Robust Auto Tool Change for Industrial Robots Using Visual Servoing

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

[a]Instituto IDF, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain (e-mail: pmunyoz@idf.upv.es). *Corresponding author

ARTICLE HISTORY
Compiled November 23, 2018

ABSTRACT
This work presents an automated solution for tool changing in industrial robots using visual servoing and sliding mode control. The robustness of the proposed method is due to the control law of the visual servoing, which uses the information acquired by a vision system to close a feedback control loop. Furthermore, sliding mode control is simultaneously used in a prioritized level to satisfy the constraints typically present in a robot system: joint range limits, maximum joint speeds and allowed workspace. Thus, the global control accurately places the tool in the warehouse, but satisfying the robot constraints. The feasibility and effectiveness of the proposed approach is substantiated by simulation results for a complex 3D case study. Moreover, real experimentation with a 6R industrial manipulator is also presented to demonstrate the applicability of the method for tool changing.

KEYWORDS
tool change, robot system, visual servoing, sliding mode control

1. Introduction

The automation of industrial processes has allowed, among other things, to reduce human exposure to repetitive and/or dangerous tasks, as well as to increase the productivity and quality of the manufactured products. This automation has been largely linked to the technological breakthrough of complex sensors such as vision and complex actuators such as robots. Even so, there are still non-automated processes within the production lines due to their complexity.

A good example of this situation is the tool change task performed by a robot, which consists in switching between different tools to complete a particular set of tasks. Regardless of the working environment, nowadays the change is pre-programmed and several problems may arise: 1) discrepancies in the tool position within the warehouse along time with respect to the first calibration; 2) misplacement of the tool in the warehouse, which consequence is the requirement of a tool checkup sub-routine, slowing down the process of tool change. 3) Both the tool and its warehouse must be placed imperiously in fixed positions within the robot workspace. Recent advances in tool management for the automation of manufacturing systems have been reported in several works, such as (Bi & Zhang, 2001), (Pehlivan & Summers, 2008) and (Hashemi et al., 2014), which highlights the significance of the tool change procedure and the necessity of improving it.

This paper proposes the use of visual feedback control, also known as Visual Servoing (VS), to provide a robust solution to overcome the aforementioned problems. VS technique, studied for more than 30 years, refers to the motion control of a robot system using visual feedback signals from a vision device (Chaumette & Hutchinson, 2008). 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. Basically, VS control laws can be divided into two categories: those that carry out VS in operational space, namely Position-Based VS (PBVS); and those that carry out visual servoing in the image space, known as Image-Based VS (IBVS). The main difference is that PBVS schemes reconstruct the relative pose of the object with respect to the camera, while IBVS schemes are based on the comparison of visual features in the image for current and desired poses. Another aspect to take into account is the position of the vision sensor within the VS system. There are two main cases known as *eye-in-hand* configuration, when the camera moves attached to the robot end-effector, and *eye-to-hand* configuration, when the camera is outside the robot system. In our case, the *eye-to-hand* configuration is chosen to have a broader view of the entire workspace, allowing us to detect not only the position of the tool and the warehouse, but also possible obstacles to be avoided.

To the best of the authors knowledge, VS has not yet been used to address the problem tackle in this work. However, some solutions based on computer vision can be found in literature to improve the process of automatic robot tool change. For instance, a calibration method is presented in (Gordic & Ongaro, 2016) to correct image distortion in order to obtain an accurate location of the tool center point. Similarly, other works (Motta, de Carvalho, & McMaster, 2001) (Du & Zhang, 2013) (Yin, Ren, Zhu, Yang, & Ye, 2013) proposed techniques for modeling and performing robot calibration processes using a vision-based measurement system. However, none of the mentioned approaches consider a control loop using the visual information, like proposed in this work.

In general, to accomplish a specific task, e.g., tool changing operations, the robot has to fulfill a number of *constraints*, such as not exceeding the *joint range* limits, not exceeding the maximum *joint speeds* and not leaving the *allowed workspace*. Typically, the allowed workspace is given by: the workspace limits of the robot; obstacles in the environment that must be avoided; a possible predefined area to confine the robot in a limited region to avoid unnecessary or not desired movements; etc. However, the control law given by conventional VS can lead to a trajectory that would not satisfy these constraints, for instance due to a large motion in a positioning task, due to modeling errors or because a moving target temporarily leaves the robot workspace.

In general, the robot motion should be conditioned to fulfill the constraints but without aborting the VS task. Different approaches have been presented to deal with constraints in VS. For instance, combining different VS approaches: IBVS and PBVS (Chesi, Hashimoto, Prattichizzo, & Vicino, 2004; Hafez & Jawahar, 2007; Kermorgant & Chaumette, 2011), PBVS and backward motion (Gans & Hutchinson, 2007), hybrid VS and translational movements (Kim, Lovelett, Wang, & Behal, 2009). Other proposals rely on path planning algorithms (Baumann, Léonard, Croft, & Little, 2010; Huang, Zhang, & Fang, 2014; Kazemi, Gupta, & Mehrandezh, 2013; Zhong, Zhong, & Peng, 2015) or online corrective terms (Chen, Dawson, Dixon, & Chitrakaran, 2007; Corke & Hutchinson, 2001). This work deals with constraints using sliding mode control (SMC) (Edwards & Spurgeon, 1998; Menani, Mohammadridha, Magdelaine, Abdelaziz, & Moog, 2017; Mobayen, Tchier, & Ragoub, 2017; Muñoz-

Vázquez, Parra-Vega, & Sánchez-Orta, 2017). The generic visual servoing architecture, such as the one used in (Ouyang, Zhang, Gupta, & Zhao, 2007), consists of two levels of control: visual feedback and joint level control. This work considers the same architecture, but redefining the first level with two objectives, visual feedback and constraints fulfillment, which are handled using task-priority redundancy resolution (Nakamura, Hanafusa, & Yoshikawa, 1987) and SMC. The proposed SMC algorithm for constraints fulfillment does only activate when the robot system is about to violate the constraints and allows to reach the limits smoothly depending on a free design parameter, whereas the task-priority strategy allows to hierarchical satisfy the constraints while making as small as possible the reference tracking error.

This paper proposes an automatic solution to increase the robustness and flexibility of a common problem in automation, the tool change procedure in industrial robots. In particular, the novelty of using visual servoing to close the control loop in this specific problem increases the robustness, in the sense that it can deal with possible misplacement of the tool with respect to the warehouse (e.g., due to mechanical friction, due to wear of the robot over time, etc.) or due to an unexpected position of the warehouse. Moreover, visual servoing also increases the flexibility of the process, since it allows the system to deal with moving warehouses. Furthermore, the fulfillment of robot constraints is addressed in this work using SMC in order to benefit from its inherent robustness and low computational cost. This work details a general approach to obtain robust auto tool change and can be adapted to all VS configurations. To illustrate this generality, experiments with different VS configurations are presented using a real 6R industrial manipulator for a auto tool change tasks.

The structure of the paper is as follows. Next section introduces some preliminaries, 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 proposed approach is applied in Section 6 to a complex 3D case study in order to show its feasibility and effectiveness. Furthermore, real experimentation with a 6R industrial manipulator is shown in Section 7 to demonstrate the applicability of the method. Finally, some concluding remarks are given.

2. Preliminaries

Fig. 1 shows the coordinate frames involved in the eye-to-hand VS problem: F robot base frame; E robot end-effector frame; C camera frame; O object frame; O^* desired object frame. Following the standard notation (Chaumette & Hutchinson, 2008), the VS application is characterized by the visual *feature vector* $\mathbf{s}$ that depends on the robot *configuration* $\mathbf{q}$ and also explicitly on time for the general case of a *moving target* object, that is:

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

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

The first-order kinematics of 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. In this case, $\partial \mathbf{s}/\partial t$ is related to the change of the visual features (target object) in the 3D Euclidean space. When the trajectory of the target is known, $\partial \mathbf{s}/\partial t$ can be updated using its analytical expression. Otherwise, different approaches have been proposed in the literature when no a priori information is available, such as, feedforward control or motion prediction, as reviewed in (Chaumette & Hutchinson, 2008). The resulting Jacobian matrix $\mathbf{J}_s$ 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; and ${}^e\mathbf{J}_e$ is the robot Jacobian expressed in the end-effector frame. For more details on the computation of $\mathbf{J}_s$ see (Corke, 2011).

The second-order kinematics of vector $\mathbf{s}$ is given by:

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

The task carried out by the robot system consists on achieving a *reference* value for the visual feature vector $\mathbf{s}$, that is:

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

where $\mathbf{s}_{ref}(t)$ is the reference for the visual feature vector.

Both PBVS and IBVS require a *computer vision algorithm* composed of three modules: the first one performs the *image processing* for obtaining the image plane coordinates (u_i, v_i) of all the visual features; the second one carries out 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 one, which only applies for PBVS, performs the *pose estimation* of the camera (eye-in-hand) or robot (eye-to-hand) from the features of the second module. The output of the algorithm is the visual feature vector $\mathbf{s}$. This work assumes the existence of this algorithm.

This work also assumes the existence of an *underlying robot control* in charge of achieving a particular joint acceleration from an acceleration command $\ddot{\mathbf{q}}_c$. However, the actual acceleration $\ddot{\mathbf{q}}$ will not be exactly the commanded one $\ddot{\mathbf{q}}_c$ due to the dynamics and inaccuracies of the low-level control loop. It will be assumed hereinafter that the dynamics of the low-level control loop 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, where $\mathbf{d}_c$ represents the inaccuracies of the robot control.

The *goal* of this paper is to design a control system that is aware of the robot configuration and that generates the commanded acceleration to be sent to the robot joint controllers, so that: the actual visual feature vector $\mathbf{s}$ is as close as possible to the given reference value $\mathbf{s}_{ref}$; the robot remains in its allowed workspace throughout the process; and the joint range limits and the maximum joint speeds are not exceeded during the operation.

3. Background theory

This section briefly reviews previous results from literature that will be subsequently used by the proposed approach.

3.1. Task-priority scheme

In order to tackle several objectives simultaneously (Chiaverini, Oriolo, & Walker, 2008), it is useful to consider the task-priority strategy (Nakamura et al., 1987), which consists of assigning an order of priority to the given tasks. Thus, a lower-priority task is satisfied only by using the degrees of freedom in the null space of the higher-priority ones. The formulation is as follows. Let us consider M tasks which consist on calculating a command vector $\ddot{\mathbf{q}}_c$ to fulfill the following acceleration equality constraints:

$$\mathbf{A}_i \ddot{\mathbf{q}}_c = \mathbf{b}_i, \quad i = 1, \dots, M, \tag{7}$$

where matrix $\mathbf{A}_i$ and vector $\mathbf{b}_i$ of the ith task are assumed known and index i represents the priority order: $i = 1$ for highest priority and $i = M$ to lowest.

The solution $\ddot{\mathbf{q}}_{c,M}$ that hierarchically minimizes the error of equations in (7) is given by (Siciliano & Slotine, 1991):

$$\begin{aligned} \ddot{\mathbf{q}}_{c,i} &= \ddot{\mathbf{q}}_{c,i-1} + (\mathbf{A}_i \mathbf{N}_{i-1})^{\dagger} (\mathbf{b}_i - \mathbf{A}_i \ddot{\mathbf{q}}_{c,i-1}) \\ \mathbf{N}_i &= \mathbf{N}_{i-1} (\mathbf{I} - (\mathbf{A}_i \mathbf{N}_{i-1})^{\dagger} (\mathbf{A}_i \mathbf{N}_{i-1})), \quad i = 1, \dots, M, \quad \ddot{\mathbf{q}}_{c,0} = \mathbf{0},\, \mathbf{N}_0 = \mathbf{I}, \end{aligned} \tag{8}$$

where $\mathbf{I}$ and $\mathbf{0}$ denote the identity matrix and zero column vector, respectively, of suitable size, superscript $\dagger$ denotes the Moore-Penrose pseudoinverse and $\ddot{\mathbf{q}}_{c,i}$ and $\mathbf{N}_i$ are the solution vector and null-space projection matrix, respectively, for the set of first i tasks. The pseudoinverse may be computed via the singular value decomposition (SVD) method (Golub & Van Loan, 1996) and using a tolerance to set to zero the very small singular values to avoid extremely large values for the accelerations.

3.2. Sliding mode control for geometric invariance

This section reviews the principles of SMC and geometric invariance theory (Garelli, Mantz, & De Battista, 2011; Gracia, Sala, & Garelli, 2012) that will be used by the proposed method.

Let us consider a dynamical system with n_x states and n_u inputs given by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{d}) + \mathbf{g}(\mathbf{x})\, \mathbf{u}, \tag{9}$$

where $\mathbf{x}(t) \in X \subset \mathbb{R}^{n_x}$ is the state vector, $\mathbf{d}(t) \in D \subset \mathbb{R}^{n_d}$ is an unmeasured disturbance or model uncertainty, $\mathbf{u}(t) \in U \subset \mathbb{R}^{n_u}$ is the control input vector (possibly discontinuous), $\mathbf{f} : \mathbb{R}^{n_x + n_d} \to \mathbb{R}^{n_x}$ is a vector field defined in $X \bigcup D$ and $\mathbf{g} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_u}$ is a set of n_u vector fields defined in X.

Consider also that the system state vector $\mathbf{x}$ is subject to user-specified inequality constraints $\phi_i(\mathbf{x}) \leq 0$, $i = 1, \dots, N$, 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:

$$\Phi = \{\mathbf{x} \mid \phi_i(\mathbf{x}) \leq 0\}, \quad i = 1, \ldots, N. \tag{10}$$

The objective is to find a control input $\mathbf{u}$ such that the trajectories originating in Φ remain in Φ for all times t. Mathematically, the invariance of Φ is guaranteed by an input $\mathbf{u}$ such that[1]:

$$\begin{aligned} \frac{d(\phi_i(\mathbf{x}))}{dt} &= \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\dot{\mathbf{x}} = \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{f}(\mathbf{x},\mathbf{d}) + \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{g}(\mathbf{x})\,\mathbf{u} \\ &= L_f\phi_i(\mathbf{x},\mathbf{d}) + \mathbf{L_g}\phi_i(\mathbf{x})\mathbf{u} \leq 0, \qquad \forall i \mid \phi_i(\mathbf{x}) \geq 0, \end{aligned} \tag{11}$$

where ∇ denotes the gradient vector, 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 all the scalars $\mathbf{L_g}\phi_i\mathbf{u}$ are negative, i.e., any vector pointing toward the interior of the allowed region, can be used to satisfy (11). This vector may be computed, for example, by solving a linear-programming optimization problem (Dantzig, 2016). However, the goal of this work is to use a more simple strategy, involving only gradient computation and simple matrix operations. In particular, it is proposed to use the variable structure control law below to make the set Φ in (10) invariant:

$$\mathbf{u} = \begin{cases} \mathbf{0} & \text{if} \quad \max\limits_i \{\phi_i(\mathbf{x})\} < 0 \\ \mathbf{u}_c & \text{otherwise,} \end{cases} \tag{12}$$

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

$$\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{u}_c = -\mathbf{1}_b\,u^+, \tag{13}$$

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 switching gain to be chosen high enough to satisfy (11). 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 (Gracia, Garelli, & Sala, 2013):

$$u^+ > \sum_{i=1}^{b} (\max(L_f\phi_i, 0)). \tag{14}$$

When the state trajectory tries by itself to leave the allowed region Φ, the above control law (12) 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) behavior with no open-loop phase (reaching mode) (Edwards & Spurgeon, 1998; Utkin, Guldner, & Shi, 2009).

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

4. Proposed method

4.1. Overview of the method

Fig. 2 shows the overview of the proposal, where task-priority redundancy resolution is used with two hierarchical priority levels:

- The first level includes the *robot constraints*, which are a set of constraints that must be satisfied at all times for reasons of safety in order to avoid: exceeding the joint range limits; exceeding the maximum joint speeds; exceeding the workspace limits; colliding with objects in the robot environment.
- The second priority level, i.e., the one with the lowest priority, is designed for *reference tracking*: control the robot so 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 priority levels, see Fig. 2, is the robot state $\{\mathbf{q}, \dot{\mathbf{q}}\}$ and each level gives an acceleration equality $\mathbf{A}_i\ddot{\mathbf{q}}_c = \mathbf{b}_i$ (7) whose square error must be minimized. On the one hand, the acceleration equality for the first level is obtained below using the SMC theory presented in Section 3.2, in order to fulfill the corresponding constraints. On the other hand, the acceleration equality for the second level is obtained below using VS. Then, the commanded joint acceleration vector $\ddot{\mathbf{q}}_c$ is computed by the priority-task redundancy resolution block using (8) and serves as input to the joint controllers.

4.2. Lie derivatives

The theoretical developments in Section 3.2 require a dynamical system in the form of (9). In particular, it will be considered the state vector $\mathbf{x} = \begin{bmatrix}\mathbf{q}^{\mathrm{T}} & \dot{\mathbf{q}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, the disturbance $\mathbf{d} = \mathbf{d}_c$ and the input $\mathbf{u} = \ddot{\mathbf{q}}_c$. Thus, from (6), the state equation of the system is given by:

$$\dot{\mathbf{x}} = \begin{bmatrix}\mathbf{O} & \mathbf{I} \\ \mathbf{O} & \mathbf{O}\end{bmatrix}\mathbf{x} + \begin{bmatrix}\mathbf{0} \\ \mathbf{d}_c\end{bmatrix} + \begin{bmatrix}\mathbf{O} \\ \mathbf{I}\end{bmatrix}\mathbf{u}, \tag{15}$$

and, hence, the Lie derivatives for ϕ_i are given by:

$$\mathbf{L_g}\phi_i = \nabla\phi_i^{\mathrm{T}}\,\mathbf{g} = (\partial\phi_i/\partial\dot{\mathbf{q}})^{\mathrm{T}} \tag{16}$$

$$L_f\phi_i = \nabla\phi_i^{\mathrm{T}}\,\mathbf{f} = (\partial\phi_i/\partial\mathbf{q})^{\mathrm{T}}\,\dot{\mathbf{q}} + (\partial\phi_i/\partial\dot{\mathbf{q}})^{\mathrm{T}}\,\mathbf{d}_c. \tag{17}$$

4.3. Robot constraints

The first level includes three types of constraints: those required to not exceed the joint range limits; those required to not exceed the maximum joint speeds; and those required to not leave the allowed workspace for the robot (e.g., to avoid collisions, to fulfill the workspace limits, etc.).

4.3.1. Joint range limits

The following constraints are considered for the joint limits:

$$\sigma_{R,qi}(\mathbf{q}) = -1 + \frac{|\ q_i - q_{\mathrm{mid},i}|}{\Delta q_{\max,i}/2} + m_{R,q} = -1 + |\ \widetilde{q}_i|\ + m_{R,q} \leq 0, \quad i = 1, \ldots, n, \quad (18)$$

where $q_{\mathrm{mid},i}$ and $\Delta q_{\max,i}$ 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 (e.g., SM chattering band, modeling errors, robot control inaccuracies, etc.) in order to avoid reaching the joint limits.

Since the above constraints depend only on robot configuration $\mathbf{q}$, they will be modified as follows for the sliding manifold to have relative degree one[2] with respect to the control variable $\ddot{\mathbf{q}}_c$:

$$\phi_{R,qi}(\mathbf{q}, \dot{\mathbf{q}}) = \sigma_{R,qi}(\mathbf{q}) + K_{R,qi}\frac{d\sigma_{R,qi}(\mathbf{q})}{dt} = \sigma_{R,qi} + K_{R,qi}\ \nabla\sigma_{R,qi}^{\mathrm{T}}\ \dot{\mathbf{q}} \leq 0, \quad (19)$$

where $K_{R,qi}$ is an arbitrary positive parameter that determines the rate of approach to the boundary of the original constraint, i.e., the joint limits.

4.3.2. Maximum joint speeds

The following constraints are considered for the joint speeds:

$$\phi_{R,si}(\dot{\mathbf{q}}) = -1 + \frac{|\ \dot{q}_i|}{\dot{q}_{\max,i}} + m_{R,s} = -1 + \left|\ \widetilde{\dot{q}}_i\right|\ + m_{R,s} \leq 0, \quad i = 1, \ldots, n, \quad (20)$$

where $\dot{q}_{\max,i}$ and $-\dot{q}_{\max,i}$ are the maximum and minimum 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. Note that both speed limits have been considered symmetric. If that would not be case the constraint (20) can be easily split into two constraints for maximum and minimum speeds.

4.3.3. Robot workspace

In order to avoid that points of the robot enter or leave certain predefined regions, the robot workspace must be constrained. Some examples requiring this type of constraint are the following: for certain applications, the robot must be confined in a limited region depending on the tasks to avoid unnecessary or not desired movements, e.g., the camera retreat phenomenon (Chaumette, 1998); in a collision avoidance problem, the region defined by the obstacle must be avoided; and, in general, the predefined workspace limits for the robot must be fulfilled. Thus, 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 $\Phi_{WS}(\mathbf{p}_j) = \{\mathbf{p}_j \,|\, \sigma_{R,wsi}(\mathbf{p}_j) \leq 0\ \forall\, i\}$, where $\sigma_{R,wsi}$ is the constraint function of the object representing the obstacle or the workspace limits, e.g., this function could be the negative value of the distance from position $\mathbf{p}_j$ to the boundary surface of an obstacle. Thus, the allowed C-space results in

[2]From (6), it follows that $\dot{\phi}_{R,qi}$ (and $\ddot{\mathbf{q}}$) explicitly depends on signal $\ddot{\mathbf{q}}_c$, i.e., the sliding manifold has relative degree one with respect to the discontinuous action $\mathbf{u}$, as required by SM theory (Edwards & Spurgeon, 1998).

$\Phi_{CS}(\mathbf{q}) = \{\mathbf{q} \,|\, \sigma_{R,wsi}(\mathbf{l}_j(\mathbf{q})) = \sigma_{R,wsij}(\mathbf{q}) \leq 0 \;\forall\, i,j\}$, where $\mathbf{l}_j$ is the kinematic function of the Cartesian position of point j.

As in Section 4.3.1, constraint functions $\sigma_{R,wsij}$ depend on the robot configuration $\mathbf{q}$ and, hence, they will be modified as follows for the sliding manifold to have relative degree one with respect to the control variable $\ddot{\mathbf{q}}_c$:

$$\begin{aligned}\phi_{R,wsij}(\mathbf{q},\dot{\mathbf{q}}) =& \sigma_{R,wsij}(\mathbf{q}) + K_{R,wsi}\frac{d\sigma_{R,wsij}(\mathbf{q})}{dt}\\ =& \sigma_{R,wsij} + K_{R,wsi}\,\nabla\sigma_{R,wsij}^{\mathrm{T}}\,\dot{\mathbf{q}} \leq 0,\end{aligned} \tag{21}$$

where $K_{R,wsi}$ is an arbitrary positive parameter that determines the rate of approach to the boundary surface of object i.

The infinite number of robot points to consider in the foregoing expression can be reduced to a set of *characteristic points* such that the distance from every point on the links boundary surface to the closest robot characteristic point is less than a selected value, which is used to enlarge the workspace constrained region. Regarding the workspace limits' constraint, typically only the robot end-effector is considered.

4.3.4. Equality for the robot constraints

The derivatives of the constraint functions $\phi_{R,qi}$, $\phi_{R,si}$, and $\phi_{R,wsij}$ are needed to compute the Lie derivatives $\{\mathbf{L_g}\phi_{R,qi}, \mathbf{L_g}\phi_{R,si}, \mathbf{L_g}\phi_{R,wsij}\}$ and $\{L_f\phi_{R,qi}, L_f\phi_{R,si}, L_f\phi_{R,wsij}\}$ with (16)–(17). From (18)–(21), these derivatives result in:

$$(\partial\phi_{R,qi}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{R,qi}^{\mathrm{T}} + K_{R,qi}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma R,qi} \tag{22}$$

$$(\partial\phi_{R,si}/\partial\mathbf{q})^{\mathrm{T}} = 0 \tag{23}$$

$$(\partial\phi_{R,wsij}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{R,wsij}^{\mathrm{T}} + K_{R,wsi}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma R,wsij} \tag{24}$$

$$(\partial\phi_{R,qi}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = K_{R,qi}\nabla\sigma_{R,qi}^{\mathrm{T}} = K_{R,qi}\begin{bmatrix}0 & \cdots & \mathrm{sign}(\tilde{q}_i) & \cdots & 0\end{bmatrix} \tag{25}$$

$$(\partial\phi_{R,si}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = \nabla\phi_{R,si}^{\mathrm{T}} = \begin{bmatrix}0 & \cdots & \mathrm{sign}(\tilde{\dot{q}}_i) & \cdots & 0\end{bmatrix} \tag{26}$$

$$(\partial\phi_{R,wsij}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = K_{R,wsi}\nabla\sigma_{R,wsij}^{\mathrm{T}} \tag{27}$$

where $\mathbf{H}_{\sigma R,qi}$, and $\mathbf{H}_{\sigma R,wsij}$ denote the Hessian matrix of second-order derivatives of $\sigma_{R,qi}$, and $\sigma_{R,wsij}$, respectively, sign$(\cdot)$ represents the *sign* function and only the ith element in (25) and (26) is not zero.

Therefore, according to (16) and (25)–(27), equation (13) for this level results in:

$$\begin{aligned}\begin{bmatrix}\mathbf{K}_{R,q}\,\nabla\boldsymbol{\sigma}_{R,q}^{\mathrm{T}}\\ \nabla\boldsymbol{\phi}_{R,s}^{\mathrm{T}}\\ \mathbf{K}_{R,ws}\,\nabla\boldsymbol{\sigma}_{R,ws}^{\mathrm{T}}\end{bmatrix} =& -\begin{bmatrix}\mathbf{1}_{b,q}u_{R,q}^{+}\\ \mathbf{1}_{b,s}u_{R,s}^{+}\\ \mathbf{1}_{b,ws}u_{R,ws}^{+}\end{bmatrix}\\ =& \,\mathbf{L_g}\boldsymbol{\phi}_R\ddot{\mathbf{q}}_c = -\,\mathbf{u}_R^{+},\end{aligned} \tag{28}$$

where $\mathbf{K}_{R,q}$,and $\mathbf{K}_{R,ws}$ are diagonal matrices with diagonal entries $K_{R,qi}$, and $K_{R,wsi}$, respectively; matrices $\{\nabla\boldsymbol{\sigma}_{R,q}, \nabla\boldsymbol{\phi}_{R,s}, \nabla\boldsymbol{\sigma}_{R,ws}\}$ contain the vectors $\{\nabla\sigma_{R,qi}, \nabla\phi_{R,si}, \nabla\sigma_{R,wsij}\}$, see (25)–(27), of all *active* constraints; $\{u_{R,q}^{+}, u_{R,s}^{+}, u_{R,ws}^{+}\}$ are the chosen value of switching gain for each type of robot constraint; and $\{\mathbf{1}_{b,q}, \mathbf{1}_{b,s}, \mathbf{1}_{b,ws}\}$ are column vectors with all its components equal to one and their size is equal to the number of active constraints of each type.

Thus, by comparing (28) with (7), it is obtained that $\mathbf{A}_1 = \mathbf{L_g}\boldsymbol{\phi}_R$ and $\mathbf{b}_1 = -\mathbf{u}_R^+$.

4.3.5. Gradient vectors for the robot constraints

According to (28), only the gradient vectors of the active constraints (i.e., those with $\phi_{R,i} \geq 0$) are required to compute the control action of the first level. In particular, the gradient vectors $\nabla\sigma_{R,qi}$ and $\nabla\phi_{R,si}$ for the joint range and joint speed constraints are straightforward obtained from (25) and (26), respectively, as:

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

$$\nabla\phi_{R,si} = \begin{bmatrix}0 & \cdots & \text{sign}(\widetilde{\dot{q}}_i) & \cdots & 0\end{bmatrix}^{\mathrm{T}}, \tag{30}$$

whereas the gradient vectors $\nabla\sigma_{R,wsj}$ for the workspace constraints are obtained as follows:

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

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.

4.4. Reference tracking

This work considers the classical operational space robot control (Siciliano, Sciavicco, Villani, & Oriolo, 2009) that, taking into account (4) and (6), is given by:

$$\mathbf{J}_s\ddot{\mathbf{q}}_c = \ddot{\mathbf{s}}_c - (\mathbf{J}_s\mathbf{d}_c + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{s}}/\partial t), \tag{32}$$

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

Furthermore, the classical acceleration-based kinematic controller is considered for trajectory tracking (Khalil & Dombre, 2002), that is:

$$\ddot{\mathbf{s}}_c = \ddot{\mathbf{s}}_{ref} - K_{T,p}\mathbf{e} - K_{T,v}\dot{\mathbf{e}}, \tag{33}$$

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

It is interesting to remark that the acceleration-based robot control given by (32)–(33) has already been used in VS applications by (Fakhry & Wilson, 1996) for PBVS and by (Keshmiri, Xie, & Mohebbi, 2014) for IBVS. Note that this control requires the discrete-time derivatives $\{\dot{s}, \partial\dot{\mathbf{s}}/\partial t, \dot{\mathbf{J}}_s\}$, which can be computed using numerical differentiation, e.g., the well-known backward Euler approximation, see the actual implementation of the Appendix. Note also that some kind of filtering should be previously applied to the actual variable when non-negligible noise is present.

4.4.1. Adaptive gain for the kinematic controller

In this work, the gains of the kinematic controller are selected as follows. On the one hand, the correction gain of the velocity error is chosen to obtain an overdamped response, i.e., $K_{T,v} > 2\sqrt{K_{T,p}}$. On the other hand, the correction gain of the position error is designed depending on the position error as follows:

$$K_{T,p}(\mathbf{e}) = K_{T,p}(\infty) - (K_{T,p}(\mathbf{0}) - K_{T,p}(\infty))e^{-\dfrac{\mathbf{e}\,\dot{K}_{T,p}(\mathbf{0})}{K_{T,p}(\mathbf{0}) - K_{T,p}(\infty)}}, \tag{34}$$

where the design parameters $K_{T,p}(\mathbf{0})$, $K_{T,p}(\infty)$ and $\dot{K}_{T,p}(\mathbf{0})$ represent the gain for zero error, the gain for infinite error, and the time-derivative of the gain for zero error, respectively. The advantage of this adaptive expression is that allows to use a smaller gain at the beginning when the initial error is large in order to obtain a smooth behavior and a larger gain at the end when the final error is small in order to achieve quickly the reference value.

4.4.2. Visual feature and Jacobian matrix

The typical visual feature vector $\mathbf{s}$ used in IBVS and PBVS (Chaumette & Hutchinson, 2008) are considered in this work.The Jacobian matrix $\mathbf{J}_s$ required for the control law (32) of the reference tracking is computed from (3) using the values of the interaction matrix $\mathbf{L}_s$, the transformation matrix ${}^c\mathbf{V}_e$ from the camera to the robot end-effector and the robot Jacobian ${}^e\mathbf{J}_e$. Matrix ${}^e\mathbf{J}_e$ is obtained from the robot model, ${}^c\mathbf{V}_e$ is updated with the pose estimation algorithms and the robot model, whereas the interaction matrix $\mathbf{L}_s$ represents the well-known interaction matrix typically used in VS (Chaumette & Hutchinson, 2008).

4.5. Chattering

Discrete-time implementations of the proposed 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* (Edwards & Spurgeon, 1998). The upper bound for the chattering band of the method can be obtained from the Euler-integration of (13), yielding:

$$\triangle\boldsymbol{\phi} = T_s\,|\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{u}_c| = T_s\,u^+\,\mathbf{1}_b, \tag{35}$$

where T_s is the sampling period of the robot system and the value of u^+ is $\{u^+_{R,q}, u^+_{R,s}, u^+_{R,o}, u^+_{R,ws}\}$ for the robot and workspace constraints. This chattering amplitude must be lower than the error allowed in the fulfillment of the constraints, i.e., the safety margins $\{m_{R,q}, m_{R,s}, m_{R,o}, m_{R,ws}\}$.

5. Additional remarks

5.1. Advantages and disadvantages of the proposal

Advantages of the proposed approach:

- Using visual servoing to close the control loop in the specific problem of tool changing increases the task robustness, in the sense that it can deal with possible misplacement of the tool with respect to the warehouse (e.g., due to mechanical friction, due to wear of the robot over time, etc.) or due to an unexpected position of the warehouse.

- Visual servoing increases the flexibility of the process, since it allows the system to deal with moving warehouses.
- SMC is used in a prioritized level to satisfy the robot constraints (range limits, speed limits and allowed workspace), whose main benefits are robustness and low computational cost, inherent to SMC approaches.

Main limitations of the method:

- 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* (Gracia et al., 2012). In general, these failure situations could be anticipated by evaluating a priori the robot task with the complete geometric data of the problem, if available. Moreover, in some cases, this failure situations can be avoided using high-level planning (Latombe, 1991), although its complexity and computational cost are substantially greater than those of the method proposed in this work, see the Appendix.
- Like other SMC applications, the proposed method has the *chattering* drawback, see Section 4.5. Nevertheless, the chattering problem becomes negligible for reasonable fast sampling rates, see (35).

5.2. Guidelines for the paramaters design

- *Safety margins for the constraints*: The value of $\{m_{R,q}, m_{R,s}, m_{R,ws}\}$ should be as small as possible in order to fully utilize the available allowed space (e.g., the ellipsoid representing the workspace limits for the workspace constraint), but not too small to cater for possible errors and inaccuracies (SM chattering band, modeling errors, robot control inaccuracies, etc.) in order to avoid accidentally exceeding the boundary of the allowed space.
- *Constraint approaching parameters*: The value of constraint approaching parameters $\{K_{R,qi}, K_{R,wsi}\}$ for the constraints (join limits and workspace constraints), can be seen as the *time constant* of the braking process when approaching the boundary of the original constraints σ_i. Hence, when approaching the constraint boundary at high velocity, it is reached in approximately $3K_i$ seconds and the velocity perpendicular to the constraint boundary is also reduced to zero after that time.
- *Control action amplitude*: The value of $\{u^+_{R,q}, u^+_{R,s}, u^+_{R,ws}\}$ should be as close as possible to the lower bound given by (14) to have reduced chattering band and high chattering frequency (Section 4.5).
- *Sampling time*: The sampling time T_s should be small enough to have small chattering band (35). The minimum possible value is determined by the computation time of one iteration of the proposed algorithm, which is around 15 microseconds for the case study in Section 6 (see the Appendix).

6. Simulation

A three-dimensional (3D) case study is presented in this section to demonstrate the general feasibility and effectiveness of the proposed method. The proposal is tested for *eye-to-hand* camera-robot-warehouse configuration and IBVS, although other VS controllers could be considered. The simulation results presented in this section were obtained using MATLAB®. Details of pseudo-code and computing time for actual implementation of the proposed strategy appears in an Appendix at the end of the paper.

In the proposed case study, a classical 6R serial manipulator with spherical wrist in ceiling position is considered. Fig. 3 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, camera frame C, object frame O, initial object frame O, desired object frame O^{*1} for the first phase (positioning task) and desired object frame O^{*2} for the second phase (tracking task). The desired trajectory for the moving object during the tracking phase is also represented in Fig. 3 as a dotted line.

The robot Jacobian ${}^{e}\mathbf{J}_e$ can be readily obtained (Siciliano et al., 2009) taking into account the Denavit-Hartenberg parameters of the 6R robot shown in Table 1.

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 1.: Denavit-Hartenberg parameters for the 6R robot.

The constraint functions $\sigma_{R,wso}$ for the ellipsoid representing a forbidden area and $\sigma_{R,wsl}$ for the ellipsoid representing the workspace limits are given by:

$$\sigma_{R,wso}(\mathbf{p}_j) = 1 - \left\| \begin{bmatrix} \frac{x_j - x_o}{r_{ox}} & \frac{y_j - y_o}{r_{oy}} & \frac{z_j - z_o}{r_{oz}} \end{bmatrix}^{\mathrm{T}} \right\|_2 \tag{36}$$

$$\sigma_{R,wsl}(\mathbf{p}_j) = 1 - \left\| \begin{bmatrix} \frac{x_j - x_l}{r_{lx}} & \frac{y_j - y_l}{r_{ly}} & \frac{z_j - z_l}{r_{lz}} \end{bmatrix}^{\mathrm{T}} \right\|_2 , \tag{37}$$

where $\{\mathbf{r}_o, \mathbf{r}_l\}$ and $\{\mathbf{p}_o, \mathbf{p}_l\}$ are the radii and centers, respectively, of the ellipsoids, $m_{R,wso}$ and $m_{R,wsl}$ are the safety margins for the constraints and $\mathbf{p}_j$ is the Cartesian position of the considered point of the robot.

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

$$\frac{\partial \sigma_{R,wso}}{\partial \mathbf{p}_j} = -\frac{1}{1 - \sigma_{R,wso}(\mathbf{p}_j)} \begin{bmatrix} \frac{x_j - x_o}{r_{ox}^2} & \frac{y_j - y_o}{r_{oy}^2} & \frac{z_j - z_o}{r_{oz}^2} \end{bmatrix}^{\mathrm{T}} \tag{38}$$

$$\frac{\partial \sigma_{R,wsl}}{\partial \mathbf{p}_j} = -\frac{1}{1 - \sigma_{R,wsl}(\mathbf{p}_j)} \begin{bmatrix} \frac{x_j - x_l}{r_{lx}^2} & \frac{y_j - y_l}{r_{ly}^2} & \frac{z_j - z_l}{r_{lz}^2} \end{bmatrix}^{\mathrm{T}} . \tag{39}$$

Simulation was run under the following conditions:

i) Parameters used for the camera: focal lengths $f_x = 640$ and $f_y = 480$ pixels;

principal point $[u_0, v_0] = [320, 240]$.

ii) Parameters used for the joint limit constraints: safety margin $m_{R,q} = 0$; constraint approaching parameter $K_{R,qi} = 0.2$; control action amplitude $u^+_{R,q} = 50$; mid position and maximum range of motion for the fifth joint $q_{\text{mid},5} = -1.16$ rad and $\Delta q_{\text{max},5} = 0.46$ rad, respectively. The range limit constraint for the remaining joints are omitted for simplicity.

iii) Parameters used for the joint speed constraints: safety margin $m_{R,s} = 0$; control action amplitude $u^+_{R,s} = 10$; and maximum speed for all the joints $\dot{q}_{\text{max},i} = 0.7$ rad/s.

iv) Parameters used for the robot workspace limits constraint: safety margin $m_{R,wsl} = 0$; constraint approaching parameter $K_{R,wsl} = 0.3$; control action amplitude $u^+_{R,wsl} = 8$; radius of the ellipsoid object $\mathbf{r}_l = \begin{bmatrix} 0.75 & 0.75 & 0.80 \end{bmatrix}^{\mathrm{T}}$ m; and center of the ellipsoid object $\mathbf{p}_l = \begin{bmatrix} 0 & 0 & -0.6 \end{bmatrix}^{\mathrm{T}}$ m.

v) Parameters used for the robot constraint for collision avoidance: safety margin $m_{R,wso} = 0$; constraint approaching parameter $K_{R,wso} = 0.1$; control action amplitude $u^+_{R,wso} = 15$; radius of the ellipsoid object $\mathbf{r}_o = \begin{bmatrix} 0.20 & 0.50 & 0.20 \end{bmatrix}^{\mathrm{T}}$ m; and center of the ellipsoid object $\mathbf{p}_o = \begin{bmatrix} 0.34 & -0.62 & -0.95 \end{bmatrix}^{\mathrm{T}}$ m.

For simplicity, only the Cartesian position $\mathbf{p}_e$ of the robot end-effector will be evaluated as point $\mathbf{p}_j$ in the constraint for collision avoidance and in the workspace limits constraint. The Jacobian matrix ${}^0\mathbf{J}_{pe}$ for this point, which is needed to compute the gradient vectors in (31), can be readily obtained (Siciliano et al., 2009) taking into account the Denavit-Hartenberg parameters of the 6R robot shown in Table 1.

vi) Parameters used for the kinematic controller (33)-(34): correction gain for the velocity error $K_{T,v} = 3\sqrt{K_{T,p}}$; parameters of the adaptive position gain for the positioning phase $K_{T,p}(\mathbf{0}) = 10$, $K_{T,p}(\infty) = 0$ and $\dot{K}_{T,p}(\mathbf{0}) = 20$; and parameters of the adaptive position gain for the tracking phase $K_{T,p}(\mathbf{0}) = 50$, $K_{T,p}(\infty) = 1$ and $\dot{K}_{T,p}(\mathbf{0}) = 10$.

vii) The initial configuration considered for the robot is given by the joint position vector $\mathbf{q}(0) = \begin{bmatrix} -0.87 & -0.83 & 2.30 & -0.87 & -1.16 & -1.27 \end{bmatrix}^{\mathrm{T}}$ rad, yielding an initial robot pose given by the transformation matrix ${}^F\mathbf{M}_E(0) = \begin{bmatrix} 0.43 & -0.32 & -0.26 & -1.7741 & 0.1006 & -1.8989 \end{bmatrix}^{\mathrm{T}}$, where it has been used the following *compact notation* for the homogeneous transformation matrix: the first three elements are the Cartesian coordinates in meters and the last three elements are the roll, pitch and roll angles, respectively, in radians.

viii) The target object has four markers given by the following points with respect to the object frame: ${}^O\mathbf{p}_1 = \begin{bmatrix} -0.03 & -0.03 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^O\mathbf{p}_2 = \begin{bmatrix} 0.03 & -0.03 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^O\mathbf{p}_3 = \begin{bmatrix} 0.03 & 0.03 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^O\mathbf{p}_4 = \begin{bmatrix} -0.03 & 0.03 & 0 \end{bmatrix}^{\mathrm{T}}$ m, that is, the four markers are the vertices of a square with a side length of 6 centimeters.

ix) The object is positioned in the tool, and its position and orientation with respect to the end-effector frame is given by the transformation matrix ${}^E\mathbf{M}_O = \begin{bmatrix} 0 & 0 & 0.1 & 0 & -\pi/2 & 0 \end{bmatrix}^{\mathrm{T}}$ in compact notation.

x) The desired object frame for the positioning task (first phase of the simulation) is given by the transformation matrix ${}^F\mathbf{M}_{O^{*1}} = \begin{bmatrix} 0.4 & 0 & -1.09 & -\pi/2 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ in compact notation. For the tracking task (second phase of the simulation), a moving target object is considered, starting from ${}^F\mathbf{M}_{O^{*1}}$ and with the circular trajectory given by the transformation matrix ${}^F\mathbf{M}_{O^{*2}}(t) =$

$\begin{bmatrix} x_c + r\cos(t+\alpha) & y_c + r\sin(t+\alpha) & -1.09 & -\pi/2 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ in compact notation, where $[x_c, y_c] = [0.412, 0.222]$ m is the center, $r = 0.23$ m is the radius, and $\alpha = 265$ is the phase of the circular trajectory. The analytical expression of the trajectory is used to update $\partial\mathbf{s}/\partial t$.

xi) The algorithm was computed with a sampling time T_s of 1 milliseconds and the disturbance vector $\mathbf{d}_c$ has been considered zero.

The results of the simulation are depicted at different figures.

Fig. 4 shows that the error is made zero for both phases. Note that the positioning task ends at around time instant 6.5 s, and the error during the tracking task is affected because of the robot workspace constraint.

Fig. 5 shows that: all the constraints are fulfilled, i.e., $\max(\phi_i) \leq 0$ (see third plot); the joint limit constraint for the fifth joint (dark line in the first plot) becomes active during the first phase; the speed constraint becomes active during both phases and for up to five different joints (see second plot); the constraint for collision avoidance becomes active around time interval 3s-4s (see fourth plot) during the first phase; and the workspace constraint becomes active during the both phases (see fourth plot). It is interesting to remark that in some phases of the simulation there are up to three active constraints at a time.

Fig. 6 shows the trajectories followed by the visual features in the image plane. Finally, Fig. 7 depicts six snapshot frames of a 3D representation of the robot at different time instants, whereas a detail view of the ellipsoid obstacle is shown in Fig. 8, where it can be seen that the constraint for collision avoidance is fulfilled.

A video of this simulation can be played at https://media.upv.es/player/?id=28f5b5a0-17f4-11e7-a875-bd62e853f1c3.

7. Real experimentation

Real experiments for automatic tool change with a 6R industrial manipulator are presented in this section to demonstrate the applicability of the proposed method. The experimentation has been carried out with the following setup (see Fig. 9): a Kuka Agilus R900 sixx manipulator in ceiling position (i.e., the same robot simulated above) equipped with a robot controller that allows external real-time communication using the Ethernet UDP protocol; a general purpose web cam for image acquisition; and an external PC with Ubuntu 12.04 OS prompted with real time kernel that implements the computer vision algorithms and the control algorithms proposed in this work. The position of the markers are updated using the dot tracker in ViSP (Visual Servoing Platform) (Marchand, Spindler, & Chaumette, 2005).

Two different experiments are conducted: the first one uses a static warehouse, whilst the second one uses a moving warehouse. For the first one, IBVS is considered with the eye-to-hand configuration. IBVS has its main advantage in the fact that it is inherently robust to camera calibration and target modeling errors (Hutchinson, Hager, & Corke, 1996). This is suitable for compensating robot positioning errors with respect to an static warehouse, which reference features are previously computed by the well-known *teaching-by-showing* method (Chaumette & Hutchinson, 2008; Hutchinson et al., 1996). However, in the case of a moving warehouse, this method is not feasible in practice and therefore PBSV has to be used. Hence, the second experiment with a moving warehouse considers PBVS to overcome the aforementioned IBVS limitation.

The experiments were run under the following conditions:

i) Parameters used for the kinematic controller: correction gain for the position error $K_{T,p} = 0.025$ and velocity error $K_{T,v} = 3\sqrt{K_{T,p}}$.

ii) The control period T_s is set to 0.1 seconds due to the requirements of image acquisition and processing.

iii) 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.

iv) Four markers define the warehouse and the tool, representing the vertices of a square with a side length of 17 centimeters in both cases.

v) For the first experiment, an obstacle has been considered and the robot constraint used for collision avoidance has the following parameters: safety margin $m_{R,wso} = 0.1$; constraint approaching parameter $K_{R,wso} = 0.1$; control action amplitude $u^+_{R,wso} = 15$; an ellipsoid enveloping the obstacle is defined with the following parameters: radius $\mathbf{r}_o = \begin{bmatrix} 0.30 & 0.45 & 0.60 \end{bmatrix}^{\mathrm{T}}$ m; and center of the ellipsoid obstacle $\mathbf{p}_o = \begin{bmatrix} 0.20 & 0.40 & -1.15 \end{bmatrix}^{\mathrm{T}}$ m. Note that this ellipsoid is known a priori and is used to represent the obstacle in order to illustrate the behavior of the proposed visual servoing with constraints. In general, the online detection and accurate 3D positioning of the obstacles in the environment may require additional sensing and algorithms. However, this is out of the scope of this work.

   As in the simulation case, only the Cartesian position $\mathbf{p}_e$ of the robot end-effector will be evaluated as point $\mathbf{p}_j$ in the constraint for collision avoidance. The Jacobian matrix ${}^0\mathbf{J}_{pe}$ for this point, which is needed to compute the gradient vectors in (31), can be readily obtained (Siciliano et al., 2009) taking into account the Denavit-Hartenberg parameters of the 6R robot shown in Table 1.

In the first experiment, two cases are considered: the first one uses no algorithm to avoid the obstacle, while the second one uses the proposed SMC method to satisfy constraints. The video for both cases can be played at (video at double speed) `https://media.upv.es/player/?id=934ea160-14a6-11e7-a875-bd62e853f1c3`. Fig. 10 shows that the error in the second case is made zero, thus the VS task is accomplished, and subsequently the robot places the tool in the warehouse, see the video mentioned above. Fig. 11 shows the trajectories followed by the visual features in the image plane. Note that, although the dot tracker identifies the markers in the warehouse, this information is not used in this experiment. The ending of the servoing is defined in the image using the *teaching-by-showing* method, as aforementioned. Fig. 12 shows that the robot workspace constraint becomes active around time interval 5s–12s, whereas Fig. 13 shows a detail view of the ellipsoid defined to envelope the obstacle and how the constraint for collision avoidance is fulfilled.

It is interesting to remark that, despite that the sampling time of the real platform used for experimentation is not very small, 0.1 s, the performance of the proposed SMC algorithm is satisfactory.

The second experiment has been conducted to illustrate the flexibility of the proposed method with a moving warehouse. This experiment shows how the robot successfully follows the warehouse and performs the auto tool change task. The video of this experiment can be played at `https://media.upv.es/player/?id=a1770050-bbdc-11e8-a361-599725480ca3`. Fig. 14 shows several frames from the video: Fig. 14(a-c) (interval 10s–49s in the video) show how the robot follows the moving warehouse; Fig. 14(d-e) (interval 50s–1m07s in the video) correspond to the process of placing the first tool in the warehouse; Fig. 14(f-i) (interval 1m08s–1m33s in the video) show how the warehouse is moved again before taking the second tool; and Fig. 14(j-k) (interval 1m34s–1m55s in the video) show how the tool change is successfully completed.

8. Conclusions

An automated approach for tool changing in industrial robots using visual servoing and sliding mode control has been presented. In particular, the main task used image-based visual servoing to properly place the tool in the warehouse, whereas sliding mode control was used in a prioritized level to satisfy the robot constraints (range limits, speed limits and allowed workspace).

The robustness and flexibility achieved with the proposed method is mainly due to the control law of the image-based visual servoing, which uses the information acquired by a vision system to close a feedback control loop. The proposed approach only requires a few program lines and has a short computation load, see the Appendix.

The feasibility and effectiveness of the proposed approach was illustrated in simulation for a complex 3D case study. Furthermore, the applicability of the method was demonstrated with real experimentation using a conventional 6R industrial manipulator for tool changing.

Appendix. 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 robot constraints: $\{\phi_{R,qi}(\mathbf{q},\dot{\mathbf{q}}),\phi_{R,si}(\dot{\mathbf{q}}),\phi_{R,wsij}(\mathbf{q},\dot{\mathbf{q}})\}$ and $\{\nabla\sigma_{R,qi}(\mathbf{q}),\nabla\phi_{R,si}(\dot{\mathbf{q}}),\nabla\sigma_{R,wsij}(\mathbf{q})\}$.
- 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 (Section 3.1): $(\cdot)^{\dagger}$.
- 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 15 microseconds for the case study example in Section 6.

```
Algorithm executed at sampling time of $T_s$ seconds
1  while $s < s_{end}$ do
2      $[\mathbf{q}, \dot{\mathbf{q}}]$ =GetRobotState();
3      $\dot{\mathbf{s}} = (\mathbf{s} - \mathbf{s}_{prev})/T_s$ ;                    // Discrete-time derivative
4      $\dot{\mathbf{s}}_{ref} = (\mathbf{s}_{ref} - \mathbf{s}_{ref,prev})/T_s$ ;           // Discrete-time derivative
5      $\ddot{\mathbf{s}}_{ref} = (\dot{\mathbf{s}}_{ref} - \dot{\mathbf{s}}_{ref,prev})/T_s$ ;           // Discrete-time derivative
6      $\dot{\mathbf{J}}_s = (\mathbf{J}_s - \mathbf{J}_{s,prev})/T_s$ ;                 // Discrete-time derivative
7      $\ddot{\mathbf{s}}_c = \ddot{\mathbf{s}}_{ref} - K_{T,p}(\mathbf{s} - \mathbf{s}_{ref}) - K_{T,v}(\dot{\mathbf{s}} - \dot{\mathbf{s}}_{ref})$ ;          // Eq. (33)
8      $\mathbf{A}_1 = \begin{bmatrix} \mathbf{K}_{R,q}\,\nabla\boldsymbol{\sigma}_{R,q}^{\mathrm{T}} \\ \nabla\boldsymbol{\phi}_{R,s}^{\mathrm{T}} \\ \mathbf{K}_{R,ws}\,\nabla\boldsymbol{\sigma}_{R,ws}^{\mathrm{T}} \end{bmatrix}$ with the gradients of all active constraints:
         $\phi_{R,qi} > 0,\ \phi_{R,si} > 0,\ \phi_{R,wsij} > 0$ ;                  // Eq. (28)
9      $\mathbf{b}_1 = -\mathbf{u}_R^+$ ;                                       // Eq. (28)
10     $\mathbf{A}_2 = \mathbf{J}_s$ ;                                         // Eq. (32)
11     $\mathbf{b}_2 = \ddot{\mathbf{s}}_c - \dot{\mathbf{J}}_s\dot{\mathbf{q}} - \partial\dot{\mathbf{s}}/\partial t$ ;                     // Eq. (32)
12     $\ddot{\mathbf{q}}_{c,1} = \mathbf{A}_1^{\dagger}\mathbf{b}_1$ ;                                  // Eq. (8), $i = 1$
13     $\mathbf{N}_1 = \mathbf{I} - \mathbf{A}_1^{\dagger}\mathbf{A}_1$ ;                               // Eq. (8), $i = 1$
14     $\ddot{\mathbf{q}}_{c,2} = \ddot{\mathbf{q}}_{c,1} + (\mathbf{A}_2\mathbf{N}_1)^{\dagger}(\mathbf{b}_2 - \mathbf{A}_2\ddot{\mathbf{q}}_{c,1})$ ;         // Eq. (8), $i = 2$
15     SendToJointControllers($\ddot{\mathbf{q}}_{c,2}$);
16     $\mathbf{s}_{prev} = \mathbf{s}$ ;                                    // For next iteration
17     $\mathbf{s}_{ref,prev} = \mathbf{s}_{ref}$ ;                               // For next iteration
18     $\dot{\mathbf{s}}_{ref,prev} = \dot{\mathbf{s}}_{ref}$ ;                               // For next iteration
19     $\mathbf{J}_{s,prev} = \mathbf{J}_s$ ;                                  // For next iteration
20 end
```

Acknowledgements

This work was supported in part by the Spanish Government under Grant BES-2010-038486 and Project DPI2017-87656-C2-1-R.

References

Baumann, M., Léonard, S., Croft, E. a., & Little, J. J. (2010, feb). Path Planning for Improved Visibility Using a Probabilistic Road Map. *IEEE Trans. on Robotics*, *26*(1), 195–200.

Bi, Z. M., & Zhang, W. J. (2001, jan). Flexible fixture design and automation: Review, issues and future directions. *International Journal of Production Research*, *39*(13), 2867–2894.

Chaumette, F. (1998). Potential problems of stability and convergence in image-based and position-based visual servoing. In D. J. Kriegman, G. D. Hager, & A. S. Morse (Eds.), *The confluence of vision and control* (Vol. 237, pp. 66–78). London: Springer London.

Chaumette, F., & Hutchinson, S. (2008). Visual servoing and visual tracking. *Springer Handbook of robotics*, 563-583.

Chen, J., Dawson, D. M., Dixon, W. E., & Chitrakaran, V. K. (2007). Navigation function-based visual servo control. *Automatica*, *43*(7), 1165–1177.

Chesi, G., Hashimoto, K., Prattichizzo, D., & Vicino, A. (2004, oct). Keeping Features in the Field of View in Eye-In-Hand Visual Servoing: A Switching Approach. *IEEE Trans. on Robotics*, *20*(5), 908–913.

Chiaverini, S., Oriolo, G., & Walker, I. (2008). Kinematically redundant manipulators. *Springer Handbook of Robotics*, 245-268.

Corke, P. (2011). *Robotics, vision and control: Fundamental algorithms in MATLAB.* Berlin, Germany: Springer-Verlag.

Corke, P., & Hutchinson, S. (2001). A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, *17*(4), 507–515.

Dantzig, G. (2016). *Linear programming and extensions.* Princeton university press.

Du, G., & Zhang, P. (2013, December). Online robot calibration based on vision measurement. *Robot. Comput.-Integr. Manuf.*, *29*(6), 484–492.

Edwards, C., & Spurgeon, S. (1998). *Sliding mode control: Theory and applications* (1st ed.). UK: Taylor & Francis.

Fakhry, H., & Wilson, W. (1996). A modified resolved acceleration controller for position-based visual servoing. *Mathematical and Computer Modelling*, *24*(5-6), 1-9.

Gans, N. R., & Hutchinson, S. a. (2007, jun). Stable Visual Servoing Through Hybrid Switched-System Control. *IEEE Trans. on Robotics*, *23*(3), 530–540.

Garelli, F., Mantz, R., & De Battista, H. (2011). *Advanced control for constrained processes and systems.* London, UK: IET, Control Engineering Series.

Golub, G., & Van Loan, C. (1996). *Matrix computations* (3rd ed.). Baltimore, MD: The Johns Hopkins University InPress.

Gordic, Z., & Ongaro, C. (2016). Calibration of robot tool centre point using camera-based system. *Serbian Journal of Electrical Engineering*, *13*(1), 9–20.

Gracia, L., Garelli, F., & Sala, A. (2013). Reactive sliding-mode algorithm for collision avoidance in robotic systems. *IEEE Transactions on Control Systems Technology*, *21*(6), 2391-2399.

Gracia, L., Sala, A., & Garelli, F. (2012). A path conditioning method with trap avoidance. *Robotics and Autonomous Systems*, *60*(6), 862-873.

Hafez, A. H. A., & Jawahar, C. (2007, apr). Visual Servoing by Optimization of a 2D/3D Hybrid Objective Function. In *Proceedings 2007 ieee int. conference on robotics and automation* (pp. 1691–1696). IEEE.

Hashemi, H., Shaharouna, A. M., Sudin, I., Bijan, G., Zafar, N., & Saeed, S. (2014). Fixture design automation and optimization techniques: review and future trends. *International Journal of Engineering*, *27*(11).

Huang, Y., Zhang, X., & Fang, Y. (2014). Vision-based minimum-time planning of mobile robots with kinematic and visibility constraints. *IFAC Proceedings Volumes*, *47*(3), 11878–11883.

Hutchinson, S., Hager, G. D., & Corke, P. I. (1996, Oct). A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, *12*(5), 651-670.

Kazemi, M., Gupta, K. K., & Mehrandezh, M. (2013, oct). Randomized Kinodynamic Planning for Robust Visual Servoing. *IEEE Trans. on Robotics*, *29*(5), 1197–1211.

Kermorgant, O., & Chaumette, F. (2011, sep). Combining IBVS and PBVS to ensure the visibility constraint. In *2011 ieee/rsj int. conference on intelligent robots and systems* (pp. 2849–2854). IEEE.

Keshmiri, M., Xie, W., & Mohebbi, A. (2014). Augmented image-based visual servoing of a manipulator using acceleration command. *IEEE Transactions on Industrial Electronics*, *61*(10), 5444-5452.

Khalil, W., & Dombre, E. (2002). *Modeling, identification and control of robots.* Bristol, PA: Taylor & Francis Inc.

Kim, D., Lovelett, R., Wang, Z., & Behal, A. (2009). A region-based switching scheme for practical visual servoing under limited FOV and dynamically changing features. In *Iasted 14th int. conf. robotic applications.*

Latombe, J.-C. (1991). *Robot motion planning.* Boston: Kluwer.

Marchand, E., Spindler, F., & Chaumette, F. (2005, dec). ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics & Automation Magazine*, *12*(4), 40–52.

Menani, K., Mohammadridha, T., Magdelaine, N., Abdelaziz, M., & Moog, C. (2017). Positive sliding mode control for blood glucose regulation. *International Journal of Systems Science*, *48*(15), 3267-3278.

Mobayen, S., Tchier, F., & Ragoub, L. (2017). Design of an adaptive tracker for n-link rigid robotic manipulators based on super-twisting global nonlinear sliding mode control. *International Journal of Systems Science*, *48*(9), 1990-2002.

Motta, J. M. S., de Carvalho, G. C., & McMaster, R. (2001). Robot calibration using a 3d vision-based measurement system with a single camera. *Robotics and Computer-Integrated Manufacturing*, *17*(6), 487 - 497.

Muñoz-Vázquez, A., Parra-Vega, V., & Sánchez-Orta, A. (2017). A novel continuous fractional sliding mode control. *International Journal of Systems Science*, *48*(13), 2901-2908.

Nakamura, Y., Hanafusa, H., & Yoshikawa, T. (1987). Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, *6*(2), 3-15.

Ouyang, P., Zhang, W., Gupta, M. M., & Zhao, W. (2007, dec). Overview of the development of a visual based automated bio-micromanipulation system. *Mechatronics*, *17*(10), 578–588.

Pehlivan, S., & Summers, J. D. (2008). A review of computer-aided fixture design with respect to information support requirements. *International Journal of Production Research*, *46*(4), 929–947.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, planning and control*. London, UK: Springer-Verlag.

Siciliano, B., & Slotine, J. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *Proceedings of the fifth international conference on advanced robotics (icar'91)* (pp. 1211-1216, 1991). Pisa, Italy.

Utkin, V., Guldner, J., & Shi, J. (2009). *Sliding mode control in electro-mechanical systems* (2nd ed.). London: Taylor & Francis.

Yin, S., Ren, Y., Zhu, J., Yang, S., & Ye, S. (2013). A vision-based self-calibration method for robotic visual inspection systems. *Sensors (Basel, Switzerland)*, *13*(12), 16565–16582.

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

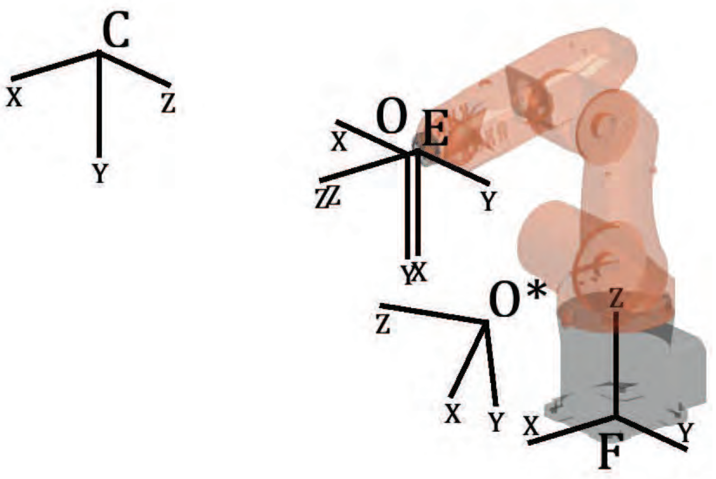


Figure 1.: Frames involved in eye-to-hand visual servoing.

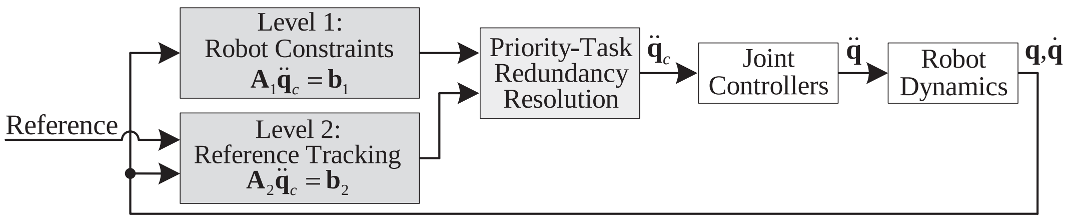

Level 1:
Robot Constraints
$\mathbf{A}_1\ddot{\mathbf{q}}_c = \mathbf{b}_1$
Level 2:
Reference Tracking
$\mathbf{A}_2\ddot{\mathbf{q}}_c = \mathbf{b}_2$
Reference
Priority-Task Redundancy Resolution
$\ddot{\mathbf{q}}_c$
Joint Controllers
$\ddot{\mathbf{q}}$
Robot Dynamics
$\mathbf{q},\dot{\mathbf{q}}$

Figure 2.: Overview of the proposed approach.

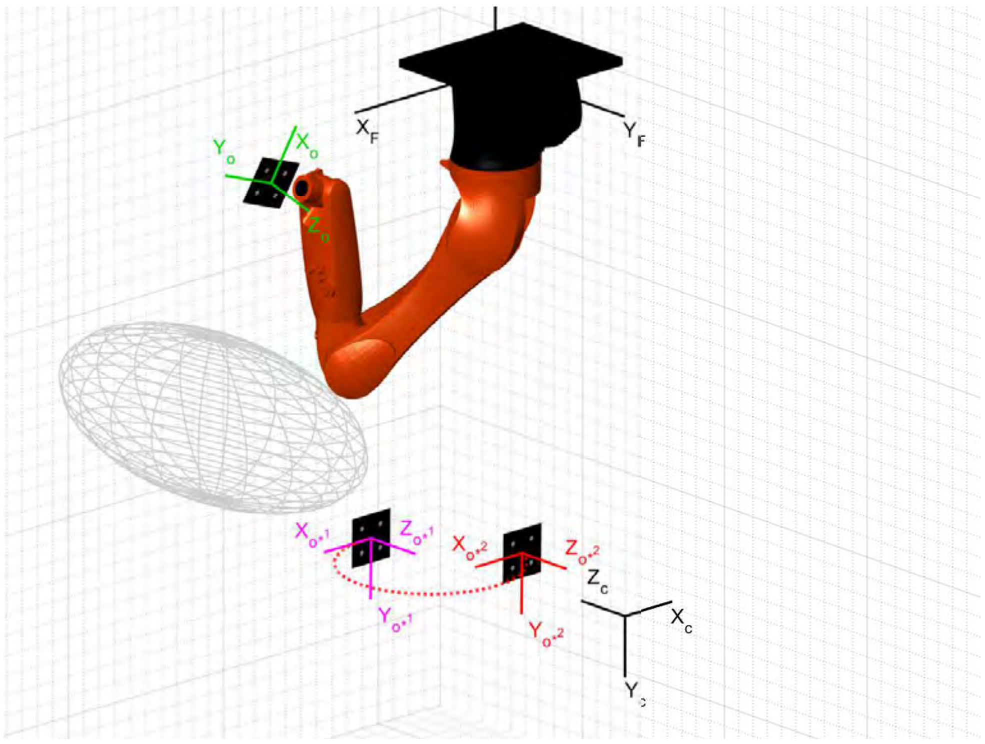


Figure 3.: System used for 3D simulation: 6R robot, target object with four markers, sphere representing a forbidden area and coordinate frames.

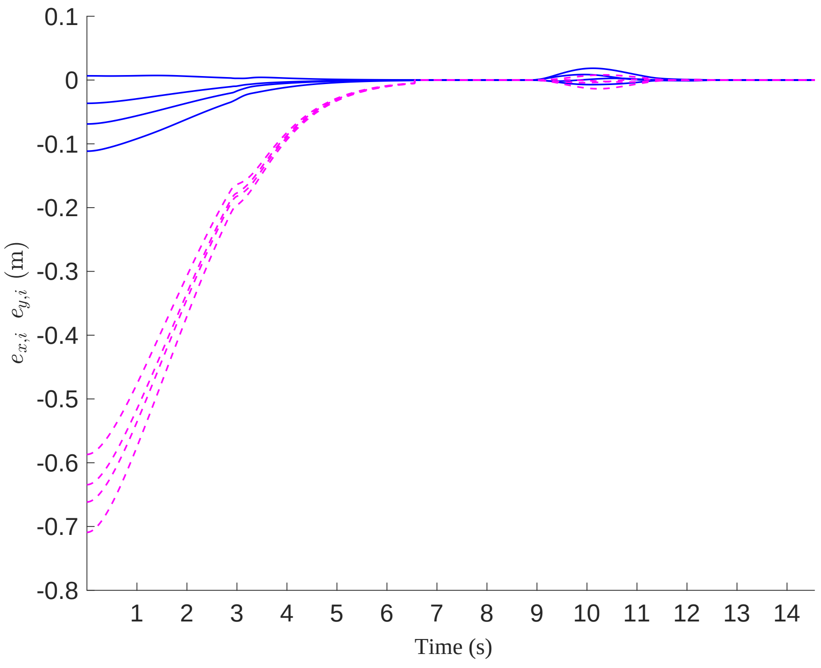


Figure 4.: Visual features error: $e_{x,i}$ (solid, blue) and $e_{y,i}$ (dashed, magenta).

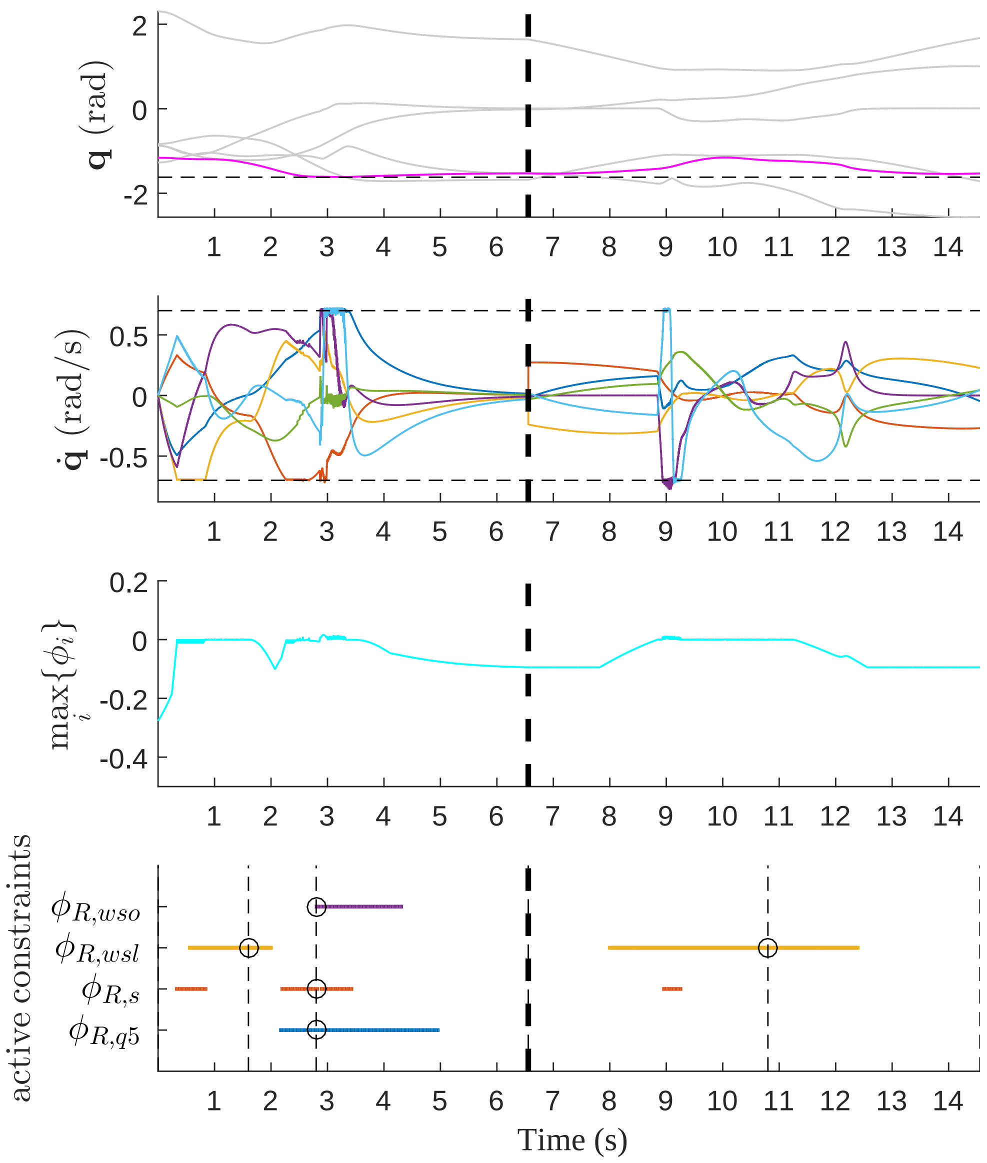


Figure 5.: From top to bottom plots: (1) joint postions $\mathbf{q}$ (the horizontal dashed line represents the joint limit for the active constraint, which is depicted with a dark line); (2) joint speeds $\dot{\mathbf{q}}$ (the horizontal dashed lines represent the speed limit for all the joints); (3) 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. 7 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 ${}^{F}\mathbf{M}_{C^{*1}}$ to ${}^{F}\mathbf{M}_{C^{*2}}$.

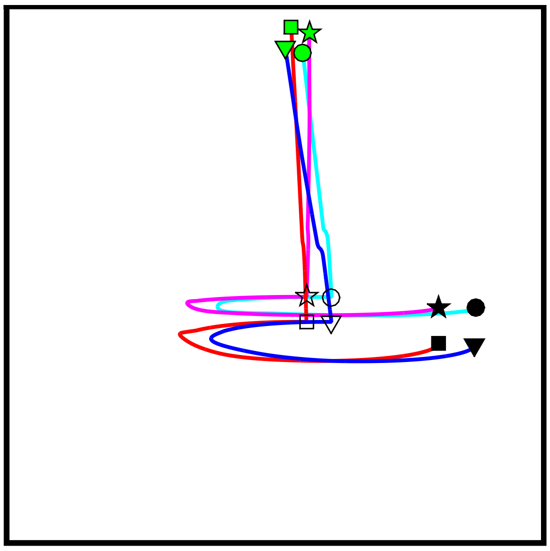


Figure 6.: Feature trajectories in the image plane for the case study.

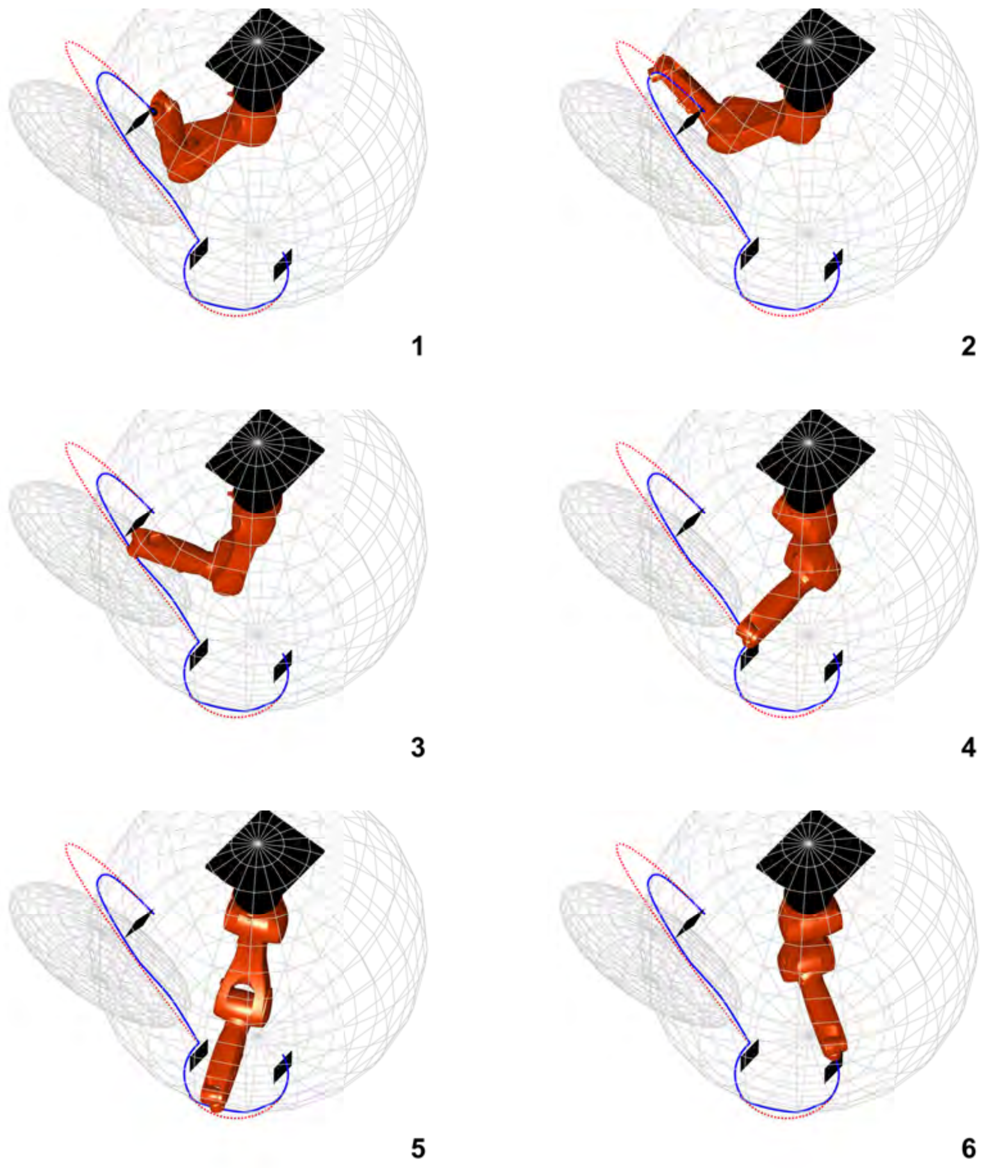


Figure 7.: Sequence of frames (frames 1 to 6) showing the robot configuration during the simulation, the actual path follwed by the robot end-effector (solid, blue) and the ideal path for no constraints (dotted, red). The active constraints at each frame are shown in Fig. 5.

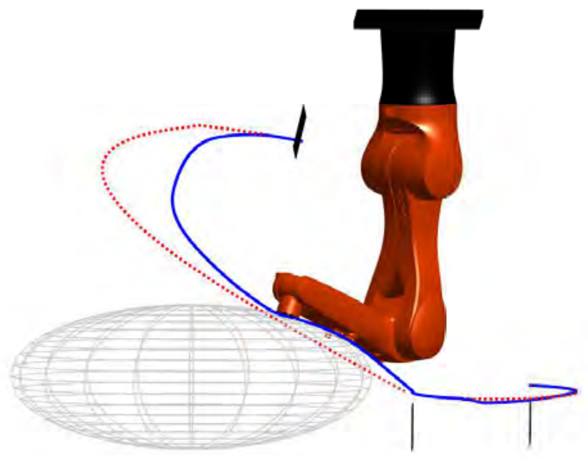


Figure 8.: Detailed view of the fulfillment of the constraint for the ellipsoid obstacle.

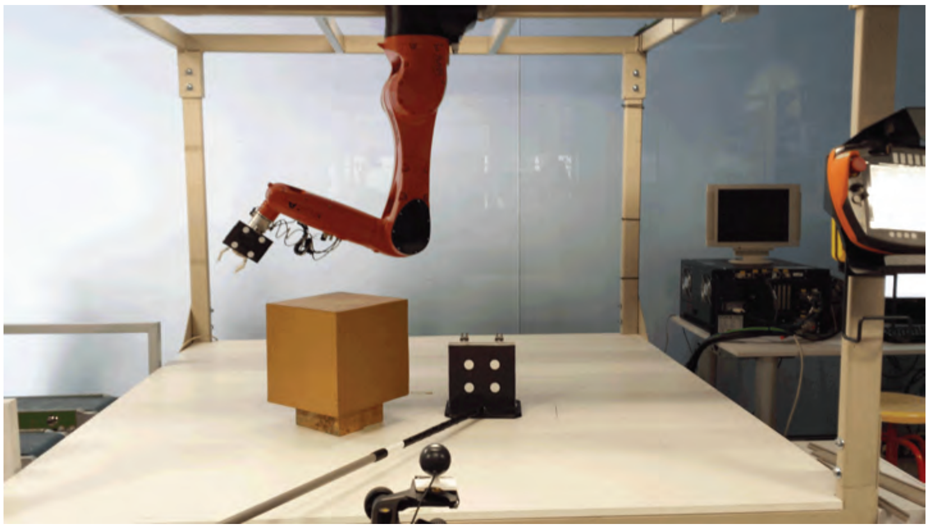


Figure 9.: Experimental setup: 6R serial industrial manipulator in ceiling position with markers in the tool and the warehouse, camera out of the robot (eye-to-hand), obstacle and robot PC.

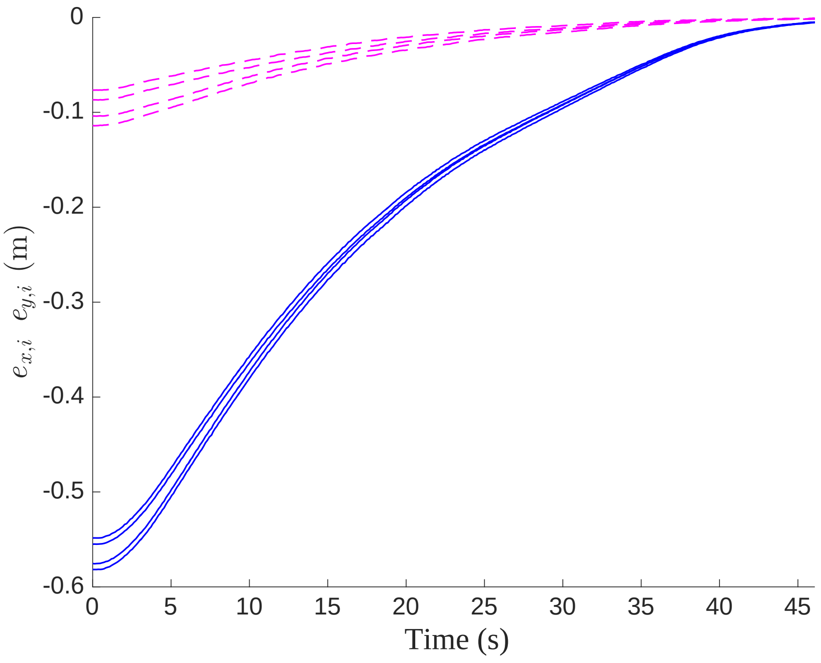


Figure 10.: Visual features error: $e_{x,i}$ (solid, blue) and $e_{y,i}$ (dashed, magenta).

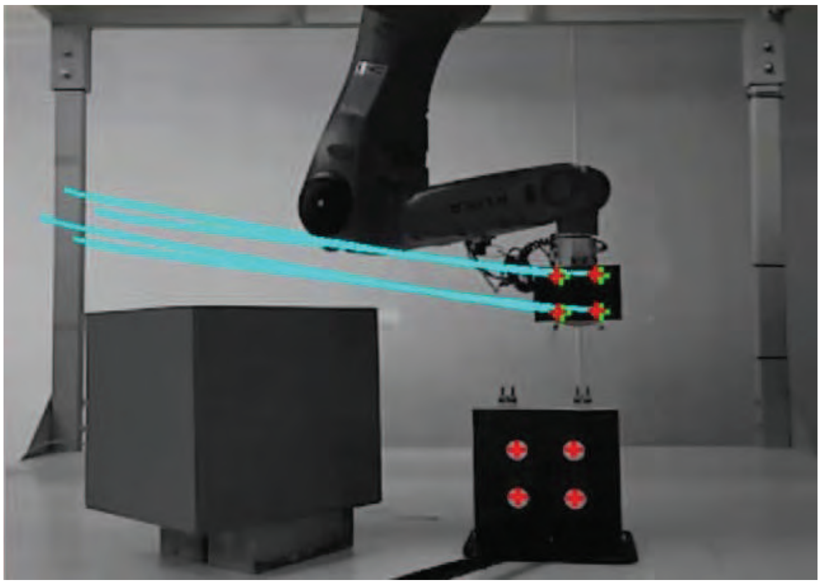


Figure 11.: Feature trajectories in the image plane for the first experiment.

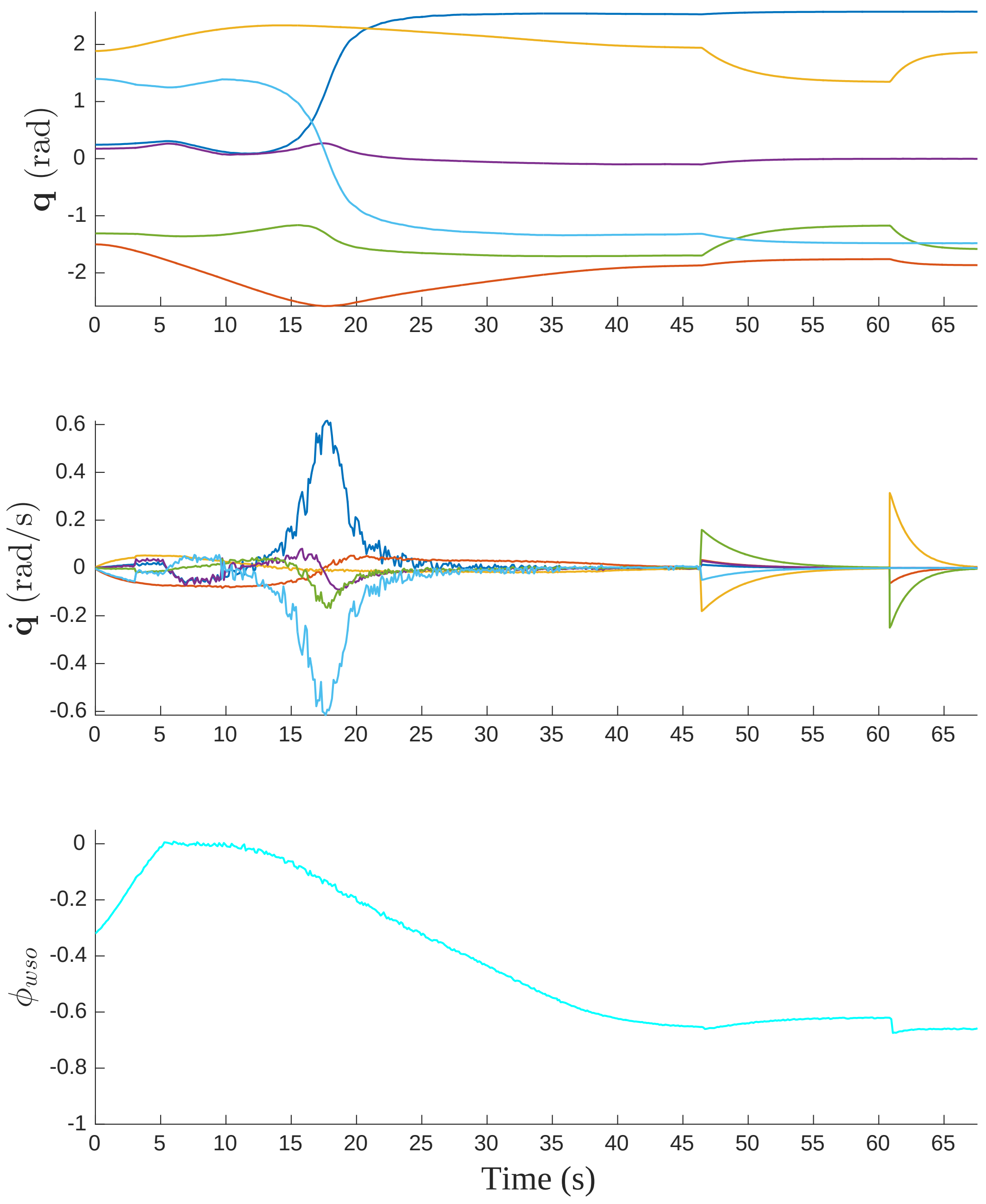


Figure 12.: From top to bottom plots: (1) joint positions $\mathbf{q}$; (2) joint speeds $\dot{\mathbf{q}}$; (3) constraint function ϕ_{wso}. Once the VS task is accomplished, around time instant 46 s, it can be observed the robot movement to place the tool in the warehouse and to go back.

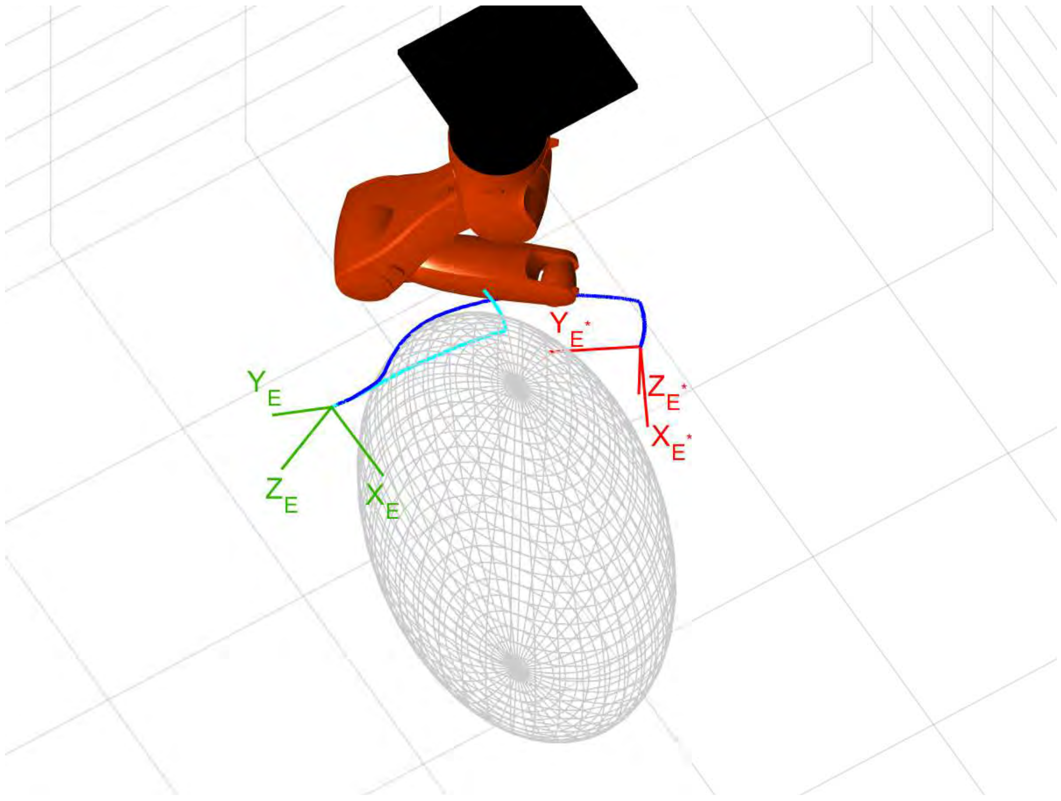
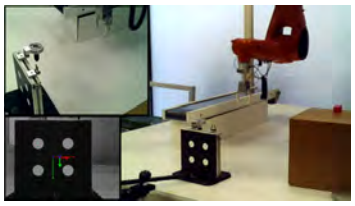
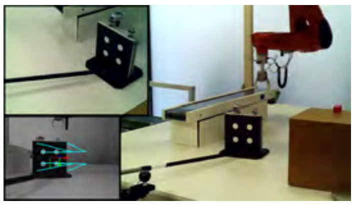
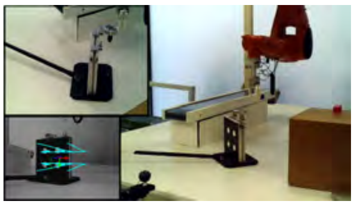
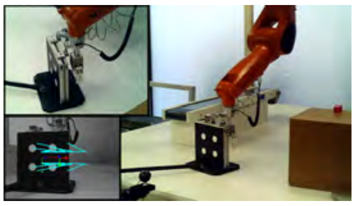


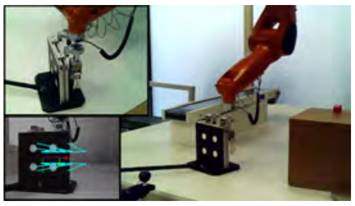
Figure 13.: Detailed view of the fulfillment of the constraint for the obstacle avoidance in the first experiment. The lines represent the trajectory of the robot end-effector when the proposed SMC method is used (dark-blue) or not (light-cyan) from initial pose (E-frame) to final pose (E*-frame). A graphical model of the robot is drawn at time instant 30 s. When the SMC method is not active the safety zone enveloping the obstacle is invaded resulting in a collision and the task is aborted.

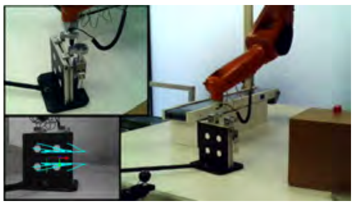

(a) 10s

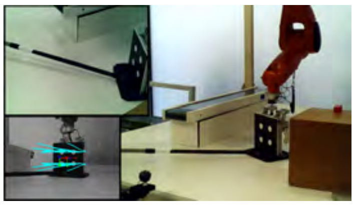

(b) 26s

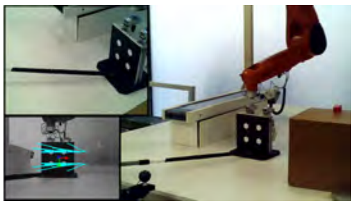

(c) 33s

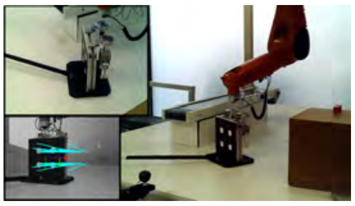

(d) 50s

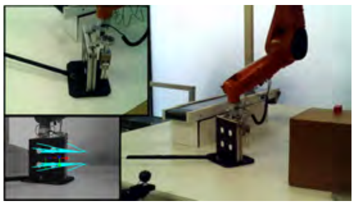

(e) 1m07s


(f) 1m08s


(g) 1m17s


(h) 1m23s


(i) 1m33s


(j) 1m44s

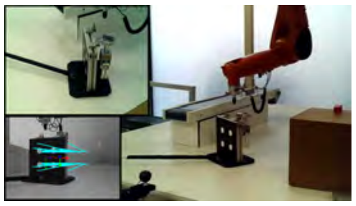

(k) 1m55s

Figure 14.: Frames of the video of the second experiment. The time instant is indicated for each frame.