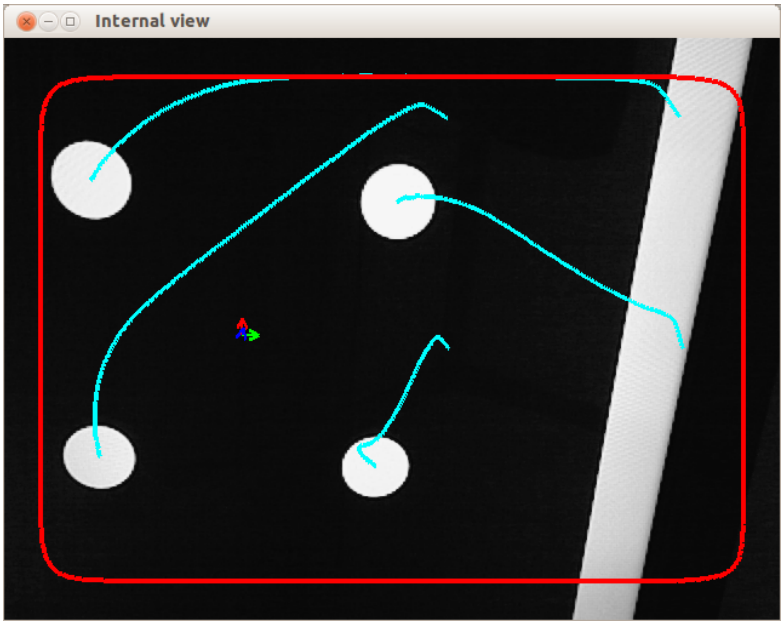

Fig. 15. Image features trajectories for the PBVS experiment with no errors in the gradient vector. Snapshot in the desired pose.

they are the only variables where a little difference can be appreciated.

11. Conclusions

An approach for constrained visual servoing has been presented using sliding mode concepts. In particular, the proposal used non-conventional sliding control to satisfy the typical robot constraints (range limits, speed limits and forbidden areas in the robot workspace) and to satisfy the visibility constraints (camera field-of-view and occlusions) of the visual servoing application. Moreover, another task with low-priority has been considered to track the target object.

On the one hand, the main advantages of the proposed approach are: low computational cost (see Appendix B), robustness and fully utilization of the allowed space for the constraints. On the other hand, like other sliding mode control applications, the proposed method has the chattering drawback.

The feasibility and effectiveness of the proposed approach was illustrated in simulation for a simple 2D case and a complex 3D case study. Furthermore, real experimentation with a conventional 6R industrial manipulator was also

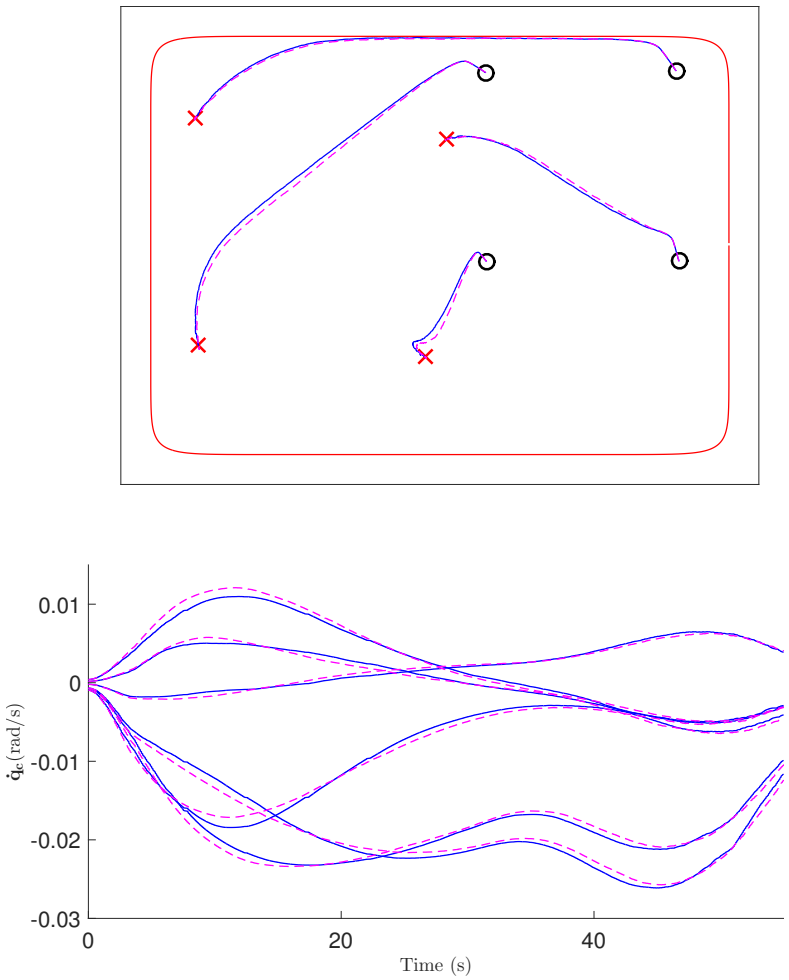

Fig. 16. Comparison of PBVS experiments without and with errors in the gradient vector. (1) Image features trajectories, from initial (circle) to desired (cross) pose ; (2) commanded joint speeds $\dot{\mathbf{q}}_c$. Blue-solid line experiment with no errors and magenta-dashed line experiment with errors.

presented to demonstrate its applicability and robustness. In particular, it is interesting to remark that, despite that the sampling time of the real platform used for experimentation was not very small, 0.1 s, the performance of the proposed SM algorithm was satisfactory even for a 30% gradient vector error.

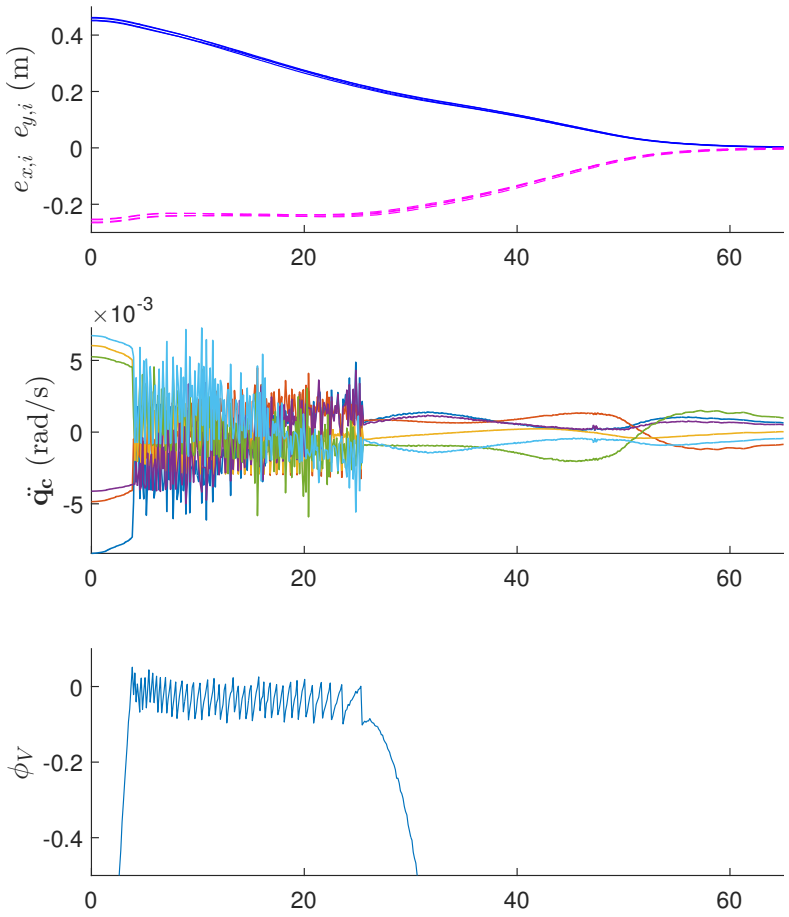

Fig. 17. IBVS experiment with no errors in the gradient vector. From top to bottom plots: (1) visual feature errors; (2) control action $\ddot{\mathbf{q}}_c$; (3) constraint function vector ϕ_V.

Appendix A. Proof of condition (12)

Proof. From (9) and (11), the column vector $\dot{\boldsymbol{\phi}}$ composed of the constraint function derivatives $\dot{\phi}_i$ is given by:

$$\dot{\boldsymbol{\phi}} = L_f \boldsymbol{\phi} - \mathbf{z}\, u^+, \tag{A.1}$$

where $\mathbf{z}$ is a column vector with the ith-component $z_i = 1$ if $\phi_i \geq 0$ and $z_i = 0$ otherwise.

Assuming that $\boldsymbol{\phi}(0) > \mathbf{0}$, the goal is to show that convergence to point $\boldsymbol{\phi} = \mathbf{0}$ is achieved in finite time. For this purpose, let $V = \mathbf{z}^{\mathrm{T}}\, \boldsymbol{\phi}$ be a

Fig. 18. Image features trajectories for the IBVS experiment with no errors in the gradient vector. Snapshot in the desired pose.

Lyapunov function candidate. Vector $\boldsymbol{\phi}$ can be generically partitioned into two subvectors $\boldsymbol{\phi} = [\boldsymbol{\phi}^{a\,\mathrm{T}} \quad \boldsymbol{\phi}^{N-a\,\mathrm{T}}]^{\mathrm{T}}$, where SM occurs in the manifold given by $\boldsymbol{\phi}^a = \mathbf{0}_a$, whereas $\boldsymbol{\phi}^{N-a} > \mathbf{0}_{N-a}$. Obviously, one of these two subvectors may be empty at a certain time. Since $\mathbf{z}^{N-a}$ is a constant column vector with all its elements equal to 1 , the time derivative of V results in:

$$\begin{aligned}\dot{V} = \frac{d}{dt}\left(\mathbf{z}^{\mathrm{T}}\,\boldsymbol{\phi}\right) =& \frac{d}{dt}\left(\begin{bmatrix}\mathbf{z}^a\\ \mathbf{z}^{N-a}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\boldsymbol{\phi}^a\\ \boldsymbol{\phi}^{N-a}\end{bmatrix}\right)\\ =& \begin{bmatrix}\dot{\mathbf{z}}^a\\ \mathbf{0}_{N-a}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{0}_a\\ \boldsymbol{\phi}^{N-a}\end{bmatrix} + \mathbf{z}^{\mathrm{T}}\,\dot{\boldsymbol{\phi}} = \mathbf{z}^{\mathrm{T}}\,\dot{\boldsymbol{\phi}}.\end{aligned} \tag{A.2}$$

Replacing vector $\dot{\boldsymbol{\phi}}$ with its value from (A.1), it is obtained:

$$\dot{V} = \mathbf{z}^{\mathrm{T}}\,L_f\boldsymbol{\phi} - \mathbf{z}^{\mathrm{T}}\,\mathbf{z}\,u^{+}. \tag{A.3}$$

The components of vector $\mathbf{z}$ range from 0 to 1, hence the upper bound of the first term in (A.3) is given by $z_i = 1$ if $L_f\phi_i > 0$ and $z_i = 0$ if $L_f\phi_i < 0$,

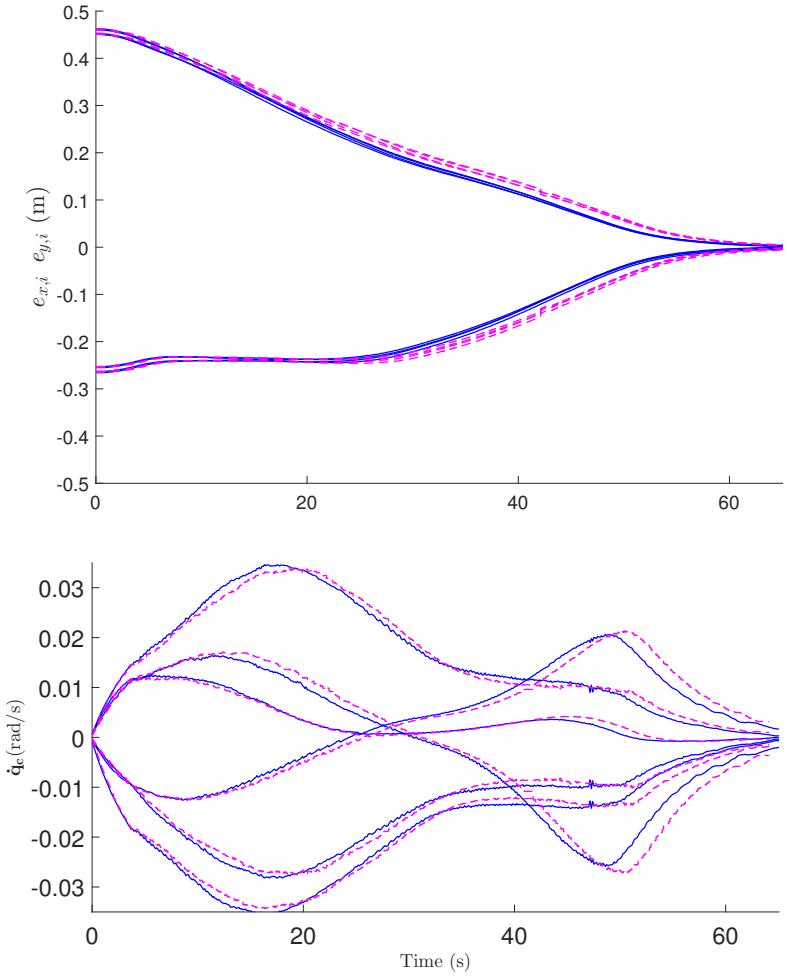

Fig. 19. Comparison of IBVS experiments without and with errors in the gradient vector. (1) visual feature errors; (2) commanded joint speeds $\dot{\mathbf{q}}_c$. Blue-solid experiment with no errors and magenta-dashed experiment with errors.

that is:

$$\mathbf{z}^{\mathrm{T}}\, L_f \boldsymbol{\phi} \leq \sum_{i=1}^{N} (\max(L_f \phi_i, 0)). \tag{A.4}$$

Assuming that $u^+ > 0$, the second term in (A.3) is negative and its upper bound is given by:

$$-\,\mathbf{z}^{\mathrm{T}}\,\mathbf{z}\,u^+ = -\|\mathbf{z}\|_2^2\; u^+ \leq -u^+ \quad \text{where} \quad \|\mathbf{z}\|_2 \geq 1 \quad \forall\, \boldsymbol{\phi} \neq \mathbf{0}_N, \tag{A.5}$$

because if vector $\boldsymbol{\phi}^{N-a}$ is not empty at least one component of vector $\mathbf{z}$ is equal to 1.

From (A.4) and (A.5), the upper bound of $\dot{V}$ results in:

$$\dot{V} \leq \sum_{i=1}^{N} \left(\max(L_f \phi_i, 0)\right) - u^{+}. \tag{A.6}$$

Therefore, if u^{+} fulfills (12) the Lyapunov function decays at a finite rate, it vanishes and collective SM in the intersection of the N constraints occurs after a finite time interval. □

Appendix B. Computer Implementation

The pseudo-code of the proposed method is shown below. The algorithm is executed at a sampling time of T_s seconds and uses the following auxiliary functions:

- Constraint functions and gradient vectors for the visibility and robot constraints: $\{\phi_{V,i}(\mathbf{q}, \dot{\mathbf{q}}), \phi_{R,qi}(\mathbf{q}, \dot{\mathbf{q}}), \phi_{R,si}(\dot{\mathbf{q}}), \phi_{R,oij}(\mathbf{q}, \dot{\mathbf{q}})\}$ and $\{\nabla\sigma_{V,i}(\mathbf{q}), \nabla\sigma_{R,qi}(\mathbf{q}), \nabla\phi_{R,si}(\dot{\mathbf{q}}), \nabla\sigma_{R,oij}(\mathbf{q})\}$.
- Jacobian matrix: $\mathbf{J}_s(\mathbf{q}, t)$.
- Visual feature vector (which is obtained with the computer vision algorithm described in Section 2) and its reference: $\mathbf{s}(\mathbf{q}, t)$ and $\mathbf{s}_{ref}(t)$.
- Moore-Penrose pseudoinverse function (using a tolerance to set to zero the very small singular values): $(\cdot)^{\dagger}$.
- Robot sensors: *GetRobotState*(), which returns the current robot state given by $\mathbf{q}$ and $\dot{\mathbf{q}}$.
- Actuators: *SendToJointControllers*($\ddot{\mathbf{q}}_c$), which sends the current commanded joint acceleration vector to the joint controllers.

The computation time per iteration of the algorithm in a computer with Intel Core i7-4710HQ processor at 2.5 GHz clock frequency using MATLAB® R2015b (compiled C-MEX-file) was around 20 microseconds for the case study example in Section 9.

Algorithm executed at sampling time of T_s seconds

$[\mathbf{q}, \dot{\mathbf{q}}]$ =`GetRobotState()`;

$\dot{\mathbf{s}} = (\mathbf{s} - \mathbf{s}_{prev})/T_s$; `// Derivative`

$\dot{\mathbf{s}}_{ref} = (\mathbf{s}_{ref} - \mathbf{s}_{ref,prev})/T_s$; `// Derivative`

$\ddot{\mathbf{s}}_{ref} = (\dot{\mathbf{s}}_{ref} - \dot{\mathbf{s}}_{ref,prev})/T_s$; `// Derivative`

$\dot{\mathbf{J}}_s = (\mathbf{J}_s - \mathbf{J}_{s,prev})/T_s$; `// Derivative`

$\ddot{\mathbf{s}}_c = \ddot{\mathbf{s}}_{ref} - K_{T,p}(\mathbf{s} - \mathbf{s}_{ref}) - K_{T,v}(\dot{\mathbf{s}} - \dot{\mathbf{s}}_{ref})$; `// Eq.` (42)

$\mathbf{A}_1 = \begin{bmatrix} \mathbf{K}_V \, \nabla \boldsymbol{\sigma}_V^{\mathrm{T}} \\ \mathbf{K}_{R,q} \, \nabla \boldsymbol{\sigma}_{R,q}^{\mathrm{T}} \\ \nabla \boldsymbol{\phi}_{R,s}^{\mathrm{T}} \\ \mathbf{K}_{R,o} \, \nabla \boldsymbol{\sigma}_{R,o}^{\mathrm{T}} \end{bmatrix}$ with the gradients of all active constraints:

$\phi_{V,i} > 0$, $\phi_{R,qi} > 0$, $\phi_{R,si} > 0$, $\phi_{R,oij} > 0$; `// Eq.` (40)

$\mathbf{b}_1 = -\mathbf{u}_1^+$; `// Eq.` (40)

$\mathbf{A}_2 = \mathbf{J}_s$; `// Eq.` (41)

$\mathbf{b}_2 = \ddot{\mathbf{s}}_c - \dot{\mathbf{J}}_s \dot{\mathbf{q}} - \partial \dot{\mathbf{s}} / \partial t$; `// Eq.` (41)

$\ddot{\mathbf{q}}_c = \mathbf{A}_1^{\dagger} \mathbf{b}_1 + (\mathbf{A}_2 (\mathbf{I} - \mathbf{A}_1^{\dagger} \mathbf{A}_1))^{\dagger} (\mathbf{b}_2 - \mathbf{A}_2 \mathbf{A}_1^{\dagger} \mathbf{b}_1)$; `// Eq.` (15)

$\mathbf{s}_{prev} = \mathbf{s}$; `// For next iteration`

$\mathbf{s}_{ref,prev} = \mathbf{s}_{ref}$; `// For next iteration`

$\dot{\mathbf{s}}_{ref,prev} = \dot{\mathbf{s}}_{ref}$; `// For next iteration`

$\mathbf{J}_{s,prev} = \mathbf{J}_s$; `// For next iteration`

`SendToJointControllers(`$\ddot{\mathbf{q}}_c$`);`

Acknowledgements

This work was supported in part by the Spanish Government under grants BES-2010-038486 and Project DPI2013-42302-R, and the Generalitat Valenciana under grant VALi+d APOSTD/2016/044.

References

[1] F. Chaumette, S. Hutchinson, Visual servoing and visual tracking, Springer Handbook of robotics (2008) 563–583.

[2] L. Gracia, F. Garelli, A. Sala, Reactive sliding-mode algorithm for collision avoidance in robotic systems, IEEE Trans. on Control Systems Technology 21 (2013) 2391–2399.

[3] N. Cazy, P. Wieber, P. Giordano, F. Chaumette, Visual servoing when visual information is missing: Experimental comparison of visual feature prediction schemes, in: Proceedings of the IEEE Int. Conference on Robotics and Automation, Seattle WA, USA, pp. 6031–6036.

[4] G. Garcia, J. Pomares, F. Torres, P. Gil, Event-based visual servoing with features' prediction, in: ROBOT2013: First Iberian Robotics Conference, Advances in Robotics, Springer Int. Publishing, Madrid, Spain, 2014, pp. 679–691.

[5] N. Garcia-Aracil, E. Malis, R. Aracil-Santonja, C. Perez-Vidal, Continuous visual servoing despite the changes of visibility in image features, IEEE Trans. on Robotics 21 (2005) 1214–1220.

[6] D. Kragic, H. I. Christensen, Survey on Visual Servoing for Manipulation, Technical Report, COMPUTATIONAL VISION AND ACTIVE PERCEPTION LABORATORY, 2002.

[7] G. Chesi, K. Hashimoto, D. Prattichizzo, A. Vicino, Keeping Features in the Field of View in Eye-In-Hand Visual Servoing: A Switching Approach, IEEE Trans. on Robotics 20 (2004) 908–913.

[8] N. R. Gans, S. a. Hutchinson, Stable Visual Servoing Through Hybrid Switched-System Control, IEEE Trans. on Robotics 23 (2007) 530–540.

[9] D. Kim, R. Lovelett, Z. Wang, A. Behal, A region-based switching scheme for practical visual servoing under limited FOV and dynamically changing features, in: IASTED 14th Int. Conf. Robotic Applications.

[10] L. Deng, F. Janabi-Sharifi, Hybrid motion control and planning strategies for visual servoing, IEEE Trans. on Industrial Electronics 52 (2005) 1024–1040.

[11] O. Kermorgant, F. Chaumette, Combining IBVS and PBVS to ensure the visibility constraint, in: 2011 IEEE/RSJ Int. Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 2849–2854.

[12] A. H. A. Hafez, C. Jawahar, Visual Servoing by Optimization of a 2D/3D Hybrid Objective Function, in: Proceedings 2007 IEEE Int. Conference on Robotics and Automation, IEEE, 2007, pp. 1691–1696.

[13] V. Kyrki, D. Kragic, H. I. Christensen, New shortest-path approaches to visual servoing, 2004.

[14] M. Baumann, S. Léonard, E. a. Croft, J. J. Little, Path Planning for Improved Visibility Using a Probabilistic Road Map, IEEE Trans. on Robotics 26 (2010) 195–200.

[15] G. Chesi, Y. S. Hung, Global path-planning for constrained and optimal visual servoing, IEEE Trans. on Robotics 23 (2007) 1050–1060.

[16] G. Chesi, Visual Servoing Path Planning via Homogeneous Forms and LMI Optimizations, IEEE Trans. on Robotics 25 (2009) 281–291.

[17] M. Kazemi, K. K. Gupta, M. Mehrandezh, Randomized Kinodynamic Planning for Robust Visual Servoing, IEEE Trans. on Robotics 29 (2013) 1197–1211.

[18] G. Garcia, J. Pomares, F. Torres, Automatic robotic tasks in unstructured environments using an image path tracker, Control Engineering Practice 17 (2009) 597–608.

[19] Y. Huang, X. Zhang, Y. Fang, Vision-based minimum-time planning of mobile robots with kinematic and visibility constraints, IFAC Proceedings Volumes 47 (2014) 11878–11883.

[20] P. Corke, S. Hutchinson, A new partitioned approach to image-based visual servo control, IEEE Trans. on Robotics and Automation 17 (2001) 507–515.

[21] Y. Mezouar, F. Chaumette, Path planning for robust image-based control, IEEE Trans. on Robotics and Automation 18 (2002) 534–549.

[22] N. J. Cowan, J. D. Weingarten, D. E. Koditschek, Visual servoing via navigation functions, IEEE Trans. on Robotics and Automation 18 (2002) 521–533.

[23] J. Chen, D. M. Dawson, W. E. Dixon, V. K. Chitrakaran, Navigation function-based visual servo control, Automatica 43 (2007) 1165–1177.

[24] A. Hajiloo, M. Keshmiri, W. F. Xie, T. T. Wang, Robust Online Model Predictive Control for a Constrained Image-Based Visual Servoing, IEEE Trans. on Industrial Electronics 63 (2016) 2242–2250.

[25] G. Allibert, E. Courtial, F. Chaumette, Predictive Control for Constrained Image-Based Visual Servoing, IEEE Trans. on Robotics 26 (2010) 933–939.

[26] S. Heshmati-alamdari, G. K. Karavas, A. Eqtami, M. Drossakis, K. J. Kyriakopoulos, Robustness analysis of model predictive control for constrained Image-Based Visual Servoing, in: 2014 IEEE Int. Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 4469–4474.

[27] X. Song, F. Miaomiao, CLFs-based optimization control for a class of constrained visual servoing systems, ISA Trans. 67 (2017) 507–514.

[28] B. J. Nelson, P. K. Khosla, Strategies for Increasing the Tracking Region of an Eye-in-Hand System by Singularity and Joint Limit Avoidance, The Int. Journal of Robotics Research 14 (1995) 255–269.

[29] F. Chaumette, T. Marchand, A redundancy-based iterative approach for avoiding joint limits: application to visual servoing, IEEE Trans. on Robotics and Automation 17 (2001) 719–730.

[30] X. Zhong, X. Zhong, X. Peng, Robots visual servo control with features constraint employing Kalman-neural-network filtering scheme, Neurocomputing 151 (2015) 268–277.

[31] G. Chesi, A. Vicino, Visual servoing for large camera displacements, Robotics, IEEE Trans. on 20 (2004) 724–735.

[32] N. Mansard, F. Chaumette, Task sequencing for sensor-based control, IEEE Trans. on Robotics 23 (2007) 60–72.

[33] E. Rimon, D. Koditschek, Exact robot navigation using artificial potential functions, IEEE Trans. on Robotics and Automation 8 (1992) 501–518.

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

[35] P. Zanne, G. Morel, F. Piestan, Robust vision based 3D trajectory tracking using sliding mode control, in: Proceedings 2000 ICRA. Millennium Conference. IEEE Int. Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), volume 3, IEEE, 2000, pp. 2088–2093.

[36] J. Kim, D. Kim, S. Choi, S. Won, Image-based visual servoing using sliding mode control, in: Proceedings of the SICE-ICASE Int. Joint Conference, pp. 4996–5001.

[37] T. R. Oliveira, A. J. Peixoto, A. C. Leite, L. Hsu, Sliding mode control of uncertain multivariable nonlinear systems applied to uncalibrated robotics visual servoing, 2009 American Control Conference (2009) 71–76.

[38] T. R. Oliveira, A. C. Leite, A. J. Peixoto, L. Hsu, Overcoming limitations of uncalibrated robotics visual servoing by means of sliding mode control and switching monitoring scheme, Asian Journal of Control 16 (2014) 752–764.

[39] V. Parra-Vega, S. Arimoto, Y.-H. Liu, G. Hirzinger, P. Akella, Dynamic sliding pid control for tracking of robot manipulators: theory and experiments, IEEE Trans. on Robotics and Automation 19 (2003) 967–976.

[40] F. Li, H.-L. Xie, Sliding mode variable structure control for visual servoing system, Int. Journal of Automation and Computing 7 (2010) 317–323.

[41] M. Parsapour, S. RayatDoost, H. Taghirad, A 3d sliding mode control approach for position based visual servoing system, Scientia Iranica. Transaction B, Mechanical Engineering 22 (2015) 844–853.

[42] M. Parsapour, H. D. Taghirad, Kernel-based sliding mode control for visual servoing system, IET Computer Vision 9 (2015) 309–320.

[43] W. Burger, E. Dean-Leon, G. Cheng, Robust second order sliding mode control for 6D position based visual servoing with a redundant mobile manipulator, in: 2015 IEEE-RAS 15th Int. Conference on Humanoid Robots (Humanoids), IEEE, 2015, pp. 1127–1132.

[44] H. M. Becerra, C. Sagüés, Sliding Mode Control for Visual Servoing of Mobile Robots using a Generic Camera, in: Prof. Andrzej Bartoszewicz (Ed.), Sliding Mode Control, 2011, pp. 221–236.

[45] H. M. Becerra, G. López-Nicolás, C. Sagüés, A sliding-mode-control law for mobile robots based on epipolar visual servoing from three views, IEEE Trans. on Robotics 27 (2011) 175–183.

[46] J. Xin, B.-J. Ran, X.-M. Ma, Robot visual sliding mode servoing using SIFT features, in: 2016 35th Chinese Control Conference (CCC), IEEE, 2016, pp. 4723–4729.

[47] Yu, Stability Analysis of Visual Servoing with Sliding-mode Estimation and Neural Compensation, Int. Journal of Control, Automation, and Systems 4 (2013) 545–558.

[48] Y. Nakamura, H. Hanafusa, T. Yoshikawa, Task-priority based redundancy control of robot manipulators, The Int. Journal of Robotics Research 6 (1987) 3–15.

[49] P. Corke, Robotics, Vision and Control: Fundamental Algorithms in MATLAB, Springer-Verlag, Berlin, Germany, 2011.

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

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

[52] Wikipedia website, Superellipse, https://en.wikipedia.org/wiki/Superellipse (Accessed 2017/04/10).

[53] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer-Verlag, London, UK, 2009.

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

[55] H. Fakhry, W. Wilson, A modified resolved acceleration controller for position-based visual servoing, Mathematical and Computer Modelling 24 (1996) 1–9.

[56] M. Keshmiri, W. Xie, A. Mohebbi, Augmented image-based visual servoing of a manipulator using acceleration command, IEEE Trans. on Industrial Electronics 61 (2014) 5444–5452.

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

[58] E. Marchand, F. Spindler, F. Chaumette, ViSP for visual servoing: a generic software platform with a wide class of robot control skills, IEEE Robotics & Automation Magazine 12 (2005) 40–52.

[59] Web link of the video for the PBVS experiment considering FOV constraints (video at double speed), https://media.upv.es/player/?id=56805190-8411-11e7-90ea-23686ce0f1be, 2017.

[60] Web link of the video for the IBVS experiment considering occlusion constraints (video at double speed), https://media.upv.es/player/?id=2dd4d490-8412-11e7-90ea-23686ce0f1be, 2017.