ROBOT VISUAL SERVOING USING DISCONTINUOUS CONTROL

PhD Thesis

Author:
Pau Muñoz Benavent
***Supervisors*:**
Josep Tornero Montserrat
Luis Gracia Calandín

Departament d'Enginyeria de Sistemes i Automàtica
Universitat Politècnica de València

May 2017

This work has been supported by the Spanish Government (DIVISAMOS Project DPI-2009-14744-C03-01), and the Generalitat Valenciana (PROMETEO Program).

The author received the *Formación de Personal Investigador (FPI)* grant (BES-2010-038486) for predoctoral researchers by the Spanish Government.

A short-term research stay was done in the Vision for Robotics (V4R) group of the Technische Universität Wien, under the *Estancias Breves* program by the Spanish Government, with grant number EEBB-I-14-08482. The author would like to acknowledge the collaboration of the group.

Als meus

Agraïments

En primer lloc, m'agradaria donar les gràcies als directos, Josep i Luis, pels seus consells i les seues aportacions a aquest treball de tesi. Ha sigut una experiència molt enriquidora poder aprendre al seu costat.

Voldria mostrar també el meu agraïment a Mario i Vicent Franch per fer més fàcil i agradable el meu treball, a Vicent Girbés i Ernesto per tots aquestos anys de doctorat que hem passat junts i tot el que això implica, i en general a tots els membres de l'Institut de Disseny i Fabricació de la UPV.

I, molt especialment, voldria donar les gràcies als meus, família i amics, per la confiança en mi que sempre han mostrat i per animar-me a continuar endavant. Pel seu suport incondicional.

Moltes gràcies, aquesta tesi té molt de vosaltres.

Resum

Aquest treball presenta diferents propostes per a tractar problemes habituals en el control de robots per realimentació visual, basades en l'aplicació de mètodes de control discontinus. La viabilitat i eficàcia de les propostes es fonamenta amb resultats en simulació i amb experiments reals utilitzant un robot manipulador industrial 6R.

Les principals contribucions són:

– **Invariància geomètrica utilitzant control en mode lliscant (Capítol 3):** la invariància d'alt ordre definida ací és utilitzada després pels mètodes proposats, per a tractar problemes en control per realimentació visual. S'aporten proves teòriques de la condició d'invariància.

– **Compliment de restriccions en control per realimentació visual (Capítol 4):** aquesta proposta utilitza mètodes de control en mode lliscant per a satisfer restriccions mecàniques i visuals en control per realimentació visual, mentre una tasca secundària s'encarrega del seguiment de l'objecte. Els principals avantatges de la proposta són: baix cost computacional, robustesa i plena utilització de l'espai disponible per a les restriccions.

– **Canvi de ferramenta robust per a un robot industrial mitjançant control per realimentació visual (Capítol 4):** el control per realimentació visual i el mètode proposat per al compliment de les restriccions s'apliquen a una solució automatitzada per al canvi de ferramenta en robots industrials. La robustesa de la proposta radica en l'ús del control per realimentació visual, que utilitza informació del sistema de visió per a tancar el llaç de control. A més, el control en mode lliscant s'utilitza simultàniament en un nivell de prioritat superior per a satisfer les restriccions. Així doncs, el control és capaç de deixar la ferramenta en

l'intercanviador de ferramentes de forma precisa, a la vegada que satisfà les restriccions del robot.

- **Controlador en mode lliscant per a seguiment de referència (Capítol 5):** es proposa un enfocament basat en el control en mode lliscant per a seguiment de referència en robots manipuladors industrials controlats per realimentació visual. La novetat de la proposta radica en la introducció d'un controlador en mode lliscant que utilitza senyal de control discontínua d'alt ordre, i.e. acceleracions o jerks de les articulacions, per a obtindre un comportament més suau i assegurar l'estabilitat del sistema robòtic, la qual cosa es demostra amb una prova teòrica.
- **Control per realimentació visual mitjançant PWM i PFM en mètodes completament desacoblats (Capítol 6):** es proposa un control discontinu basat en modulació de l'ample i la freqüència del pols per a mètodes completament desacoblats de control per realimentació visual basats en posició, amb l'objectiu d'aconseguir el mateix temps de convergència per als moviments de rotació i translació de la càmera.

A més, es presenten també altres resultats obtinguts en aplicacions de control per realimentació visual.

Resumen

Este trabajo presenta diferentes propuestas para tratar problemas habituales en el control de robots por realimentación visual, basadas en la aplicación de métodos de control discontinuos. La viabilidad y eficacia de las propuestas se fundamenta con resultados en simulación y con experimentos reales utilizando un robot manipulador industrial 6R.

Las principales contribuciones son:

- **Invariancia geométrica utilizando control en modo deslizante (Capítulo 3):** la invariancia de alto orden definida aquí es utilizada después por los métodos propuestos, para tratar problemas en control por realimentación visual. Se apuertan pruebas teóricas de la condición de invariancia.

- **Cumplimiento de restricciones en control por realimentación visual (Capítulo 4):** esta propuesta utiliza métodos de control en modo deslizante para satisfacer restricciones mecánicas y visuales en control por realimentación visual, mientras una tarea secundaria se encarga del seguimiento del objeto. Las principales ventajas de la propuesta son: bajo coste computacional, robustez y plena utilización del espacio disponible para las restricciones.

- **Cambio de herramienta robusto para un robot industrial mediante control por realimentación visual (Capítulo 4):** el control por realimentación visual y el método propuesto para el cumplimiento de las restricciones se aplican a una solución automatizada para el cambio de herramienta en robots industriales. La robustez de la propuesta radica en el uso del control por realimentación visual, que utiliza información del sistema de visión para cerrar el lazo de control. Además, el control en modo deslizante se utiliza simultáneamente en un nivel de prioridad superior para satisfacer las restricciones. Así pues, el control es capaz de dejar la herramienta en el intercambiador de herramientas de forma precisa, a la par que satisface las restricciones del robot.

– **Controlador en modo deslizante para seguimiento de referencia (Capítulo 5):** se propone un enfoque basado en el control en modo deslizante para seguimiento de referencia en robots manipuladores industriales controlados por realimentación visual. La novedad de la propuesta radica en la introducción de un controlador en modo deslizante que utiliza la señal de control discontinua de alto orden, i.e. aceleraciones o jerks de las articulaciones, para obtener un comportamiento más suave y asegurar la estabilidad del sistema robótico, lo que se demuestra con una prueba teórica.

– **Control por realimentación visual mediante PWM y PFM en métodos completamente desacoplados (Capítulo 6):** se propone un control discontinuo basado en modulación del ancho y frecuencia del pulso para métodos completamente desacoplados de control por realimentación visual basados en posición, con el objetivo de conseguir el mismo tiempo de convergencia para los movimientos de rotación y traslación de la cámara .

Además, se presentan también otros resultados obtenidos en aplicaciones de control por realimentación visual.

Abstract

This work presents different proposals to deal with common problems in robot visual servoing based on the application of discontinuous control methods. The feasibility and effectiveness of the proposed approaches are substantiated by simulation results and real experiments using a 6R industrial manipulator.

The main contributions are:

– **Geometric invariance using sliding mode control (Chapter 3):** the defined higher-order invariance is used by the proposed approaches to tackle problems in visual servoing. Proofs of invariance condition are presented.

– **Fulfillment of constraints in visual servoing (Chapter 4):** the proposal uses sliding mode methods to satisfy mechanical and visual constraints in visual servoing, while a secondary task is considered to properly 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.

– **Robust auto tool change for industrial robots using visual servoing (Chapter 4):** visual servoing and the proposed method for constraints fulfillment are applied to an automated solution for tool changing in industrial robots. The robustness of the proposed method is due to the control law of the visual servoing, which uses the information acquired by the vision system to close a feedback control loop. Furthermore, sliding-mode control is simultaneously used in a prioritized level to satisfy the aforementioned constraints. Thus, the global control accurately places the tool in the warehouse, but satisfying the robot constraints.

– **Sliding mode controller for reference tracking (Chapter 5):** an approach based on sliding mode control is proposed for reference tracking in robot visual servoing using industrial robot manipulators. The novelty of the proposal is the introduction of a sliding mode controller that uses a high-order discontinuous control signal, i.e., joint accelerations or joint jerks, in order to obtain a smoother behavior and ensure the robot system stability, which is demonstrated with a theoretical proof.

– **PWM and PFM for visual servoing in fully decoupled approaches (Chapter 6):** discontinuous control based on pulse width and pulse frequency modulation is proposed for fully decoupled position-based visual servoing approaches, in order to get the same convergence time for camera translation and rotation.

Moreover, other results obtained in visual servoing applications are also described.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid progress and cost reduction in digital imaging, vision systems became widely used in robotics. Compared to others sensors, cameras produce the highest bandwidth of data. The need to create high-level understanding of the data acquired with digital images or videos led to the development of the research field of computer vision. Visual servoing or visual servo control represents the use of this visual feedback information to control the motion of a robot. Visual servoing relies on techniques from image processing, computer vision and control theory.

Visual servoing has a great potential to increase the flexibility of robotic systems in dealing with changing and less-structured environments. Examples include applications in mobile robots (Becerra et al. (2011)), humanoid robots (Burger et al. (2015)), underwater vehicles (Lots et al. (2001)) and unmanned aerial vehicles (Mejias et al. (2006)). However, visual servoing is mostly used in robot manipulators. Relevant reference works in the field are Hutchinson et al. (1996), Chaumette and Hutchinson (2008) and Corke (2011), among others.

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.

Motivation

One of the major problems in visual servoing consists in satisfying mechanical and visual constraints, regardless of the workspace in which visual servoing control laws are computed.

Mechanical constraints include:
- Joint range limits.
- Maximum joint speeds.
- robot workspace limits.
- Task limits, defined for certain applications if the robot has to be confined in a predefined space.
- Forbidden areas, defined to avoid collisions with object in the workspace.

Visibility constraints are one of the critical issues of visual servoing. Indeed, if a large part of the target leaves the camera field of view (FOV) during the servo or the target object is occluded, the visual information can no more be computed.

The violation of any of these constraints can lead to the failure of the visual servoing control task.

Another important issue in visual servoing is the robustness of the control law to modeling errors. Potential sources of error are: target estimation, camera calibration, target and robot modeling and low-level controller.

These issues motivated the initiation of the research, which reaches a milestone with the completion of the present work.

Thesis Outline

The present thesis contains eight chapters, including introduction and conclusion chapters. It is organized as follows:

- **Chapter 2: State of the art.** In this chapter, some fundamental concepts about visual servoing, sliding mode control and task-priority based redundancy resolution are introduced.

- **Chapter 3: Geometric invariance using sliding mode control.** This chapter reviews the principles of sliding mode control and geometric invariance theory that will be subsequently used by the proposed approaches to tackle problems in visual servoing.

- **Chapter 4: Fulfillment of constraints in visual servoing using sliding mode control.** This chapter addresses the problem of mechanical and visual constraints in visual servoing. In particular, the proposal uses sliding mode methods to satisfy motion constraints (joint limits, joint speed limits, forbidden areas to avoid collisions, task space limits and robot workspace limits) and visibility constraints (camera field-of-view and occlusions) of visual servoing applications. Moreover, another task with low-priority is considered to properly track the target object. The feasibility and effectiveness of the proposed approach is illustrated in simulation for a simple 2D case and complex 3D case studies. Furthermore, real experimentation with a conventional 6R industrial manipulator is also presented to demonstrate its applicability and robustness.

- **Chapter 5: High-Order Sliding-Mode Control for Reference Tracking in Visual Servoing.** In this chapter, the aim is to develop a sliding mode controller for reference tracking in visual servoing that uses a high-order discontinuous control signal in order to obtain a smoother behavior and ensure the robot system stability. In particular, two sliding-mode controls have been obtained depending on whether the joint accelerations or the joint jerks are considered as the discontinuous control action. Both sliding-mode controls have been compared theoretically and in simulation to their equivalent continuous counterparts. The applicability and feasibility of the proposed approach is substantiated by experimental results using a conventional 6R industrial manipulator for positioning and tracking tasks.

- **Chapter 6: PWM and PFM for visual servoing in fully decoupled approaches.** The proposal in this chapter copes with the idea of having the same convergence time for all components of the error vector by inserting an error weighting matrix in the control law without overstretching any part of the visual servoing system. The procedure is applied to fully-decoupled visual servoing approaches, i.e. those with block-diagonal interaction matrix. In particular, in the fully decoupled position-based visual servoing approach, the coefficients of the gain matrix are tuned to get the same convergence time for camera translation and rotation. Finally, novel PWM and PFM visual servoing techniques are presented, consisting in modulating, in pulse width (PWM) and pulse frequency (PFM) with high-frequency signals.

– **Chapter 7: Other results in visual servoing applications.** During the PhD, the author of this thesis has also contributed to the development of other visual servoing techniques and applications, together with members of the same robotics and automation research group. Three of these co-works are detailed in this chapter.

The main contents of this thesis are represented using a mind map in Figure 1.1.

In the next chapter, a basic state of the art of the fields of research involved in the thesis is presented.

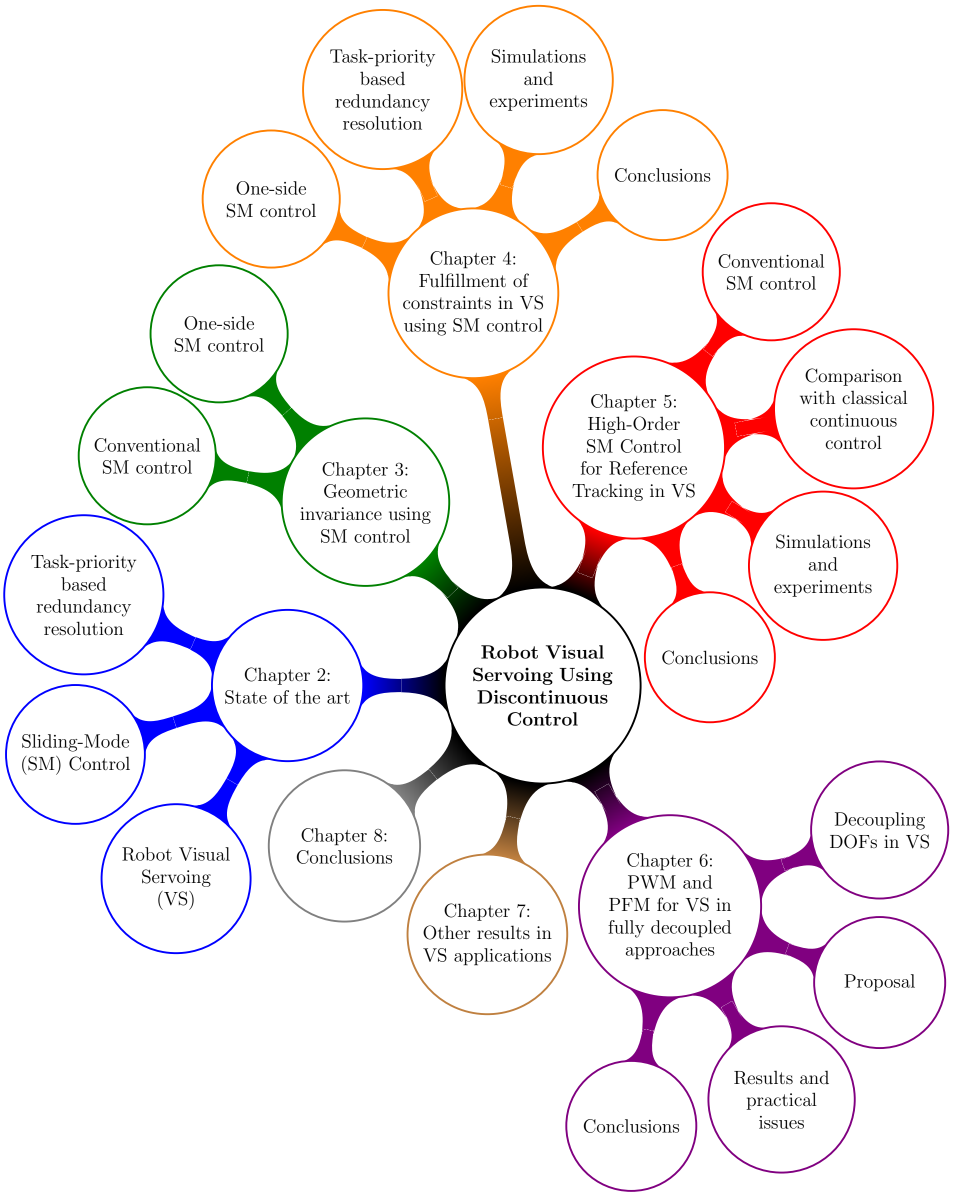

Task-priority based redundancy resolution
Simulations and experiments
One-side SM control
Conclusions
Chapter 4: Fulfillment of constraints in VS using SM control
Conventional SM control
Comparison with classical continuous control
One-side SM control
Chapter 5: High-Order SM Control for Reference Tracking in VS
Conventional SM control
Chapter 3: Geometric invariance using SM control
Simulations and experiments
Task-priority based redundancy resolution
Conclusions
Chapter 2: State of the art
Robot Visual Servoing Using Discontinuous Control
Sliding-Mode (SM) Control
Decoupling DOFs in VS
Chapter 8: Conclusions
Robot Visual Servoing (VS)
Chapter 6: PWM and PFM for VS in fully decoupled approaches
Chapter 7: Other results in VS applications
Proposal
Results and practical issues
Conclusions

Figure 1.1: Mind map representing the main contents of this thesis.

Chapter 2

State of the Art

This section briefly reviews previous results from literature that will be subsequently used by the proposed approach. State of the art for each chapter is analyzed with details in their respective introductions. The main contents of this chapter are represented using a mind map in Figure 2.1.

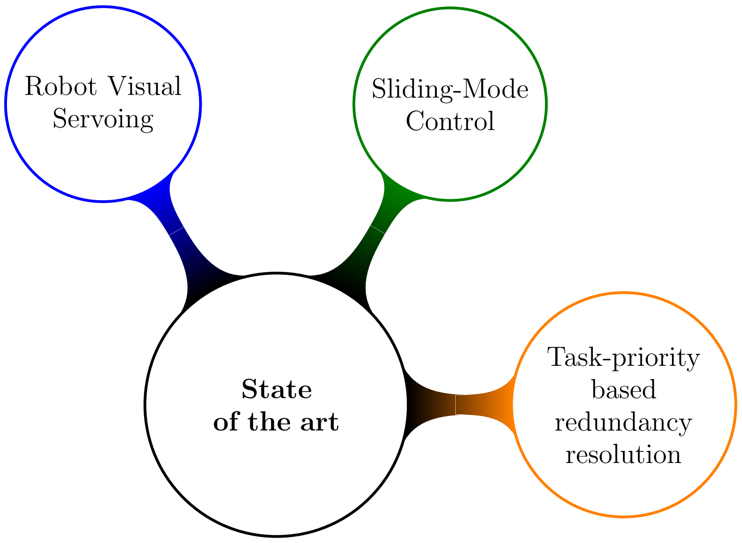


Figure 2.1: Mind map representing the main contents of this chapter.

2.1 Robot visual servoing

Visual servoing (VS) or visual servo control represents the use of feedback information extracted from a computer vision system to control the motion of a robot or any mechanical system Weiss et al. (1987). The aim of the control scheme is to minimize the difference between the measure of a set of *visual features* $\mathbf{s}$ and its desired values $\mathbf{s}^*$ or $\mathbf{s}_{ref}$.

The task in VS applications, which consists on achieving a reference value for the visual feature vector $\mathbf{s}$, can be written with the following equations:

$$\mathbf{s}(t) = \mathbf{s}_{ref}(t), \tag{2.1}$$

$$\mathbf{e} = \mathbf{s}(t) - \mathbf{s}_{ref}(t) = \mathbf{0}, \tag{2.2}$$

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

Although VS relies also on image processing and computer vision, this work focuses its research on control techniques. The main concepts of control theory for VS are reviewed bellow.

2.1.1 Basic notation

All along this document the following notation is used:

- **Coordinate frames transformations.** A leading superscript denotes the frame with respect to which a set of coordinates (subscript) is defined. For example, the coordinate vector ${}^{F_1}R_{F_2}$ represents the rotation coordinates of the origin of frame F_2 expressed relative to frame F_1.
- **Homogeneous transformations in compact notation.** ${}^{F_1}\mathbf{M}_{F_2} = \begin{bmatrix} x & y & z & \alpha & \gamma & \theta \end{bmatrix}^{\mathrm{T}}$ is the compact notation adopted for detailing the values of the homogeneous transformation matrix from frame F_1 to frame F_2, where x, y and z are the Cartesian coordinates in meters, and α, γ and θ are the roll, pitch and yaw angles, respectively, in radians.

2.1.2 Kinematics

When used with industrial manipulators, as it is mainly the case of this work, the visual feature vector $\mathbf{s}$ depends on the *robot configuration* $\mathbf{q}$ and also explicitly on time for the general case of a *moving target* object, that is:

$$\mathbf{s} = \mathbf{l}(\mathbf{q}, t), \tag{2.3}$$

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:

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{l}(\mathbf{q}, t)^{\mathrm{T}}}{\partial \mathbf{q}} \dot{\mathbf{q}} + \frac{\partial \mathbf{l}(\mathbf{q}, t)}{\partial t} = \mathbf{J}_s \dot{\mathbf{q}} + \partial \mathbf{s}/\partial t, \tag{2.4}$$

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 different Jacobian matrices.

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

$$\ddot{\mathbf{s}} = \mathbf{J}_s \ddot{\mathbf{q}} + \dot{\mathbf{J}}_s \dot{\mathbf{q}} + \partial \dot{\mathbf{s}}/\partial t. \tag{2.5}$$

$$\dddot{\mathbf{s}} = \mathbf{J}_s \dddot{\mathbf{q}} + 2\dot{\mathbf{J}}_s \ddot{\mathbf{q}} + \ddot{\mathbf{J}}_s \dot{\mathbf{q}} + \partial \ddot{\mathbf{s}}/\partial t. \tag{2.6}$$

Free flying camera. In the special case of a free flying camera, i.e. no robot is considered, the first-order kinematics is rewritten as:

$$\dot{\mathbf{s}} = \mathbf{L}_s \, \boldsymbol{\tau} + \partial \mathbf{s}/\partial t, \tag{2.7}$$

where $\mathbf{L}_s$ is the so-called *interaction matrix* and $\boldsymbol{\tau} = [\mathbf{v}, \boldsymbol{\omega}]$ is the camera kinematic screw.

2.1.3 Classical continuous control laws

Consider the first-order differential equation for the error:

$$\mathbf{e} + K_a \dot{\mathbf{e}} = \mathbf{0}, \tag{2.8}$$

where K_a is a positive parameter that determines the time constant.

Substituting the first-order kinematics of the robot system (2.4) and using (2.2), the following equation is obtained:

$$\mathbf{e} + K_a \dot{\mathbf{e}} = \mathbf{e} + K_a (\mathbf{J}_s \dot{\mathbf{q}} + \partial \mathbf{s}/\partial t - \dot{\mathbf{s}}_{ref}) = \mathbf{0}, \tag{2.9}$$

and the robot joint velocity vector results in:

$$\dot{\mathbf{q}} = -\mathbf{J}_s^{\dagger} (K_a^{-1} \mathbf{e} + \partial \mathbf{e}/\partial t - \dot{\mathbf{s}}_{ref}), \tag{2.10}$$

which is the most typical control law used in VS (Chaumette and Hutchinson (2008)) in order to obtain an exponential decrease of the tracking error. Superscript † denotes the Moore-Penrose pseudoinverse[1] (Golub and Van Loan (1996)).

Consider now the second-order differential equation for the error:

$$\mathbf{e} + K_{j1}\dot{\mathbf{e}} + K_{j2}\ddot{\mathbf{e}} = \mathbf{0}, \tag{2.11}$$

where K_{j1} and K_{j2} are positive parameters that determine the time constant.

Substituting the second-order kinematics of the robot system (2.5) and using (2.2), the following equation is obtained:

$$\mathbf{e} + K_{j1}\dot{\mathbf{e}} + K_{j2}(\mathbf{J}_s + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{e}}/\partial t - \dot{\mathbf{s}}_{ref}) = \mathbf{0}, \tag{2.12}$$

and the robot joint acceleration vector results in:

$$\ddot{\mathbf{q}} = -\mathbf{J}_s^{\dagger}(K_{j2}^{-1}\mathbf{e} + K_{j2}^{-1}K_{j1}\dot{\mathbf{e}} + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{e}}/\partial t - \dot{\mathbf{s}}_{ref}), \tag{2.13}$$

which represents the classical operational space robot control (Siciliano et al. (2009)) that has already been used in VS appliactions by Fakhry and Wilson (1996) and Keshmiri et al. (2014).

The partial derivative $\partial\mathbf{e}/\partial t$ and $\partial\dot{\mathbf{e}}/\partial t$ are typically estimated (see Chaumette and Hutchinson (2008)) using the first-order and second-order kinematics given by (2.4) and (2.5), yielding:

$$\partial\mathbf{e}/\partial t = \dot{\mathbf{e}} - \mathbf{J}_s\dot{\mathbf{q}}, \tag{2.14}$$

$$\partial\dot{\mathbf{e}}/\partial t = \ddot{\mathbf{e}} - \mathbf{J}_s\ddot{\mathbf{q}} - \dot{\mathbf{J}}_s\dot{\mathbf{q}}. \tag{2.15}$$

2.1.4 Robot-camera configurations

Two main configurations are commonly used, depending on the position of the camera in the scene with respect to the robot: *eye-in-hand* configuration, when the camera is rigidly attached to the robot end-effector; and *eye-to-hand* configuration, when the camera is placed out of the robot.

[1]Pseudoinverse may be computed via the singular value decomposition (SVD) method Golub and Van Loan (1996) and using a tolerance to set to zero the very small singular values in order to avoid extremely large values for the commanded accelerations.

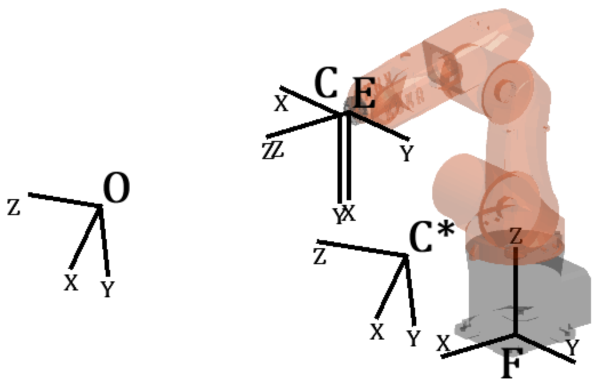


Figure 2.2: Frames involved in eye-in-hand VS.

Eye-in-hand configuration. Fig. 2.2 shows the coordinate frames involved in the eye-in-hand 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.

Eye-to-hand configuration. Fig. 2.3 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.

The resulting Jacobian matrices are expressed in 2.16 and 2.17 for the eye-in-hand and eye-to-hand configurations respectively

$$\mathbf{J}_s(\mathbf{q},t) = \mathbf{L}_s(\mathbf{q},t)\,{}^c\mathbf{V}_e\,{}^e\mathbf{J}_e(\mathbf{q}), \tag{2.16}$$

$$\mathbf{J}_s(\mathbf{q},t) = -\mathbf{L}_s(\mathbf{q},t)\,{}^c\mathbf{V}_e\,{}^e\mathbf{J}_e(\mathbf{q}), \tag{2.17}$$

where $\mathbf{L}_s$ is the the 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.

For all the experiments in this work, it is used a general purpose web-cam and a classical 6R serial robot with spherical wrist: the Kuka Agilus KR6

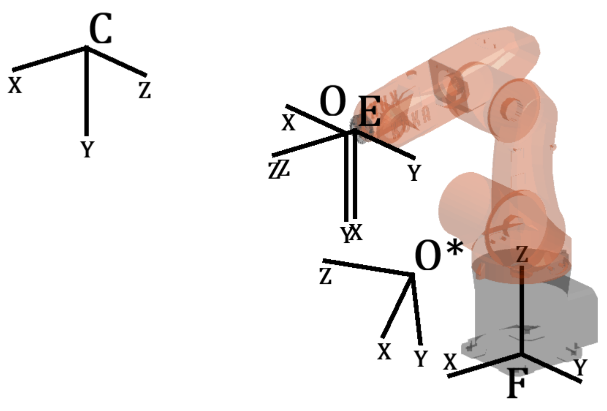


Figure 2.3: Frames involved in eye-to-hand VS.

R900 sixx manipulator. The robot is ceiling-mounted and its Jacobian matrix ${}^{e}\mathbf{J}_{e}$ can be readily obtained (Siciliano et al. (2009)) taking into account the Denavit-Hartenberg (DH) parameters shown in Table 2.1. Further information about the experimental platform can be found in Appendix A.

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 2.1: Denavit-Hartenberg parameters for the Kuka Agilus KR6 R900 sixx 6R robot.

2.1.5 Visual servoing schemes

During the last three decades, many visual servoing schemes have been proposed in the literature, most of them differing in the selection of the visual features. However, all of them can be classified in two main categories, depending on the workspace in which the control is set: image-based or 2D visual servoing (IBVS) (Weiss et al. (1987), Feddema and Mitchell (1989), Espiau et al. (1992) , Hashimoto et al. (1991)) and position-based or 3D visual servoing (PBVS) (Wilson et al. (1996)).

2.1.5.1 Position Based Visual Servoing

In Position Based Visual Servoing (PBVS) approaches the visual features vector $\mathbf{s}$ is defined in terms of 3D pose of the camera with respect to some of the coordinate frames in Fig. 2.2 or Fig. 2.3. Computing this pose from a set of image features necessitates the camera intrinsic parameters and the 3-D model of the object observed to be known (Chaumette and Hutchinson (2008)). $\mathbf{s}$ and $\mathbf{s}_{ref}$ are defined as $\mathbf{s} = [\mathbf{t}, {}^{c^*}\theta\mathbf{u}_c]$ and $\mathbf{s}_{ref} = [\mathbf{t}^*, \mathbf{0}]$, where $\mathbf{t}$ and $\mathbf{t}^*$ are translation vectors and ${}^{c^*}\theta\mathbf{u}_c$ gives the angle/axis parameterization for the rotation between desired camera frame, C^*, and current camera frame, C. Different PBVS approaches are obtained, depending on the selected coordinate frames for $\mathbf{t}$ and $\mathbf{t}^*$.

If $\mathbf{t} = {}^{c^*}\mathbf{t}_c$ and $\mathbf{t}^* = \mathbf{0}$ are considered for the translation vectors, $\mathbf{s}$, $\mathbf{s}_{ref}$ and $\mathbf{L}_s$ are given by:

$$\mathbf{s} = \begin{bmatrix} {}^{c^*}\mathbf{t}_c^{\mathrm{T}} & {}^{c^*}\theta\mathbf{u}_c^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{2.18}$$

$$\mathbf{s}_{ref} = \mathbf{0} \tag{2.19}$$

$$\mathbf{L}_s = \begin{bmatrix} {}^{c^*}\mathbf{R}_c & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}. \tag{2.20}$$

If $\mathbf{t} = {}^{c}\mathbf{t}_o$ and $\mathbf{t}^* = {}^{c^*}\mathbf{t}_o$ are considered for the translation vectors, $\mathbf{s}$, $\mathbf{s}_{ref}$ and $\mathbf{L}_s$ are given by:

$$\mathbf{s} = \begin{bmatrix} {}^{c}\mathbf{t}_o^{\mathrm{T}} & {}^{c^*}\theta\mathbf{u}_c^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{2.21}$$

$$\mathbf{s}_{ref} = \begin{bmatrix} {}^{c^*}\mathbf{t}_o^{\mathrm{T}} & \mathbf{0} \end{bmatrix}^{\mathrm{T}} \tag{2.22}$$

$$\mathbf{L}_s = \begin{bmatrix} -\mathbf{I} & [{}^{c}\mathbf{t}_o]_x \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}. \tag{2.23}$$

where $[{}^{c}\mathbf{t}_o]_x$ is the skew-symmetric matrix associated to vector ${}^{c}\mathbf{t}_o$.

In both cases

$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\operatorname{sinc}(\theta)}{(\operatorname{sinc}(\theta/2))^2}\right)[\mathbf{u}]_\times^2, \tag{2.24}$$

where sinc(·) represents the *sinus cardinal* function, i.e., $\operatorname{sinc}(x) = \sin(x)/x$.

In the angle/axis parameterization for the rotation, $\mathbf{u} = [u_x\ u_y\ u_z]^{\mathrm{T}}$ is a unit vector representing a rotation axis and θ is the rotation angle, being $\theta\mathbf{u}$ one of the minimal representation for the orientation. In this sense, a rotation matrix $\mathbf{R}$ can be obtained from the $\theta\mathbf{u}$ representation using the Rodrigues' rotation formula:

$$\mathbf{R} = \mathbf{I}_3 + (1 - \cos\theta)\mathbf{u}\mathbf{u}^{\mathrm{T}} + (\sin\theta)[\mathbf{u}]_\times, \tag{2.25}$$

where $\mathbf{I}_3$ is the identity matrix of dimension 3×3 and $[\mathbf{u}]_\times$ the skew matrix:

$$[\mathbf{u}]_\times = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}. \tag{2.26}$$

In the same way, $\theta\mathbf{u}$ can be obtained from the elements $\mathbf{R}_{ij}$ of rotation matrix $\mathbf{R}$ as follows:

$$\cos\theta = \frac{\mathbf{R}_{11} + \mathbf{R}_{22} + \mathbf{R}_{33} - 1}{2} \tag{2.27}$$

$$\sin\theta = \frac{\sqrt{(\mathbf{R}_{32} - \mathbf{R}_{23})^2 + (\mathbf{R}_{13} - \mathbf{R}_{31})^2 + (\mathbf{R}_{21} - \mathbf{R}_{12})^2}}{2} \tag{2.28}$$

$$\theta = \arctan\left(\frac{\sin\theta}{\cos\theta}\right) \tag{2.29}$$

$$\mathbf{u} = \frac{1}{2\sin\theta}\begin{bmatrix} \mathbf{R}_{32} - \mathbf{R}_{23} \\ \mathbf{R}_{13} - \mathbf{R}_{31} \\ \mathbf{R}_{21} - \mathbf{R}_{12} \end{bmatrix} \tag{2.30}$$

$$\theta\mathbf{u} = \theta\,\mathbf{u}, \tag{2.31}$$

where function arctan(·) performs the four-quadrant inverse tangent.

2.1.5.2 Image Based Visual Servoing

In Image Based Visual Servoing (IBVS) the control law is directly computed in the image space. Traditional image-based control schemes use the pixel coordinates of *image features* to define the visual features vector $\mathbf{s}$. Many geomteric and non-geometric features have been proposed. For example, Tahri et al. (2004), Chaumette (2004) and Zhao et al. (2015) use image moments; Bakthavatchalam et al. (2014) coined the photogrametric moments; Hafez et al. (2008) use mixture of Gaussian features.

However, this work considers the most commonly used image features, i.e., image points. To control six degrees of freedom, more than three points are usually considered, to avoid singularities in the interaction matrix and the existence of global minima (Chaumette and Hutchinson (2008)). In our case, four points forming the vertices of a square are always considered.

Thus $\mathbf{s}$, $\mathbf{s}_{ref}$ and $\mathbf{L}_s$ are given by:

$$\mathbf{s} = [u_i \quad v_i]^{\mathrm{T}}, \tag{2.32}$$

$$\mathbf{s}_{ref} = [u_i^* \quad v_i^*]^{\mathrm{T}}, \tag{2.33}$$

$$\mathbf{L}_{s,i} = \begin{bmatrix} -f/Z & 0 & u_i/Z & u_i v_i/f & -(f^2+u_i^2)/f & v_i \\ 0 & -f/Z & v_i/Z & (f^2+v_i^2)/f & -u_i v_i/f & -u_i \end{bmatrix}, \tag{2.34}$$

$$\mathbf{L}_s = \begin{bmatrix} \mathbf{L}_{s,1} \\ \cdots \\ \mathbf{L}_{s,4} \end{bmatrix}, \tag{2.35}$$

where f is the focal length of the camera lens in pixels, (u_i, v_i) and (u_i^*, v_i^*) are the current and desired pixel coordinates of the image feature i in the image plane with respect to a coordinate system UV located in its center and Z is the distance from the image plane to the target object.

2.1.5.3 PBVS-IBVS comparison

As explained before, control law in PBVS is carried out in the operational space, while in IBVS is directly computed in the image space. These different approaches lead to different evolution of the variables during the servoing, although the choice of the appropriate approach is application dependent. In this section the main differences are outlined, for a complete comparison between IBVS and PBVS see Janabi-Sharifi et al. (2011).

– PBVS requires 3D pose estimation of the object with respect to the camera-robot system. A 3D model of the object is necessary to estimate the pose from the image features. On the contrary, in IBVS only the camera intrinsic parameters are needed to update the visual features $\mathbf{s}$. Distance from the camera to the object Z is needed to compute the interaction matrix, although estimations and approximations that not require a full 3D model of the object can be used (Chaumette and Hutchinson (2008)).

– IBVS is inherently robust to camera calibration and target modeling errors (Hutchinson et al. (1996); Chaumette and Hutchinson (2008); Hafez et al. (2008); Kermorgant and Chaumette (2011)).

– The 3D parameter estimation affects the accuracy of the reached pose in PBVS, whereas in IBVS it affects the camera motion but not the convergence (Chaumette and Hutchinson (2008)).

– In IBVS a significant coupling between the end-effector translation and rotation motion control is present. In particular, the common choice of points as visual features and the coupling between the third and sixth columns in the interaction matrix, produce unnecessary translation motion when large rotation errors are considered, problem known as *camera retreat* (Chaumette (1998)).

– IBVS can experience task singularities, which is avoided using more than three points as image features, and local minima.

– As PBVS control law is done in the 3D space and no control is done in the 2D image plane, the camera trajectory can make the image features leave the camera field of view (FOV). For this reason, camera FOV constraint in PBVS is addressed in the experiment in Section 4.7.

– As IBVS control law is done in the 2D image plane and no control is done in the 3D space, the robot end-effector can exceed the allowed workspace. This fact is worsened by the effect of *camera retreat*. For this reason, workspace limits constraint in IBVS is addresses in the experiment in Section 4.8.

2.1.5.4 VS schemes combining PBVS and IBVS

Many approaches have been proposed, based on the idea of combining advantages of PBVS and IBVS while trying to avoid their shortcomings (Kragic et al. (2002)): authors in Corke and Hutchinson (2001) present a partitioned approach for IBVS to isolate motion related to the optic axis; authors in Malis et al. (1999), Morel et al. (2000) and Chaumette and Malis (2000) propose the so-called hybrid approaches; authors in Chesi et al. (2004) presented a switching method between IBVS and PBVS; authors in Gans and Hutchinson (2007) introduced a switching approach which uses the classic PBVS control law and backward motion along the camera optical axis; authors in Kim et al. (2009) proposed a switching approach using Hybrid Visual Servoing (HVS) control laws and pure translation motions; authors in Deng and Janabi-Sharifi (2005) introduced a path planning and PBVS-IBVS switching method in order to deal with image singularities and local minima; authors in Kermorgant and Chaumette (2011) presented a combination approach which uses 2D and 3D information from IBVS and PBVS; and authors in Hafez and Jawahar (2007) proposed a combination method based on weighting IBVS and PBVS control strategies with a 5D objective function.

2.1.6 Constraints in visual servoing

In visual servoing, system evolution in two different spaces must be taken into account: Cartesian space (3D) and image space (2D). Both spaces constraints must be considered to evaluate overall performance of the servoing: visibility, joints and workspace constraints.

Many works have been proposed to deal with this issue, e.g., (Corke and Hutchinson (2001), Chesi et al. (2004), Deng and Janabi-Sharifi (2005) and Hajiloo et al. (2016)). Chapter 4 is fully dedicated to the study of constraints in VS, and an approach for constraints fulfillment based on sliding mode techniques is proposed.

2.1.7 Robot control

This work assumes the existence of an underlying robot control in charge of achieving a particular joint velocity, joint acceleration or joint jerk (depending on the control case) from a command. Nevertheless, the actual joint value will not be exactly the commanded one due to the dynamics of the low-level control loop and the inaccuracies because of disturbances. However, in this work it is assumed that the dynamics of the low-level control loop is fast enough compared to that of the joint commanded variable so that the relationships below hold approximately true, avoiding the need of including extra state variables. Therefore, depending on the control application (velocity, acceleration or jerk) the following equations are considered:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_c + \mathbf{d}_c \tag{2.36}$$

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_c + \mathbf{d}_c \tag{2.37}$$

$$\dddot{\mathbf{q}} = \dddot{\mathbf{q}}_c + \mathbf{d}_c, \tag{2.38}$$

where subscript c is used for the commanded variable and $\mathbf{d}_c$ represents the inaccuracies of the low-level control loop due.

2.1.8 Computer vision algorithm

This work assumes existence of the computer vision algorithm, necessary to update the visual features $\mathbf{s}$ and interaction matrix $\mathbf{L}_s$ from the image features. ViSP (Visual Servoing Platform) software package (Marchand et al. (2005)) is used for this purpose. This algorithm is composed of three parts:

i) Image processing for obtaining the image plane coordinates (u_i, v_i) of all the image feature points,

ii) 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,

iii) Pose estimation of the camera (eye-in-hand) or robot (eye-to-hand) for PBVS, and depth estimation Z for IBVS (if needed).

2.2 Sliding mode control

This section offers a brief introduction to Sliding Mode (SM) control, refer to Edwards and Spurgeon (1998) for further details. SM control is a Variable Structure Control (VSC) method based on output feedback and a high-frequency switching control action. Its primary function consists in performing a switching between two different structures in order to get a desired new dynamics in the system, known as *sliding-mode dynamics*. This feature allows the system to have an enhanced robust performance, including insensitivity to parametric uncertainties and rejection to disturbances that verify the so-called *matching condition* (Kunusch et al. (2012)).

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

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{d}) + \mathbf{g}(\mathbf{x})\,\mathbf{u}, \tag{2.39}$$

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.

Let ϕ be defined as a constraint function designed according to the control requirements. Then:

$$\phi = \{\mathbf{x} \mid \phi(\mathbf{x}) = 0\} \tag{2.40}$$

defines a region of the state space compatible with the constraint function called *sliding manifold* or *switching surface*.

To guarantee that the system converges to ϕ and remains there henceforth, a discontinuous control action is proposed, which takes its value depending on the sign of ϕ:

$$\mathbf{u} = \begin{cases} u^+ & \text{if} \quad \phi(\mathbf{x}) > 0 \\ u^- & \text{if} \quad \phi(\mathbf{x}) < 0, \end{cases} \tag{2.41}$$

where u^+ and u^- are constants, which must be capable of forcing the state trajectory to the switching surface and of maintaining a sliding mode condition. When the surface is reached, the rate of change of ϕ must guarantee the crossing of the surface from both sides, see Fig. 2.4.

Discrete-time implementations of any practical SM control makes the system leave the ideal SM and oscillate with finite frequency and amplitude inside a band around $\phi = 0$, namely *chattering* (Edwards and Spurgeon (1998)). This can degrade the system performance or may even lead to instability. Nevertheless, the chattering band decreases when fast sampling rates can be considered.

2.3 Task-priority based redundancy resolution

Redundancy resolution in robotic systems allows to tackle several (possibly incompatible) objectives simultaneously (Chiaverini et al. (2008)). In particular, it is useful to consider the task-priority strategy (Gracia et al. (2014)),

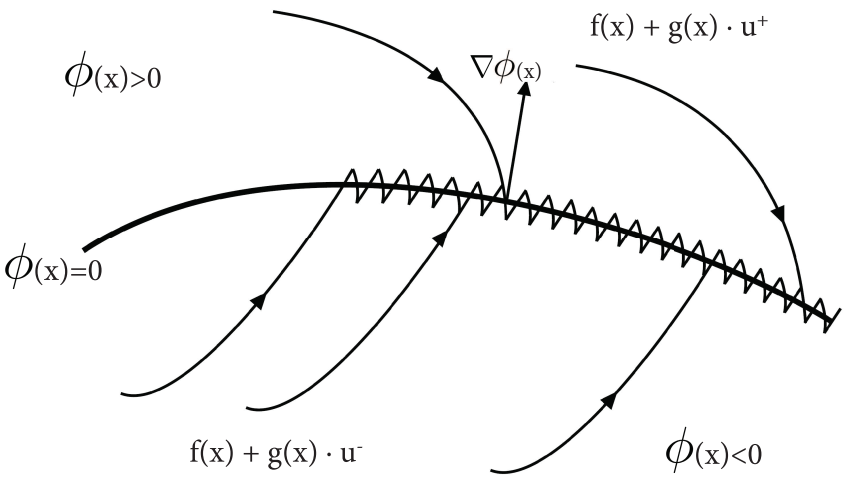


Figure 2.4: Sliding manifold or switching surface.

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 (Nakamura et al. (1987)). When an exact solution is not possible for a given task at a particular priority level, its error is minimized. The formulation for this approach is detailed below. Let us consider M tasks which consist on calculating a command vector $\ddot{\mathbf{q}}_c$ (i.e., the commanded joint acceleration vector) in order to fulfill the following acceleration equality constraints:

$$\mathbf{A}_i\ddot{\mathbf{q}}_c = \mathbf{b}_i, \quad i = 1, \ldots, M, \tag{2.42}$$

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 (2.42) is given by the following recursive formulation, proposed in Siciliano and Slotine (1991):

$$\begin{aligned} \ddot{\mathbf{q}}_{c,i} &= \ddot{\mathbf{q}}_{c,i-1} + (\mathbf{A}_i\mathbf{N}_{i-1})^\dagger(\mathbf{b}_i - \mathbf{A}_i\ddot{\mathbf{q}}_{c,i-1}) \\ \mathbf{N}_i &= \mathbf{N}_{i-1}(\mathbf{I} - (\mathbf{A}_i\mathbf{N}_{i-1})^\dagger(\mathbf{A}_i\mathbf{N}_{i-1})), \quad i = 1, \ldots, M, \quad \ddot{\mathbf{q}}_{c,0} = \mathbf{0}, \, \mathbf{N}_0 = \mathbf{I}, \end{aligned} \tag{2.43}$$

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 (Golub and Van Loan (1996)), 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.

2.3.1 Regularization

Note that excessively large values of $\ddot{\mathbf{q}}_{c,i}$ are obtained around the singularities of matrix $\mathbf{A}_i\mathbf{N}_{i-1}$. If minor deviations can be allowed (for instance, in lower-level tasks), this issue can be overcome using matrix regularization for $\mathbf{A}_i\mathbf{N}_{i-1}$ in the computation of $\ddot{\mathbf{q}}_{c,i}$ in (2.43). For instance, the regularized Moore-Penrose pseudoinverse of a matrix $\mathbf{H}$ using the classical damped least-squares (DLS) solution (Deo and Walker (1995)) results in:

$$\mathbf{H}^{\sharp} = \mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{H}^{\mathrm{T}} + \lambda^2\mathbf{I})^{-1} = (\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda^2\mathbf{I})^{-1}\mathbf{H}^{\mathrm{T}}, \tag{2.44}$$

where λ is a nonnegative damping factor and superscript $\sharp$ denotes the regularized pseudoinverse. Note that, it is computationally more efficient to use the second expression above if $\mathbf{H}$ is a matrix with more rows than columns and to use the first one otherwise. The value chosen for the damping factor can be a small constant (Wampler (1986)) or other more sophisticated proposals in the literature can also be considered (Nakamura and Hanafusa (1986)).

In the next chapter, the principles of sliding mode control and geometric invariance are reviewed.

Chapter 3

Geometric invariance using sliding mode control

This chapter reviews the principles of sliding mode control and geometric invariance theory (Garelli et al. (2011)), that will be subsequently used by the proposed approach to tackle problems in visual servoing. The main contents of this chapter are represented using a mind map in Figure 3.1.

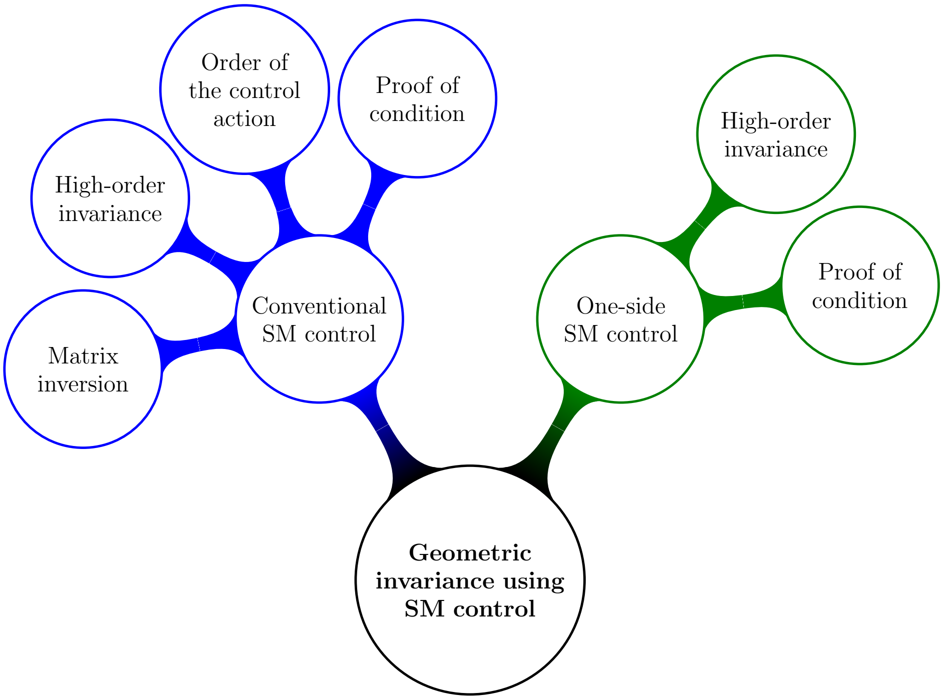


Figure 3.1: Mind map representing the main contents of this chapter.

3.1 Conventional sliding mode

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

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{d}) + \mathbf{g}(\mathbf{x})\,\mathbf{u}, \tag{3.1}$$

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 equality constraints $\phi_i(\mathbf{x}) = 0$, $i = 1, \dots, N$, where $\phi_i(\mathbf{x})$ is the ith equality constraint function. Thus, the region Φ of the state space compatible with the constraints on state $\mathbf{x}$ is given by:

$$\Phi = \{\mathbf{x} \,|\, \phi_i(\mathbf{x}) = 0\}, \quad i = 1, \dots, N. \tag{3.2}$$

The objective is to find a control input $\mathbf{u}$ such that the system converges to Φ in finite time and remains there henceforth. Mathematically, this is guaranteed by an input $\mathbf{u}$ such that[1]:

$$\begin{aligned} \frac{d(\phi_i(\mathbf{x}))}{dt} &= \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\dot{\mathbf{x}} = \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{f}(\mathbf{x},\mathbf{d}) + \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{g}(\mathbf{x})\,\mathbf{u} \\ &= L_f\phi_i(\mathbf{x},\mathbf{d}) + \mathbf{L_g}\phi_i(\mathbf{x})\mathbf{u} = \begin{cases} < 0 & \text{if} \quad \phi_i(\mathbf{x}) > 0 \\ 0 & \text{if} \quad \phi_i(\mathbf{x}) = 0 \\ > 0 & \text{if} \quad \phi_i(\mathbf{x}) < 0, \end{cases} \\ & \quad i = 1, \dots, N \end{aligned} \tag{3.3}$$

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.

To satisfy (3.3), this work proposes a simple strategy, involving simple matrix operations, given by the variable structure control law below:

$$\mathbf{u} = -\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\,\mathrm{sign}(\boldsymbol{\phi})\,u^{+}, \tag{3.4}$$

where matrix $\mathbf{L_g}\boldsymbol{\phi}$ contains the row vectors $\mathbf{L_g}\phi_i$ of all constraints, $\boldsymbol{\phi}$ is a column vector with all the constraint functions ϕ_i, $\mathrm{sign}(\cdot)$ represents the *sign*

[1]Note that it is assumed that the constraint function ϕ_i is differentiable.

function (typically used in SM control) and u^+ is a positive constant to be chosen high enough to satisfy (3.3). In particular, a *sufficient*, but not necessary, condition to achieve this is that:

$$u^+ > \|L_f\boldsymbol{\phi}\|_1 \Big/ \mathrm{eig}_{\min}(\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}), \tag{3.5}$$

where $L_f\boldsymbol{\phi}$ is a column vector containing the elements $L_f\phi_i$ of all constraints, $\|\cdot\|_1$ represents the 1-norm (also known as the Taxicab norm) and function $\mathrm{eig}_{\min}(\cdot)$ computes the minimum eigenvalue of a matrix.

See *proof* in 3.1.4.

3.1.1 Using matrix inversion

Alternatively, instead of using the transpose of matrix $\mathbf{L_g}\boldsymbol{\phi}$ in (3.4), it could be also utilized the Moore-Penrose pseudoinverse (Golub and Van Loan (1996)), but at the expense of *higher computational load* and condition that $\mathbf{L_g}\boldsymbol{\phi}$ must be now *full row rank*. In this case, the control law and the lower bound condition for u^+ result in:

$$\mathbf{u} = -\mathbf{L_g}\boldsymbol{\phi}^{\dagger}\,\mathrm{sign}(\boldsymbol{\phi})\,u^+, \tag{3.6}$$

$$u^+ > \|L_f\boldsymbol{\phi}\|_1, \tag{3.7}$$

where superscript † denotes the Moore-Penrose pseudoinverse

The control law given by (3.4), or alternatively (3.6), firstly makes the system converge to Φ in finnite time. This initial phase is known as open loop or reaching phase (Edwards and Spurgeon (1998)). And, secondly, once the region Φ has been achieved, the control law will make $\mathbf{u}$ switch at a theoretically infinite frequency in order to keep the system on the so-called sliding surface Φ. This final phase is known as SM phase (Edwards and Spurgeon (1998)). Moreover, a continuous equivalent control (Utkin et al. (2009)) can be obtained for the SM phase, i.e., the control required to keep the system on the sliding surface. Hence, the SM generated by (3.4) or (3.6) produces such control action without explicit knowledge of it and with a low computational cost, which is a typical advantage of SM strategies (Utkin et al. (2009)).

3.1.2 Higher-order invariance

The above SM method produces a non-smooth $\mathbf{u}$. If a smooth control action is wished, the following approach can be used. Firstly, the initial constraint

$\phi(\mathbf{x}) = 0$ is transformed to $\overline{\phi} = \phi(\mathbf{x}) + K\dot{\phi}(\mathbf{x}) = 0$. Thus, we obtain $\overline{\phi} = \phi(\mathbf{x}) + K\nabla\phi \cdot \dot{\mathbf{x}} = \phi(\mathbf{x}) + K\nabla\phi \cdot (\mathbf{f}(\mathbf{x}, \mathbf{d}) + \mathbf{g}(\mathbf{x})\mathbf{u})$. Hence, an augmented state is considered $\bar{\mathbf{x}}^{\mathrm{T}} = [\mathbf{x}^{\mathrm{T}}\, \mathbf{u}^{\mathrm{T}}]$, which includes the input $\mathbf{u}$, so that $\overline{\phi}$ is a function of the augmented state, i.e, $\overline{\phi}(\bar{\mathbf{x}}, \mathbf{d})$. Then, taking time derivatives[2], we obtain $\dot{\overline{\phi}} = (\partial\overline{\phi}/\partial\bar{\mathbf{x}})^{\mathrm{T}}\dot{\bar{\mathbf{x}}} + (\partial\overline{\phi}/\partial\mathbf{d})^{\mathrm{T}}\dot{\mathbf{d}}$. Thus, since $\dot{\mathbf{u}}$ appears in $\dot{\bar{\mathbf{x}}}$, $\overline{\phi}$ is relative degree one in $\dot{\mathbf{u}}$, so considering $\dot{\mathbf{u}}$ as the "new" input (which will have a switching behavior when using SM approaches), the actual control $\mathbf{u}$ will be now smooth[3]. Hence, the fulfillment of the new constraint $\overline{\phi} = 0$ (e.g., using SM control) gives rise to an exponential decrease of the original constraint ϕ towards zero, i.e., $\phi(t) = \phi(0)e^{-t/K}$, where K represents the time-constant of the first-order system relating both constraints. Therefore, K is a free design parameter to establish the rate of approach to the original constraint in order to reach it in a controlled fashion, i.e., the larger K is, the slower ϕ changes, approaching the original case.

3.1.3 Order of the control action

In order to use the SM method above, the time-derivative of $\boldsymbol{\phi}$ must explicitly depend on the control action $\mathbf{u}$, see (3.3). That is, the sliding manifold must have *relative degree one* with respect to the control variable, as required by SM control theory (Edwards and Spurgeon (1998)). Therefore, when the constraint function vector $\boldsymbol{\phi}$ is defined, the order of the corresponding discontinuous control action in (3.4) or (3.6) is also established. Nevertheless, if the order of the actual control action vector for the system at hand does not match the order of the mentioned discontinuous control action, a *filter* with the right order can be used between both signals to meet the relative degree condition above.

For example: a) if the constraint function depends on the joint positions and velocities, the discontinuous control action is an acceleration or second-order signal; b) if the actual control action is the joint velocity vector, a first-order filter (or a pure integrator) has to be applied to the discontinuous control signal in order to compute the actual control action, which is continuous.

Note that in any case, the order of the actual control action must be equal

[2]Note that to use this approach the original constraint function ϕ_i needs to be *twice* differentiable.

[3]The assumption that $\dot{\mathbf{d}}$ is bounded is also needed as $\dot{\mathbf{d}}$ appears in $\dot{\overline{\phi}}$, e.g., this assumption is fulfilled if $\mathbf{d}$ has finite bandwidth.

to or lower than the order of the discontinuous control signal. If that would not be the case, the higher-order SM described in Section 3.1.2 may be used to increase the order of the discontinuous control signal.

It is important to remark that, if needed, the filter has to be properly designed since it limits the bandwidth of the controlled system.

3.1.4 Proof of condition Eq. (3.5)

From (3.3) and (3.4), the column vector $\dot{\boldsymbol{\phi}}$ composed of the constraint function derivatives $\dot{\phi}_i$ is given by:

$$\dot{\boldsymbol{\phi}} = L_f\boldsymbol{\phi} - \left(\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\right)\,\mathbf{z}\,u^{+}, \tag{3.8}$$

where $\mathbf{z}$ is a column vector with the ith-component $z_i = \operatorname{sign}(\phi_i)$.

The goal of this proof is to show that $\boldsymbol{\phi} = \mathbf{0}$ is an asymptotically stable equilibrium point with finite time convergence. For this purpose, let $V = \mathbf{z}^{\mathrm{T}}\,\boldsymbol{\phi}$ be a 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 the components of vector $\boldsymbol{\phi}^{N-a}$ are not zero. Obviously, one of these two subvectors may be empty at a certain time. According to the continuous equivalent control (Edwards and Spurgeon (1998)), vector $\mathbf{z}^a$ must be replaced by the function $\mathbf{z}^a_{eq}$ such that $\dot{\boldsymbol{\phi}}^a = \mathbf{0}_a$. Because $\boldsymbol{\phi}^a = \mathbf{0}_a$ in SM, the time derivative of V results in:

$$\begin{aligned}\dot{V} = \frac{d}{dt}\left(\mathbf{z}^{\mathrm{T}}\,\boldsymbol{\phi}\right) &= \frac{d}{dt}\left(\begin{bmatrix}\mathbf{z}^a_{eq}\\ \pm\mathbf{1}_{N-a}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\boldsymbol{\phi}^a\\ \boldsymbol{\phi}^{N-a}\end{bmatrix}\right)\\ &= \begin{bmatrix}\dot{\mathbf{z}}^a_{eq}\\ \mathbf{0}_{N-a}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{0}_a\\ \boldsymbol{\phi}^{N-a}\end{bmatrix} + \mathbf{z}^{\mathrm{T}}\,\dot{\boldsymbol{\phi}} = \mathbf{z}^{\mathrm{T}}\,\dot{\boldsymbol{\phi}},\end{aligned} \tag{3.9}$$

where $\pm\mathbf{1}$ represents a column vector with all its elements equal to 1 or -1.

Replacing vector $\dot{\boldsymbol{\phi}}$ with its value from (3.8), it is obtained:

$$\dot{V} = \mathbf{z}^{\mathrm{T}}\,L_f\boldsymbol{\phi} - \mathbf{z}^{\mathrm{T}}\,\left(\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\right)\,\mathbf{z}\,u^{+}. \tag{3.10}$$

The components of vector $\mathbf{z}$ range from -1 to 1, hence the upper bound of the first term in (3.10) is given by $z_i = 1$ if $L_f\phi_i > 0$ and $z_i = -1$ if $L_f\phi_i < 0$, that is:

$$\mathbf{z}^{\mathrm{T}}\,L_f\boldsymbol{\phi} \leq \sum_{i=1}^{N} |L_f\phi_i| = \|L_f\boldsymbol{\phi}\|_1 \tag{3.11}$$

where $|\cdot|$ represents the absolute value function.

Assuming that $u^+ > 0$, the second term in (3.10) is negative, since matrix $\left(\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\right)$ is positive definite, and its upper bound is given by:

$$-\,\mathbf{z}^{\mathrm{T}}\,\left(\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\right)\,\mathbf{z}\,u^+ \leq -\mathrm{eig}_{\min}\left(\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\right)\,\|\mathbf{z}\|_2^2\;u^+, \tag{3.12}$$

$$\text{where}\qquad \|\mathbf{z}\|_2 \geq 1 \quad \forall\,\boldsymbol{\phi} \neq \mathbf{0}_N, \tag{3.13}$$

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

From (3.11), (3.12) and (3.13), the upper bound of the time derivative of the Lyapunov function V results in:

$$\dot{V} \leq \|L_f\boldsymbol{\phi}\|_1 - \mathrm{eig}_{\min}\left(\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\right)\;u^+. \tag{3.14}$$

Therefore, if u^+ fulfills (3.5) 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. That is, the origin $\boldsymbol{\phi} = \mathbf{0}_N$ is an asymptotically stable equilibrium point with finite time convergence.

3.2 One-side sliding mode

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

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{d}) + \mathbf{g}(\mathbf{x})\,\mathbf{u}, \tag{3.15}$$

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 $\phi_i(\mathbf{x}) \leq 0$, $i = 1,\ldots,N$, 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:

$$\Phi = \left\{\mathbf{x} \,|\, \phi_i(\mathbf{x}) \leq 0\right\}, \quad i = 1,\ldots,N. \tag{3.16}$$

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 Eq. (3.15) 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[4]:

$$\begin{aligned}\frac{d(\phi_i(\mathbf{x}))}{dt} &= \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\dot{\mathbf{x}} = \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{f}(\mathbf{x},\mathbf{d}) + \nabla\phi_i^{\mathrm{T}}(\mathbf{x})\mathbf{g}(\mathbf{x})\,\mathbf{u} \\ &= L_f\phi_i(\mathbf{x},\mathbf{d}) + \mathbf{L_g}\phi_i(\mathbf{x})\mathbf{u} \le 0, \quad \forall i \mid \phi_i(\mathbf{x}) \ge 0, \end{aligned} \tag{3.17}$$

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 $\phi_i(\mathbf{x}) \ge 0$ 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 Eq. (3.17). 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:

$$\mathbf{u} = \begin{cases} \mathbf{0} & \text{if} \quad \max_i\{\phi_i(\mathbf{x})\} < 0 \\ \mathbf{u}_c & \text{otherwise,} \end{cases} \tag{3.18}$$

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

$$\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{u}_c = -\mathbf{1}_b\,u^+, \tag{3.19}$$

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 a positive constant to be chosen high enough to satisfy Eq. (3.17). 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:

$$u^+ > \sum_{i=1}^{b}(\max(L_f\phi_i, 0)). \tag{3.20}$$

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

See *proof* in 3.2.1.

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

When the state trajectory tries by itself to leave the allowed region Φ, the above control law in Eq. (3.18) 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) (Edwards and Spurgeon (1998)). In fact, this approach can be coined as *one-side* sliding control. Once SM is established on the boundary of Φ by the control action $\mathbf{u}$, a continuous equivalent control (Edwards and Spurgeon (1998)) can be obtained, i.e., the control required to keep the system on the boundary of Φ. Hence, the SM generated by Eq. (3.18) produces such control action without explicit knowledge of it and with a low computational cost, which is a typical advantage of SM strategies (Utkin et al. (2009)).

The above SM method produces a non-smooth $\mathbf{u}$. If a smooth control action is wished, the *higher-order invariance* detailed in Section 3.1.2 can be used.

3.2.1 Proof of condition Eq. (3.20)

From Eqs. (3.17) and (3.19), the column vector $\dot{\boldsymbol{\phi}}$ composed of the constraint function derivatives $\dot{\phi}_i$ is given by:

$$\dot{\boldsymbol{\phi}} = L_f \boldsymbol{\phi} - \mathbf{z}\, u^+, \tag{3.21}$$

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

The goal of this proof is to show that $\boldsymbol{\phi} = \mathbf{0}$ is an asymptotically stable equilibrium point with finite time convergence. For this purpose, let $V = \mathbf{z}^{\mathrm{T}}\, \boldsymbol{\phi}$ be a 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 the components of vector $\boldsymbol{\phi}^{N-a}$ are not zero. Obviously, one of these two subvectors may be empty at a certain time. According to the continuous equivalent control (Edwards and Spurgeon (1998)), vector $\mathbf{z}^a$ must be replaced by the function $\mathbf{z}^a_{eq}$ such that $\dot{\boldsymbol{\phi}}^a = \mathbf{0}_a$. Because $\boldsymbol{\phi}^a = \mathbf{0}_a$ in SM, the time derivative of V results in:

$$\begin{aligned}\dot{V} = \frac{d}{dt}\left(\mathbf{z}^{\mathrm{T}}\, \boldsymbol{\phi}\right) =& \frac{d}{dt}\left(\begin{bmatrix} \mathbf{z}^a_{eq} \\ \mathbf{z}^{N-a} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{\phi}^a \\ \boldsymbol{\phi}^{N-a} \end{bmatrix}\right) \\ =& \begin{bmatrix} \dot{\mathbf{z}}^a_{eq} \\ \mathbf{0}_{N-a} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathbf{0}_a \\ \boldsymbol{\phi}^{N-a} \end{bmatrix} + \mathbf{z}^{\mathrm{T}}\, \dot{\boldsymbol{\phi}} = \mathbf{z}^{\mathrm{T}}\, \dot{\boldsymbol{\phi}},\end{aligned} \tag{3.22}$$

where $\mathbf{z}^{N-a}$ is a constant column vector with all its elements equal to 1 or 0.

Replacing vector $\dot{\boldsymbol{\phi}}$ with its value from Eq. (3.21), it is obtained:

$$\dot{V} = \mathbf{z}^{\mathrm{T}}\, L_f \boldsymbol{\phi} - \mathbf{z}^{\mathrm{T}}\, \mathbf{z}\, u^+. \tag{3.23}$$

The components of vector $\mathbf{z}$ range from 0 to 1, hence the upper bound of the first term in Eq. (3.23) is given by $z_i = 1$ if $L_f\phi_i > 0$ and $z_i = 0$ if $L_f\phi_i < 0$, that is:

$$\mathbf{z}^{\mathrm{T}}\, L_f \boldsymbol{\phi} \leq \sum_{i=1}^{N} (\max(L_f\phi_i, 0)). \tag{3.24}$$

Assuming that $u^+ > 0$, the second term in Eq. (3.23) is negative and its upper bound is given by:

$$-\,\mathbf{z}^{\mathrm{T}}\, \mathbf{z}\, u^+ = -\|\mathbf{z}\|_2^2\; u^+ \leq -u^+, \tag{3.25}$$

$$\text{where} \qquad \|\mathbf{z}\|_2 \geq 1 \quad \forall\, \boldsymbol{\phi} \neq \mathbf{0}_N, \tag{3.26}$$

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

From Eqs. (3.24), (3.25) and (3.26), the upper bound of the time derivative of the Lyapunov function V results in:

$$\dot{V} \leq \sum_{i=1}^{N} (\max(L_f\phi_i, 0)) - u^+. \tag{3.27}$$

Therefore, if u^+ fulfills Eq. (3.20) 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. That is, the origin $\boldsymbol{\phi} = \mathbf{0}_N$ is an asymptotically stable equilibrium point with finite time convergence.

In the next chapter, an approach based on sliding-mode ideas is proposed to satisfy different types of constraints in visual servoing.

Chapter 4

Fulfillment of constraints in visual servoing using sliding mode control

4.1 Introduction

As stated in Section 2.1, two classic visual servoing approaches are defined depending on the workspace in which the control law is computed (Chaumette and Hutchinson (2008)): Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS). Regardless of the workspace in where VS control laws are computed, the following mechanical constraints can be violated: *joint range limits*; *maximum joint speeds*; *workspace limits*; *task space limits*; and *forbidden areas*, defined to avoid collisions between the robot manipulator and objects in the environment. Furthermore, since the VS control law depends on the visual feedback, it is convenient to consider the so-called *visibility constraints* to ensure the visibility of the image features of the detected object, i.e., to avoid that the image features leave the camera field of view (FOV) and to avoid occlusions with the obstacles in the robot's environment[1].

Due to the fact that the violation of any of the aforementioned mechanical

[1]Some approaches (Garcia-Aracil et al. (2005); Garcia et al. (2014); Cazy et al. (2015)) 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.

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 (Kragic et al. (2002)): authors in Chesi et al. (2004) presented a switching method between IBVS and PBVS; authors in Gans and Hutchinson (2007) introduced a switching approach which uses the classic PBVS control law and backward motion along the camera optical axis; authors in Kim et al. (2009) proposed a switching approach using Hybrid Visual Servoing (HVS) control laws and pure translation motions; authors in Deng and Janabi-Sharifi (2005) introduced a path planning and PBVS-IBVS switching method in order to deal with image singularities and local minima; authors in Kermorgant and Chaumette (2011) presented a combination approach which uses 2D and 3D information from IBVS and PBVS to ensure the *visibility constraint*; and authors in Hafez and Jawahar (2007) 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 Deng and Janabi-Sharifi (2005) commented above, authors in Kyrki et al. (2004) presented a shortest-path method to guarantee both shortest Cartesian trajectory and object visibility; authors in Baumann et al. (2010) presented a path planning method which uses a probabilistic road map; authors in Chesi and Hung (2007) introduced a global path planning method to take into account visibility, workspace and joint constraints; authors in Chesi (2009) addressed the issue with a path planning approach based on the use of homogeneous forms and linear matrix inequalities (LMIs); authors in Kazemi et al. (2013) proposed a path planning approach using search trees and IBVS trajectory tracking; authors in Garcia et al. (2009) introduced a time-independent path tracking in the image and 3D space approach for unstructured environments; authors in Huang et al. (2014) 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 (GPM).

Furthermore, there are some proposals relying on the *online corrective* terms: authors in Corke and Hutchinson (2001) introduced a partitioned approach to IBVS control with the combination of a potential function for giving solution to the *visibility constraint* issue; authors in Mezouar and Chaumette (2002) developed a path-following IBVS controller that utilizes a potential function to incorporate *motion* constraints; and authors in Cowan et al. (2002) and Chen et al. (2007) 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 Hajiloo et al. (2016), Allibert et al. (2010a) and Heshmati-alamdari et al. (2014) introduced control laws based on model predictive control (MPC) frameworks, whilst authors in Song and Miaomiao (2017) on control Lyapunov functions (CLF). Moreover, authors in Nelson and Khosla (1995) and Chaumette and Marchand (2001) 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 Zhong et al. (2015), authors dealt with the visibility constraint problem using a neural network approach which assists a Kalman filter (NNAKF), whilst in Chesi and Vicino (2004), 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 Garcia-Aracil et al. (2005), the camera invariant VS approach is redefined to take into account the changes of visibility in image features, and in Mansard and Chaumette (2007), 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 chapter 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* (Rimon and Koditschek (1992)). 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 (SM) control (Edwards and Spurgeon (1998); Gracia et al. (2012a)). Concretely, in VS field of research SM control has been used mainly to increase the robustness against errors while executing the main robot control task (Zanne et al. (2000); Kim et al. (2006); Oliveira et al. (2009, 2014); Parra-Vega et al. (2003); Li and Xie (2010); Parsapour et al. (2015); Burger et al.

(2015); Becerra and Sagüés (2011); Becerra et al. (2011); Xin et al. (2016); Yu (2013)). However, SM techniques have not yet been used in VS to fulfill constraints.

Besides the SM algorithm to fulfill the constraints, another task with low-priority (Nakamura et al. (1987)) is considered to make as small as possible the reference tracking error in order to properly track the target object.

An industrial application using the proposed approach is also presented in this chapter: VS to provide a robust solution to the process of automated tool change carried out by manipulator robots.

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. Regardless of the working environment, nowadays the change is pre-programmed, requiring imperiously both the tool and its warehouse to be placed in fixed positions within the robot workspace. Using this procedure, 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.

For this application, IBVS scheme and eye-to-hand configuration are selected. 3D parameter estimation affects the accuracy of the reached pose in PBVS, whereas in IBVS it affects the camera motion but not the convergence (Chaumette and Hutchinson (2008)). Therefore, as the pose estimation accuracy is essential in a tool change procedure, the IBVS is the more appropriate method. Moreover, IBVS is inherently robust to camera calibration and target modeling errors **?**Hutchinson1996). 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 author knowledge, VS has not yet been used to address this problem. 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 and Ongaro (2016) to correct image distortion in order to obtain an accurrate location of the tool center point. Similarly, other works (Motta et al. (2001) Du and Zhang (2013) Yin et al. (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, as discussed before, 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. A specific phenomenon of IBVS control that could contribute to the unfulfillment of the mentioned constraints is discussed below.

Beside the IBVS advantages mentioned above, this technique has the drawback that a significant coupling between the end-effector translation and rotation motion control is present. In particular, the common choice of points as visual features and the coupling between the third and sixth columns in the interaction matrix, produce unnecessary translation motion when large rotation errors are considered, problem known as *camera retreat* (Chaumette (1998)). Some proposals deal with the camera retreat problem, for example a partitioned IBVS proposed to isolate the camera axis motions from the other camera DOF (Corke and Hutchinson (2001)), a weighted/parameterized IBVS (Nematollahi and Janabi-Sharifi (2009)) and a hybrid VS combining IBVS and PBVS (Malis et al. (1999)). The camera retreat problem shows its extremest version when a pure rotation error of $180°$ around the camera optical axis is considered, inducing a singularity in the interaction matrix (Corke and Hutchinson (2001)). In short, the camera retreat problem induces large motions, which could contribute to infringe the constraints mentioned above: leaving the allowed workspace, exceeding the joint range limits and exceeding the maximum joint speeds.

The fulfillment of constraints in this robot auto tool change industrial application is also addressed with the SM control proposed in this chapter.

Problem definition. The goal of the proposed approach 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 is as close as possible to the given reference value;

– the visual features of the target object remain visible throughout the process;

– the joint range limits and the maximum joint speeds are not exceeded;

– the robot workspace and task space are not exceeded;

– and the robot does not collide during the motion with the objects of the environment that are located within its workspace.

The main contents of this chapter are represented using a mind map in Figure 4.1.

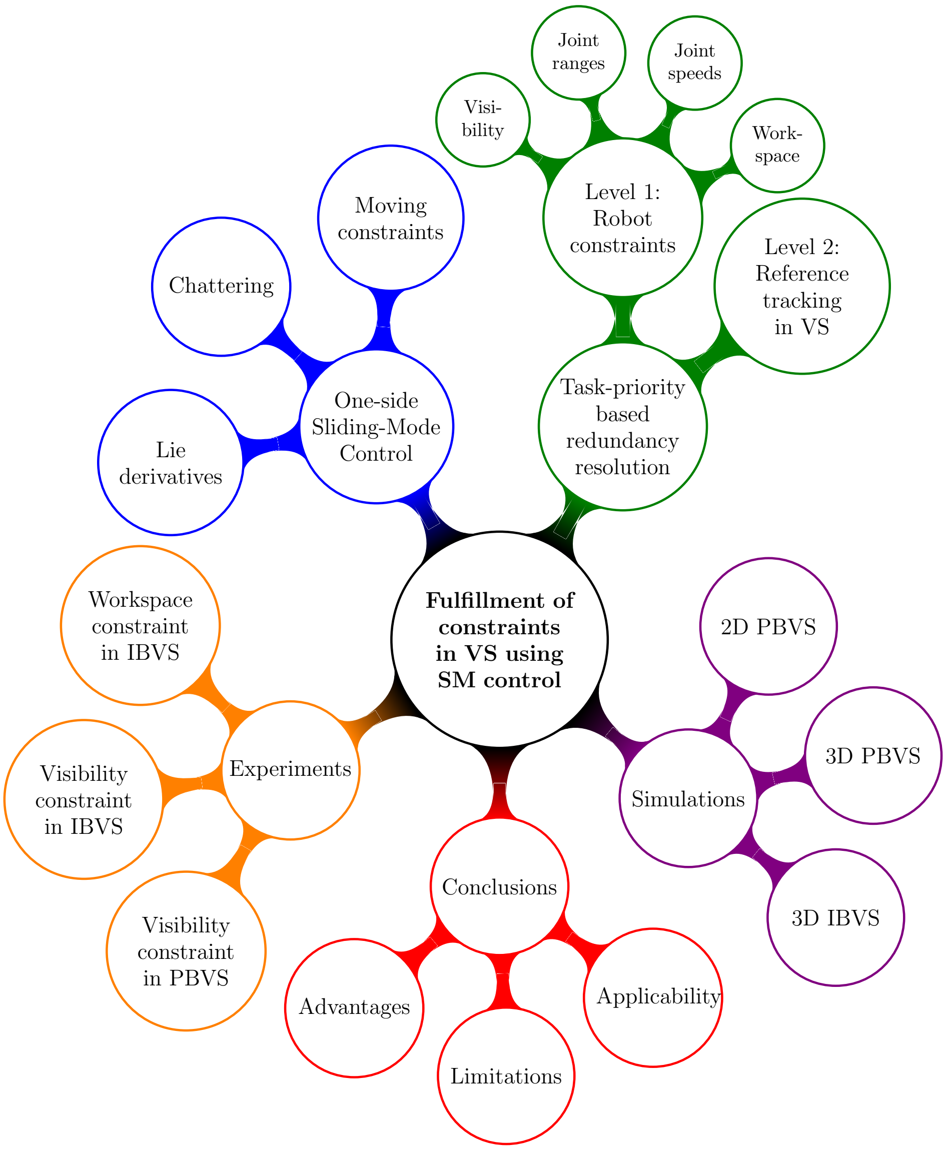

Fulfillment of constraints in VS using SM control
One-side Sliding-Mode Control
Moving constraints
Chattering
Lie derivatives
Task-priority based redundancy resolution
Level 1: Robot constraints
Visi- bility
Joint ranges
Joint speeds
Work- space
Level 2: Reference tracking in VS
Experiments
Workspace constraint in IBVS
Visibility constraint in IBVS
Visibility constraint in PBVS
Conclusions
Advantages
Limitations
Applicability
Simulations
2D PBVS
3D PBVS
3D IBVS

Figure 4.1: Mind map representing the main contents of this chapter.

4.2 Preliminaries

The content of this preliminaries section has been presented with full details in previous sections, but a small reminder is done here for better readability of this chapter.

Underlying robot controller. Robot joint acceleration controller is considered in this chapter:

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_c + \mathbf{d}_c, \tag{4.1}$$

where subscript c is used for the commanded variable and $\mathbf{d}_c$ represents the inaccuracies of the low-level control loop.

Kinematics. The first- and second-order kinematics of the visual features vector $\mathbf{s}$ are:

$$\dot{\mathbf{s}} = \mathbf{J}_s\dot{\mathbf{q}} + \partial\mathbf{s}/\partial t \tag{4.2}$$

$$\ddot{\mathbf{s}} = \mathbf{J}_s\ddot{\mathbf{q}} + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{s}}/\partial t, \tag{4.3}$$

where $\partial\mathbf{s}/\partial t$ is due to the target motion and $\mathbf{J}_s$ is the Jacobian matrix.

Dynamic model of robot manipulators. In this work, the dynamic model of the robot manipulators is not explicitly obtained, although it is relevant and should be considered in the following cases:

– For an optimal design of the underlying robot joint controller.

– To dynamically set the limits of the speed and acceleration constraints. The proposed method would still work if those limits are calculated online, as explained in Section 4.3.6.2, with the only condition of being first-order differentiable (what is reasonable in a real physic system).

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

$$\mathbf{s}(\mathbf{q}, t) = \mathbf{s}_{ref}(t), \tag{4.4}$$

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

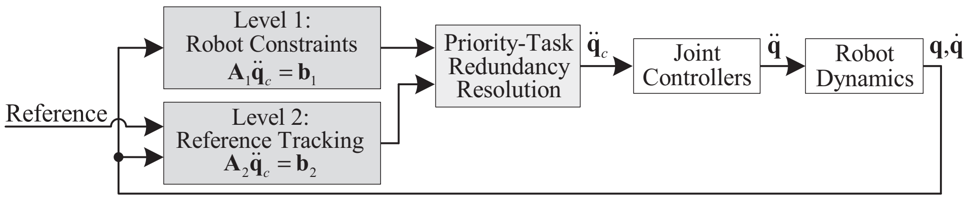

Level 1:
Robot Constraints
$\mathbf{A}_1\ddot{\mathbf{q}}_c = \mathbf{b}_1$
Level 2:
Reference Tracking
$\mathbf{A}_2\ddot{\mathbf{q}}_c = \mathbf{b}_2$
Reference
Priority-Task Redundancy Resolution
$\ddot{\mathbf{q}}_c$
Joint Controllers
$\ddot{\mathbf{q}}$
Robot Dynamics
$\mathbf{q},\dot{\mathbf{q}}$

Figure 4.2: Overview of the proposed approach.

Simulations. The simulation results presented below were obtained using MATLAB®. Details of pseudo-code and computing time for actual implementation of the proposed strategy appears in Section 4.3.7.

4.3 Proposed approach

The approach developed in this section to address the problem defined in Section 4.1 is based on task-priority redundancy resolution, geometric invariance and SM ideas reviewed in previous section.

4.3.1 System tasks

Fig. 4.2 shows the overview of the proposal, where task-priority redundancy resolution is used with two hierarchical priority levels:

– The first level includes the *visibility* and *motion constraints*. Motion constraints 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 robot workspace limits; exceeding a predefined task space; colliding with objects in the robot environment. Visibility constraints are defined to avoid that the image features leave the camera FOV and to avoid occlusions with the obstacles in the robot's workspace, since these features are required to compute the robot control law.

– 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 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:

$$\mathbf{A}_1\ddot{\mathbf{q}}_c = \mathbf{b}_1 \tag{4.5}$$

$$\mathbf{A}_2\ddot{\mathbf{q}}_c = \mathbf{b}_2, \tag{4.6}$$

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 geometric invariance theory and one-side SM control presented in Section 3.2, 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 (see 2.3) as follows:

$$\ddot{\mathbf{q}}_c = \mathbf{A}_1^\dagger\mathbf{b}_1 + (\mathbf{A}_2(\mathbf{I} - \mathbf{A}_1^\dagger\mathbf{A}_1))^\dagger(\mathbf{b}_2 - \mathbf{A}_2\mathbf{A}_1^\dagger\mathbf{b}_1), \tag{4.7}$$

where $\mathbf{I}$ denotes the identity matrix of suitable size and superscript $\dagger$ denotes the well-known Moore-Penrose pseudoinverse[2].

4.3.2 Lie derivatives

In order to use the theory in Section 3.2, a dynamical system in the form of (3.15) 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 (4.1) the state equation results in:

$$\dot{\mathbf{x}} = \begin{bmatrix}\mathbf{O} & \mathbf{I}\\ \mathbf{O} & \mathbf{O}\end{bmatrix}\mathbf{x} + \begin{bmatrix}\mathbf{0}\\ \mathbf{d}_c\end{bmatrix} + \begin{bmatrix}\mathbf{O}\\ \mathbf{I}\end{bmatrix}\mathbf{u}, \tag{4.8}$$

and, therefore, the Lie derivatives in (3.17) for the constraint function ϕ_i are given by:

$$\mathbf{L}_{\mathbf{g}}\phi_i = \nabla\phi_i^{\mathrm{T}}\,\mathbf{g} = (\partial\phi_i/\partial\dot{\mathbf{q}})^{\mathrm{T}} \tag{4.9}$$

$$L_f\phi_i = \nabla\phi_i^{\mathrm{T}}\,\mathbf{f} = (\partial\phi_i/\partial\mathbf{q})^{\mathrm{T}}\,\dot{\mathbf{q}} + (\partial\phi_i/\partial\dot{\mathbf{q}})^{\mathrm{T}}\,\mathbf{d}_c. \tag{4.10}$$

[2]Pseudoinverse may be computed via the singular value decomposition (SVD) method (Golub and Van Loan (1996)) and using a tolerance to set to zero the very small singular values in order to avoid extremely large values for the commanded accelerations.

4.3.3 Level 1: robot constraints

The first level includes *visibility* and *motion constraints.* To satisfy these constraints, the SM control of Section 3.2 is considered with the dynamical system and Lie derivatives Eqs. (4.9) and (4.10).

4.3.3.1 Visibility constraints

FOV constraints. To avoid that the image features of the target object leave the camera FOV, a constraint is defined for each image feature in order to confine them within the image plane limits. For this purpose, the formula of the *superellipse*, also known as a Lamé curve, is considered: $|x/a|^n + |y/b|^n = 1$, that for $n > 2$ is called hyperellipse and looks like a rectangle with rounded corners, see Fig. 4.3. Therefore, the visibility constraint $\sigma_{V,i}$ for the target's image features i is given by:

$$\begin{aligned}\sigma_{V,i}(u_i, v_i) = & \left|\frac{u_i}{\frac{W_V}{2}(1-m_V)}\right|^{n_V} + \left|\frac{v_i}{\frac{H_V}{2}(1-m_V)}\right|^{n_V} \\ & - 1 \leq 0,\end{aligned} \tag{4.11}$$

where u_i and v_i are the pixel coordinates of the image feature i in the image plane with respect to a coordinate system UV located in the center of the image plane and parallel to the rectangle representing the image plane limits (thick dark line in Fig. 4.3); 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. 4.3); W_V and H_V are the width and height, respectively, of the rectangle representing the image plane limits in pixels; and m_V is the safety margin for the visibility constraints to cater for possible errors and inaccuracies in order to prevent that the target's image features accidentally leave the camera FOV, e.g., Fig. 4.3 uses a safety margin of 5%.

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 3.1.2 for the sliding manifold to have relative degree one with respect to the control variable $\ddot{\mathbf{q}}_c$, that is:

$$\begin{aligned}\phi_{V,i}(\mathbf{q}, \dot{\mathbf{q}}) = & \sigma_{V,i}(\mathbf{q}) + K_{V,i}\frac{d\sigma_{V,i}(\mathbf{q})}{dt} \\ = & \sigma_{V,i} + K_{V,i}\, \nabla\sigma_{V,i}^{\mathrm{T}}\, \dot{\mathbf{q}} \leq 0,\end{aligned} \tag{4.12}$$

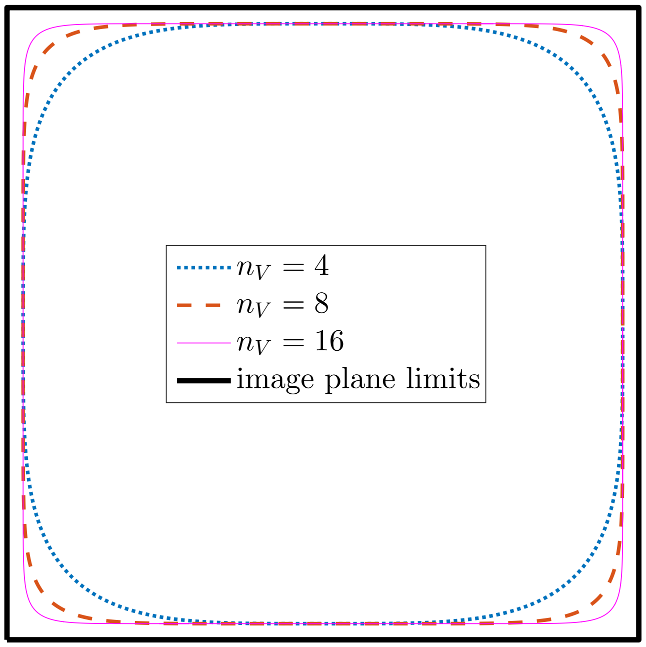


Figure 4.3: Boundary of the visibility constraint (i.e., $\sigma_{V,i} = 0$) for a safety margin m_V of 5% and diffent values of n_V.

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

The gradient vectors $\nabla\sigma_{V,i}$ for the visibility constraints result in:

$$\begin{aligned}\nabla\sigma_{V,i} &= \begin{bmatrix}\partial u_i/\partial\mathbf{q} & \partial v_i/\partial\mathbf{q}\end{bmatrix}\begin{bmatrix}\partial\sigma_{V,i}/\partial u_i\\ \partial\sigma_{V,i}/\partial v_i\end{bmatrix}\\ &=(\mathbf{L}_x\,{}^c\mathbf{V}_e\,{}^e\mathbf{J}_e)^{\mathrm{T}}\begin{bmatrix}\partial\sigma_{V,i}/\partial u_i\\ \partial\sigma_{V,i}/\partial v_i\end{bmatrix},\end{aligned} \tag{4.13}$$

where the partial derivatives $\partial\sigma_{V,i}/\partial u_i$ and $\partial\sigma_{V,i}/\partial v_i$ are straightforward obtained from Eq. (4.11) as:

$$\frac{\partial \sigma_{V,i}}{\partial u_i} = \frac{n_V \mathrm{sign}(u_i)\,|u_i|^{n_V-1}}{\left(\frac{W_V}{2}(1-m_V)\right)^{n_V}} \tag{4.14}$$

$$\frac{\partial \sigma_{V,i}}{\partial v_i} = \frac{n_V \mathrm{sign}(v_i)\,|v_i|^{n_V-1}}{\left(\frac{H_V}{2}(1-m_V)\right)^{n_V}}, \tag{4.15}$$

and $\mathbf{L}_x$ represents the well-known interaction matrix typically used in IBVS, which is given by:

$$L_x = \begin{bmatrix} -\frac{f}{Z} & 0 & \frac{u_i}{Z} & \frac{u_i v_i}{f} & -\frac{f^2+u_i^2}{f} & v_i \\ 0 & -\frac{f}{Z} & \frac{v_i}{Z} & \frac{f^2+v_i^2}{f} & -\frac{u_i v_i}{f} & -u_i \end{bmatrix}, \tag{4.16}$$

where f is the focal length of the camera lens in pixels and Z is the distance from the image plane to the target object, which is estimated by the computer vision algorithm described in Section 2.1.8.

Occlusion constraints. The image features may be occluded in the image plane by the obstacles in the robot's workspace that are closer to the camera than the target object. To avoid this situation, a constraint can be used to guarantee that the image features do not enter the area defined by these obstacles in the image plane, which represents a forbidden area. The procedure is as follows: in the first place, computer vision algorithms 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 constrain, function $\sigma_{V,i}$ is negative for a point outside the enclosed area and positive otherwise. The gradient vector computation is given by (4.13), (4.16) 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.

4.3.3.2 Constraints for the joint range limits

The following constraints are considered for the joint limits:

$$\begin{aligned}\sigma_{R,qi}(\mathbf{q}) =& -1 + \frac{|\,q_i - q_{\mathrm{mid},i}|}{\Delta q_{\max,i}/2} + m_{R,q} \\ =& -1 + |\,\widetilde{q}_i| \ + m_{R,q} \leq 0, \quad i = 1, \ldots, n,\end{aligned} \tag{4.17}$$

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 proposed in Section 3.1.2 for the sliding manifold to have relative degree one[3] with respect to the control variable $\ddot{\mathbf{q}}_c$, that is:

$$\begin{aligned}\phi_{R,qi}(\mathbf{q}, \dot{\mathbf{q}}) =& \sigma_{R,qi}(\mathbf{q}) + K_{R,qi} \frac{d\sigma_{R,qi}(\mathbf{q})}{dt} \\ =& \sigma_{R,qi} + K_{R,qi} \, \nabla\sigma_{R,qi}^{\mathrm{T}} \, \dot{\mathbf{q}} \leq 0,\end{aligned} \tag{4.18}$$

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.

The gradient vectors $\nabla\sigma_{R,qi}$ for the joint range constraints are straightforward obtained from Eq. (4.17) as:

$$\nabla\sigma_{R,qi} = \begin{bmatrix} 0 & \cdots & \mathrm{sign}(\widetilde{q}_i) & \cdots & 0 \end{bmatrix}^{\mathrm{T}}. \tag{4.19}$$

4.3.3.3 Constraints for the maximum joint speeds

The following constraints are considered for the joint speeds:

$$\begin{aligned}\phi_{R,si}(\dot{\mathbf{q}}) =& -1 + \frac{|\,\dot{q}_i|}{\dot{q}_{\max,i}} + m_{R,s} \\ =& -1 + \left|\,\widetilde{\dot{q}}_i\right| \ + m_{R,s} \leq 0, \quad i = 1, \ldots, n,\end{aligned} \tag{4.20}$$

[3]From Eq. (4.1), 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 and Spurgeon (1998)).

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 to cater for possible errors and inaccuracies in order to avoid reaching the maximum joint speeds.

In this case, higher-order invariance is not required 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., $\dot{\phi}_{R,si}$ (and $\ddot{\mathbf{q}}$) explicitly depends on signal $\ddot{\mathbf{q}}_c$, as required by SM theory (Edwards and Spurgeon (1998)).

The gradient vectors $\nabla\phi_{R,si}$ for joint speed constraints are straightforward obtained from Eq. (4.20) as:

$$\nabla\phi_{R,si} = \begin{bmatrix} 0 & \cdots & \operatorname{sign}(\widetilde{\dot{q}}_i) & \cdots & 0 \end{bmatrix}^{\mathrm{T}}. \tag{4.21}$$

4.3.3.4 Workspace constraints: object collision avoidance, task space limits and robot workspace limits

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 perform and unnecessary or not desired movements (like the camera retreat phenomenon (Chaumette (1998)) mentioned in Section 4.1) should be controlled; 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 $\Phi_{WS}(\mathbf{p}_j) = \{\mathbf{p}_j \mid \sigma_{R,wsi}(\mathbf{p}_j) \leq 0 \;\forall\, i\}$, 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. Hence, the allowed C-space results in $\Phi_{CS}(\mathbf{q}) = \{\mathbf{q} \mid \sigma_{R,wsi}(\mathbf{l}_j(\mathbf{q})) = \sigma_{R,wsij}(\mathbf{q}) \leq 0 \;\forall\, i,j\}$, where $\mathbf{l}_j$ is the kinematic function of the Cartesian position of point j.

As before, since constraint functions $\sigma_{R,wsij}$ depend on the robot configuration $\mathbf{q}$, they will be modified as indicated in Section 3.1.2 for the sliding

[4]For simplicity, both speed limits are considered symmetric. If that would not be case, constraints in Eq. (4.20) can be readily split into two constraints for maximum and minimum speeds. Details omitted for brevity.

manifold to have relative degree one with respect to the control variable $\ddot{\mathbf{q}}_c$, that is:

$$\begin{aligned}\phi_{R,wsij}(\mathbf{q},\dot{\mathbf{q}}) =& \sigma_{R,wsij}(\mathbf{q}) + K_{R,wsi}\frac{d\sigma_{R,wsij}(\mathbf{q})}{dt} \\ =& \sigma_{R,wsij} + K_{R,wsi}\,\nabla\sigma_{R,wsij}^{\mathrm{T}}\,\dot{\mathbf{q}} \leq 0, \end{aligned} \tag{4.22}$$

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 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. Regarding the workspace limits' constraint, typically only the robot end-effector is considered.

4.3.3.5 Acceleration equality for Level 1

The partial derivatives of the robot constraint functions $\phi_{V,i}$, $\phi_{R,qi}$, $\phi_{R,si}$ and $\phi_{R,wsij}$ are needed to compute the Lie derivatives $\{\mathbf{L_g}\phi_{V,i}, \mathbf{L_g}\phi_{R,qi}, \mathbf{L_g}\phi_{R,si}, \mathbf{L_g}\phi_{R,wsij}\}$ and $\{L_f\phi_{V,i}, L_f\phi_{R,qi}, L_f\phi_{R,si}, L_f\phi_{R,wsij}\}$ with (4.9)–(4.10). From Eqs. (4.12), (4.18), (4.20) and (4.22), these partial derivatives result in:

$$(\partial\phi_{V,i}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{V,i}^{\mathrm{T}} + K_{V,i}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma V,i} \tag{4.23}$$

$$(\partial\phi_{R,qi}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{R,qi}^{\mathrm{T}} + K_{R,qi}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma R,qi} \tag{4.24}$$

$$(\partial\phi_{R,si}/\partial\mathbf{q})^{\mathrm{T}} = 0 \tag{4.25}$$

$$(\partial\phi_{R,wsij}/\partial\mathbf{q})^{\mathrm{T}} = \nabla\sigma_{R,wsij}^{\mathrm{T}} + K_{R,wsi}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{H}_{\sigma R,wsij} \tag{4.26}$$

$$(\partial\phi_{V,i}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = K_{V,i}\nabla\sigma_{V,i}^{\mathrm{T}} \tag{4.27}$$

$$\begin{aligned}(\partial\phi_{R,qi}/\partial\dot{\mathbf{q}})^{\mathrm{T}} =& K_{R,qi}\nabla\sigma_{R,qi}^{\mathrm{T}} \\ =& K_{R,qi}\begin{bmatrix}0 & \cdots & \mathrm{sign}(\widetilde{q}_i) & \cdots & 0\end{bmatrix}\end{aligned} \tag{4.28}$$

$$\begin{aligned}(\partial\phi_{R,si}/\partial\dot{\mathbf{q}})^{\mathrm{T}} =& \nabla\phi_{R,si}^{\mathrm{T}} \\ =& \begin{bmatrix}0 & \cdots & \mathrm{sign}(\widetilde{\dot{q}}_i) & \cdots & 0\end{bmatrix}\end{aligned} \tag{4.29}$$

$$(\partial\phi_{R,wsij}/\partial\dot{\mathbf{q}})^{\mathrm{T}} = K_{R,wsi}\nabla\sigma_{R,wsij}^{\mathrm{T}} \tag{4.30}$$

where $\mathbf{H}_{\sigma V,i}$, $\mathbf{H}_{\sigma R,qi}$ and $\mathbf{H}_{\sigma R,wsij}$ denote the Hessian matrix of second-order partial derivatives of $\sigma_{V,i}$, $\sigma_{R,qi}$ and $\sigma_{R,wsij}$, respectively, sign$(\cdot)$ represents the *sign* function and all the elements except the ith of row vectors in Eqs. (4.28) and (4.29) are zero.

Thus, according to (4.9) and (4.27)–(4.30), equation (3.19) for the first priority level is given by:

$$\begin{bmatrix} \mathbf{K}_V \ \nabla\boldsymbol{\sigma}_V^{\mathrm{T}} \\ \mathbf{K}_{R,q} \ \nabla\boldsymbol{\sigma}_{R,q}^{\mathrm{T}} \\ \nabla\boldsymbol{\phi}_{R,s}^{\mathrm{T}} \\ \mathbf{K}_{R,ws} \ \nabla\boldsymbol{\sigma}_{R,o}^{\mathrm{T}} \end{bmatrix} = - \begin{bmatrix} \mathbf{1}_{b,V} u_V^+, \\ \mathbf{1}_{b,q} u_{R,q}^+ \\ \mathbf{1}_{b,s} u_{R,s}^+ \\ \mathbf{1}_{b,ws} u_{R,ws}^+ \end{bmatrix}$$
$$= \mathbf{L_g}\boldsymbol{\phi}_R \ddot{\mathbf{q}}_c = - \mathbf{u}_R^+, \tag{4.31}$$

where $\mathbf{K}_V$, $\mathbf{K}_{R,q}$ and $\mathbf{K}_{R,ws}$ are diagonal matrices with diagonal entries $K_{V,i}$, $K_{R,qi}$ and $K_{R,wsij}$, respectively; matrices $\{\nabla\boldsymbol{\sigma}_V, \nabla\boldsymbol{\sigma}_{R,q}, \nabla\boldsymbol{\phi}_{R,s}, \nabla\boldsymbol{\sigma}_{R,ws}\}$ contain the vectors $\{\nabla\sigma_{V,i}, \nabla\sigma_{R,qi}, \nabla\phi_{R,si}, \nabla\sigma_{R,wsij}\}$, see (4.27)–(4.30), of all *active* constraints; $\{u_V^+, u_{R,q}^+, u_{R,s}^+, u_{R,ws}^+\}$ are the chosen value of u^+ for each type of robot constraint; and $\{\mathbf{1}_{b,V}, \mathbf{1}_{b,q}, \mathbf{1}_{b,s}, \mathbf{1}_{b,ws}\}$ are column vectors with all its components equal to one and their size is equal to the number of active constraints of each type.

Therefore, by comparing the acceleration equality (4.31) for the first level with equation (4.5), it is obtained that $\mathbf{A}_1 = \mathbf{L_g}\boldsymbol{\phi}_R$ and $\mathbf{b}_1 = -\mathbf{u}_R^+$.

4.3.3.6 Gradient vectors for Level 1

According to (4.31), 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 $\nabla\sigma_{V,i}$, $\nabla\sigma_{R,qi}$ and $\nabla\phi_{R,si}$ for visibility constraints, the joint range and joint speed constraints are straightforward obtained from (4.27), (4.28) and (4.29), respectively, as:

$$\nabla\sigma_{V,i} = \begin{bmatrix} \partial u_i/\partial\mathbf{q} & \partial v_i/\partial\mathbf{q} \end{bmatrix} \begin{bmatrix} \partial\sigma_{V,i}/\partial u_i \\ \partial\sigma_{V,i}/\partial v_i \end{bmatrix} = (\mathbf{L}_x \, {}^c\mathbf{V}_e \, {}^e\mathbf{J}_e)^{\mathrm{T}} \begin{bmatrix} \partial\sigma_{V,i}/\partial u_i \\ \partial\sigma_{V,i}/\partial v_i \end{bmatrix} \tag{4.32}$$

$$\nabla\sigma_{R,qi} = \begin{bmatrix} 0 & \cdots & \mathrm{sign}(\widetilde{q}_i) & \cdots & 0 \end{bmatrix}^{\mathrm{T}} \tag{4.33}$$

$$\nabla\phi_{R,si} = \begin{bmatrix} 0 & \cdots & \mathrm{sign}(\widetilde{\dot{q}}_i) & \cdots & 0 \end{bmatrix}^{\mathrm{T}}. \tag{4.34}$$

The partial derivatives $\partial\sigma_{V,i}/\partial u_i$ and $\partial\sigma_{V,i}/\partial v_i$ are straightforward obtained from (4.11) as:

$$\frac{\partial\sigma_{V,i}}{\partial u_i} = \frac{n_V \mathrm{sign}(u_i)\,|u_i|^{n_V-1}}{\left(\frac{W_V}{2}(1-m_V)\right)^{n_V}} \tag{4.35}$$

$$\frac{\partial\sigma_{V,i}}{\partial v_i} = \frac{n_V \mathrm{sign}(v_i)\,|v_i|^{n_V-1}}{\left(\frac{H_V}{2}(1-m_V)\right)^{n_V}}, \tag{4.36}$$

and $\mathbf{L}_x$ represents the well-known interaction matrix typically used in IBVS (4.16).

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

$$\nabla\sigma_{R,wsij} = (\partial\mathbf{p}_j/\partial\mathbf{q})\,(\partial\sigma_{R,wsi}/\partial\mathbf{p}_j) = {}^0\mathbf{J}_{pj}^{\mathrm{T}}\,(\partial\sigma_{R,wsi}/\partial\mathbf{p}_j)\,, \tag{4.37}$$

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.

4.3.4 Level 2: reference tracking

For the reference tracking, this work considers the classical operational space robot control (Siciliano et al. (2009)), that taking into account (4.3) and (4.1), results in:

$$\mathbf{J}_s\ddot{\mathbf{q}}_c = \ddot{\mathbf{s}}_c - (\mathbf{J}_s\mathbf{d}_c + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{s}}/\partial t), \tag{4.38}$$

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

Moreover, considering the classical acceleraton-based kinematic controller used for trajectory tracking (Khalil and Dombre (2002)), 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:

$$\ddot{\mathbf{s}}_c = \ddot{\mathbf{s}}_{ref} - K_{T,p}\mathbf{e} - K_{T,v}\dot{\mathbf{e}}, \tag{4.39}$$

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.

It is interesting to remark that the acceleration-based robot control given by (4.38)–(4.39) has already been used in VS appliactions by Fakhry and Wilson (1996) for PBVS and by Keshmiri et al. (2014) for IBVS.

4.3.4.1 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:

$$\begin{aligned} K_{T,p}(\mathbf{e}) = &- (K_{T,p}(\mathbf{0}) - K_{T,p}(\infty))e^{-\dfrac{\mathbf{e}\,\dot{K}_{T,p}(\mathbf{0})}{K_{T,p}(\mathbf{0}) - K_{T,p}(\infty)}} \\ &+ K_{T,p}(\infty), \end{aligned} \tag{4.40}$$

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 the previous 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 promptly the reference value.

4.3.4.2 Visual features and Jacobian matrix for PBVS

The typical visual feature vector and interaction matrix used in PBVS are considered:

$$\mathbf{s} = \begin{bmatrix} {}^{C*}\mathbf{t}_C^{\mathrm{T}} & {}^{C*}\theta\mathbf{u}_C^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{4.41}$$

$$\mathbf{L}_s = \begin{bmatrix} {}^{C*}\mathbf{R}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}. \tag{4.42}$$

The Jacobian matrix $\mathbf{J}_s$ required for the control law (4.38) of the reference tracking is computed 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 ${}^{e}\mathbf{J}_e$. For further details see Section 2.1.

4.3.4.3 Visual features and Jacobian matrix for IBVS

The typical visual feature vector and interaction matrix used in IBVS are considered:

$$\mathbf{s} = [u_i \quad v_i]^{\mathrm{T}}, \tag{4.43}$$

$$L_s = \begin{bmatrix} -\dfrac{f}{z} & 0 & \dfrac{u_i}{z} & \dfrac{u_i v_i}{f} & -\dfrac{f^2+u_i^2}{f} & v_i \\ 0 & -\dfrac{f}{z} & \dfrac{v_i}{z} & \dfrac{f^2+v_i^2}{f} & -\dfrac{u_i v_i}{f} & -u_i \end{bmatrix}. \tag{4.44}$$

The Jacobian matrix $\mathbf{J}_s$ required for the control law (4.38) of the reference tracking is again computed using $\mathbf{L}_s$, ${}^c\mathbf{V}_e$ and ${}^e\mathbf{J}_e$. For further details see Section 2.1.

4.3.5 Chattering

Discrete-time implementations of any practical SM control 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 and Spurgeon (1998)). In a similar manner to other robotic applications based on SM control (Gracia et al. (2012b, 2013)), 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 (3.19), that is:

$$\triangle\boldsymbol{\phi} = T_s \left|\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{u}_c\right| = T_s\, u^+\, \mathbf{1}_b, \tag{4.45}$$

where T_s is the sampling time of the robotic system and the value of u^+ is $\{u^+_{R,q}, u^+_{R,s}, u^+_{R,o}, u^+_V\}$ for the robot and visibility constraints. This chattering amplitude must be lower than the error allowed in the fulfillment of the constraints. For this purpose, the safety margins $\{m_{R,q}, m_{R,s}, m_{R,o}, m_V\}$ for the constraints should be cautiously chosen. Not also that, a big number could be chosen for u^+ in order to ensure that it is greater than the lower bound given by (3.20) and (4.10), as usual in SM applications. Nevertheless, from (4.45), such big numbers may induce unnecessary chattering amplitude, so it is a design trade off.

4.3.6 Additional remarks

4.3.6.1 Guidelines for the paramaters design

Hyperellipse smoothing parameter. The value of n_V should be large enough to fully utilize the rectangle representing the image plane limits (i.e., the rounded corners of the hyperellipse get less smooth), but not too large to avoid the numerical instability that large exponents may cause. For instance, typical values range from 4 to 16.

Safety margins for the constraints. The value of $\{m_V, m_{R,q}, m_{R,s}, m_{R,ws}\}$ should be as small as possible in order to fully utilize the available allowed space (e.g., the rectangle representing the image plane limits for the visibility 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 (e.g., the camera FOV).

Constraint approaching parameters. The value of constraint approaching parameters $\{K_{V,i}, K_{R,qi}, K_{R,wsij}\}$ for the 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 $\{u_V^+, u_{R,q}^+, u_{R,s}^+, u_{R,ws}^+\}$ should be as close as possible to the lower bound given by (3.20) (perhaps with some margin) to have reduced chattering band and high chattering frequency (Section 4.3.5).

Sampling time. The sampling time T_s should be small enough to have small chattering band (4.45). The minimum possible value is determined by the computation time of one iteration of the proposed algorithm (see Section 4.3.7).

4.3.6.2 Moving constraints

The proposed approach can also be used if there are moving constraints, e.g., moving obstacles for the collision avoidance constraints or a *moving target object* for the visibility constraints. In that case ϕ_i also depends explicitly on time and, therefore, the derivative of ϕ_i in equation (3.17) 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 (4.9) and (4.10), 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 u^+ is changed when moving constraints are considered and, therefore, the iterative computation of the algorithm remains the same and a large-enough constant u^+ will suffice for practical implementation.

4.3.6.3 Time derivatives

The proposed approach requires the first-order time derivatives of some variables: $\sigma_{V,i}$ for the SM control of the visibility constraints; and $\{\mathbf{q}, \mathbf{e}, \mathbf{J}_s, \partial\mathbf{s}/\partial t\}$ for the VS tracking controller. However, this situation is not new in VS applications: Fakhry and Wilson (1996) and Keshmiri et al. (2014) proposed for PBVS and IBVS, respectively, the same acceleration-based robot control used in this work. As in many other applications, the simplest way to deal with this issue consists in using numerical differentiation, e.g., the well-known backward Euler approximation. However, some kind of filtering should be previously applied to the actual variable when non-negligible noise is present. It is important to remark that, the low-pass filter used for noise reduction must not limit the band-width of the control law. That is, the bandwidth of the control law should not exceed the bandwidth of the low-pass filter. For the experiments in next sections, no kind of filtering was applied since the sampling time was relatively large and the influence of the noise on the numerical time derivatives was negligible.

4.3.7 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 motion and visibility constraints: $\{\phi_{V,i}(\mathbf{q},\dot{\mathbf{q}}), \phi_{R,qi}(\mathbf{q},\dot{\mathbf{q}}), \phi_{R,si}(\dot{\mathbf{q}}), \phi_{R,wsij}(\mathbf{q},\dot{\mathbf{q}})\}$ and $\{\nabla\sigma_{V,i}(\mathbf{q}), \nabla\sigma_{R,qi}(\mathbf{q}), \nabla\phi_{R,si}(\dot{\mathbf{q}}), \nabla\sigma_{R,wsij}(\mathbf{q})\}$.

– Jacobian matrix: $\mathbf{J}_s(\mathbf{q}, t)$.

– Visual feature vector and its reference: $\mathbf{s}(\mathbf{q}, t)$ and $\mathbf{s}_{ref}(t)$.

– Moore-Penrose pseudoinverse function: $(\cdot)^\dagger$.

– 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 4.5.

Algorithm executed at sampling time of T_s seconds

```
1  while $s < s_{end}$ do
2      $[\mathbf{q}, \dot{\mathbf{q}}]$ =GetRobotState();
3      $\dot{\mathbf{s}} = (\mathbf{s} - \mathbf{s}_{prev})/T_s$ ;                                        // Derivative
4      $\dot{\mathbf{s}}_{ref} = (\mathbf{s}_{ref} - \mathbf{s}_{ref,prev})/T_s$ ;                        // Derivative
5      $\ddot{\mathbf{s}}_{ref} = (\dot{\mathbf{s}}_{ref} - \dot{\mathbf{s}}_{ref,prev})/T_s$ ;            // Derivative
6      $\dot{\mathbf{J}}_s = (\mathbf{J}_s - \mathbf{J}_{s,prev})/T_s$ ;                                  // 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. (4.39)
8      $\mathbf{A}_1 = \begin{bmatrix} \mathbf{K}_V \, \nabla\boldsymbol{\sigma}_V^{\mathrm{T}} \\ \mathbf{K}_{R,q} \, \nabla\boldsymbol{\sigma}_{R,q}^{\mathrm{T}} \\ \nabla\boldsymbol{\phi}_{R,s}^{\mathrm{T}} \\ \mathbf{K}_{R,ws} \, \nabla\boldsymbol{\sigma}_{R,ws}^{\mathrm{T}} \end{bmatrix}$ with the gradients of all active constraints:
       $\phi_{V,i} > 0$, $\phi_{R,qi} > 0$, $\phi_{R,si} > 0$, $\phi_{R,wsij} > 0$ ;                  // Eq. (4.31)
9    $\mathbf{b}_1 = -\mathbf{u}_1^+$ ;                                                                   // Eq. (4.31)
10   $\mathbf{A}_2 = \mathbf{J}_s$ ;                                                                       // Eq. (4.38)
11   $\mathbf{b}_2 = \ddot{\mathbf{s}}_c - \dot{\mathbf{J}}_s\dot{\mathbf{q}} - \partial\dot{\mathbf{s}}/\partial t$ ;   // Eq. (4.38)
12   $\ddot{\mathbf{q}}_c = \mathbf{A}_1^\dagger\mathbf{b}_1 + (\mathbf{A}_2(\mathbf{I} - \mathbf{A}_1^\dagger\mathbf{A}_1))^\dagger(\mathbf{b}_2 - \mathbf{A}_2\mathbf{A}_1^\dagger\mathbf{b}_1)$ ;   // Eq. (4.7)
13   $\mathbf{s}_{prev} = \mathbf{s}$ ;                                                 // For next iteration
14   $\mathbf{s}_{ref,prev} = \mathbf{s}_{ref}$ ;                                       // For next iteration
15   $\dot{\mathbf{s}}_{ref,prev} = \dot{\mathbf{s}}_{ref}$ ;                           // For next iteration
16   $\mathbf{J}_{s,prev} = \mathbf{J}_s$ ;                                             // For next iteration
17   SendToJointControllers($\ddot{\mathbf{q}}_c$);
18 end
```

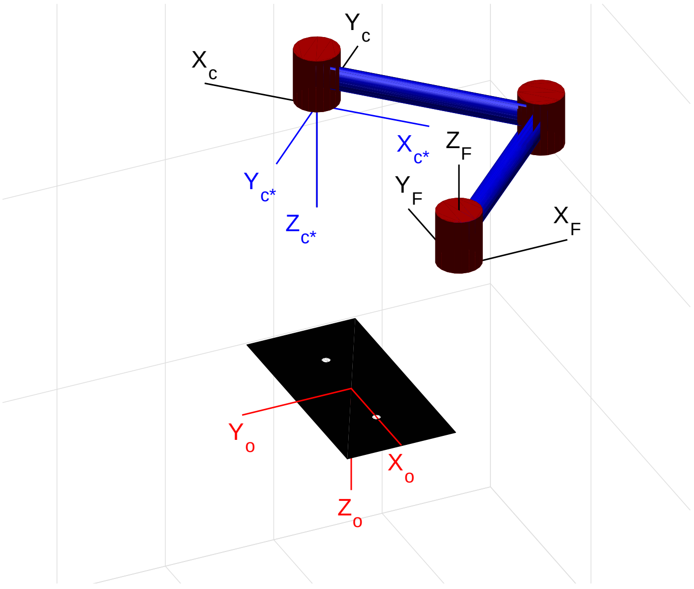


Figure 4.4: System used for 2D simulation: 3R planar robot, target object with two features and coordinate frames.

4.4 Simulation 2D PBVS: first example

As first example, a simple two-dimensional (2D) robot system is considered for better illustration of the main features of the algorithm and its comparison to the conventional potential field-based approach used in Mezouar and Chaumette (2002).

The robot 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. Fig. 4.4 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 (Siciliano et al. (2009)) taking into account the DH parameters shown in Table 4.1 for this robot.

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 4.1: Denavit-Hartenberg parameters for the simulated 3R planar robot.

4.4.1 Potential Field-Based Method

For the acceleration-based robot control, the conventional potential field-based method is given by Latombe (1991):

$$\ddot{\mathbf{q}}_c = F_{att} + \sum_i \left(F_{rep,i}\right), \tag{4.46}$$

where F_{att} is the *attractive force* to the reference and $F_{rep,i}$ is the *repulsive force* from the ith-constraint. Thus, the sum of all "forces" determines the magnitude and direction of the control action.

In order to perform a fair comparison between the proposed approach and the conventional potential field-based method, the attractive force F_{att} is obtained computing $\ddot{\mathbf{q}}_c$ from Eq. (4.38) and the repulsive forces are computed with the commonly used expression shown below (Khatib (1986)):

$$F_{rep,i} = \begin{cases} \xi\left(\rho_i^{-1} - \rho_0^{-1}\right)\rho_i^{-2}\ \nabla\rho_i & \text{if } \rho_i < \rho_0 \\ 0 & \text{otherwise,} \end{cases} \tag{4.47}$$

where ξ is a positive constant that represents the gain of the repulsive field, ρ_i represents the distance to the boundary of the ith-constraint and ρ_0 is a positive constant denoting the distance of influence of the constraints. For simplicity, the equivalence $\rho_i = -\sigma_i$ is considered for the simulations.

It is interesting to remark that, the value of parameter ρ_0 cannot be too small (e.g., in order to fully utilize the available workspace) since some distance of influence is required to correct the robot motion, particularly when the robot approaches the constraint boundary at high speed. In this regard, note that the potential field-based method does not consider the robot speed in contrast to the proposed approach.

4.4.2 Simulation conditions and parameter values

In order to better illustrate the behavior of the proposed SM algorithm, only the visibility FOV constraints described in Section 4.3.3 are considered, i.e., no joint limits, maximum joint speeds or workspace constraints are included. Furthermore, the potential field-based approach described in Section 4.4.1 is also simulated for comparison purposes.

Simulation was run under the following conditions:

i) Parameters used for the camera: focal length $f = 400$ pixels; width and height of the rectangle representing the image plane limits $W_V = H_V = 512$ pixels; and camera to end-effector transformation matrix ${}^{c}\mathbf{M}_e = \mathbf{I}_4$ where $\mathbf{I}_4$ represents the identity matrix of dimension 4, i.e., the camera pose is equivalent to the end-effector pose.

ii) Parameters used for the visibility constraints: safety margin $m_v = 5\%$; hyperellipse smoothing parameter $n_V = 16$; constraint approaching parameter $K_{V,i} = 0.2$; and control action amplitude $u_V^+ = 5$.

iii) Parameters used for the kinematic controller: correction gain for the velocity error $K_{T,v} = 3\sqrt{K_{T,p}}$ (note that an overdamped response is obtained for $K_{T,v} > 2\sqrt{K_{T,p}}$); and the correction gain for the position error uses an *exponential*[5] waveform depending on the magnitude of the position error: gain for zero error $K_{T,p}(\mathbf{0}) = 20$, gain for infinity error $K_{T,p}(\infty) = 0$ and time-derivative of the gain for zero error $\dot{K}_{T,p}(\mathbf{0}) = 40$.

iv) Parameters used for the potential field-based approach: gain of the repulsive field $\xi = 0.2$ and distance of influence of the repulsive field $\rho_0 = 0.2$.

v) The initial configuration considered for the robot is given by the joint position vector $\mathbf{q}(0) = \begin{bmatrix}\pi/4 & \pi/2 & 0\end{bmatrix}^{\mathrm{T}}$ rad, yielding an initial camera pose given by the transformation matrix
${}^{F}\mathbf{M}_C(0) = \begin{bmatrix}0 & 1.4142 & 0 & \pi & 0 & -3\pi/4\end{bmatrix}^{\mathrm{T}}$.

vi) A static target object is considered with the pose given by the transformation matrix ${}^{F}\mathbf{M}_O = \begin{bmatrix}-0.0707 & 0.9192 & -1.1 & \pi & 0 & \pi/2\end{bmatrix}^{\mathrm{T}}$, with

[5]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.

two markers[6] given by the following points with respect to the object frame: ${}^{O}\mathbf{p}_1 = \begin{bmatrix} -0.25 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ m and ${}^{O}\mathbf{p}_2 = \begin{bmatrix} 0.25 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ m.

vii) The desired camera frame is given by the transformation matrix ${}^{F}\mathbf{M}_{C^*} = \begin{bmatrix} 0 & 1.4142 & 0 & \pi & 0 & \pi/4 \end{bmatrix}^{\mathrm{T}}$. That is, the initial error for the camera position is zero and for the camera orientation is π radians in the Z-axis, see Fig. 4.4.

viii) The algorithm was computed with a sampling time T_s of 5 milliseconds and the disturbance vector $\mathbf{d}_c$ has been considered zero.

4.4.3 Simulation results

Fig. 4.5 shows comparatively the simulated behavior of the VS system regarding the trajectory followed by the image features for three cases: considering no visibility constraint; using the conventional potential field-based method; and using the proposed SM approach. In the first case, the second feature (that with the star symbol) leaves the FOV of the camera. For the potential field-based case, this feature remains in the FOV but the available allowed space is not fully utilized, i.e., the boundary of the hyperellipse constraint is not achieved. Finally, for the proposed SM approach the feature also remains in the FOV but, in contrast to the previous method, the allowed space is fully utilized. Furthermore, the feature trajectory for the proposed method is smoother than that for the potential field-based approach. In particular, note that the boundary of the hyperellipse constraint is reached smoothly.

Fig. 4.6 depicts the detail of different parameters of the proposed strategy. The figure shows that: the initial error for the camera orientation is made zero; a position error arises when the visibility constraint is activated, which subsequently is made zero; the visibility constraint is hit at the time intervals 3-5 s and 8-9 s for the second image feature; and the visibility constraints are fulfilled, i.e., $\max(\phi_i) \leq 0$.

[6]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.

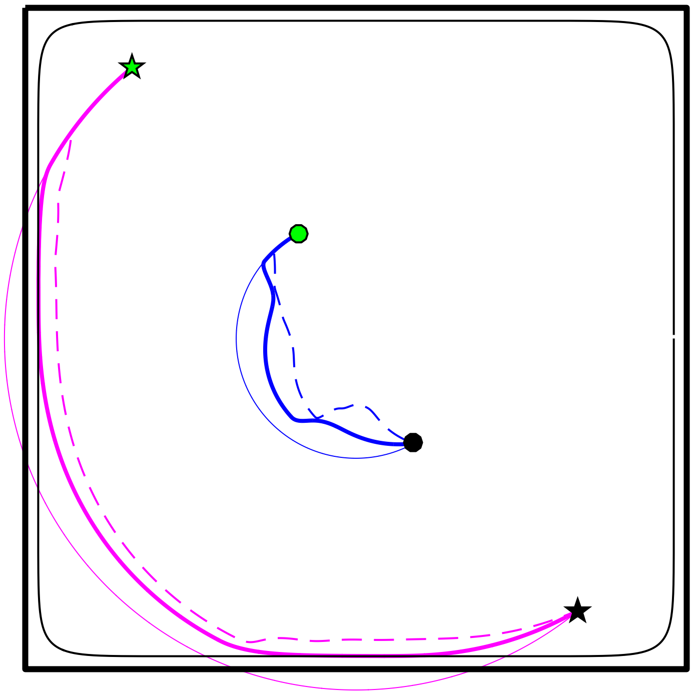


Figure 4.5: Feature trajectories in the image plane for the first example: considering no visibility constraint (thin solid), using the potential field-based method (dashed) and using the proposed SM approach (thick solid).

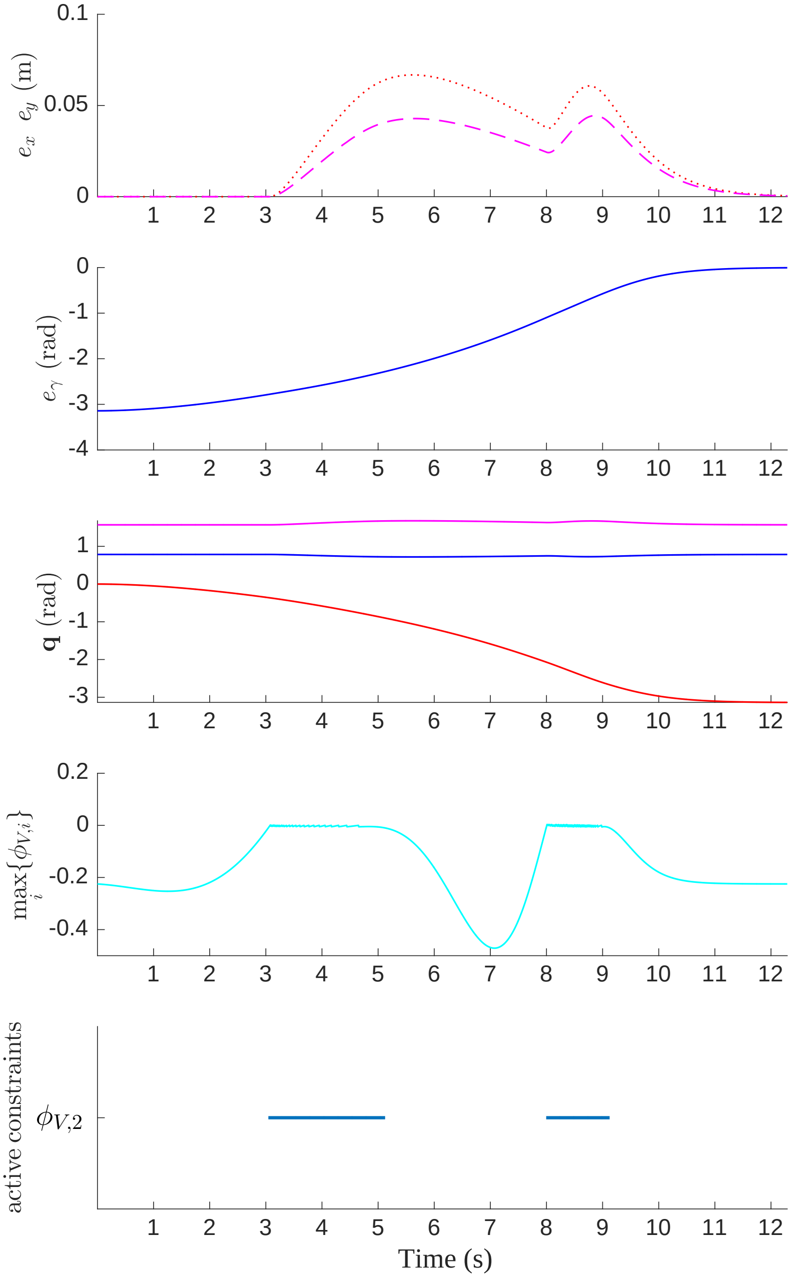


Figure 4.6: From top to bottom plots: (1) Camera position error for X- and Y-axes; (2) camera angular error for the Z-axis; (3) joint values $\{q_1,q_2,q_3\}$; (4) maximum value of the visibility constraint functions $\phi_{V,i}$; (5) horizontal lines indicating when a constraint is active.

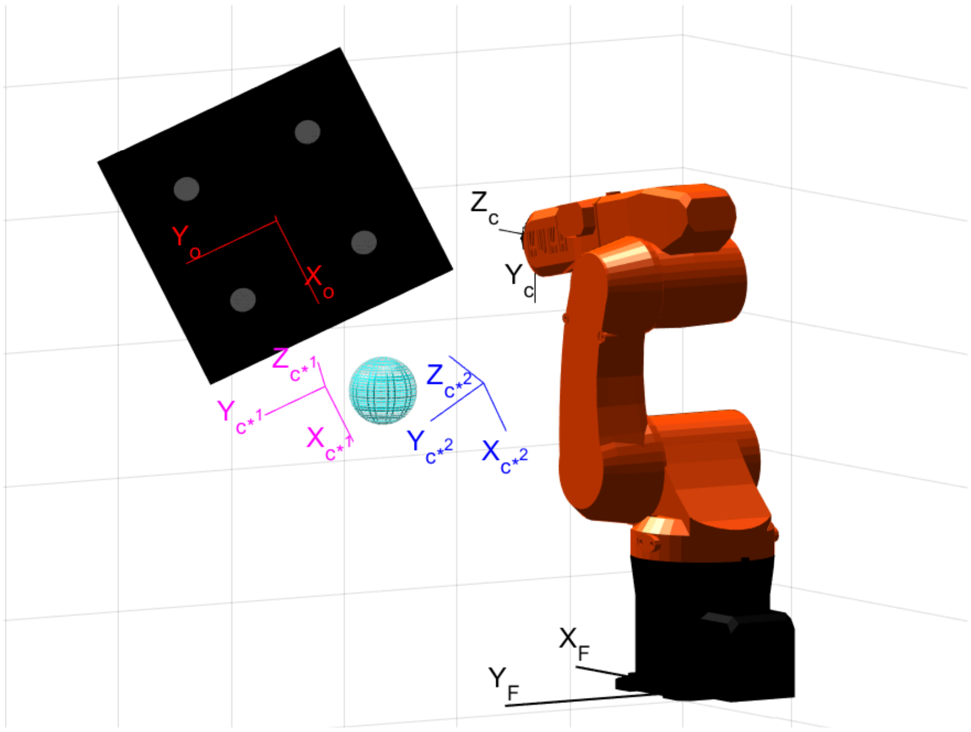


Figure 4.7: System used for 3D simulation: 6R robot, target object with four markers, sphere representing a forbidden area and coordinate frames.

4.5 Simulation 3D PBVS: case study

A three-dimensional case study in PBVS 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. Fig. 4.7 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 (Siciliano et al. (2009)) taking into account the DH parameters shown in Table 4.2 for this robot.

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 4.2: Denavit-Hartenberg parameters for the simulated 6R robot.

is given by:

$$\sigma_{R,o1}(\mathbf{p}_j) = r_s - \|\mathbf{p}_j - \mathbf{p}_s\|_2 + m_{R,o}, \tag{4.48}$$

where r_s and $\mathbf{p}_s$ are the radius and center, respectively, of the sphere obstacle, $m_{R,o}$ is the safety margin for the constraint and $\mathbf{p}_j$ is the Cartesian position of the considered point of the robot.

Therefore, the partial derivative of $\sigma_{R,o1}$ with respect to $\mathbf{p}_j$, which is needed for computing the gradient vector in Eq. (4.37), results in:

$$\frac{\partial \sigma_{R,oi}}{\partial \mathbf{p}_j} = -\frac{\mathbf{p}_j - \mathbf{p}_s}{\|\mathbf{p}_j - \mathbf{p}_s\|_2}. \tag{4.49}$$

4.5.1 Simulation conditions and parameter values

Simulation was run under the following conditions:

i) Parameters used for the camera: focal length $f = 1000$ pixels; width and height of the rectangle representing the image plane limits $W_V = H_V = 512$ pixels; and camera to end-effector transformation matrix ${}^{c}\mathbf{M}_e = \mathbf{I}_4$, i.e., the camera pose is equivalent to the end-effector pose.

ii) Parameters used for the visibility constraints: safety margin $m_v = 5\%$; hyperellipse smoothing parameter $n_V = 16$; constraint approaching parameter $K_{V,i} = 0.1$; and control action amplitude $u_V^+ = 2$.

iii) Parameters used for the joint limit constraints: safety margin $m_{R,q} = 0$; constraint approaching parameter $K_{R,qi} = 0.1$; control action amplitude

$u^{+}_{R,q} = 2$; mid position and maximum range of motion for the sixth joint $q_{\text{mid},6} = -0.5$ rad and $\Delta q_{\max,6} = 1.2$ rad, respectively. The range limit constraint for the remaining joints are omitted for simplicity.

iv) Parameters used for the joint speed constraints: safety margin $m_{R,s} = 0$; control action amplitude $u^{+}_{R,s} = 1$; and maximum speed for the second joint $\dot{q}_{\max,2} = 0.4$ rad/s. The speed constraints for the remaining joints are omitted for simplicity.

v) Parameters used for the robot constraint for collision avoidance: safety margin $m_{R,o} = 0$; constraint approaching parameter $K_{R,o1} = 0.1$; control action amplitude $u^{+}_{R,o} = 0.2$; radius of the sphere obstacle $r_s = 0.05$ m; and center of the sphere obstacle $\mathbf{p}_s = \begin{bmatrix} 0.57 & 0.2 & 0.405 \end{bmatrix}^{\mathrm{T}}$ 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 collision avoidance. The Jacobian matrix ${}^{0}\mathbf{J}_{pe}$ for this point, which is needed to compute the gradient vector in Eq. (4.37), can be readily obtained (Siciliano et al. (2009)) taking into account the DH parameters of the 6R robot shown in Table 2.1.

vi) Parameters used for the kinematic controller: correction gain for the velocity error $K_{T,v} = 3\sqrt{K_{T,p}}$; and, as before, the correction gain for the position error uses an exponential waveform depending on the magnitude of the position error: for the first phase $\{K_{T,p}(\mathbf{0}) = 30, K_{T,p}(\infty) = 0, \dot{K}_{T,p}(\mathbf{0}) = 50\}$ and for the second phase $\{K_{T,p}(\mathbf{0}) = 10, K_{T,p}(\infty) = 0, \dot{K}_{T,p}(\mathbf{0}) = 70\}$.

vii) The initial configuration considered for the robot is given by the joint position vector $\mathbf{q}(0) = \begin{bmatrix} 0 & 0 & 0.2 & 0 & 0.2 & 0 \end{bmatrix}^{\mathrm{T}}$ rad, yielding an initial camera pose given by the transformation matrix ${}^{F}\mathbf{M}_C(0) = \begin{bmatrix} 0.462 & 0 & 0.6346 & 0 & \pi/2 & -\pi/2 \end{bmatrix}^{\mathrm{T}}$ in compact notation.

viii) A static target object is considered with the pose given by the transformation matrix matrix ${}^{F}\mathbf{M}_O = \begin{bmatrix} 1.162 & 0.01 & 0.5946 & \pi/3 & \pi/3 & -\pi/2 \end{bmatrix}^{\mathrm{T}}$ in compact notation, with four markers given by the following points with respect to the object frame: ${}^{O}\mathbf{p}_1 = \begin{bmatrix} -0.1 & -0.1 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^{O}\mathbf{p}_2 = \begin{bmatrix} 0.1 & -0.1 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^{O}\mathbf{p}_3 = \begin{bmatrix} 0.1 & 0.1 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^{O}\mathbf{p}_4 = \begin{bmatrix} -0.1 & 0.1 & 0 \end{bmatrix}^{\mathrm{T}}$

m, that is, the four markers are the vertices of a square with a side length of 0.2 m.

ix) The desired camera frames for the first and second phase are given by the transformation matrices $^{F}\mathbf{M}_{C^{*1}} = \begin{bmatrix} 0.5549 & 0.3119 & 0.4203 & \pi/3 & \pi/3 & -\pi/2 \end{bmatrix}^{\mathrm{T}}$ and $^{F}\mathbf{M}_{C^{*2}} = \begin{bmatrix} 0.6 & 0 & 0.4 & 0.012 & 1.257 & -0.611 \end{bmatrix}^{\mathrm{T}}$, respectively, in compact notation.

x) The algorithm was computed with a sampling time T_s of 2 milliseconds and the disturbance vector $\mathbf{d}_c$ has been considered zero.

4.5.2 Simulation results

The results of the simulation are depicted at different figures. Fig. 4.8 shows that the position and orientation error is made zero for both phases. Note that the desired camera frame is changed from $^{F}\mathbf{M}_{C^{*1}}$ to $^{F}\mathbf{M}_{C^{*2}}$ around time instant 4 s. Fig. 4.9 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. 4.10 shows that: all the constraints are fulfilled, i.e., $\max(\phi_i) \leq 0$; the joint limit constraint for the sixth joint (dark line in the first plot) and the speed constraint for the second joint (dark line in the second plot) become active during the first phase; the constraint for collision avoidance becomes active for the time interval 6-8 s during the second phase; and the visibility 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, a detail view of the sphere obstacle is shown in Fig. 4.11, where it can be seen that the constraint for collision avoidance is fulfilled, whereas Fig. 4.12 depicts four snapshot frames of a 3D representation of the robot at different time instants .

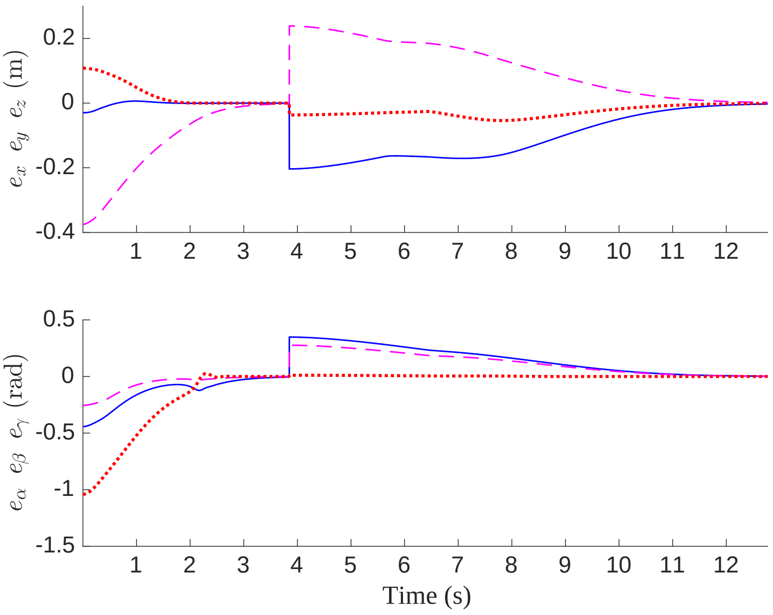


Figure 4.8: Position and orientation error: e_x and e_α (solid, blue), e_y and e_β (dashed, magenta) and e_z and e_γ (dotted, red).

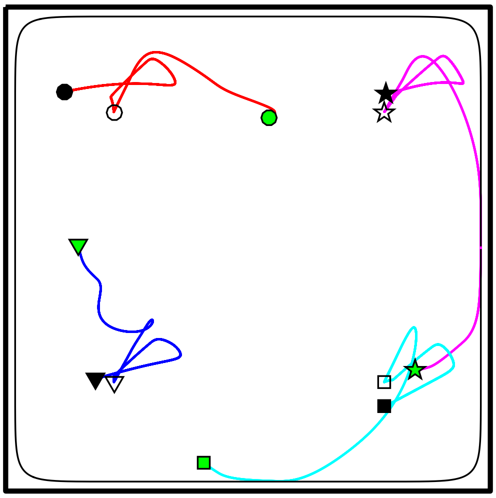


Figure 4.9: Image features trajectories for the case study.

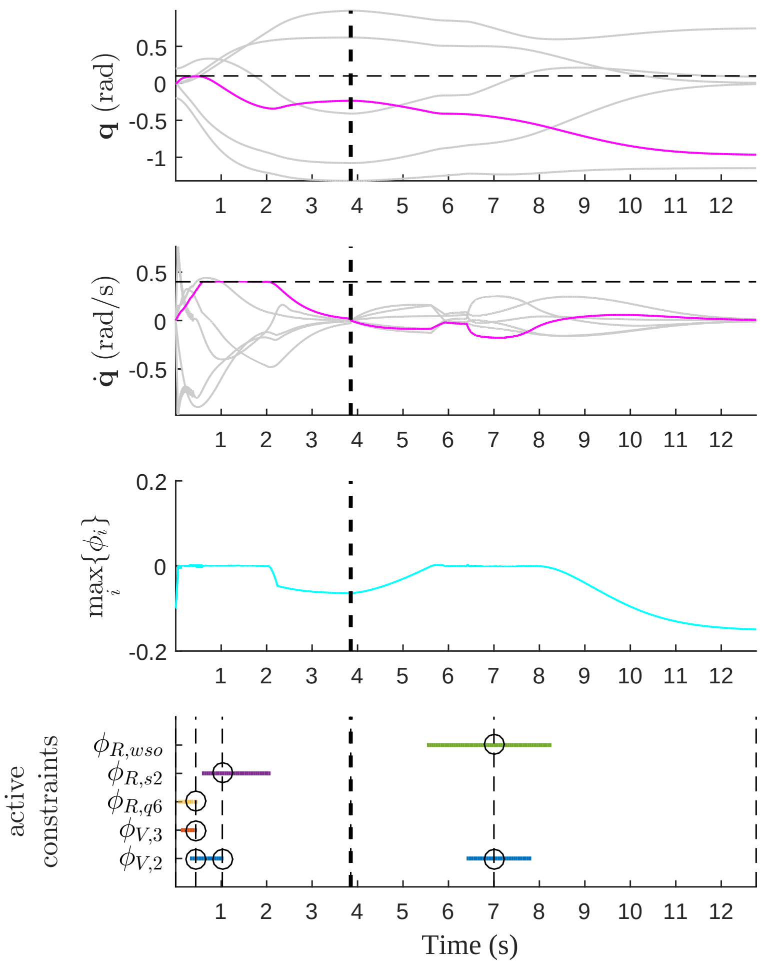


Figure 4.10: From top to bottom plots: (1) joint postions $\mathbf{q}$ and (2) joint speeds $\dot{\mathbf{q}}$ (the horizontal dashed line represents the limit for the active constraint, which is depicted with a dark line); (3) maximum value of the constraint functions ϕ_i; (4) horizontal lines indicating when a constraint is active (the dashed vertical lines correspond to the time instants of the frames in Fig. 4.12 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 ${}^{F}\mathbf{M}_{C^{*1}}$ to ${}^{F}\mathbf{M}_{C^{*2}}$.

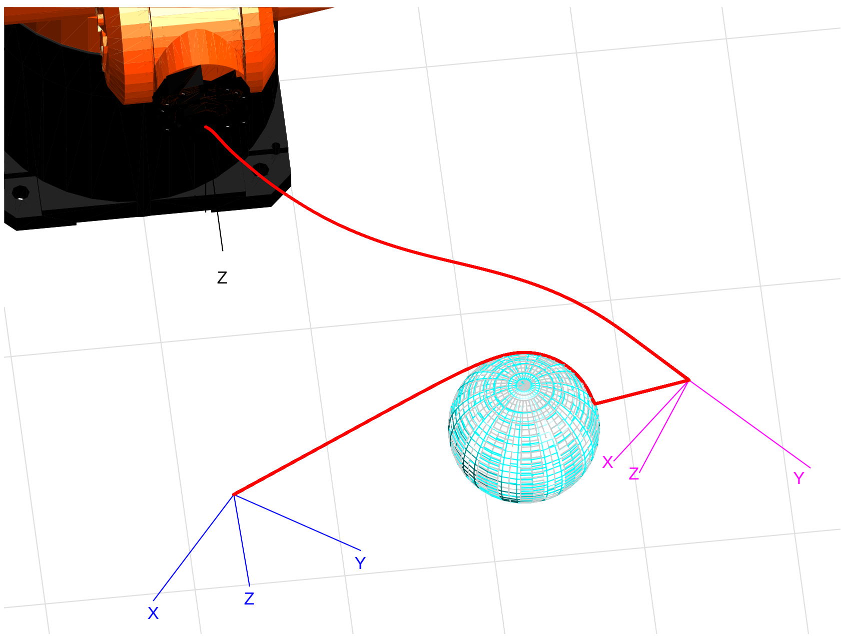


Figure 4.11: Detailed view of the fulfillment of the constraint for the sphere obstacle.

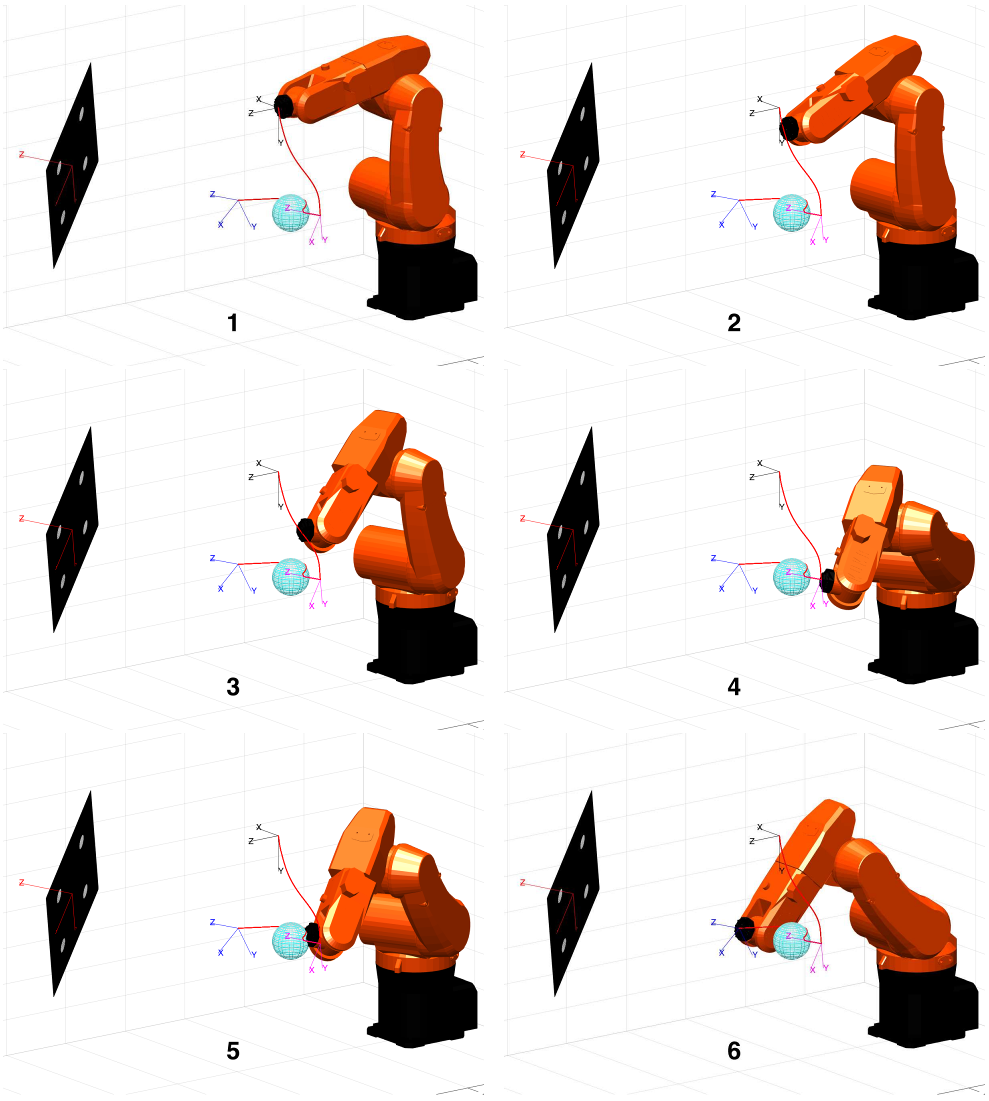


Figure 4.12: Sequence of frames (frames 1 to 6) showing the robot configuration during the simulation and the path follwed by the robot end-effector. The active constraints at each frame are shown in Fig. 4.10.

4.6 Simulation 3D IBVS: case study

In this section, a three-dimensional (3D) case study is presented. IBVS is considered with the eye-to-hand configuration.

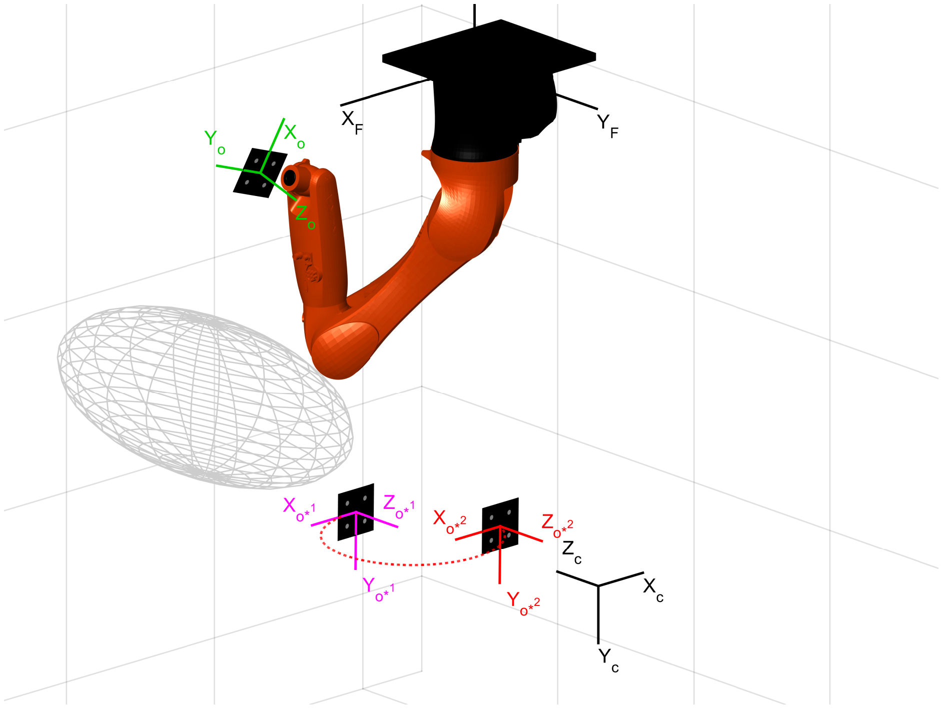


Figure 4.13: System used for 3D simulation: 6R robot, target object with four markers, sphere representing a forbidden area and coordinate frames.

In the proposed case study, a classical 6R serial manipulator with spherical wrist in ceiling position is considered. Fig. 4.13 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, and desired trajectory of the moving object and its final position O^{*2} for the second phase.

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:

$$\sigma_{R,wso}(\mathbf{p}_j) = 1 - \left\| \begin{bmatrix} \frac{x_j - x_o}{r_{ox}} & \frac{y_j - y_o}{r_{oy}} & \frac{z_j - z_o}{r_{oz}} \end{bmatrix}^{\mathrm{T}} \right\|_2 \tag{4.50}$$

$$\sigma_{R,wsl}(\mathbf{p}_j) = 1 - \left\| \begin{bmatrix} \frac{x_j - x_l}{r_{lx}} & \frac{y_j - y_l}{r_{ly}} & \frac{z_j - z_l}{r_{lz}} \end{bmatrix}^{\mathrm{T}} \right\|_2 , \tag{4.51}$$

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 (4.37), results in:

$$\frac{\partial \sigma_{R,wso}}{\partial \mathbf{p}_j} = -\frac{1}{1 - \sigma_{R,wso}(\mathbf{p}_j)} \begin{bmatrix} \frac{x_j - x_o}{r_{ox}^2} & \frac{y_j - y_o}{r_{oy}^2} & \frac{z_j - z_o}{r_{oz}^2} \end{bmatrix}^{\mathrm{T}} \tag{4.52}$$

$$\frac{\partial \sigma_{R,wsl}}{\partial \mathbf{p}_j} = -\frac{1}{1 - \sigma_{R,wsl}(\mathbf{p}_j)} \begin{bmatrix} \frac{x_j - x_l}{r_{lx}^2} & \frac{y_j - y_l}{r_{ly}^2} & \frac{z_j - z_l}{r_{lz}^2} \end{bmatrix}^{\mathrm{T}} . \tag{4.53}$$

4.6.1 Simulation conditions and parameter values

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 $\mathbf{p}_o = \begin{bmatrix}0.34 & -0.60 & -0.95\end{bmatrix}^{\mathrm{T}}$ 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 (4.37), can be readily obtained (Siciliano et al. (2009)) taking into account the Denavit-Hartenberg parameters of the Kuka Agilus KR6 R900 sixx robot shown in Table 2.1.

vi) Parameters used for the kinematic controller: 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 ${}^{F}\mathbf{M}_E(0) = \begin{bmatrix}0.43 & -0.32 & -0.26 & -1.7741 & 0.1006 & -1.8989\end{bmatrix}^{\mathrm{T}}$, in compact notation.

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 $[x_c, y_c] = [0.437, 0.219]$ m is the center, $r = 0.235$ m is the radius, and $\alpha = 260°$ is the phase of the circular trajectory.

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.

4.6.2 Simulation results

The results of the simulation are depicted at different figures.

Fig. 4.14 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. 4.15 shows the trajectories followed by the visual features in the image plane.

Fig. 4.16 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.

Finally, Fig. 4.17 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. 4.18, 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`.

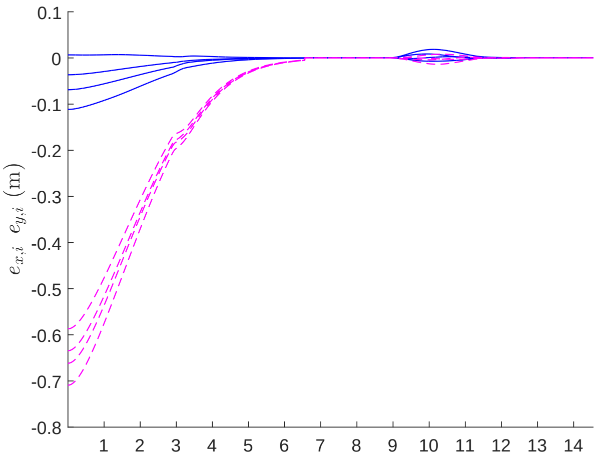


Figure 4.14: Visual features error: $e_{x,i}$ (solid, blue) and $e_{y,i}$ (dashed, magenta).

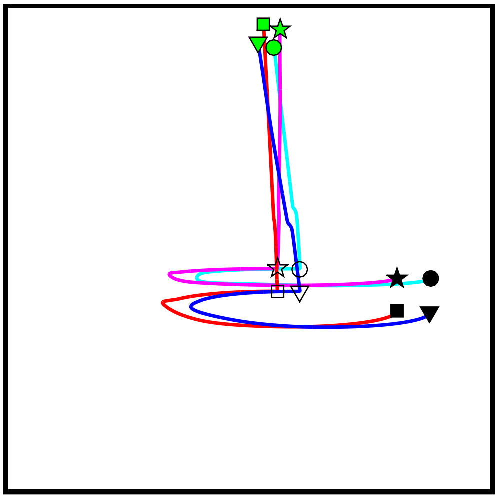


Figure 4.15: Feature trajectories in the image plane for the case study.

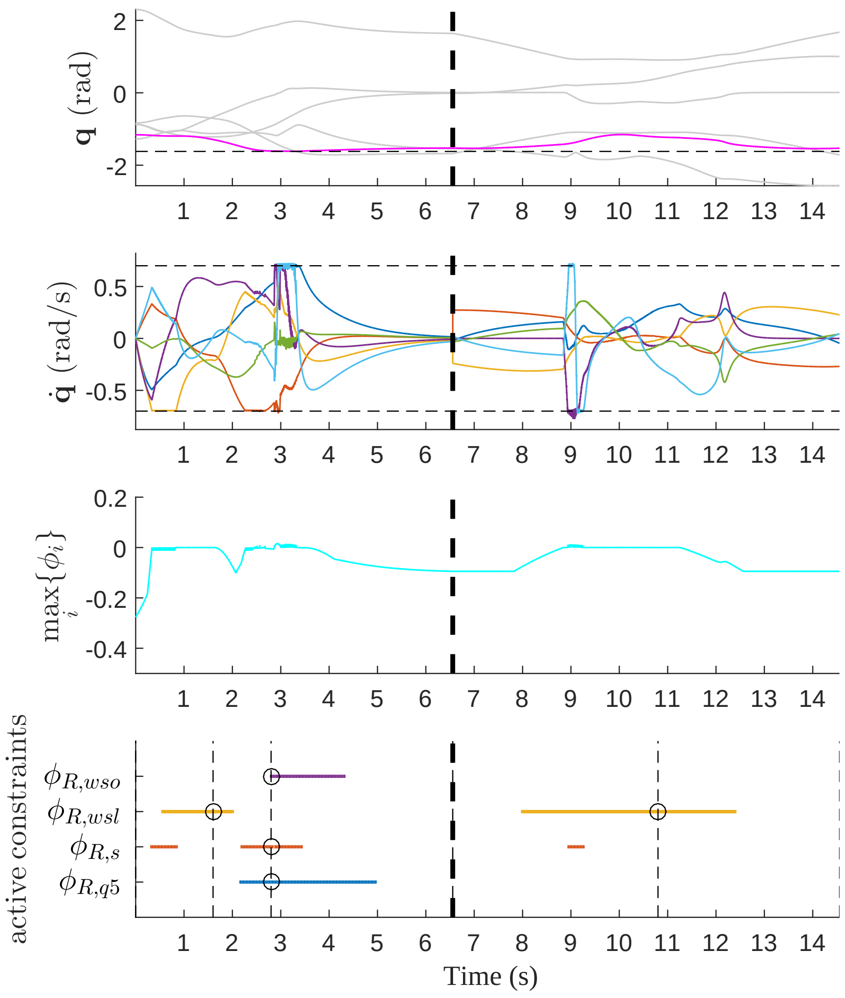


Figure 4.16: From top to bottom plots: (1) joint postions $\mathbf{q}$ (the horizontal dashed line represents the joint limit for the active constraint, which is depicted with a dark line); (2) joint speeds $\dot{\mathbf{q}}$ (the horizontal dashed lines represent the speed limit for all the joints); (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. 4.17 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 ${}^F\mathbf{M}_{C^{*2}}$.

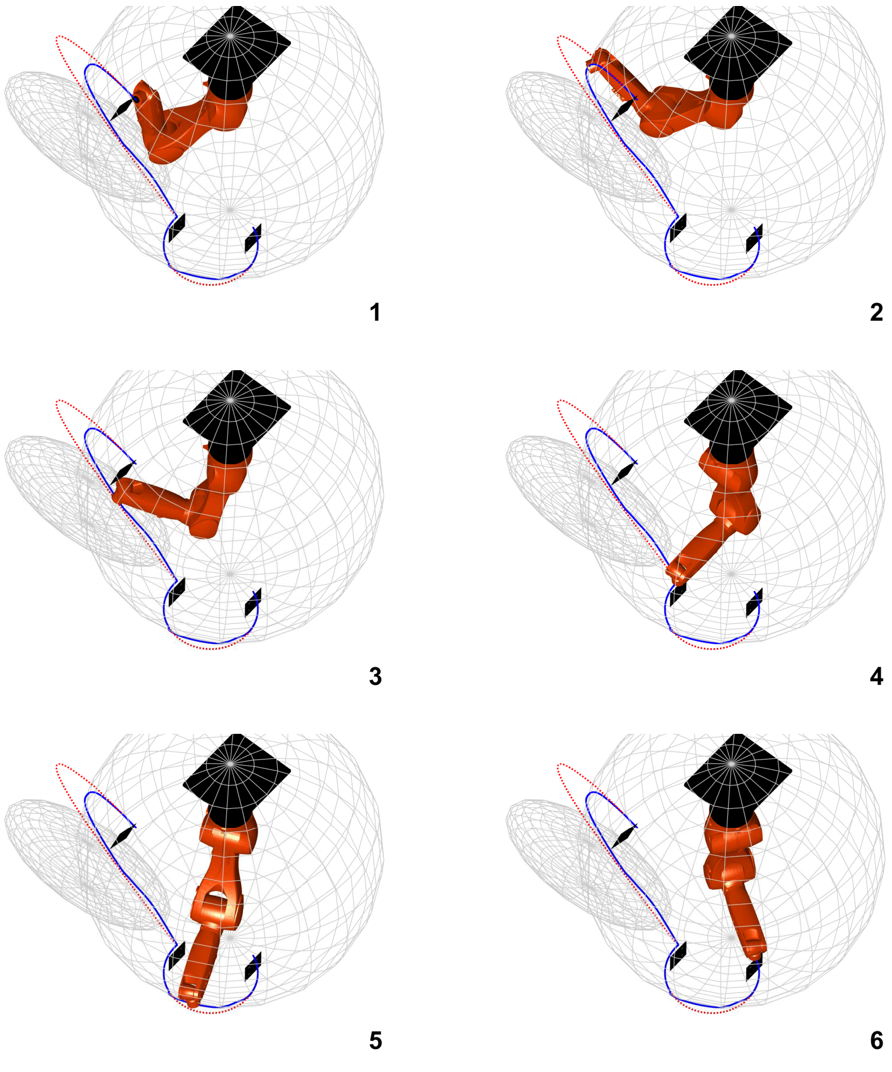


Figure 4.17: Sequence of frames (frames 1 to 6) showing the robot configuration during the simulation, the actual path follwed by the robot end-effector (solid, blue) and the ideal path for no constraints (dotted, red). The active constraints at each frame are shown in Fig. 4.16.

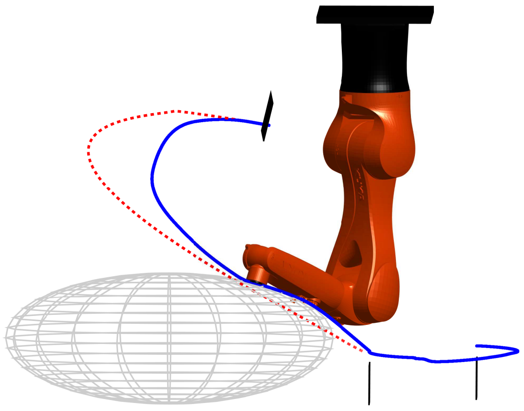


Figure 4.18: Detailed view of the fulfillment of the constraint for the ellipsoid obstacle.

4.7 Experiments: visibility constraints in PBVS and IBVS

The proposed SM method has been implemented to obtain real experiments in order to demonstrate its feasibility and robustness. The following setup has been used (see Fig. 4.19): a Kuka Agilus KR6 R900 sixx robot manipulator in ceiling-mounted position, is equipped with the Kuka Robot Sensor Interface (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) (Marchand et al. (2005)), whilst the object pose is estimated to update the visual feature vector $\mathbf{s}$ and to compute $\mathbf{L}_s$. Since the camera is rigidly attached to the end-effector of the robot, which corresponds with the eye-in-hand configuration, the twist matrix ${}^{c}\mathbf{V}_e$ is constant and can be computed from the camera to end-effector transformation matrix ${}^{c}\mathbf{M}_e$. Finally, the Jacobian matrix ${}^{e}\mathbf{J}_e$ for this robot can be readily obtained (Siciliano et al. (2009)) taking into account the DH parameters of the Kuka Agilus KR6 R900 sixx robot shown in Table 2.1.

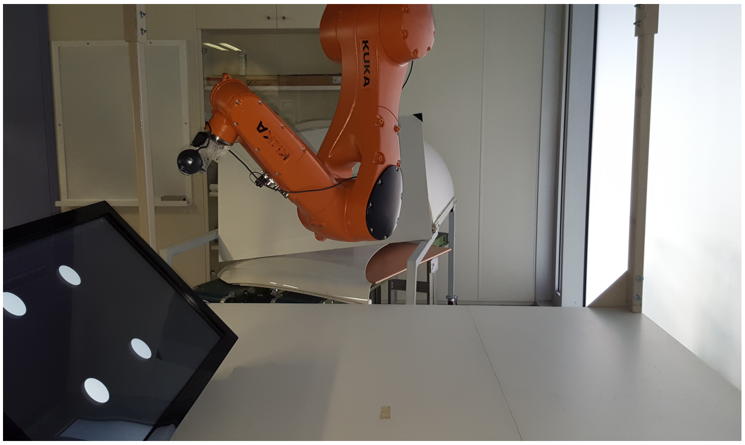


Figure 4.19: Experimental setup: 6R serial industrial manipulator in ceiling position with the camera rigidly attached to the robot end-effector (eye-in-hand) and a screen to display the object markers.

To illustrate the applicability of the method, two experiments have been considered, which consist in positioning the robot with respect to a motionless target object while fulfilling the visibility constraints:

i) **Experiment 1.** ***Visibility constraints due to the limited camera FOV in PBVS:*** in PBVS it is more likely that the image features leave the camera FOV because the control law is defined in the Cartesian space.[7].

ii) **Experiment 2.** ***Visibility constraints due to object occlusion in IBVS.***

[7]For instance, for the typical IBVS case of a motionless target object with points in the image plane as visual features, the ideal trajectory for the feature points results in a straight line from the initial to the goal position. Therefore, unless the error of the estimated interaction matrix and/or the camera intrinsic parameters are particularly large, the image features do not leave the camera FOV. For example, (Mezouar and Chaumette (2002)) considers IBVS and intentionally introduces an artificial error of 40% in the camera intrinsic parameters in order for the feature points' trajectory to leave the camera FOV.

It is important to remark that, independently of the reason of violation of the visibility constraints, the proposed method applies to any VS control domain. In fact, the setup used for a specific VS application, yields the robot control law (4.38) used in the low-priority task, but it does not modify the acceleration equality (4.31) used in the high-priority task to fulfill the visibility constraints.

4.7.1 Experimental conditions and parameter values

The experiments were run under the following conditions:

i) Three periodic threads are defined and scheduled following a fixed priority scheme, from highest to lowest: server, control and vision threads. The server period must be set to 4 milliseconds due to robot specification. Both the vision period and the control period T_s are set to 100 milliseconds to guarantee an appropriate scheduling.

ii) 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.

iii) Camera parameters: focal length $f = [710.1, 709.8]$ pixels, resolution $[W_V, H_V] = [640, 480]$ pixels, camera to end-effector transformation matrix ${}^c\mathbf{M}_e = \begin{bmatrix} 0 & 0.07 & -0.05 & 0 & 0 & -\pi/2 \end{bmatrix}^{\mathrm{T}}$, in compact notation.

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

v) Parameters used for the visibility constraints due to the limited camera FOV in PBVS: hyperellipse smoothing parameter $n_V = 16$; safety margin $m_v = 30$ pixels; constraint approaching parameter $K_{V,i} = 3$; and control action amplitude $u_V^+ = 1$.

vi) Parameters used for the visibility constraints due to object occlusion in IBVS: to represent the forbidden area in the image plane, an hyperellipse has been considered with the parameters $[x, a, y, b, n_V] = [460, 100, 375, 100, 4]$; safety margin $m_v = 0$ pixels; constraint approaching parameter $K_{V,i} = 3$; and control action amplitude $u_V^+ = 1$.

vii) Parameters used for the kinematic controller: correction gain for the position error starting at $K_{T,p} = 0.01$ and increasing linearly up to $K_{T,p} = 0.60$ after the visibility constraints are ensured to speed up convergence; correction gain for the velocity error $K_{T,v} = 3\sqrt{K_{T,p}}$.

viii) The initial configuration is given by the robot joint position vector $\mathbf{q}(0) = \begin{bmatrix} 2.67 & -1.80 & 2.14 & 0.79 & -1.44 & -0.97 \end{bmatrix}^{\mathrm{T}}$ rad, and the visual feature vector $\mathbf{s}(0) = \begin{bmatrix} -0.066 & 0.161 & 0.034 & 0.312 & 0.147 & 1.51 \end{bmatrix}^{\mathrm{T}}$ for PBVS and $\mathbf{s}(0) = [0.4601 \quad -0.2528 \quad 0.4506 \quad -0.2549 \quad 0.4528 \quad -0.2657 \quad 0.4625 \quad -0.2630]^{\mathrm{T}}$ for IBVS.

ix) 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:

$$\nabla\boldsymbol{\sigma}_{Ve} = \nabla\boldsymbol{\sigma}_V + c_e \begin{bmatrix} -1 & 1 & 1 & 1 & -1 & -1 \end{bmatrix}^{\mathrm{T}} \circ |\nabla\boldsymbol{\sigma}_V|, \qquad (4.54)$$

where $|\cdot|$ represents the absolute value function, 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., $c_e = 0.3$.

4.7.2 Experimental results

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

A video of the PBVS experiment using the gradient vector with no errors can be played at (video at double speed) `https://media.upv.es/player/?id=56805190-8411-11e7-90ea-23686ce0f1be`. For this experiment, Fig. 4.20 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. Fig. 4.21 shows position and orientation errors, control action $\ddot{\mathbf{q}}_c$ and constraint function ϕ_V.

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. 4.22 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.

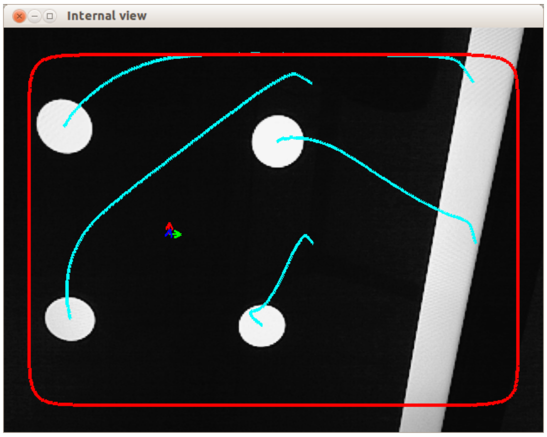


Figure 4.20: Image features trajectories for the visibility constraints experiment in PBVS with no errors in the gradient vector. Snapshot in the desired pose.

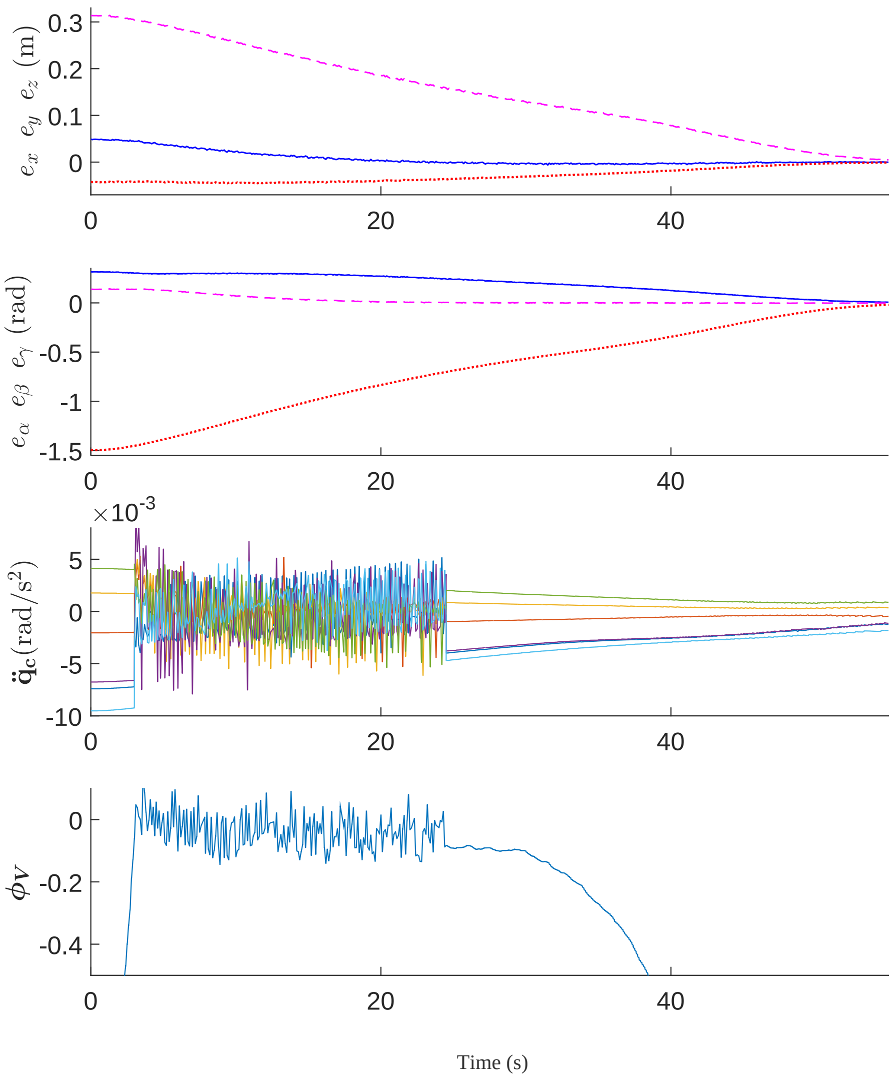
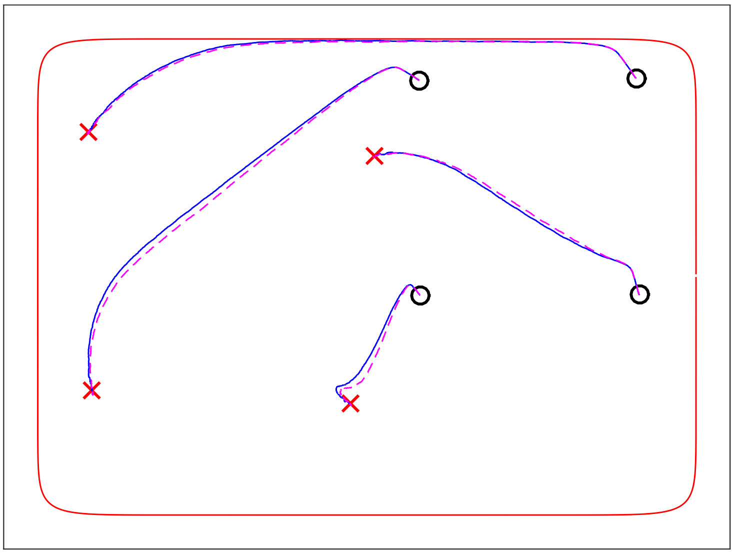


Figure 4.21: Visibility constraints experiment in PBVS with no errors in the gradient vector. From top to bottom plots: (1) position errors; (2) orientation errors; (3) control action $\ddot{\mathbf{q}}_c$; (4) constraint function vector ϕ_V.



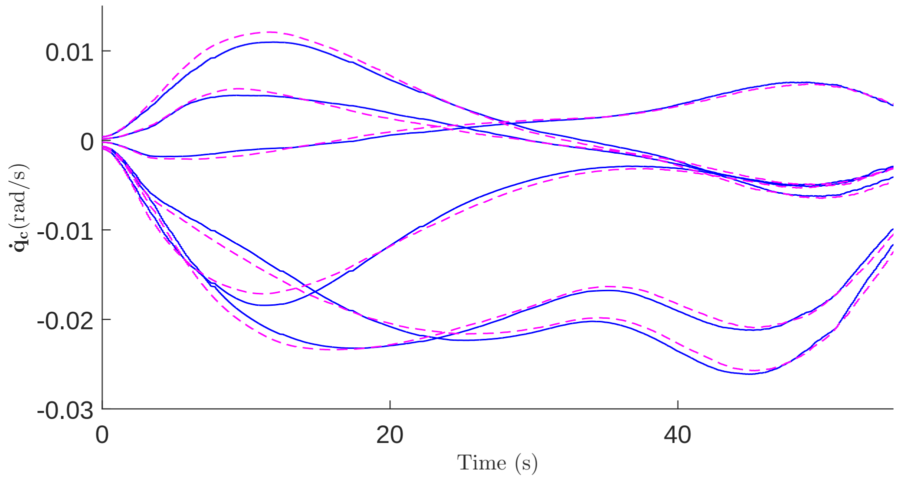


Figure 4.22: Comparison of visibility constraints experiments in PBVS, without and with errors in the gradient vector. (1) Image features trajectories, from initial (circle) to desired (cross) pose; (2) commanded joint speeds $\dot{\mathbf{q}}_c$. Solid-blue experiment with no errors and dashed-magenta experiment with errors.

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 played at (video at double speed) `https://media.upv.es/player/?id=2dd4d490-8412-11e7-90ea-23686ce0f1be`. For this experiment, Fig. 4.23 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. Fig. 4.24 shows visual feature errors, control action $\ddot{\mathbf{q}}_c$ and constraint function ϕ_V.

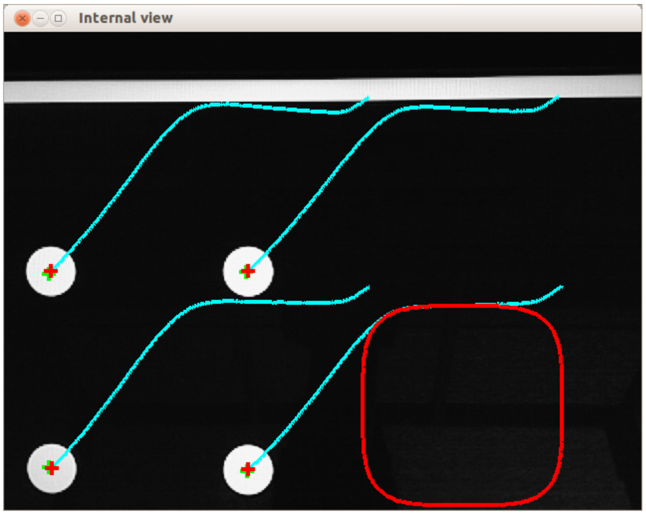


Figure 4.23: Image features trajectories for the visibility constraints experiment in IBVS with no errors in the gradient vector. Snapshot in the desired pose.

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. 4.25 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.

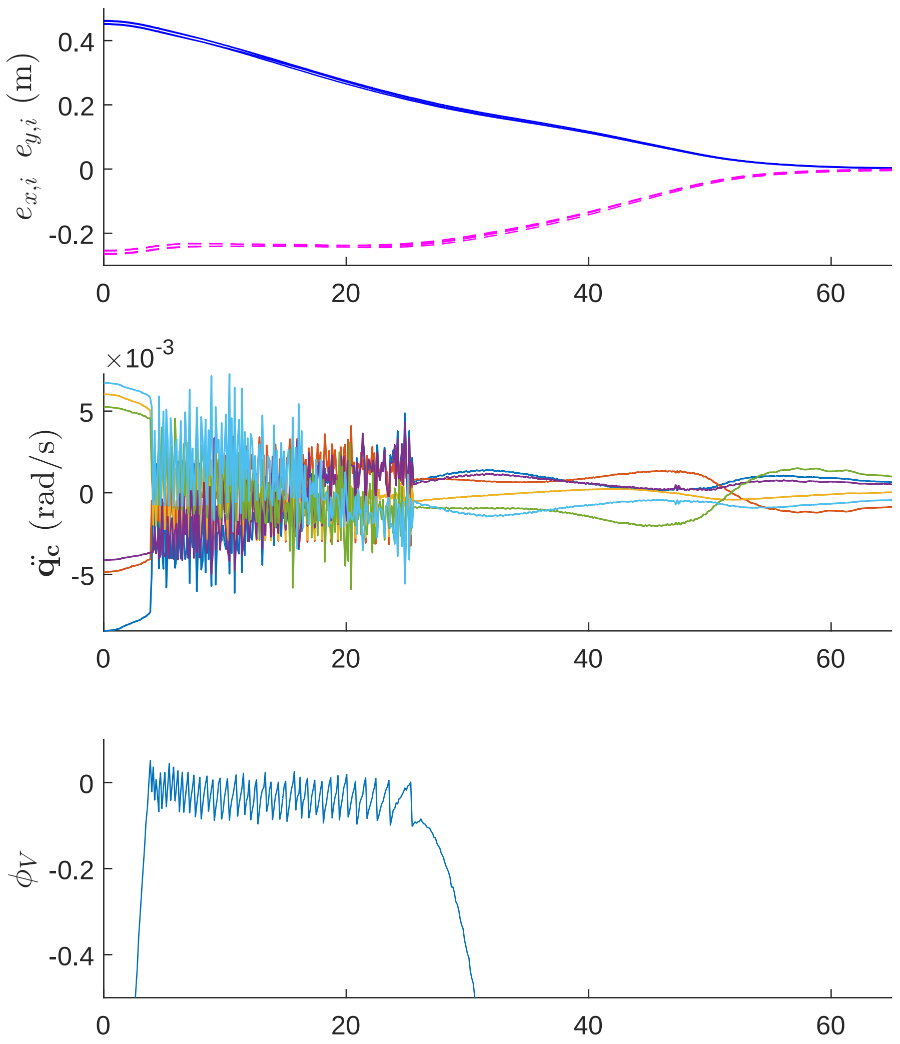


Figure 4.24: Visibility constraints experiment in IBVS with no errors in the gradient vector. From top to bottom plots: (1) visual feature errors; (2) control action $\ddot{\mathbf{q}}_c$; (3) constraint function vector ϕ_V.

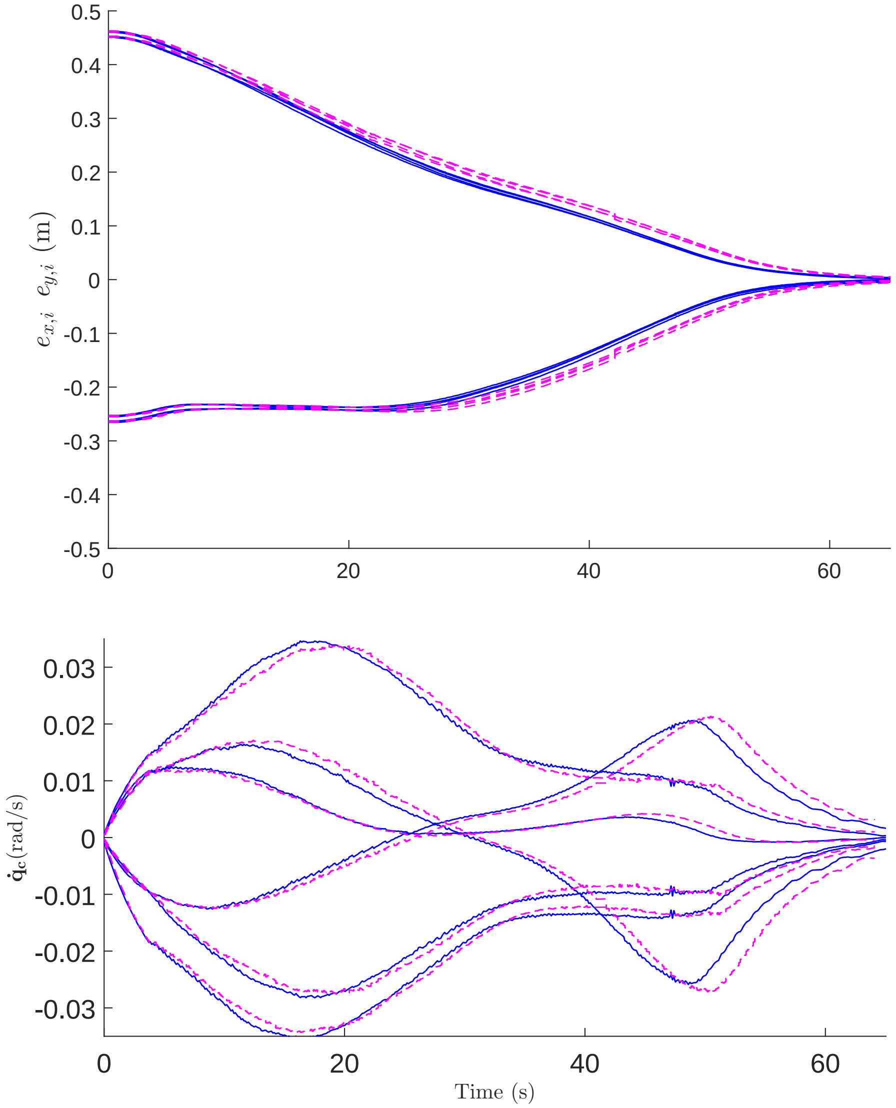


Figure 4.25: Comparison of visibility constraints experiment in IBVS, without and with errors in the gradient vector. (1) Image features trajectories, from initial (circle) to desired (cross) pose ; (2) commanded joint speeds $\dot{\mathbf{q}}_c$. Solid-blue experiment with no errors and dashed-magenta experiment with errors.

4.8 Experiment: workspace constraints in IBVS

A real experiment for automatic tool change with a 6R industrial manipulator is presented in this section to demonstrate the applicability of the proposed method. The experimentation has been carried out with the following setup (see Fig. 4.26): a Kuka Agilus KR6 R900 sixx manipulator in ceiling position 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. IBVS is considered with the eye-to-hand configuration. Moreover, the proposed SM method to satisfy constraints is illustrated with an obstacle avoidance problem. The experiment consists in moving the robot from the initial pose to the warehouse position avoiding an obstacle placed in the trajectory to the goal. The position of the markers on both the tool and the warehouse are updated using the dot tracker in ViSP (Visual Servoing Platform) (Marchand et al. (2005)), and the goal position of the tool markers is defined in the image plane relative to the warehouse markers. The pose of the tool markers is estimated to compute ${}^{c}\mathbf{V}_e$ and the robot Jacobian ${}^{e}\mathbf{J}_e$ is calculated with the joint positions read from the robot PC.

4.8.1 Experiment conditions and parameter values

Experiment was run under the following conditions:

i) Parameters used for the robot constraint for collision avoidance: 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.

   As before, 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 (4.37), can be readily obtained (Siciliano et al. (2009))

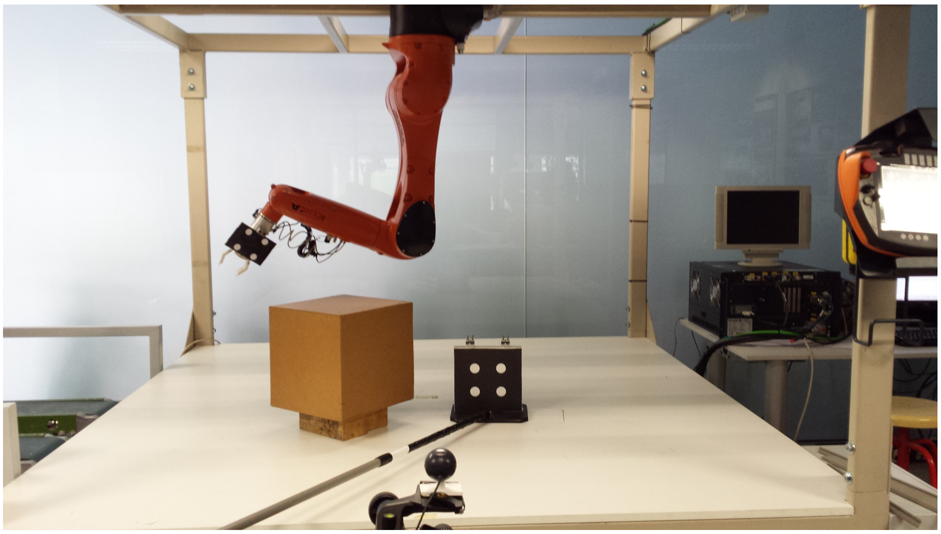


Figure 4.26: Experimental setup: 6R serial industrial manipulator in ceiling position with markers in the tool and the warehouse, camera out of the robot (eye-to-hand), obstacle and robot PC.

taking into account the Denavit-Hartenberg parameters of the 6R robot shown in Table 2.1.

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

iii) Three periodic threads are defined and scheduled following a fixed priority scheme, from highest to lowest: server, control and vision threads. The server period must be set to 4 milliseconds due to robot specification. Both the vision period and the control period T_s are set to 100 milliseconds to guarantee an appropriate scheduling.

iv) 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.

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

vi) The initial configuration considered for the robot is given by the joint position vector $\mathbf{q}(0) = \begin{bmatrix} 0.24 & -1.50 & 1.89 & 0.17 & -1.31 & 1.40 \end{bmatrix}^{\mathrm{T}}$ rad, and the ending of the servoing is defined in the image plane as the position of the tool markers relative to the warehouse markers.

vii) After the VS task is accomplished and the goal position is reached, the robot goes down to place the tool in the warehouse and back again.

4.8.2 Experimental results

A video with two experiments can be played at (video at double speed) `https://media.upv.es/player/?id=934ea160-14a6-11e7-a875-bd62e853f1c3`: the first one uses no algorithm to avoid the obstacle, while the second one uses the proposed SM method to satisfy constraints. Fig. 4.27 shows that the error in the second experiment 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. 4.28 shows the trajectories followed by the visual features in the image plane, whereas Fig. 4.29 shows that the robot workspace constraint becomes active around time interval 5s–12s. Fig. 4.30 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 SM algorithm is satisfactory.

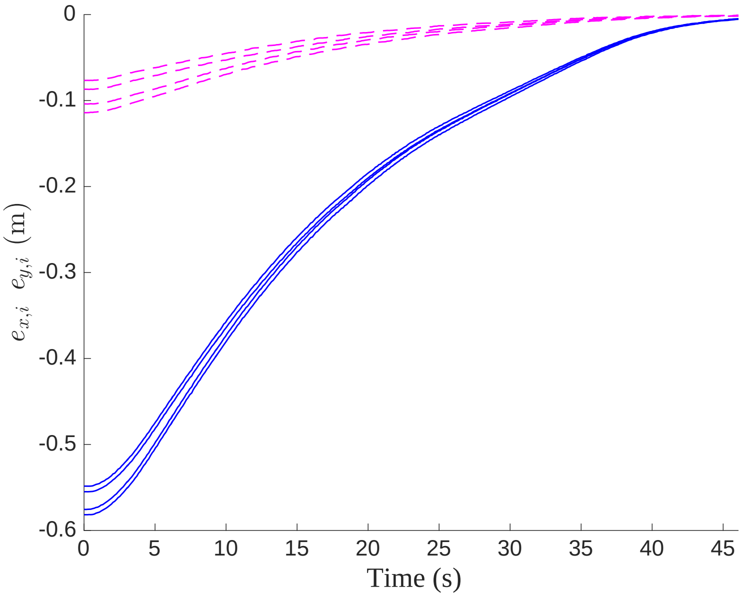


Figure 4.27: Visual features error: $e_{x,i}$ (solid, blue) and $e_{y,i}$ (dashed, magenta).

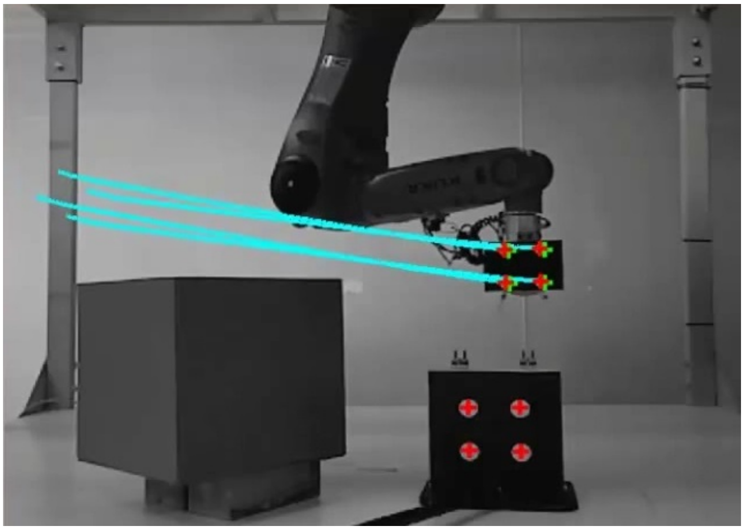


Figure 4.28: Feature trajectories in the image plane for the experiment.

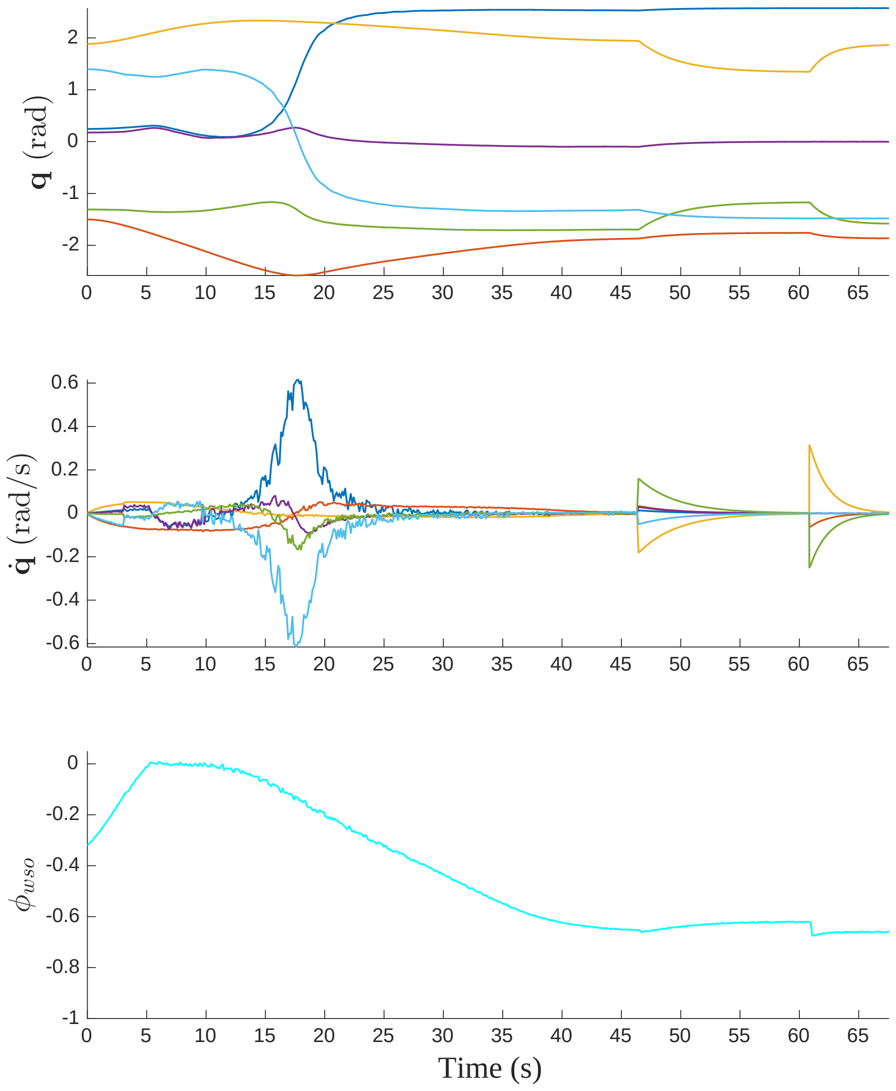


Figure 4.29: From top to bottom plots: (1) joint positions $\mathbf{q}$; (2) joint speeds $\dot{\mathbf{q}}$; (3) constraint function ϕ_{wso}. Once the VS task is accomplished, around time instant 46 s, it can be observed the robot movement to place the tool in the warehouse and to go back.

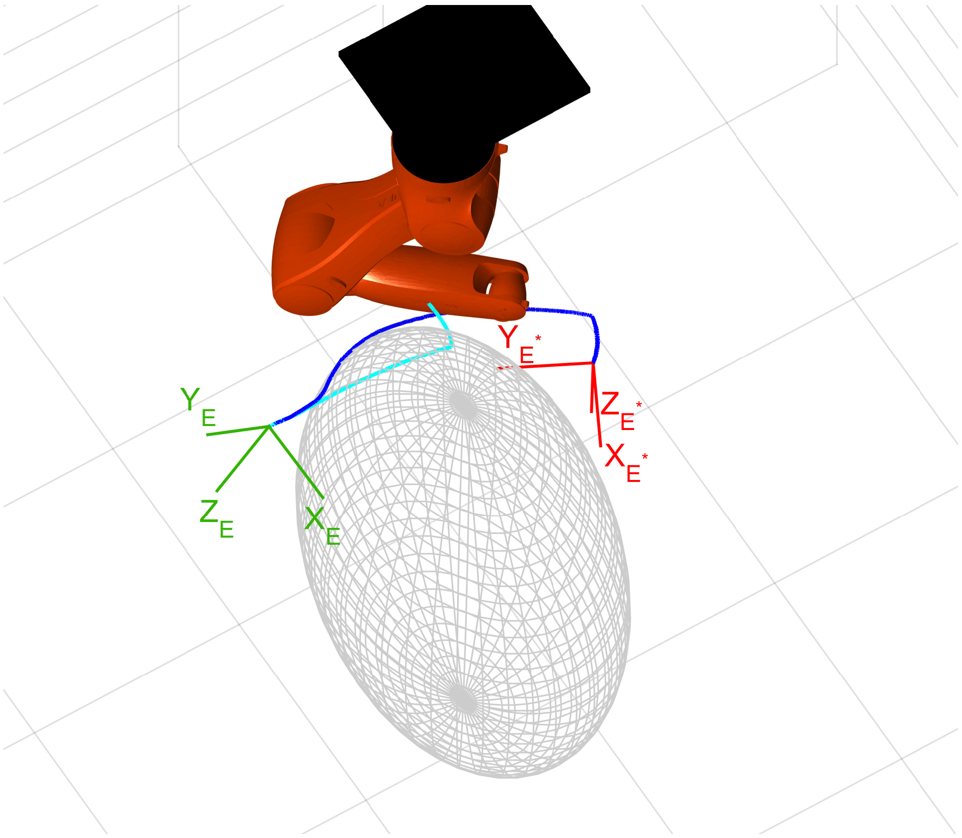


Figure 4.30: Detailed view of the fulfillment of the constraint for the obstacle avoidance in the real experiment. The lines represent the trajectory of the robot end-effector when the proposed SM method is used (dark-blue) or not (light-cyan) from initial pose (E-frame) to final pose (E*-frame). A graphical model of the robot is drawn at time instant 30 s. When the SM method is not active the safety zone enveloping the obstacle is invaded resulting in a collision and the task is aborted.

4.9 Conclusions

An approach to fulfill constraints in VS has been presented using SM concepts. In particular, the proposal uses one-side sliding control to satisfy the motion constraints (joint limits, speed limits, forbidden area to avoid collisions, task space limits and robot workspace limits) and visibility constraints (camera field-of-view and occlusions) of the VS applications. Moreover, another task with low-priority has been considered to make as small as possible the reference tracking error to properly track the target object.

Advantages of the proposed approach:

– In contrast to other similar techniques, like the *artificial potential fields* (Mezouar and Chaumette (2002)), the SM algorithm proposed to fulfill the constraints *fully utilizes* the available allowed space, e.g., the rectangle representing the image plane limits for the visibility constraint.

– 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 and, hence, the system stability is increased.

– The SM algorithm proposed to fulfill the constraints uses *partial information* of the system model, i.e., the Lie derivatives $L_f\phi_i$ (4.10) are not needed, only the Lie derivatives $\mathbf{L_g}\phi_i$ (4.9) are required. Therefore, only first order derivatives (gradient vectors, Jacobian matrices, etc.) are needed, see (4.31), and no second-order derivatives (Hessian matrices, derivative of Jacobians, etc.) are required, see (4.23)–(4.26).

– The SM algorithm is *robust* against environment modeling errors, i.e., it is not affected by the inaccuracies and uncertainties in the second-order derivatives mentioned above.

– Only requires a few program lines (see Section 4.3.7) and has *reduced computation time* since only linear algebra is used.

– Simplifies the *user interface* since the method directly deals with the fulfillment of the constraints and the tracking of the reference trajectory.

Main limitations of the method:

– The SM 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. (2012a)). Some of these situations could be avoided using a high-level planner with the complete geometric data of the problem to perform long-term planning. Nevertheless, the complexity and computational load of this planner are substantially greater than those of the method proposed in this work.

– Like other SM applications, the proposed method has the *chattering* drawback, see Section 4.3.5. Nevertheless, the chattering problem becomes negligible for reasonable fast sampling rates, see (4.45).

The feasibility and effectiveness of the proposed approach was illustrated in simulation for a simple 2D case and complex 3D case studies. Furthermore, real experimentation with a conventional 6R industrial manipulator was also presented to demonstrate its applicability and robustness, both for IBVS and PBVS. 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 for both position-based and image-based experiments even for a gradient vector error of 30%.

Moreover, an automated approach for tool changing in industrial robots using VS and SM control has been presented. In particular, the main task used IBVS in eye-to-hand configuration to properly place the tool in the warehouse, whereas SM control was used in a prioritized level to satisfy the robot constraints.

In the next chapter, an approach based on sliding-mode is proposed for reference tracking in robot visual servoing.

Chapter 5

High-Order Sliding Mode Control for Reference Tracking in Visual Servoing

5.1 Introduction

As stated in Section 2.1, two classic visual servoing (VS) approaches are defined depending on the workspace in which the control law is computed (Chaumette and Hutchinson (2008)): Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS). Independently of the workspace in which the control is carried out, another classification can be done focusing on the control law nature: *continuous* or *discontinuous control laws* (Ryan and Corless (1984)).

On the one hand, the most typical continuous control law used in VS applications for positioning or tracking tasks is based on computing a continuous joint velocity (Chaumette and Hutchinson (2008)) to be commanded to the joint actuators in order to obtain an exponential decrease of the error signal. In this respect, we can find in the literature a vast number of approaches, as for instance: classic PID controllers (Chaumette and Hutchinson (2006, 2007); Bonfe et al. (2002); Solanes et al. (2016); Elena et al. (2003)); optimal (Hashimoto et al. (1996); Chan et al. (2011); Allibert et al. (2010b); Hajiloo et al. (2016)) and robust controllers (Kragic and Christensen (2003); Mezouar and Chaumette (2000); Morel et al. (2005); Hammouda et al. (2015));

based on learning (Yang et al. (2002); Sadeghzadeh et al. (2015); Lee et al. (2017)); etc. However, other continuous approaches that are based on computing the joint accelerations to be commanded to the joint actuators can be found, either in PBVS(Fakhry and Wilson (1996)) or IBVS (Keshmiri et al. (2014)).

On the other hand, *discontinuous control laws* have been deeply studied in the context of sliding-mode (SM) control (Edwards and Spurgeon (1998)). In particular, several works have studied the use of SM for the main control law[1] of the VS system, mainly to increase its robustness against errors: authors in Zanne et al. (2000) used SM theory to design a 3D vision based controller that is robust to bounded parametric estimation errors; in Oliveira et al. (2009) and Oliveira et al. (2014), a SM control strategy based on a switching scheme and monitoring function was developed to deal with the uncertainties in the camera calibration parameters; authors in Li and Xie (2010) proposed a visual controller and a robot joint controller based on the SM control theory for a camera-in-hand planar two-link robot visual servo system in order to achieve strong robustness against parameter variations and external disturbances; in Kim et al. (2006), the SM control was applied to IBVS in order to increase the robustness on the parametric uncertainties; in Parsapour et al. (2015), a SM controller together with an estimator based on unscented Kalman observer cascading with Kalman filter demonstrated to be a stable and robust structure in PBVS, considering system uncertainties existing in the estimation model and observation noise; in Burger et al. (2015), a second order SM controller for PBVS was presented in order to control the end effector pose of a 7 DoF robot arm in eye-to-hand configuration; in Becerra and Sagüés (2011) and Becerra et al. (2011), a robust tracking control law under image noise and uncertainty of parameters was designed on the basis of SM theory for mobile robots using epipolar geometry in IBVS; in Parsapour and Taghirad (2015), the SM was integrated with kernel-based VS to improve the tracking error and expand the stability region; in Xin et al. (2016), rotation and translation SM controllers using SIFT features were designed to solve the robot VS problem; in Zhao et al. (2017), a two stage control scheme based on sliding surfaces was proposed for path-following and accurate positioning in PBVS for a robotic riveting system.

[1]SM has also been used in VS for other purposes. For instance, authors in Yu (2013) presented an approach in which a SM observer is applied to estimate the joint velocities of the VS system.

All works mentioned above use the joint velocities as the discontinuous control action for the SM controller. Therefore, the objective of this work is to develop a SM controller for VS that uses a high-order discontinuous control signal, i.e., joint accelerations or joint jerks, in order to obtain a *smoother behavior* and ensure the robot system stability. The proposed SM control approach is equivalent, in some sense, to the continuous control strategy mentioned above but has two main advantages: *robustness* and *low computational cost*; while its main limitation is the chattering drawback, although this problem becomes negligible for reasonable fast sampling rates.

The main contents of this chapter are represented using a mind map in Figure 5.1.

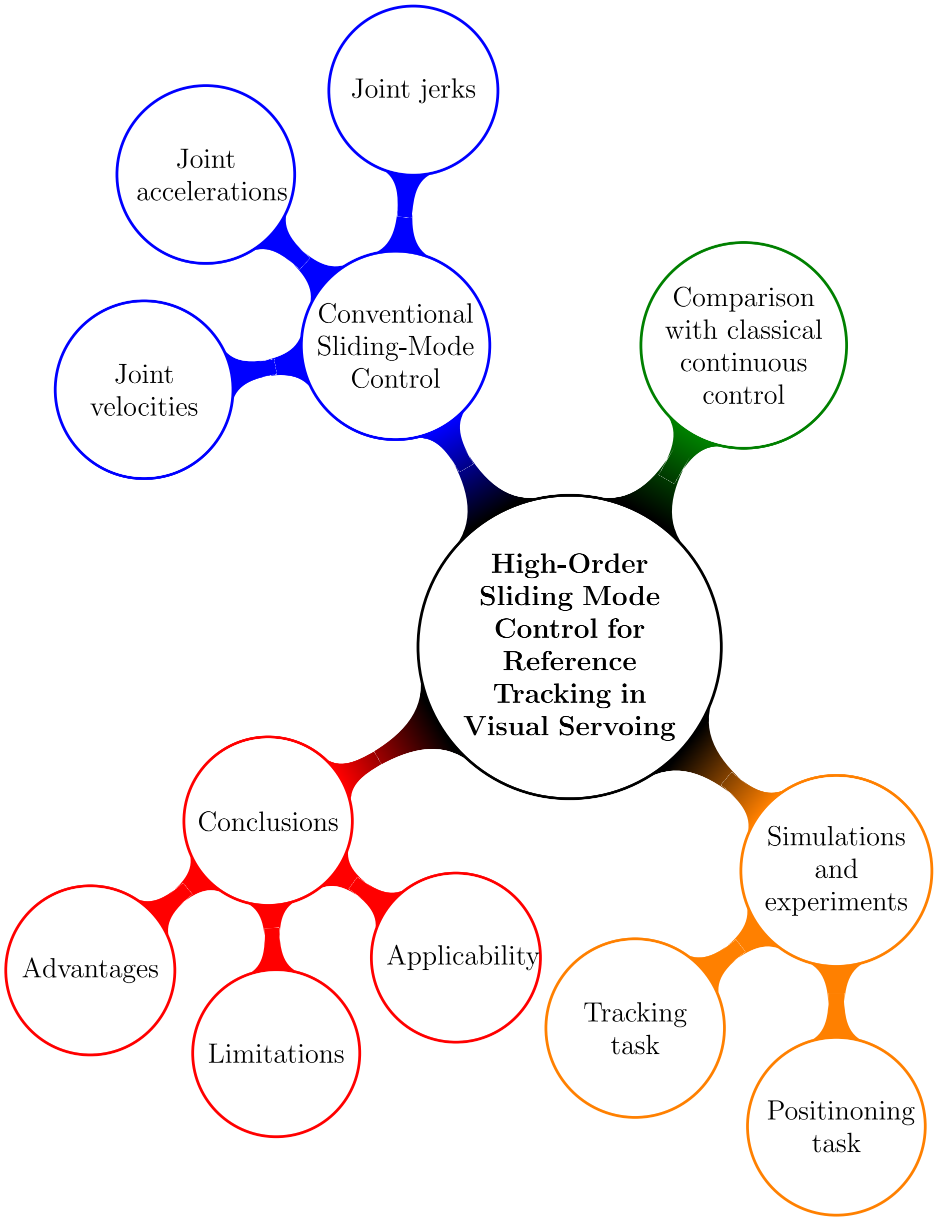

High-Order Sliding Mode Control for Reference Tracking in Visual Servoing
Conventional Sliding-Mode Control
Joint jerks
Joint accelerations
Joint velocities
Comparison with classical continuous control
Simulations and experiments
Tracking task
Positinoning task
Conclusions
Advantages
Limitations
Applicability

Figure 5.1: Mind map representing the main contents of this chapter.

5.2 Preliminaries

The content of this preliminaries section has been presented with full details in previous sections, but a small reminder is done here for better readability of this chapter.

Underlying robot controller. Depending on the control application (velocity, acceleration or jerk) the following equations are considered:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_c + \mathbf{d}_c \tag{5.1}$$

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_c + \mathbf{d}_c \tag{5.2}$$

$$\dddot{\mathbf{q}} = \dddot{\mathbf{q}}_c + \mathbf{d}_c, \tag{5.3}$$

where subscript c is used for the commanded variable and $\mathbf{d}_c$ represents the inaccuracies of the low-level control loop.

Kinematics. The first-, second- and third-order kinematics of the visual features vector $\mathbf{s}$ are:

$$\dot{\mathbf{s}} = \mathbf{J}_s\dot{\mathbf{q}} + \partial\mathbf{s}/\partial t \tag{5.4}$$

$$\ddot{\mathbf{s}} = \mathbf{J}_s\ddot{\mathbf{q}} + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{s}}/\partial t \tag{5.5}$$

$$\dddot{\mathbf{s}} = \mathbf{J}_s\dddot{\mathbf{q}} + 2\dot{\mathbf{J}}_s\ddot{\mathbf{q}} + \ddot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\ddot{\mathbf{s}}/\partial t, \tag{5.6}$$

where $\partial\mathbf{s}/\partial t$ is due to the target motion and $\mathbf{J}_s$ is the Jacobian matrix.

Dynamic model of robot manipulators. In this work, the dynamic model of the robot manipulators is not explicitly obtained, although it is relevant and should be considered for an optimal design of the underlying robot joint controller.

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

$$\mathbf{s}(\mathbf{q}, t) = \mathbf{s}_{ref}(t), \tag{5.7}$$

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

SM control laws. As stated in Section 3.1, the following control laws are proposed:

$$\mathbf{u} = -\mathbf{L_g}\boldsymbol{\phi}^{\mathrm{T}}\,\mathrm{sign}(\boldsymbol{\phi})\,u^{+} \tag{5.8}$$

$$\mathbf{u} = -\mathbf{L_g}\boldsymbol{\phi}^{\dagger}\,\mathrm{sign}(\boldsymbol{\phi})\,u^{+}, \tag{5.9}$$

where matrix $\mathbf{L_g}\boldsymbol{\phi}$ contains the row vectors $\mathbf{L_g}\phi_i$ of all constraints, $\boldsymbol{\phi}$ is a column vector with all the constraint functions ϕ_i, $\mathrm{sign}(\cdot)$ represents the *sign function* (typically used in SM control) and u^{+} is a positive constant to be chosen high enough to satisfy (3.3).

Simulations. The simulation results presented below were obtained using MATLAB®.

5.3 Proposed approach

5.3.1 Sliding mode control for reference tracking

5.3.1.1 Procedure to use sliding mode control

This section gives an overview of the required steps to use SM control. This "recipe" will be used in the next subsections.

The first step consists in defining the equality constraint to be satisfied: $\boldsymbol{\phi} = \mathbf{0}$. Typically, this constraint is chosen to be an ordinary differential equation in order to obtain the desired dynamics for the controlled system. In particular, for reference tracking, this differential equation is expressed in terms of the error variable, i.e., the difference between the current value and the reference value for the controlled variable. Note that, when the order of the differential equation is defined, the order of the corresponding discontinuous control action is also established, see Section 3.1.3. Therefore, the control design specification could be either the order of the differential equation or, alternatively, the order of the discontinuous control action.

The next step consists in obtaining the time-derivative of the constraint function vector $\boldsymbol{\phi}$ and identifying the Lie derivatives $L_f\boldsymbol{\phi}$ and $\mathbf{L_g}\boldsymbol{\phi}$ of the system at hand.

Finally, the control law given by (5.8), or alternatively (5.9) if the matrix inversion option is considered, has to be obtained.

5.3.1.2 Sliding mode control using joint velocities

This case considers the most simple equality constraint for reference tracking, which is straightforward obtained by rewriting (5.7) as:

$$\boldsymbol{\phi}_v(\mathbf{q},t) = \mathbf{s}(\mathbf{q},t) - \mathbf{s}_{ref}(t) = \mathbf{e} = \mathbf{0}, \tag{5.10}$$

where $\mathbf{e}$ represents the tracking position error of the visual feature vector $\mathbf{s}$.

Taking into account the first-order kinematics in (5.4), the time-derivative of the constraint function vector $\boldsymbol{\phi}_v$ is given by:

$$\begin{aligned}\dot{\boldsymbol{\phi}}_v = \dot{\mathbf{s}} - \dot{\mathbf{s}}_{ref} &= \mathbf{J}_s\dot{\mathbf{q}} + \partial\mathbf{s}/\partial t - \dot{\mathbf{s}}_{ref} \\ &= \mathbf{J}_s\dot{\mathbf{q}} + \partial\mathbf{e}/\partial t,\end{aligned} \tag{5.11}$$

where it can be noticed that it explicitly depends on the joint velocities. Hence, in order to satisfy the relative degree condition mentioned above, the control input vector $\mathbf{u}$ for this case is the commanded joint velocity $\dot{\mathbf{q}}_c$.

Therefore, from (5.11) and (5.1) the Lie derivatives in (3.3) for the constraint function vector $\boldsymbol{\phi}_v$ are given by:

$$\mathbf{L_g}\boldsymbol{\phi}_v = \mathbf{J}_s \tag{5.12}$$

$$L_f\boldsymbol{\phi}_v = \mathbf{J}_s\,\mathbf{d}_c + \partial\mathbf{e}/\partial t. \tag{5.13}$$

Therefore, the control law given by (5.9) (alternatively, (5.8) can be considered if matrix inversion should be avoided) results in:

$$\dot{\mathbf{q}}_c = -\mathbf{J}_s^{\dagger}\,\mathrm{sign}(\mathbf{e})\,u^+. \tag{5.14}$$

It is interesting to note that the sliding surface given by (5.10) has already been used in VS for a SM velocity controller in Zanne et al. (2000), Li and Xie (2010), Kim et al. (2006), Becerra and Sagüés (2011) and Becerra et al. (2011).

5.3.1.3 Sliding mode control using joint accelarations

The above constraint $\boldsymbol{\phi}_v$ will be modified via the higher-order SM described in Section 3.1.2 as follows:

$$\boldsymbol{\phi}_a(\mathbf{q},\dot{\mathbf{q}},t) = \boldsymbol{\phi}_v + K_a\dot{\boldsymbol{\phi}}_v = \mathbf{e} + K_a\dot{\mathbf{e}} = \mathbf{0}, \tag{5.15}$$

where K_a is a positive parameter that determines the time constant of the approach to $\boldsymbol{\phi}_v = \mathbf{0}$.

Taking into account the first- and second-order kinematics in (5.4) and (5.5), the time-derivative of the constraint function vector $\boldsymbol{\phi}_a$ is given by:

$$\begin{aligned}\dot{\boldsymbol{\phi}}_a =& \dot{\mathbf{e}} + K_a\ddot{\mathbf{e}} = \dot{\mathbf{s}} - \dot{\mathbf{s}}_{ref} + K_a(\ddot{\mathbf{s}} - \ddot{\mathbf{s}}_{ref}) \\ =& \mathbf{J}_s\dot{\mathbf{q}} + \partial\mathbf{e}/\partial t + K_a(\mathbf{J}_s\ddot{\mathbf{q}} + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{e}}/\partial t),\end{aligned} \tag{5.16}$$

where it can be noticed that it explicitly depends on the joint accelerations. Hence, in order to satisfy the relative degree condition mentioned above, the control input vector $\mathbf{u}$ for this case is the commanded joint acceleration $\ddot{\mathbf{q}}_c$.

Therefore, the main advantage of considering (5.15) instead of (5.10) is that the control is smoother: the joint velocity is continuous instead of discontinuous and the error equation is a first-order differential equation instead of an static equation.

From (5.16) and (5.2) the Lie derivatives for the constraint function vector $\boldsymbol{\phi}_a$ are given by:

$$\mathbf{L_g}\boldsymbol{\phi}_a = K_a\mathbf{J}_s \tag{5.17}$$

$$L_f\boldsymbol{\phi}_a = (\mathbf{J}_s + K_a\dot{\mathbf{J}}_s)\dot{\mathbf{q}} + K_a\mathbf{J}_s\mathbf{d}_c + \partial(\mathbf{e} + K_a\dot{\mathbf{e}})/\partial t. \tag{5.18}$$

Therefore, the control law given by (5.9) results in:

$$\ddot{\mathbf{q}}_c = -K_a^{-1}\mathbf{J}_s^{\dagger}\,\mathrm{sign}(\mathbf{e} + K_a\dot{\mathbf{e}})\,u^+. \tag{5.19}$$

5.3.1.4 Sliding mode control using joint jerks

As before, the constraint $\boldsymbol{\phi}_a$ will be modified via the higher-order SM described in Section 3.1.2 as follows:

$$\begin{aligned}\boldsymbol{\phi}_j(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, t) =& \boldsymbol{\phi}_a + K_j\dot{\boldsymbol{\phi}}_a = \mathbf{e} + K_a\dot{\mathbf{e}} + K_j\dot{\mathbf{e}} + K_aK_j\ddot{\mathbf{e}} \\ =& \mathbf{e} + K_{j1}\dot{\mathbf{e}} + K_{j2}\ddot{\mathbf{e}} = \mathbf{0},\end{aligned} \tag{5.20}$$

where K_j is a positive parameter that determines the time constant of the approach to $\boldsymbol{\phi}_a = \mathbf{0}$ and $K_{j1} = K_a + K_j$ and $K_{j2} = K_aK_j$ represent the coefficients of the differential equation above.

Taking into account (5.4), (5.5) and (5.6), the time-derivative of the constraint function vector $\boldsymbol{\phi}_j$ is given by:

$$\begin{aligned}\dot{\boldsymbol{\phi}}_j =& \dot{\mathbf{e}} + K_{j1}\ddot{\mathbf{e}} + K_{j2}\dddot{\mathbf{e}} \\ =& \dot{\mathbf{s}} - \dot{\mathbf{s}}_{ref} + K_{j1}(\ddot{\mathbf{s}} - \ddot{\mathbf{s}}_{ref}) + K_{j2}(\dddot{\mathbf{s}} - \dddot{\mathbf{s}}_{ref}) \\ =& \mathbf{J}_s\dot{\mathbf{q}} + \partial\mathbf{e}/\partial t + K_{j1}(\mathbf{J}_s\ddot{\mathbf{q}} + \dot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\dot{\mathbf{e}}/\partial t) \\ &+ K_{j2}(\mathbf{J}_s\dddot{\mathbf{q}} + 2\dot{\mathbf{J}}_s\ddot{\mathbf{q}} + \ddot{\mathbf{J}}_s\dot{\mathbf{q}} + \partial\ddot{\mathbf{e}}/\partial t),\end{aligned} \tag{5.21}$$

where it can be noticed that it explicitly depends on the joint jerks. Hence, in order to satisfy the relative degree condition mentioned above, the control input vector $\mathbf{u}$ for this case is the commanded joint jerk $\dddot{\mathbf{q}}_c$.

Therefore, as before, the main advantage of considering (5.20) instead of (5.15) is that the control is smoother: the joint acceleration is continuous instead of discontinuous and the error differential equation is of second-order instead of first-order.

From (5.21) and (5.3) the Lie derivatives for the constraint function vector $\boldsymbol{\phi}_j$ are given by:

$$\mathbf{L_g}\boldsymbol{\phi}_j = K_{j2}\mathbf{J}_s \tag{5.22}$$

$$\begin{aligned}L_f\boldsymbol{\phi}_j =& (\mathbf{J}_s + K_{j1}\dot{\mathbf{J}}_s + K_{j2}\ddot{\mathbf{J}}_s)\dot{\mathbf{q}} + (K_{j1}\mathbf{J}_s + 2K_{j2}\dot{\mathbf{J}}_s)\ddot{\mathbf{q}} \\ &+ K_{j2}\mathbf{J}_s\mathbf{d}_c + \partial(\mathbf{e} + K_{j1}\dot{\mathbf{e}} + K_{j2}\ddot{\mathbf{e}})/\partial t.\end{aligned} \tag{5.23}$$

Therefore, the control law given by (5.9) results in:

$$\dddot{\mathbf{q}}_c = -K_{j2}^{-1}\mathbf{J}_s^{\dagger}\,\mathrm{sign}(\mathbf{e} + K_{j1}\dot{\mathbf{e}} + K_{j2}\ddot{\mathbf{e}})\,u^{+}. \tag{5.24}$$

5.3.1.5 Additional remarks

Chattering. Discrete-time implementations of any practical SM control makes the system leave the ideal SM and oscillate with finite frequency and amplitude inside a band around $\boldsymbol{\phi} = \mathbf{0}$, namely *chattering* (Edwards and Spurgeon (1998)). The chattering band $\triangle\boldsymbol{\phi}$ of the proposal can be obtained using the Euler-integration of the discontinuous control action given by (5.9), that is:

$$\triangle\boldsymbol{\phi} = T_s\,|\mathbf{L_g}\boldsymbol{\phi}\,\mathbf{u}| = T_s\,u^{+}\,\mathbf{1}, \tag{5.25}$$

where T_s is the sampling time of the robot control and $\mathbf{1}$ is the column vector with all its components equal to one.

Time derivatives. The proposed approach requires the first-order time derivatives of some variables. As in many other applications, the simplest way to deal with this issue consists in using numerical differentiation, e.g., the well-known backward Euler approximation. However, some kind of filtering should be previously applied to the actual variable when non-negligible noise is present. It is important to remark that, the low-pass filter used for noise reduction must not limit the band-width of the control law In particular, for the proposed SM Approach, the theoretical frequency of the control low signal is equal to $(2T_s)^{-1}$ Hertz and, hence, the filter attenuation at this frequency should be relatively small.

5.3.2 Comparison with classical continuous control

5.3.2.1 Classical continuous control using joint velocities

Substituting the first-order kinematics of the robot system (5.4) and the low-level control equation (5.1) in the first-order differential equation of the error given by (5.15), it is obtained the following equation:

$$\begin{aligned}\mathbf{e} + K_a\dot{\mathbf{e}} =& \mathbf{e} + K_a(\mathbf{J}_s\dot{\mathbf{q}} + \partial\mathbf{s}/\partial t - \dot{\mathbf{s}}_{ref}) \\ =& \mathbf{e} + K_a(\mathbf{J}_s(\dot{\mathbf{q}}_c + \mathbf{d}_c) + \partial\mathbf{e}/\partial t) = \mathbf{0},\end{aligned} \tag{5.26}$$

and the commanded joint velocity vector results in:

$$\dot{\mathbf{q}}_c = -\mathbf{J}_s^{\dagger}(K_a^{-1}\mathbf{e} + \partial\mathbf{e}/\partial t) - \mathbf{d}_c, \tag{5.27}$$

which is the most typical control law used in VS (Chaumette and Hutchinson (2008)) in order to obtain an exponential decrease of the tracking error.

The partial derivative $\partial\mathbf{e}/\partial t$ is typically estimated (see Chaumette and Hutchinson (2008)) using the first-order kinematics given by (5.4), yielding:

$$\partial\mathbf{e}/\partial t = \dot{\mathbf{e}} - \mathbf{J}_s\dot{\mathbf{q}}, \tag{5.28}$$

and, hence, the time-derivative of the error vector $\dot{\mathbf{e}}$ is also needed like in the SM control given by (5.19).

5.3.2.2 Classical continuous control using joint accelarations

Substituting the second-order kinematics of the robot system (5.5) and the low-level control equation (5.2) in the second-order differential equation of the error given by (5.20), it is obtained the following equation:

$$\begin{aligned}\mathbf{e}+K_{j1}\dot{\mathbf{e}}+K_{j2}\ddot{\mathbf{e}}=\mathbf{e}+K_{j1}\dot{\mathbf{e}}\\ +K_{j2}(\mathbf{J}_s(\ddot{\mathbf{q}}_c+\mathbf{d}_c)+\dot{\mathbf{J}}_s\dot{\mathbf{q}}+\partial\dot{\mathbf{e}}/\partial t)=\mathbf{0},\end{aligned} \tag{5.29}$$

and the commanded joint acceleration vector results in:

$$\ddot{\mathbf{q}}_c=-\mathbf{J}_s^{\dagger}(K_{j2}^{-1}\mathbf{e}+K_{j2}^{-1}K_{j1}\dot{\mathbf{e}}+\dot{\mathbf{J}}_s\dot{\mathbf{q}}+\partial\dot{\mathbf{e}}/\partial t)-\mathbf{d}_c, \tag{5.30}$$

which represents the classical operational space robot control (Siciliano et al. (2009)) that has already been used in VS appliactions by Fakhry and Wilson (1996) for PBVS and by Keshmiri et al. (2014) for IBVS.

As above, the partial derivative $\partial\dot{\mathbf{e}}/\partial t$ can be estimated using the second-order kinematics given by (5.5), yielding:

$$\partial\dot{\mathbf{e}}/\partial t=\ddot{\mathbf{e}}-\mathbf{J}_s\ddot{\mathbf{q}}-\dot{\mathbf{J}}_s\dot{\mathbf{q}}, \tag{5.31}$$

and, hence, the second-order time-derivative of the error vector $\ddot{\mathbf{e}}$ is also needed like in the SM control given by (5.24).

5.3.2.3 Equivalences between sliding mode control and classical continuous control

Assuming that the SM control is in the SM phase, i.e., the reaching phase has finished and the system on the sliding surface, the SM controls proposed in Section 5.3.1.3 and Section 5.3.1.4 are equivalent to the classical velocity and acceleration controls described in Section 5.3.2.1 and Section 5.3.2.2, respectively, in the sense that both approaches give rise to the same first-order or second-order differential equation for the tracking error. In particular, if the SM control is in the SM phase and considering the same initial conditions, both the SM control and its continuous equivalent give rise to the same value for the joint velocities or joint accelerations, as will be shown in the simulations of Section 5.5 and Section 5.6.

However, despite the mentioned equivalences, the proposed SM strategy presents several significant advantages over its classical continuous counterpart that are discussed below.

5.4 Conditions for the simulations and experiments

The proposed approach can be used either for PBVS or IBVS. However, the simulations and experiments are focused on PBVS, since it is less robust to calibration and modeling errors than IBVS (Chaumette and Hutchinson (2008); Hafez et al. (2008); Kermorgant and Chaumette (2011)) and would get more benefit from the proposed SM approach. Furthermore, the simulations and experiments have been developed for the eye-in-hand configuration, i.e., camera rigidly attached to the robot end-effector, although the method can also be used for eye-to-hand configuration (camera does not move with the robot).

The typical visual feature vector and interaction matrix used in PBVS are considered:

$$\mathbf{s} = \begin{bmatrix} {}^{C^*}\mathbf{t}_C^{\mathrm{T}} & {}^{C^*}\theta\mathbf{u}_C^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \tag{5.32}$$

$$\mathbf{L}_s = \begin{bmatrix} {}^{C^*}\mathbf{R}_C & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}. \tag{5.33}$$

The Jacobian matrix $\mathbf{J}_s$ required for the reference tracking control laws is computed 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 ${}^{e}\mathbf{J}_e$. For further details see Section 2.1.

5.5 Simulation: positioning task

Fig. 5.2 depicts the VS application in consideration for the simulated positioning task with the following elements: 6R 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^*.

5.5.1 Conditions for the simulated positioining task

Simulation for the positioning task was run under the following conditions:

i) Parameters used for the camera: focal lengths $f_x = 640$ and $f_y = 480$ pixels; principal point $[u_0, v_0] = [320, 240]$ pixels; and camera to end-effector transformation matrix ${}^{c}\mathbf{M}_e = \mathbf{I}_4$ where $\mathbf{I}_4$ represents the identity

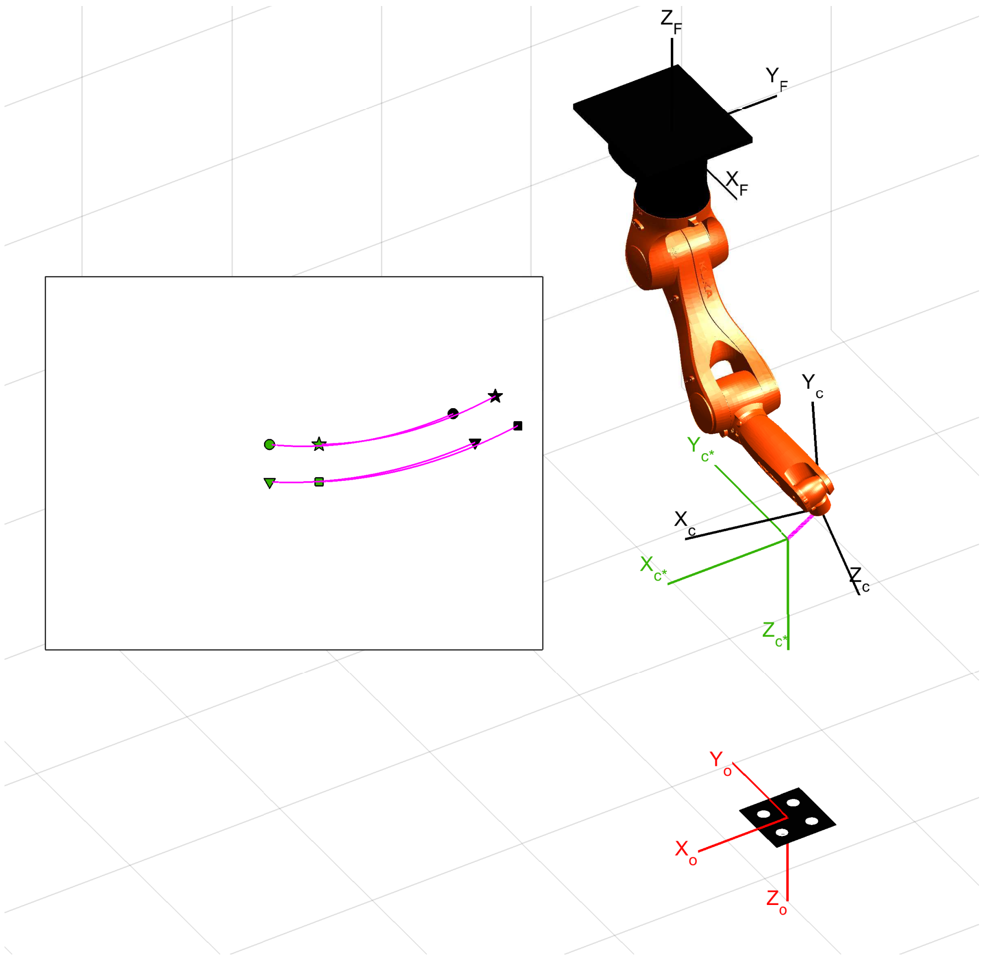


Figure 5.2: 3D Positioning task: 6R robot, target object with four markers, involved coordinate frames, and resulting 3D trajectory of the camera (magenta). Image plane on the left-hand side: initial position (black), desired position (green), and trajectory of the features (magenta). For clarity, only the trajectories for SM control using joints acceleration and $\mathbf{J}_s^\dagger$ are depicted.

matrix of dimension 4, i.e., the camera pose is equivalent to the end-effector pose.

ii) Coefficients for the error differential equation: $K_a = 5$ for the first-order differential equation; $K_{j2} = 5$ and $K_{j1} = 3\sqrt{K_{j2}}$ (i.e., a value of 1.5 for the damping ration) for the second-order differential equation.

iii) The initial configuration considered for the robot is given by the joint position vector $\mathbf{q}(0) = \begin{bmatrix} 0 & -0.82 & 0.79 & -0.35 & -1.57 & -2.09 \end{bmatrix}^{\mathrm{T}}$ rad, yielding an initial camera pose given by the transformation matrix ${}^{F}\mathbf{M}_{C}(0) = \begin{bmatrix} 0.755 & 0.027 & 0.564 & -2.7923 & -0.0250 & 2.1038 \end{bmatrix}^{\mathrm{T}}$ in compact notation.

iv) A *static* target object is considered with the pose given by the transformation matrix ${}^{F}\mathbf{M}_{O} = \begin{bmatrix} 0.755 & 0.027 & 0.564 & \pi & 0 & \pi/2 \end{bmatrix}^{\mathrm{T}}$ in compact notation and with four markers given by the following points with respect to the object frame: ${}^{O}\mathbf{p}_1 = \begin{bmatrix} -0.05 & -0.05 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^{O}\mathbf{p}_2 = \begin{bmatrix} 0.05 & -0.05 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^{O}\mathbf{p}_3 = \begin{bmatrix} 0.05 & 0.05 & 0 \end{bmatrix}^{\mathrm{T}}$ m, ${}^{O}\mathbf{p}_4 = \begin{bmatrix} -0.05 & 0.05 & 0 \end{bmatrix}^{\mathrm{T}}$ m, that is, the four markers are the vertices of a square with a side length of 0.1 m.

v) The desired camera pose is given by the transformation matrix ${}^{F}\mathbf{M}_{O} = \begin{bmatrix} 0.624 & 0.001 & 0.377 & \pi & 0 & \pi/2 \end{bmatrix}^{\mathrm{T}}$ in compact notation.

vi) Control action amplitude u^{+} for the SM control: $u^{+} = 1$ when $\mathbf{J}_s^{\dagger}$ is used and $u^{+} = 3$ when $\mathbf{J}_s^{\mathrm{T}}$ is used.

vii) For comparison purposes between the proposed SM approach and the continuous equivalent, the initial value for the joint velocities and joint accelerations is chosen so that the system starts on the sliding surface Φ and, hence, the reaching phase is not present, see Section 5.3.2.3. In particular, in order to satisfy (5.15) when joint accelerations are used, the initial value for the joint velocities is computed as $\dot{\mathbf{q}}(0) = -\mathbf{J}_s^{\dagger}\dot{\mathbf{e}}(0)$ with $\dot{\mathbf{e}}(0) = -K_{T,p}\mathbf{e}(0)$. Similarly, in order to satisfy (5.20) when joint jerks are used, the initial value for the joint velocities is set to zero $\dot{\mathbf{q}}(0) = 0$, i.e., $\dot{\mathbf{e}}(0) = 0$, and the initial value for the joint accelerations is computed as $\ddot{\mathbf{q}}(0) = -\mathbf{J}_s^{\dagger}\ddot{\mathbf{e}}(0)$ with $\ddot{\mathbf{e}}(0) = -K_{T,p}\mathbf{e}(0)$.

viii) The simulation was carried out with a sampling time T_s of 1 millisecond.

5.5.2 Simulation results for the positining task

The results of the simulation are depicted at different figures. In particular, Fig. 5.2 shows that, as expected in PBVS approaches, the resulting camera trajectory in the 3D space is a straight line from the initial to the desired camera pose, whilst the trajectory of the features in the image plane describe a non-straight line. For clarity, only the trajectories for SM control using accelerations and $\mathbf{J}_s^\dagger$ are depicted. The differences between all the approaches is shown in the following figures.

Fig. 5.3 and Fig. 5.4 show the position error $\mathbf{e}$, the integral of the control action (i.e., $\dot{\mathbf{q}}$ or $\ddot{\mathbf{q}}$) and the constraint function vector (i.e., $\boldsymbol{\phi}_a$ or $\boldsymbol{\phi}_j$) for the SM control using joint accelerations and joint jerks, respectively. Note that the behavior of the position errors and joint velocities or accelerations is very similar when using $\mathbf{J}_s^\dagger$ and $\mathbf{J}_s^{\mathrm{T}}$, whereas the difference in the chattering band (see the constraint function) is due to the value used in each case for the control action amplitude u^+.

Fig. 5.5 compares the SM controls to their respective continuous equivalents in terms of joint speeds or joint accelerations, i.e., $\dot{\mathbf{q}}$ or $\ddot{\mathbf{q}}$, and in terms of the Euclidean norm (in the sequel, unless stated otherwise, it will be assumed the Euclidean norm) of the tracking error, i.e., $\|\mathbf{e}\|$. The differences are similar for the SM control using joint accelerations and joint jerks, and the norm of the tracking error increases around one order of magnitude when the transpose of the Jacobian matrix is used instead of the pseudoinverse.

A video of this simulated tracking task for the SM control using joint accelerations and $\mathbf{J}_s^\dagger$ (the other cases are very similar) can be played at `https://media.upv.es/player/?id=29817e60-11b3-11e7-be40-11c7233e792d`.

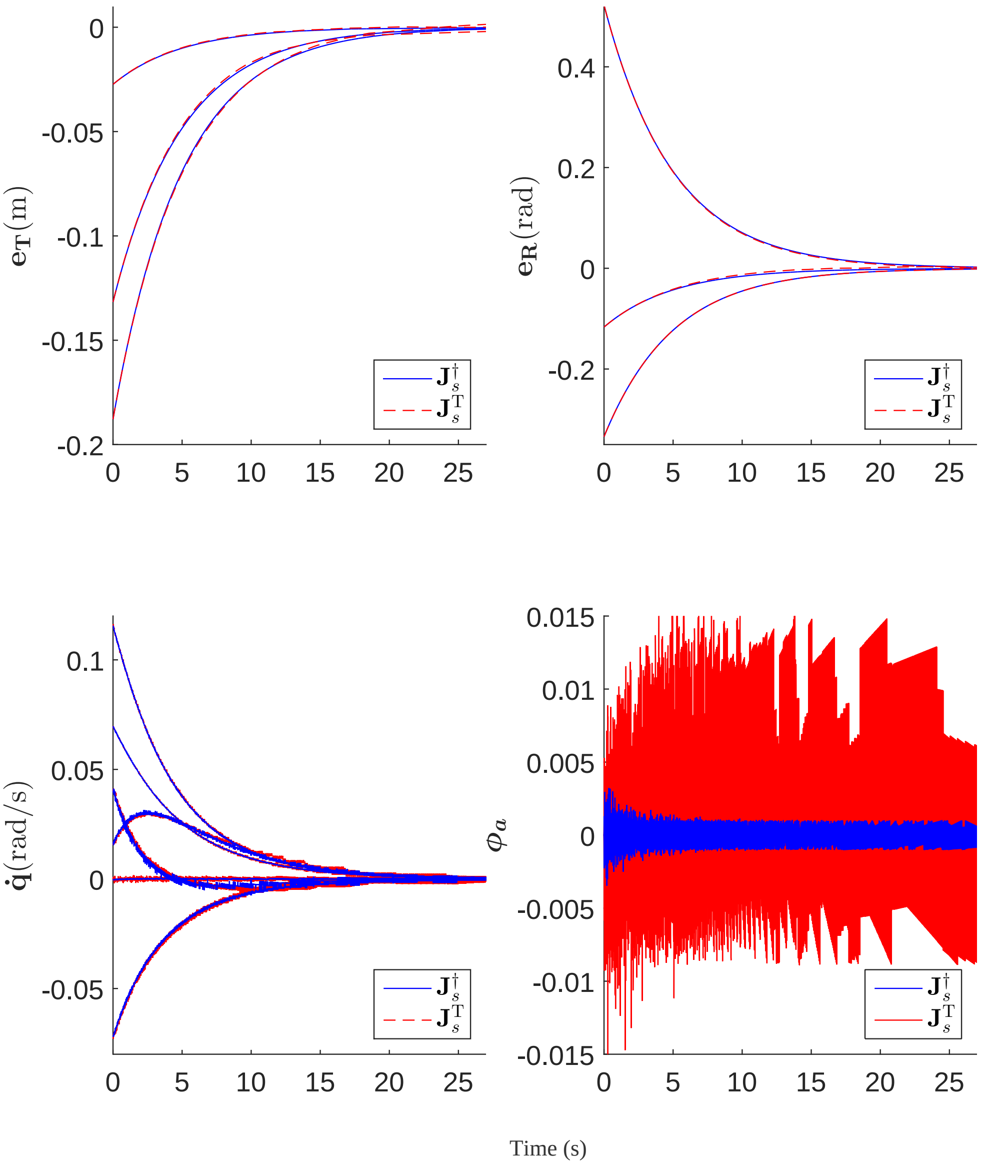


Figure 5.3: Simulated positioning task. SM control using joint accelerations. From top to bottom and left to right plots: (1) translation errors $\mathbf{e_T}$; (2) rotation errors $\mathbf{e_R}$; (3) joint speeds $\dot{\mathbf{q}}$; (4) Constraint function vector $\boldsymbol{\phi}_a$.

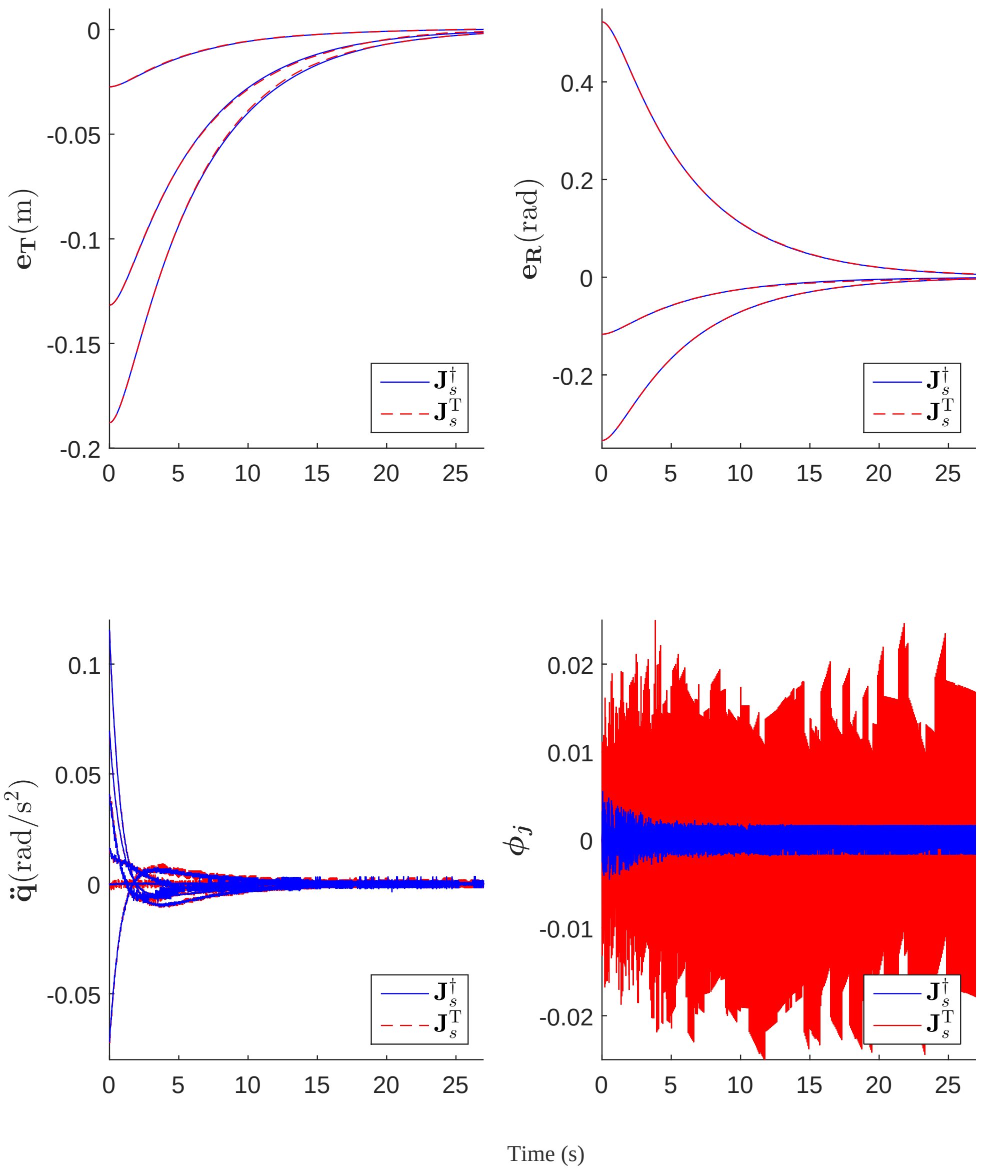


Figure 5.4: Simulated positioning task. SM control using joint jerks. From top to bottom and left to right plots: (1) translation errors $\mathbf{e_T}$; (2) rotation errors $\mathbf{e_R}$; (3) joint accelerations $\ddot{\mathbf{q}}$; (4) Constraint function vector $\boldsymbol{\phi}_a$.

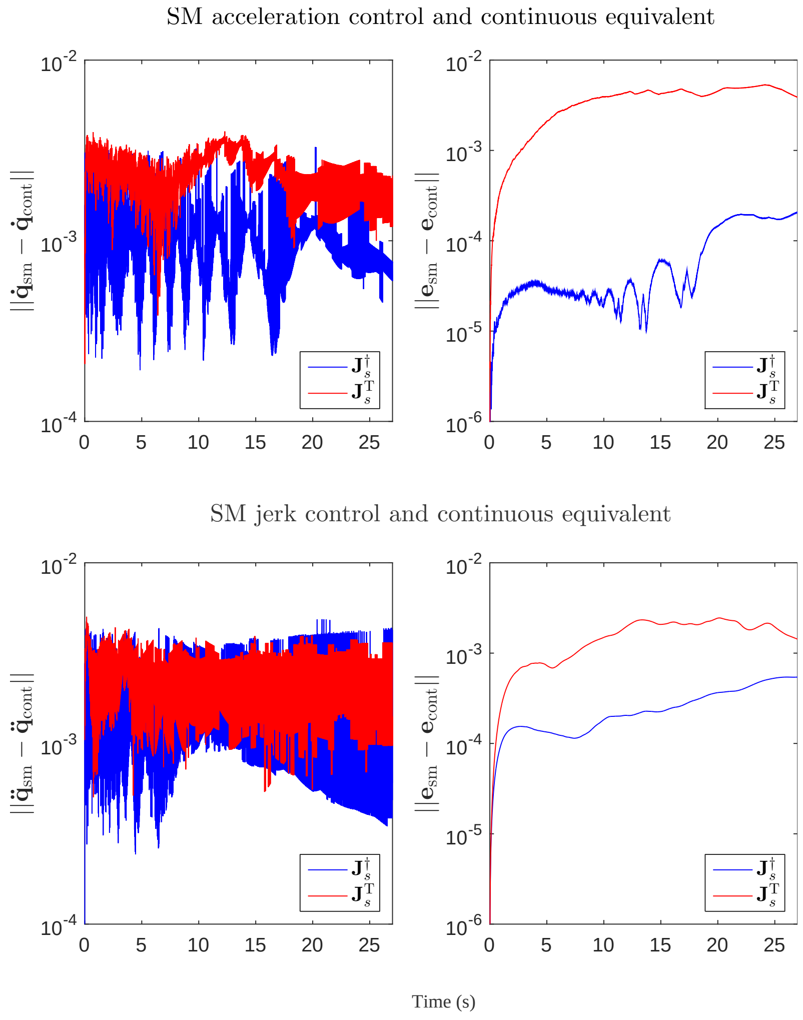


Figure 5.5: Simulated positioning task. SM control versus the continuous equivalent. From top to bottom and left to right plots: (1) Norm of the difference in $\dot{\mathbf{q}}$ between SM control using joint accelerations $\dot{\mathbf{q}}_{\mathrm{sm}}$ and the continuous equivalent $\dot{\mathbf{q}}_{\mathrm{cont}}$; (2) Norm of the difference in $\mathbf{e}$ between SM control using joint accelerations $\mathbf{e}_{\mathrm{sm}}$ and the continuous equivalent $\mathbf{e}_{\mathrm{cont}}$. (3) Norm of the difference in $\ddot{\mathbf{q}}$ between SM control using joint jerks $\ddot{\mathbf{q}}_{\mathrm{sm}}$ and the continuous equivalent $\ddot{\mathbf{q}}_{\mathrm{cont}}$; (4) Norm of the difference in $\mathbf{e}$ between SM control using joint jerks $\mathbf{e}_{\mathrm{sm}}$ and the continuous equivalent $\mathbf{e}_{\mathrm{cont}}$.

5.6 Simulation: tracking task

As above, Fig. 5.6 depicts the VS tracking application in consideration for the simulated tracking task.

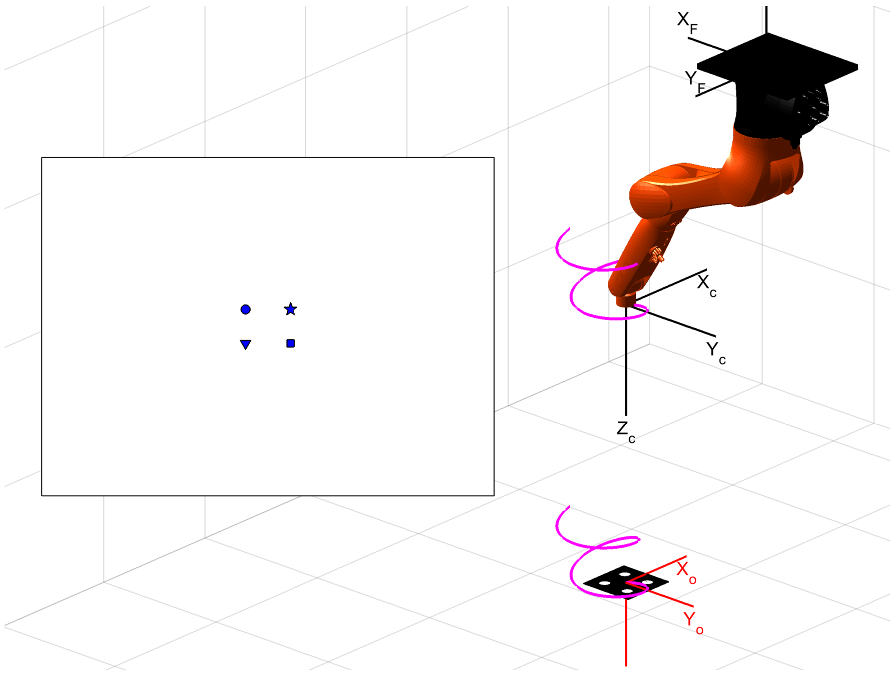


Figure 5.6: 3D Tracking task: 6R robot, target object with four markers, object trajectory, involved coordinate frames, and resulting 3D trajectory of the camera (magenta). For clarity, only the trajectories for SM control using joints acceleration and $\mathbf{J}_s^{\dagger}$ is depicted. Image plane on the left-hand side: Constant desired features position.

5.6.1 Conditions for the simulated tracking task

Simulation was run under the same conditions as the positioning task except for the following:

i) Coefficients for the error differential equation: $K_a = 0.4$ for the first-order differential equation; $K_{j2} = 0.4$ and $K_{j1} = 3\sqrt{K_{j2}}$ for the second-order differential equation.

ii) The initial configuration considered for the robot is given by the joint position vector $\mathbf{q}(0) = \begin{bmatrix} 0 & -0.40 & -0.65 & 0 & -0.52 & -1.57 \end{bmatrix}^{\mathrm{T}}$ rad, yielding an initial camera pose given by the transformation matrix ${}^{F}\mathbf{M}_{C}(0) = \begin{bmatrix} 0.624 & 0 & 0.376 & \pi & 0 & \pi/2 \end{bmatrix}^{\mathrm{T}}$ in compact notation.

iii) The desired camera pose with respect to the object is given by the transformation matrix ${}^{C^*}\mathbf{M}_{O} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ in compact notation, i.e., the camera must follow the target trajectory keeping a distance in the ${}^{C}Z$ axis equal to 1 meter.

iv) Control action amplitude u^+ for the SM control: $u^+ = 0.2$ when $\mathbf{J}_s^{\dagger}$ is used and $u^+ = 5$ when $\mathbf{J}_s^{\mathrm{T}}$ is used.

v) A *moving* target object is considered with four markers as described in previous section and with the following transformation matrix ${}^{F}\mathbf{M}_{O}(t) = \begin{bmatrix} {}^{F}x_O & {}^{F}y_O & {}^{F}z_O & \pi & 0 & \pi/2 \end{bmatrix}^{\mathrm{T}}$ in compact notation, where ${}^{F}x_O = 0.514 + 0.1\cos(t) + 0.01t + e^{-t}(0.1 - 0.09\cos(t))$, ${}^{F}y_O = -0.15(\sin(t) + e^{-t}\cos(t) - 1)$ and ${}^{F}z_O = -0.649 + (t + e^{-t}\cos(t))/40$, which basically represents an ellipsoidal movement in the horizontal axes, plus a linear displacement in Z and X axes and plus a transient component (given by the term e^{-t}) in order for the target object to have initial velocity and acceleration equal to zero.

vi) The initial value for the joint velocities and joint accelerations is set to zero, that is $\dot{\mathbf{q}}(0) = \ddot{\mathbf{q}}(0) = \mathbf{0}$. Note that for these initial conditions, as before, the SM control starts on the sliding surface Φ, i.e., (5.15) or (5.20) are satisfied due to $\mathbf{e}(0) = \mathbf{0}$, see $\{{}^{F}\mathbf{M}_{C}(0), {}^{C^*}\mathbf{M}_{O}, {}^{F}\mathbf{M}_{O}(t)\}$, and $\dot{\mathbf{e}}(0) = \ddot{\mathbf{e}}(0) = \mathbf{0}$ since the initial velocity and acceleration for the joints and the target is equal to zero.

5.6.2 Simulation results for the tracking task

Fig. 5.6 shows the object and camera trajectories during the tracking process. For clarity, only the trajectories for SM control using accelerations and $\mathbf{J}_s^{\dagger}$ is depicted. The differences between all the approaches is shown in the following figures.

Fig. 5.7 and Fig. 5.8 show the integral of the control action (i.e., $\dot{\mathbf{q}}$ or $\ddot{\mathbf{q}}$), the tracking error $\mathbf{e}$ and the constraint function vector (i.e., $\boldsymbol{\phi}_a$ or $\boldsymbol{\phi}_j$) for the SM control using joint accelerations and joint jerks, respectively. Note that the tracking errors are always lower than 0.7 millimeters and that, as before, the chattering band is greater when the transpose of the Jacobian matrix is used instead of the pseudoinverse.

Fig. 5.9 compares the SM controls to their respective continuous equivalents in terms of the joint speeds or joint accelerations, i.e., $\dot{\mathbf{q}}$ or $\ddot{\mathbf{q}}$. The differences are similar for the SM control using joint accelerations and joint jerks.

A video of this simulated tracking task for the SM control using joint accelerations and $\mathbf{J}_s^\dagger$ (the other cases are very similar) can be played at `https://media.upv.es/player/?id=0af0f070-11b3-11e7-be40-11c7233e792d`.

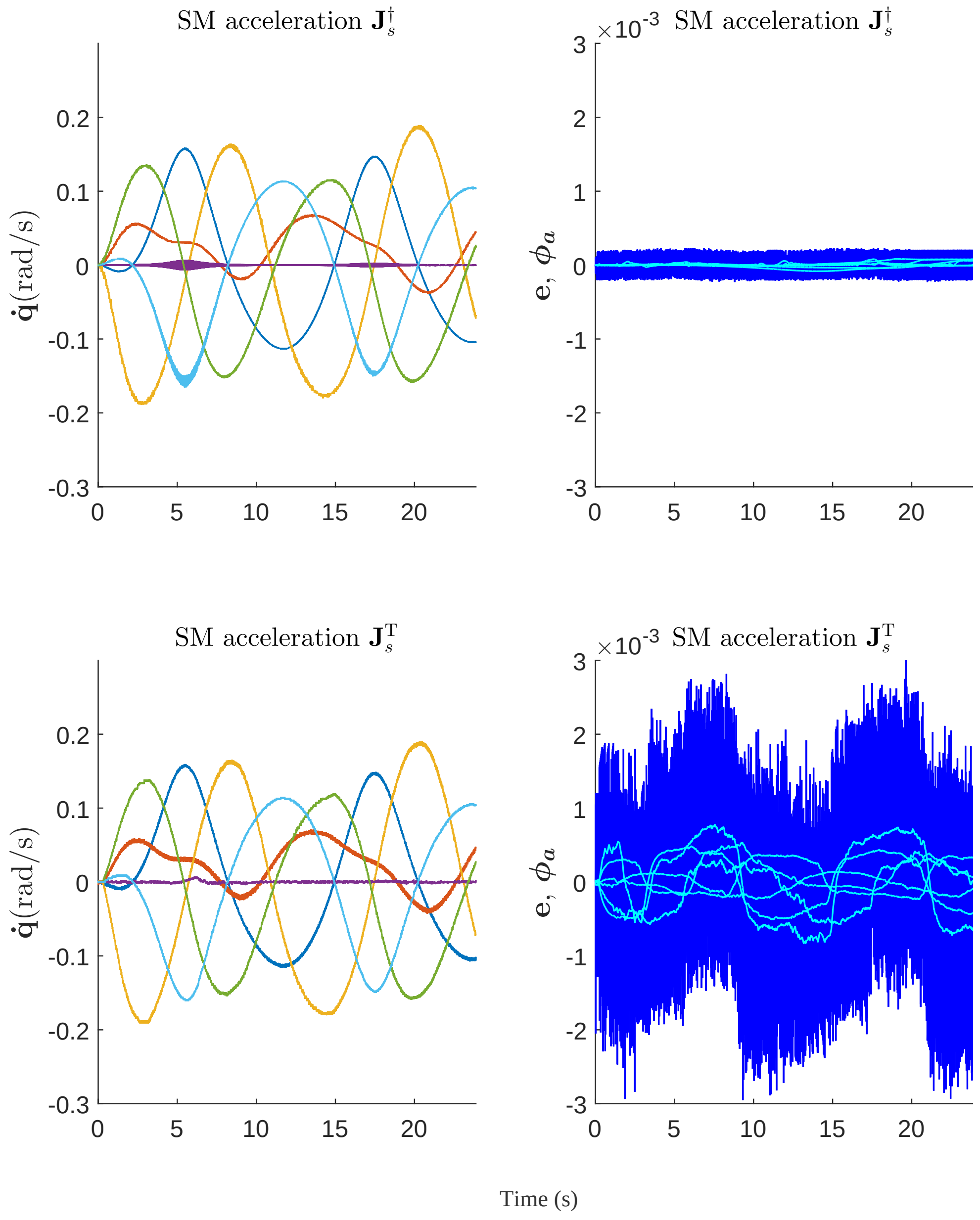


Figure 5.7: Simulated tracking task. SM control using joint accelerations. Integral of the control action, i.e., $\dot{\mathbf{q}}$, on the left-hand side, and error $\mathbf{e}$ and constraint function ϕ_a, on the right-hand side.

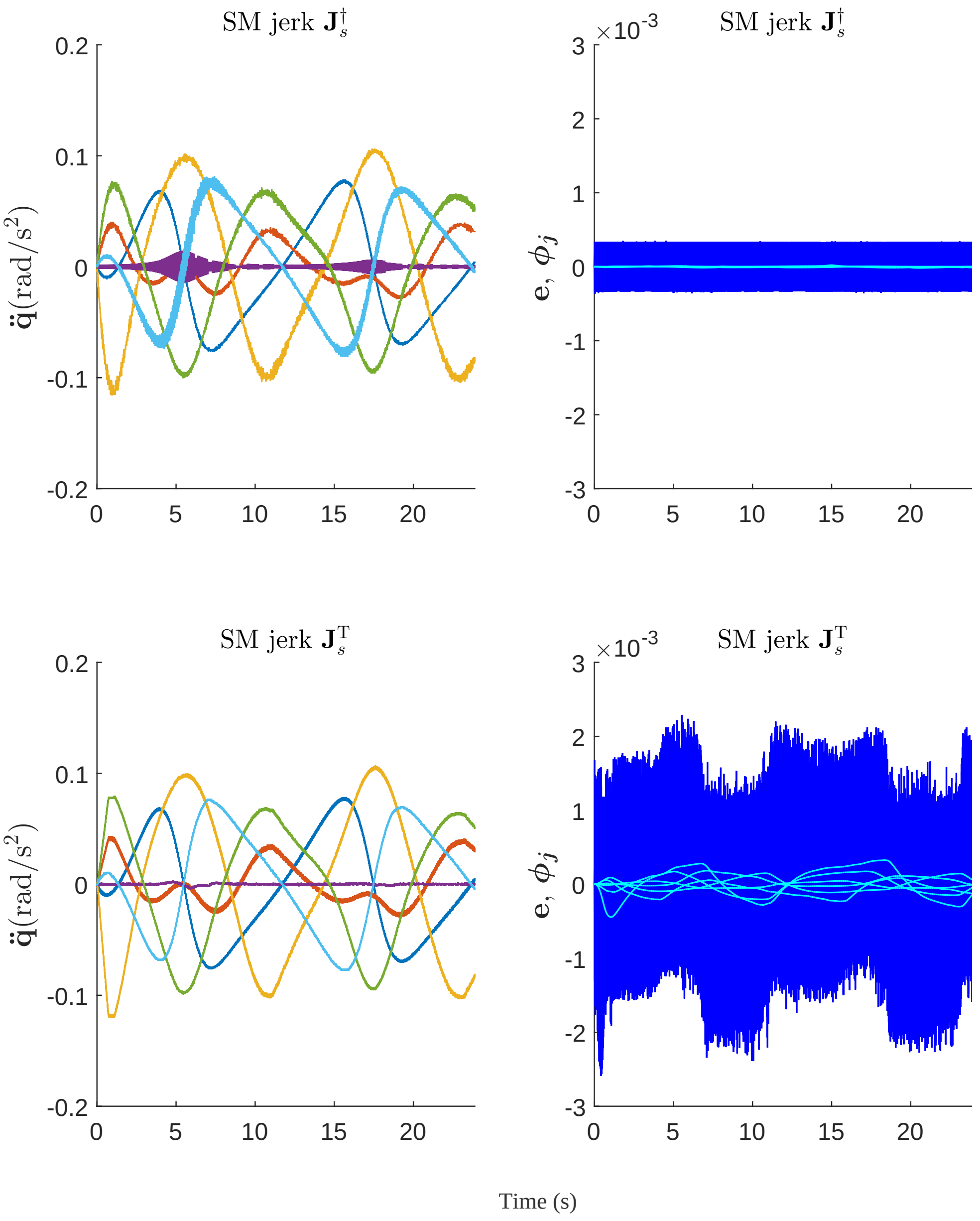
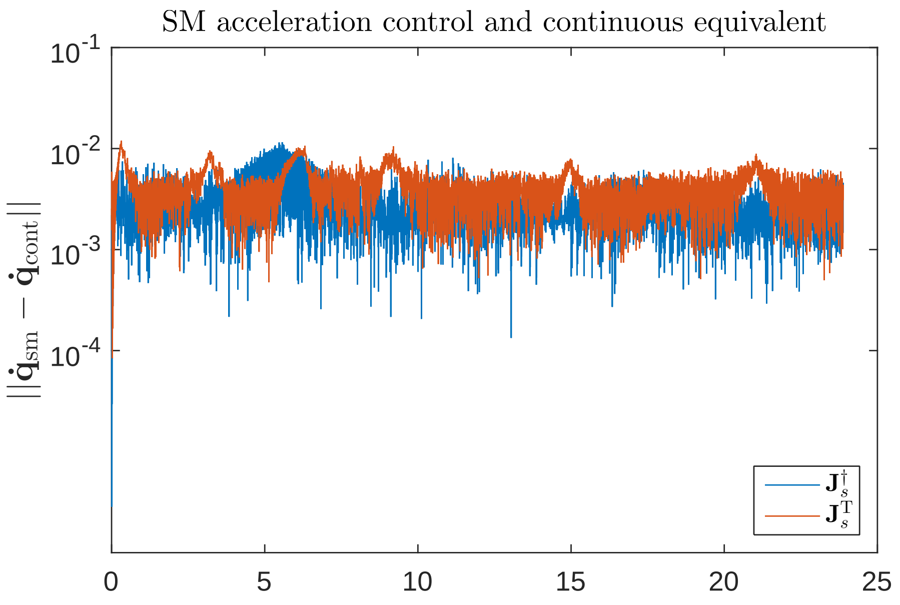


Figure 5.8: Simulated tracking task. SM control using joint jerks. Integral of the control action, i.e., $\ddot{\mathbf{q}}$, on the left-hand side, and error $\mathbf{e}$ and constraint function $\boldsymbol{\phi}_j$, on the right-hand side.



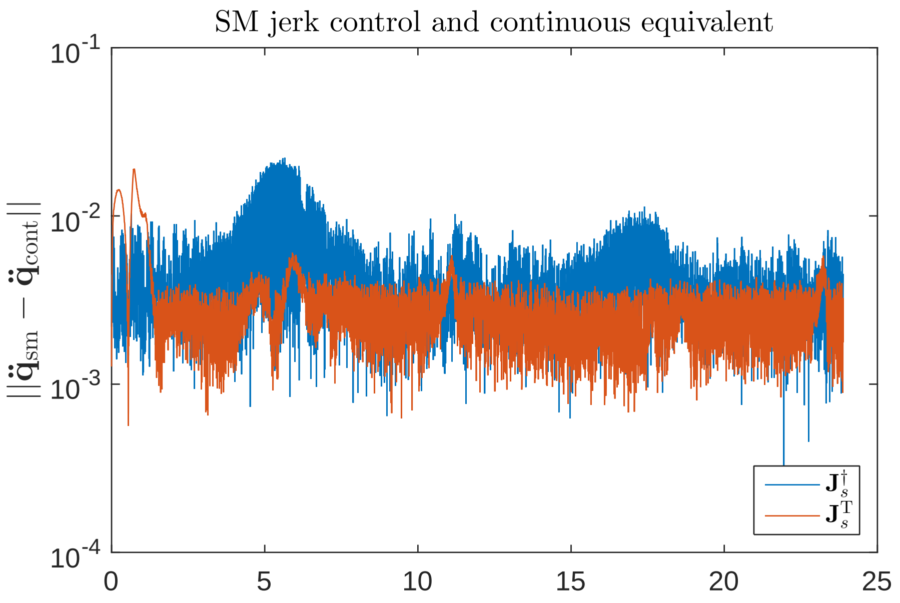


Figure 5.9: Simulated tracking task. Norm of the integral of the control action with respect to the continuous equivalent. SM control using joint accelerations, for $\mathbf{J}_s^{\dagger}$ and $\mathbf{J}_s^{\mathrm{T}}$, and SM control using joint jerks, for $\mathbf{J}_s^{\dagger}$ and $\mathbf{J}_s^{\mathrm{T}}$.

5.6.3 Robustness against errors

Three sources of “modeling errors” are considered to analyze the robustness of the proposed SM approach: low-level controller error, target estimation error and Jacobian matrix error (which includes camera, target, and robot modeling errors). The norm of the tracking error $\mathbf{e}$ is used to compare three different cases: SM control with $\mathbf{J}_s^{\dagger}$, SM control with $\mathbf{J}_s^{\mathrm{T}}$ and the equivalent continuous control. The comparison is carried out for both SM control using joint accelerations and joint jerks. Modeling errors are introduced with a signed variation in percentage of the actual value:

– Low-level controller error:

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}} + c_e \begin{bmatrix} -1 & 1 & -1 & -1 & -1 & 1 \end{bmatrix}^{\mathrm{T}} \circ |\ddot{\mathbf{q}}| \tag{5.34}$$

$$\dddot{\mathbf{q}} = \dddot{\mathbf{q}} + c_e \begin{bmatrix} -1 & 1 & -1 & -1 & -1 & 1 \end{bmatrix}^{\mathrm{T}} \circ |\dddot{\mathbf{q}}| \tag{5.35}$$

– Target motion estimation error:

$$\begin{aligned} \partial\mathbf{s}/\partial t = {} & \partial\mathbf{s}/\partial t \\ & + t_e \begin{bmatrix} -1 & 1 & -1 & -1 & -1 & 1 \end{bmatrix}^{\mathrm{T}} \circ |\partial\mathbf{s}/\partial t| \end{aligned} \tag{5.36}$$

– Jacobian matrix error:

$$\mathbf{J}_s = \mathbf{J}_s + m_e \begin{pmatrix} -1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 & 1 & 1 \end{pmatrix} \circ |\mathbf{J}_s|, \tag{5.37}$$

where $|\cdot|$ represents the absolute value function, symbol $\circ$ denotes the element-wise or Hadamard product, and c_e, t_e and m_e represent the percentage errors considered in each case.

Fig. 5.10 and Fig. 5.11 show the norm of the tracking error for (a) the ideal case, (b) a low-level controller error of $c_e = 20\%$, (c) a target motion estimation error of $t_e = 40\%$, and (d) a Jacobian matrix error of $m_e = 20\%$, when using joint accelerations and joint jerks, respectively. Note that, the equivalent continuous control for each case has very small values for the tracking error in the ideal case, but this value dramatically increases (around three orders of magnitude) in the presence of modeling errors. In contrast, the value of the tracking error for the SM control using $\mathbf{J}_s^\dagger$ approximately remains the same regardless the considered modeling errors. When $\mathbf{J}_s^\mathrm{T}$ is used for the SM control the norm of the tracking error may be larger, since it represents an additional error to the already existing error due to using matrix transpose. However, this increment is significantly lower (up to one order of magnitude) than the one experienced by the continuous equivalent.

Nevertheless, the tracking errors for the SM approaches are due to the chattering band and, therefore, they can be reduced as much as desired by lowering the sampling time. In this sense, Fig. 5.12 shows a case that combines all three errors simultaneously: $c_e = 20\%$, $t_e = 20\%$, and $m_e = 20\%$, and the effect of reducing one order of magnitude the sampling time T_s. In particular, the norm of the tracking error for the SM approaches is reduced around one order of magnitude, whereas that value for the continuous equivalent basically remains the same. This evidences that the tracking errors for the SM controls are a consequence of the chattering band and, hence, these controls are *robust* to modeling errors. In contrast, the tracking errors for the equivalent continuous controllers are not reduced by lowering the sampling time since the control law is not qualitatively robust against modeling errors.

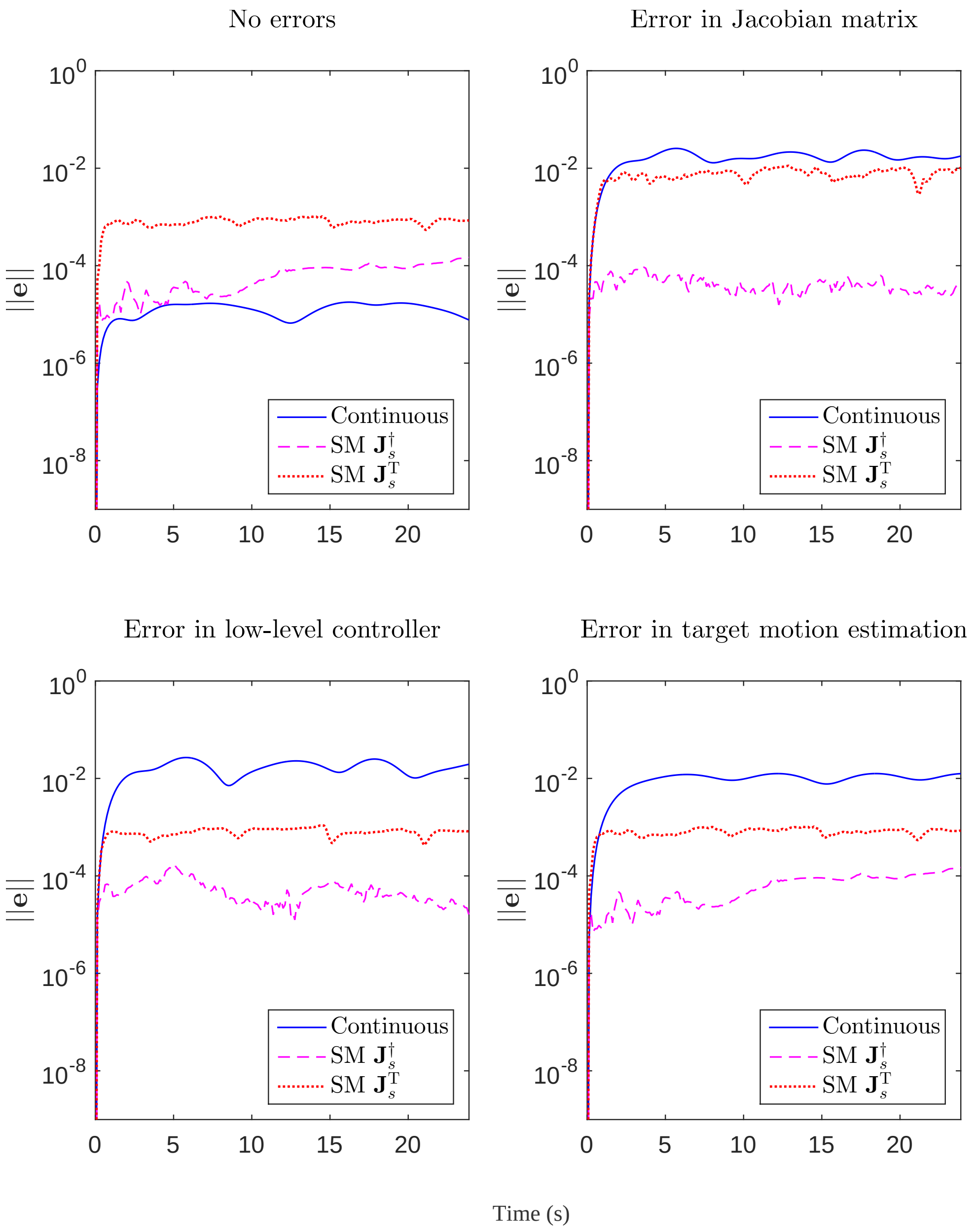


Figure 5.10: Simulated tracking task. Norm of the tracking error for SM control using joint accelerations and continuous equivalent in presence of modeling errors: Continuous (solid, blue), using $\mathbf{J}_s^\dagger$ (dashed, magenta) and using $\mathbf{J}_s^\mathrm{T}$ (dotted, red).

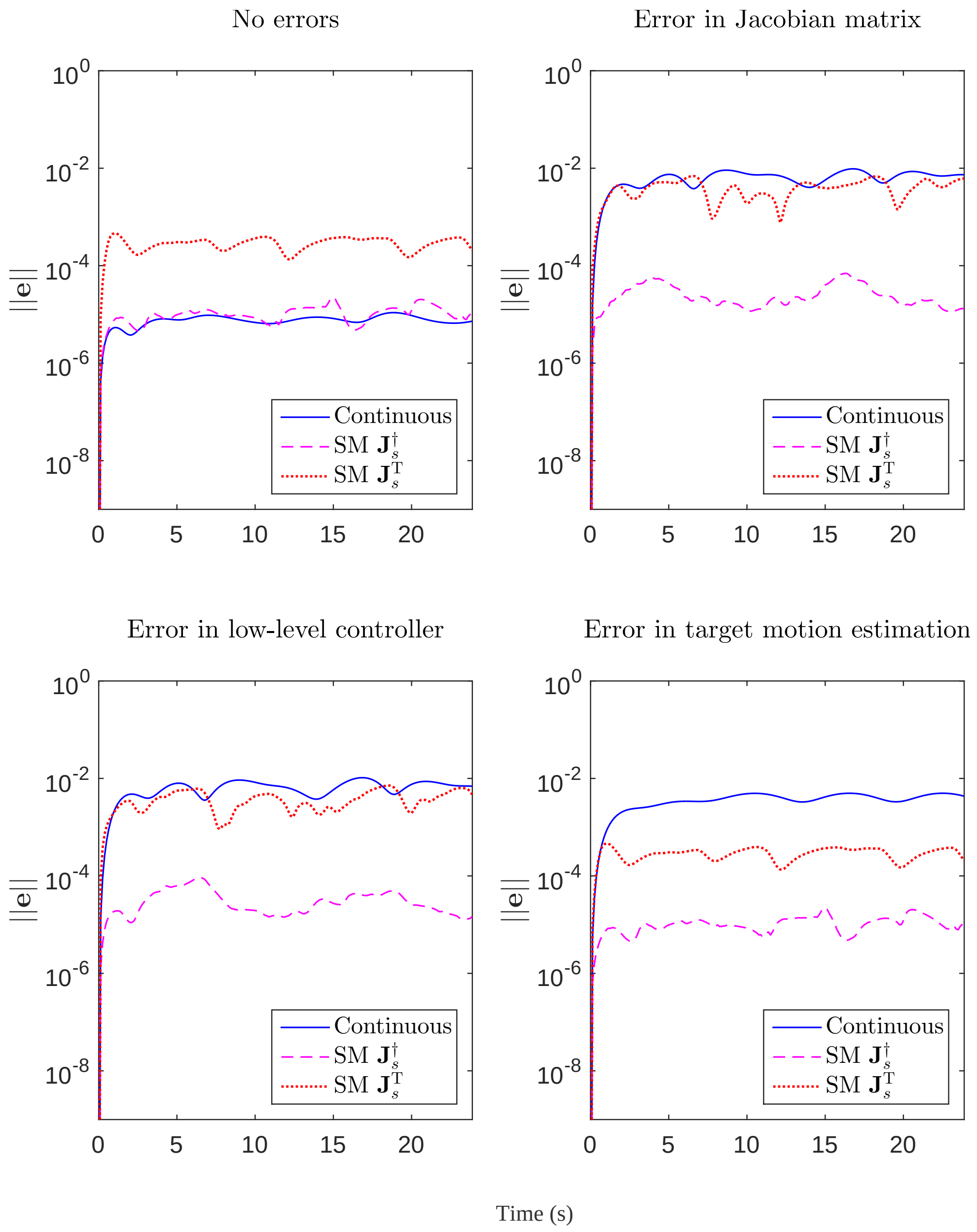
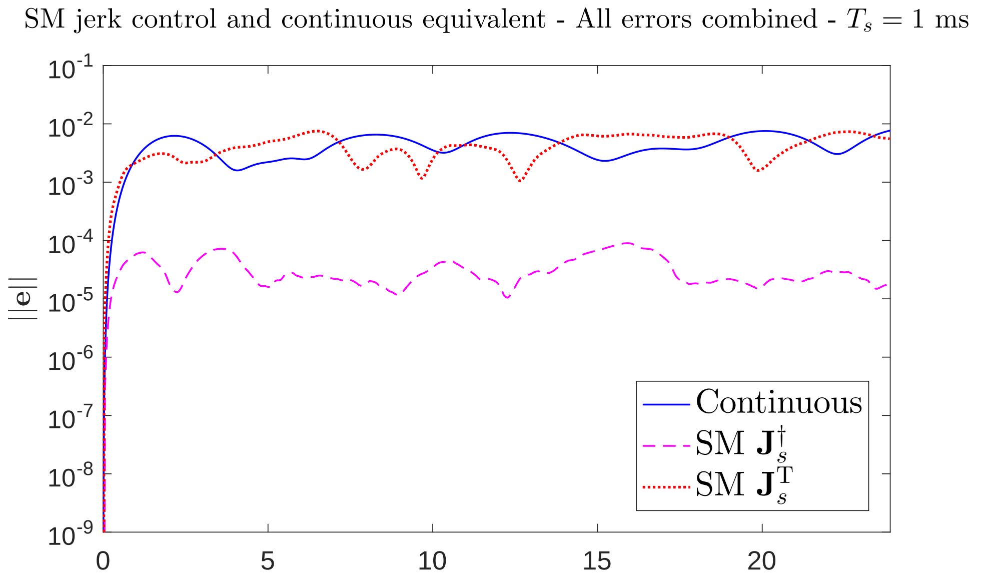


Figure 5.11: Simulated tracking task. Norm of the tracking error for SM control using joint jerks and continuous equivalent in presence of modeling errors: Continuous (solid, blue), using $\mathbf{J}_s^\dagger$ (dashed, magenta) and using $\mathbf{J}_s^\mathrm{T}$ (dotted, red).



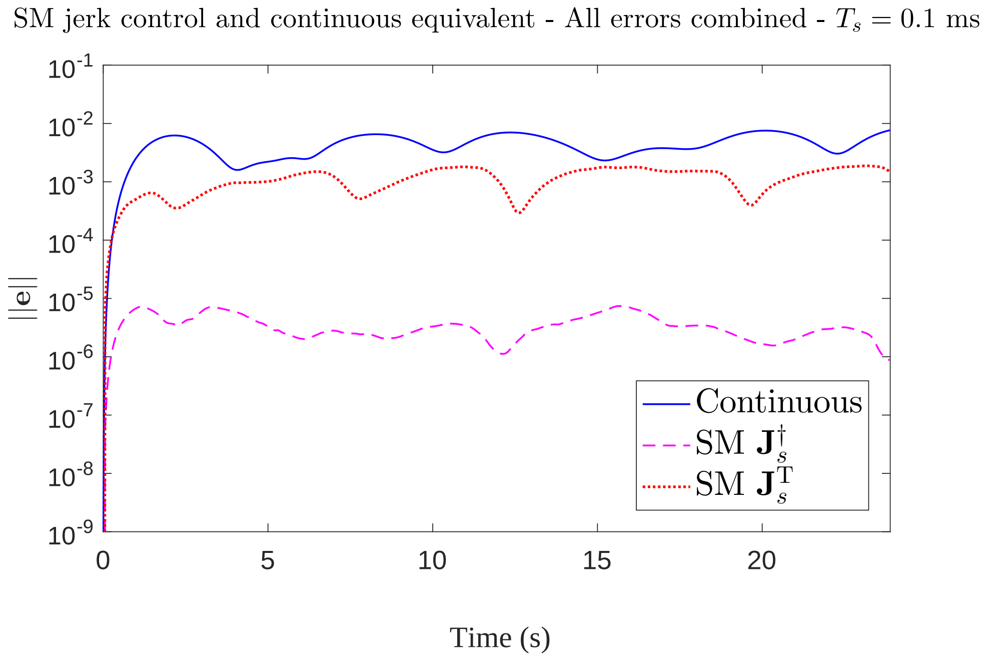


Figure 5.12: Simulated tracking task. Effect of sampling time in the tracking error for SM control using joint jerks and continuous equivalent in presence of modeling errors: Continuous (solid, blue), using $\mathbf{J}_s^\dagger$ (dashed, magenta) and using $\mathbf{J}_s^\mathrm{T}$ (dotted, red).

5.7 Experiments: positioning and tracking tasks

The proposed SM method has been implemented to obtain real experiments in order to demonstrate its feasibility and robustness. The following setup has been used (see Fig. 5.13): a Kuka Agilus KR6 R900 sixx robot manipulator in ceiling-mounted position, is equipped with the Kuka Robot Sensor Interface (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 (eye-in-hand consiguration), 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) (Marchand et al. (2005)), whilst the object pose is estimated to update the visual feature vector $\mathbf{s}$ and to compute $\mathbf{L}_s$.

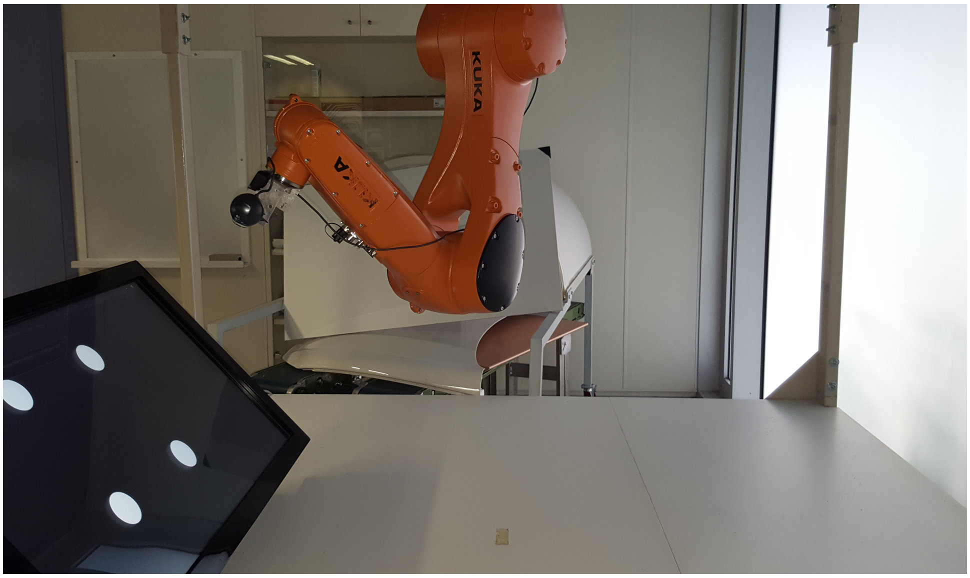


Figure 5.13: Experimental setup: 6R serial industrial manipulator in ceiling position with the camera rigidly attached to the robot end-effector (eye in hand configuration) and a screen to display the object markers.

Three experiments have been conducted to show the validity of the proposed approach: (1) a *positioning task* consisting of moving the robot from the initial to the goal position, defined with respect to a still target object; (2) a *tracking task* consisting of moving the robot to follow a linear trajectory defined by the object motion; (3) a *tracking task* consisting of moving the robot to follow a circular trajectory defined by the object motion.

5.7.1 Experiment conditions and parameter values

Both experiments were run under the following conditions:

i) The proposed SM control using joint accelerations and $\mathbf{J}_s^\dagger$ is used.

ii) Three periodic threads are defined and scheduled following a fixed priority scheme, from highest to lowest: server, control and vision threads. The server period must be set to 4 milliseconds due to robot specification. Both the vision period and the control period T_s are set to 100 milliseconds to guarantee an appropriate scheduling.

iii) The commanded joint accelerations $\ddot{\mathbf{q}}_c$ computed by the proposed algorithm are double integrated (see Section 3.1.3) to obtain the commanded joint positions $\mathbf{q}_c$ sent to the robot controller.

iv) Camera parameters: focal length $f = [710.1, 709.8]$ pixels, resolution $[W_V, H_V] = [640, 480]$ pixels, camera to end-effector transformation matrix ${}^c\mathbf{M}_e = \begin{bmatrix} 0 & 0.07 & -0.05 & 0 & 0 & -\pi/2 \end{bmatrix}^{\mathrm{T}}$ in compact notation.

v) Four markers define the object, representing the vertices of a square with a side length of 17 centimeters in both cases.

vi) Coefficient for the first-order error differential equation $K_a = 10$ and control action amplitude for the SM control $u^+ = 0.1$.

vii) A discrete first-order low-pass IIR filter (see Section 5.3.1.5) has been used with a pole at 0.4 to reduce the noise of the pose estimation signal before computing the time-derivative of the error signal.

viii) In the *positioning task*, the initial configuration is given by the robot joint position vector $\mathbf{q}(0) = \begin{bmatrix} 2.67 & -1.80 & 2.14 & 0.79 & -1.44 & -0.97 \end{bmatrix}^{\mathrm{T}}$ rad, and the visual feature vector $\mathbf{s}(0) =$ $\begin{bmatrix} 0.111 & 0.006 & 0.045 & -0.049 & -0.012 & 0.014 \end{bmatrix}^{\mathrm{T}}$.

ix) In the *tracking tasks*, the initial configuration is given by the same robot joint position vector $\mathbf{q}(0)$, and the reference trajectory for the visual feature vector $\mathbf{s}_{ref}(t) = \mathbf{s}(0) = \begin{bmatrix} 0 & 0 & 0.5 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$, i.e., the object is aligned with the camera optical axis and at a distance of 0.5 meters. In experiment (2), the target object describes a linear trajectory, whose velocity follows a trapezoidal profile with the following parameters: initial velocity equal to zero; nominal velocity equal to 0.02 m/s; and acceleration to achieve the nominal velocity equal to 0.2 m/s^2. In experiment (3), the target object describes a circular trajectory whose radius is equal to 8.5 centimeters and whose angular velocity follows a trapezoidal profile with the following parameters: initial velocity equal to zero; nominal velocity equal to 0.1 rad/s; and acceleration to achieve the nominal velocity equal to 1 rad/s^2.

5.7.2 Experimental results with no errors

For the positioning task, Fig. 5.14 shows the position $\mathbf{e_T}$ and orientation $\mathbf{e_R}$ errors, the commanded joint accelerations $\ddot{\mathbf{q}}_c$ and the constraint function $\boldsymbol{\phi}_a$ obtained using the SM control. Note that, even with noisy signals and a sampling period of 0.1 s, the robot goal position is reached with $\mathbf{e}_{T,i} < 5$ mm and $\mathbf{e}_{R,i} < 0.002$ rad. Fig. 5.15 shows the trajectory of the object markers in the image plane and the 3D camera trajectory, which is very similar to the ideal trajectory, i.e., for PBVS approaches, a straight line from the initial C to the goal C^*.

A video of this positioning experiment can be played at (video at double speed) `https://media.upv.es/player/?id=836301e0-114c-11e7-be40-11c7233e792d`.

For the linear tracking task, Fig. 5.16 shows the joint speeds $\dot{\mathbf{q}}$, the constraint function $\boldsymbol{\phi}_a$ and the tracking error $\mathbf{e}$, for the linear tracking. Note that, the tracking errors are relatively small: below 0.01 m or radians. Fig. 5.17 shows the 3D camera trajectory. The same applies for Fig. 5.18 and Fig. 5.19 for the circular tracking.

Videos of these tracking experiments can be played at (videos at double speed) `https://media.upv.es/player/?id=6785a0a0-2f96-11e7-a50a-535b6ca67416` and `https://media.upv.es/player/?id=98058050-1151-11e7-be40-11c7233e792d`.

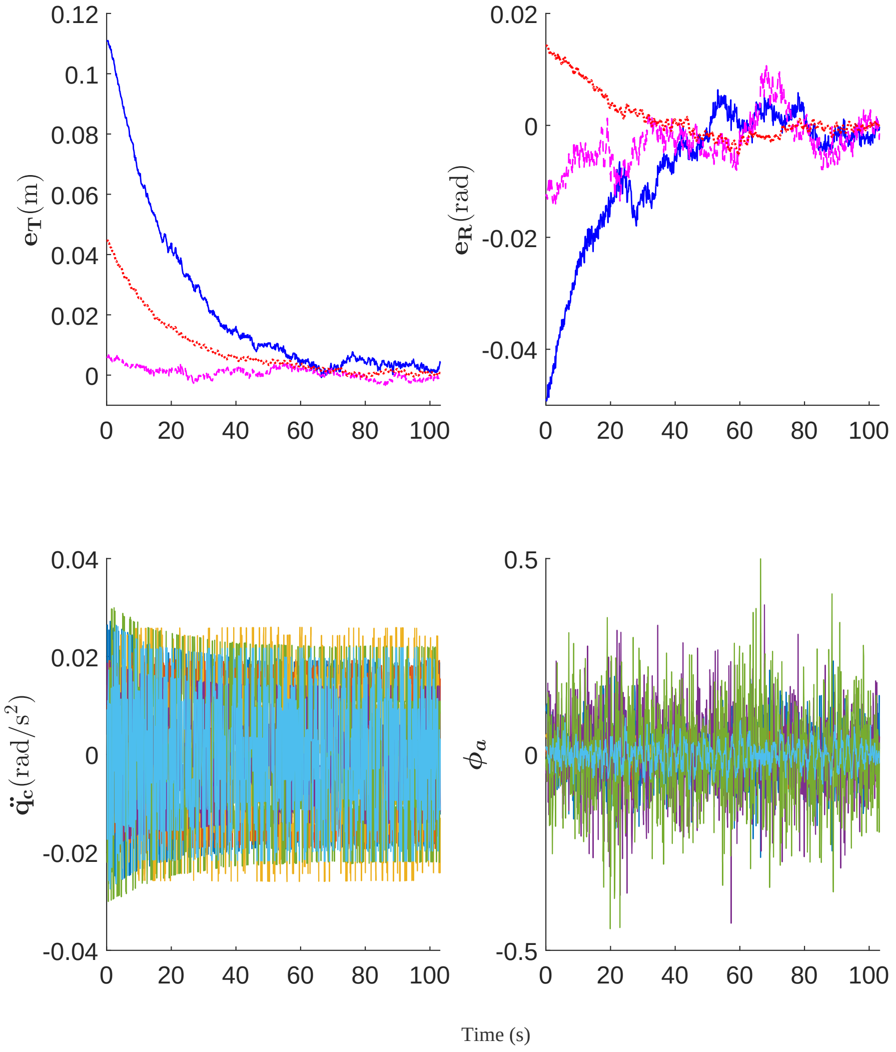
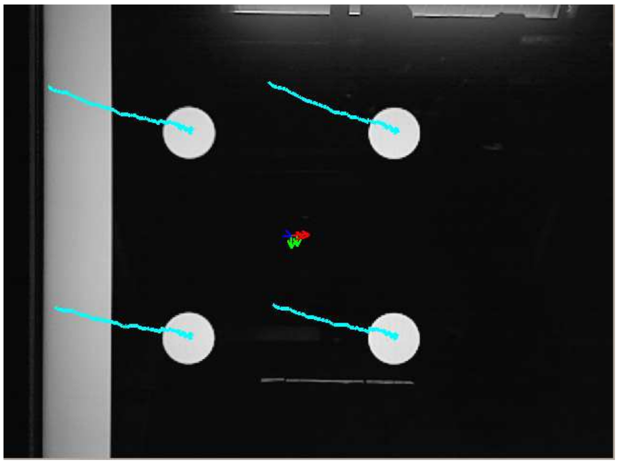


Figure 5.14: Real positioning experiment. SM control using joint accelerations and $\mathbf{J}_s^\dagger$. From top to bottom and left to right plots: (1) translation errors $\mathbf{e_T}$; (2) rotation errors $\mathbf{e_R}$; (3) commanded joint accelerations $\ddot{\mathbf{q}}_c$; (4) Constraint function vector $\boldsymbol{\phi}_a$.



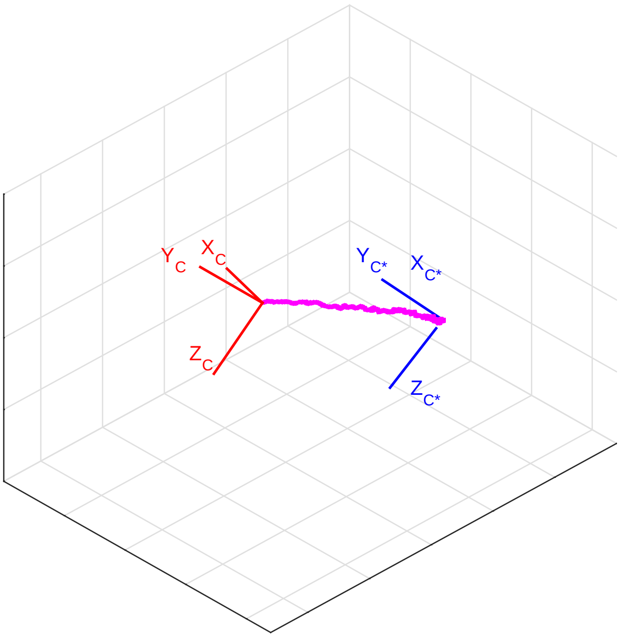


Figure 5.15: Real positioning experiment. (a) Trajectory of the object markers in the image plane; (b) 3D camera trajectory.

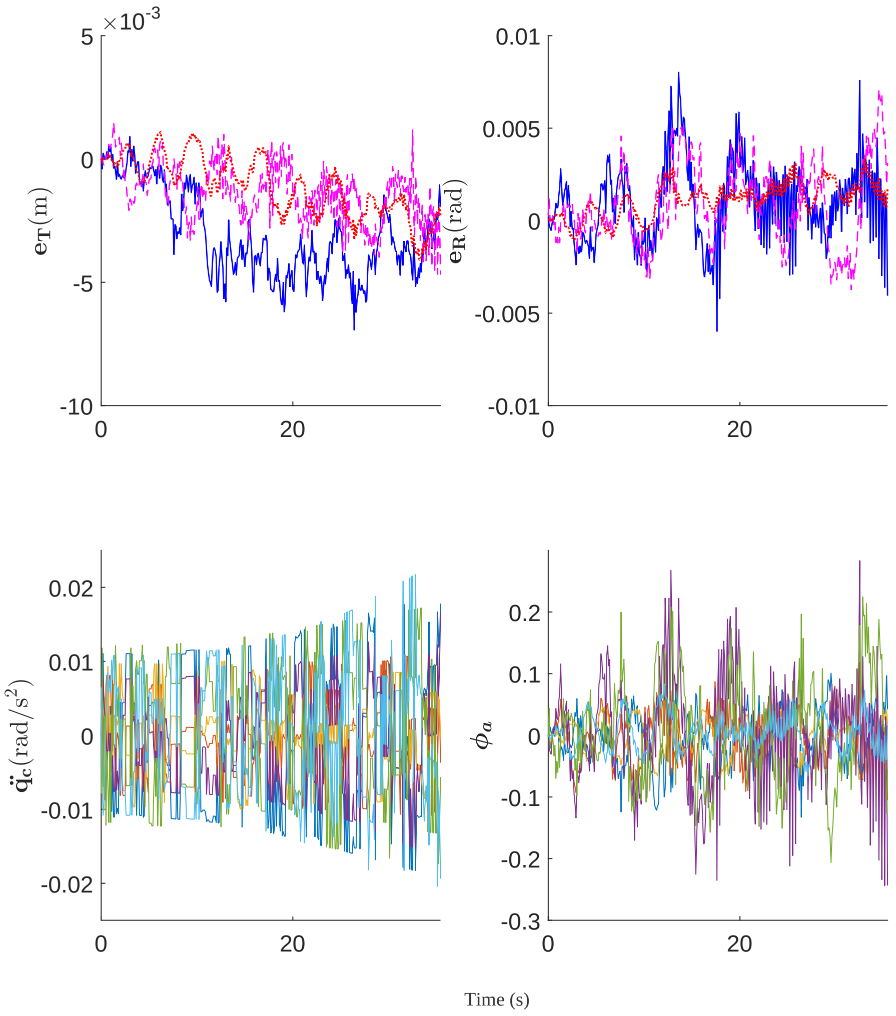


Figure 5.16: Real linear tracking experiment. SM control using joint accelerations and $\mathbf{J}_s^\dagger$. From left to right plots: (1) translation errors $\mathbf{e_T}$; (2) rotation errors $\mathbf{e_R}$; (3) commanded joint accelerations $\ddot{\mathbf{q}}_c$; (4) Constraint function vector $\boldsymbol{\phi}_a$.

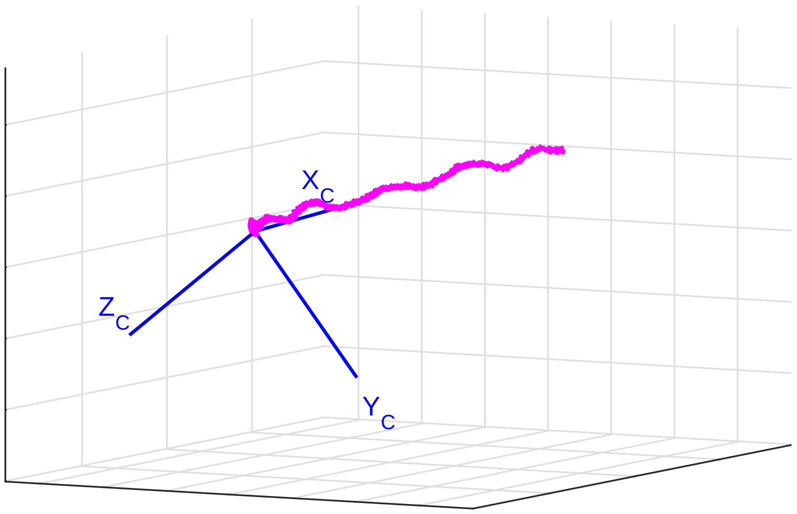


Figure 5.17: Real linear tracking experiment. 3D camera trajectory for a reference trajectory given by an almost closed circle.

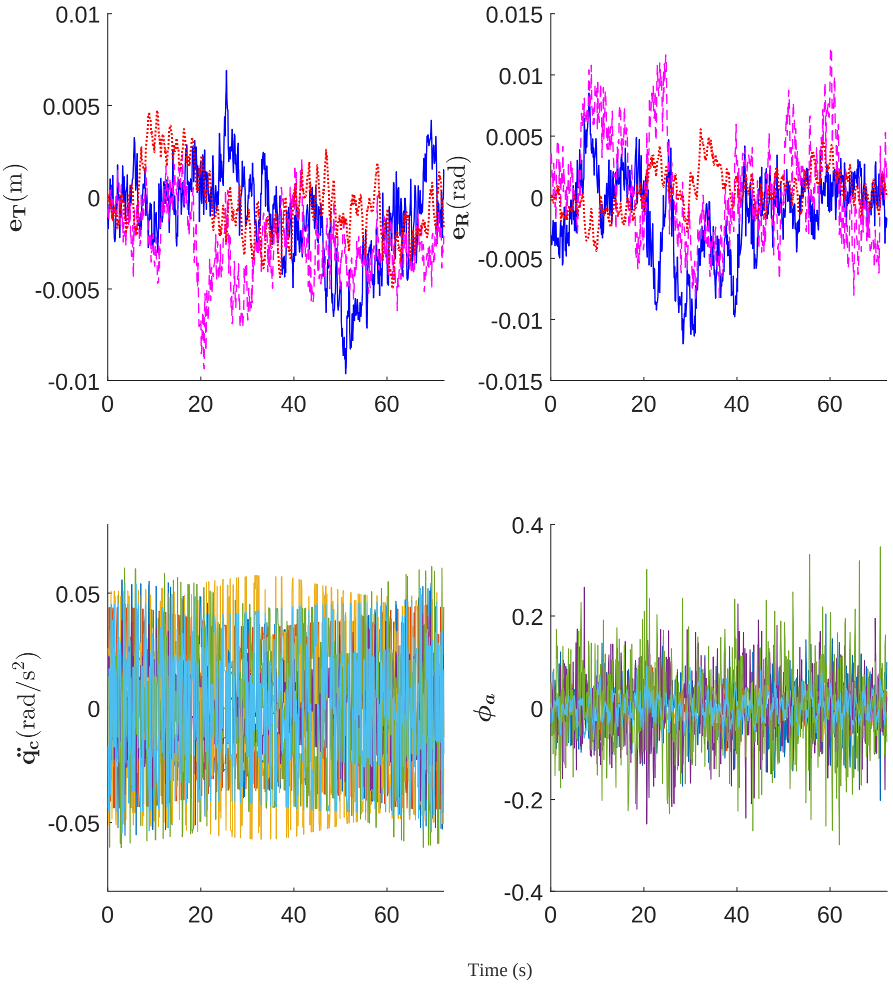


Figure 5.18: Real circular tracking experiment. SM control using joint accelerations and $\mathbf{J}_s^{\dagger}$. From left to right plots: (1) translation errors $\mathbf{e_T}$; (2) rotation errors $\mathbf{e_R}$; (3) commanded joint accelerations $\ddot{\mathbf{q}}_c$; (4) Constraint function vector $\boldsymbol{\phi}_a$.

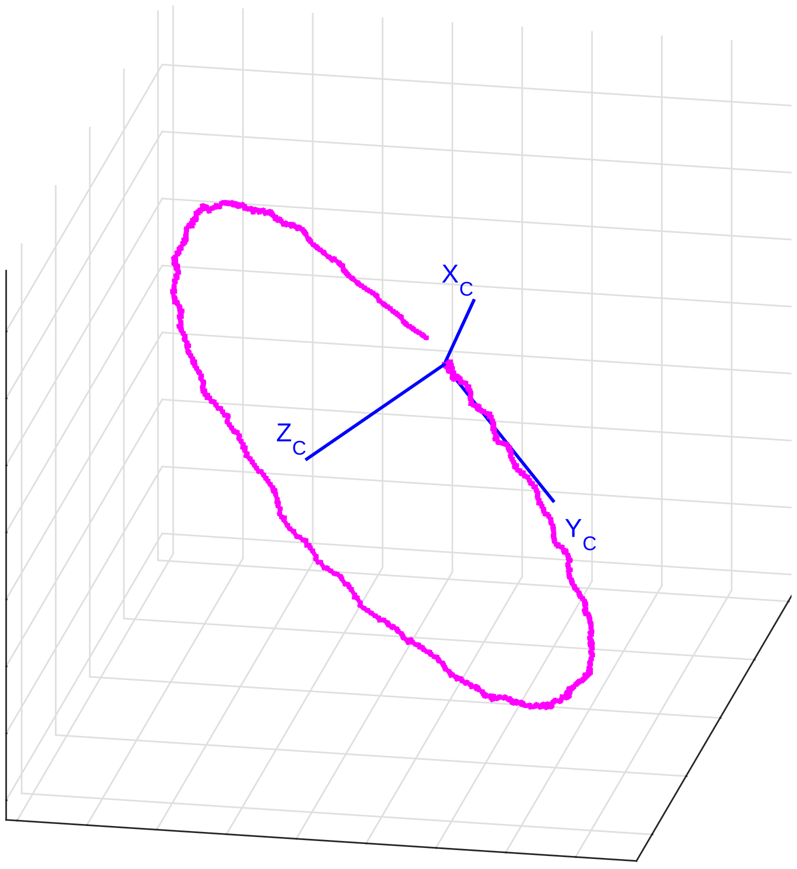


Figure 5.19: Real circular tracking experiment. 3D camera trajectory for a reference trajectory given by an almost closed circle.

5.7.3 Experimental robustness against errors

The robustness of the proposed approach is analyzed by adding a signed error in the Jacobian matrix to the previous circular tracking experiment, following eq. (5.37) with $m_e = 30\%$. Fig. 5.20 shows the tracking error $\mathbf{e}$ for SM control and the continuous equivalent both without and with errors in the Jacobian matrix. It can be seen that the tracking errors for the SM control are not significantly modified when the error in the Jacobian matrix is introduced: they are always below 0.010 m and 0.020 radians. Whereas for the continuous equivalent: position errors are approximately doubled when the error is introduced (they reached values of up to 0.015 m); while orientation errors drastically increased when the error is introduced (they reached values of up to 0.047 radians).

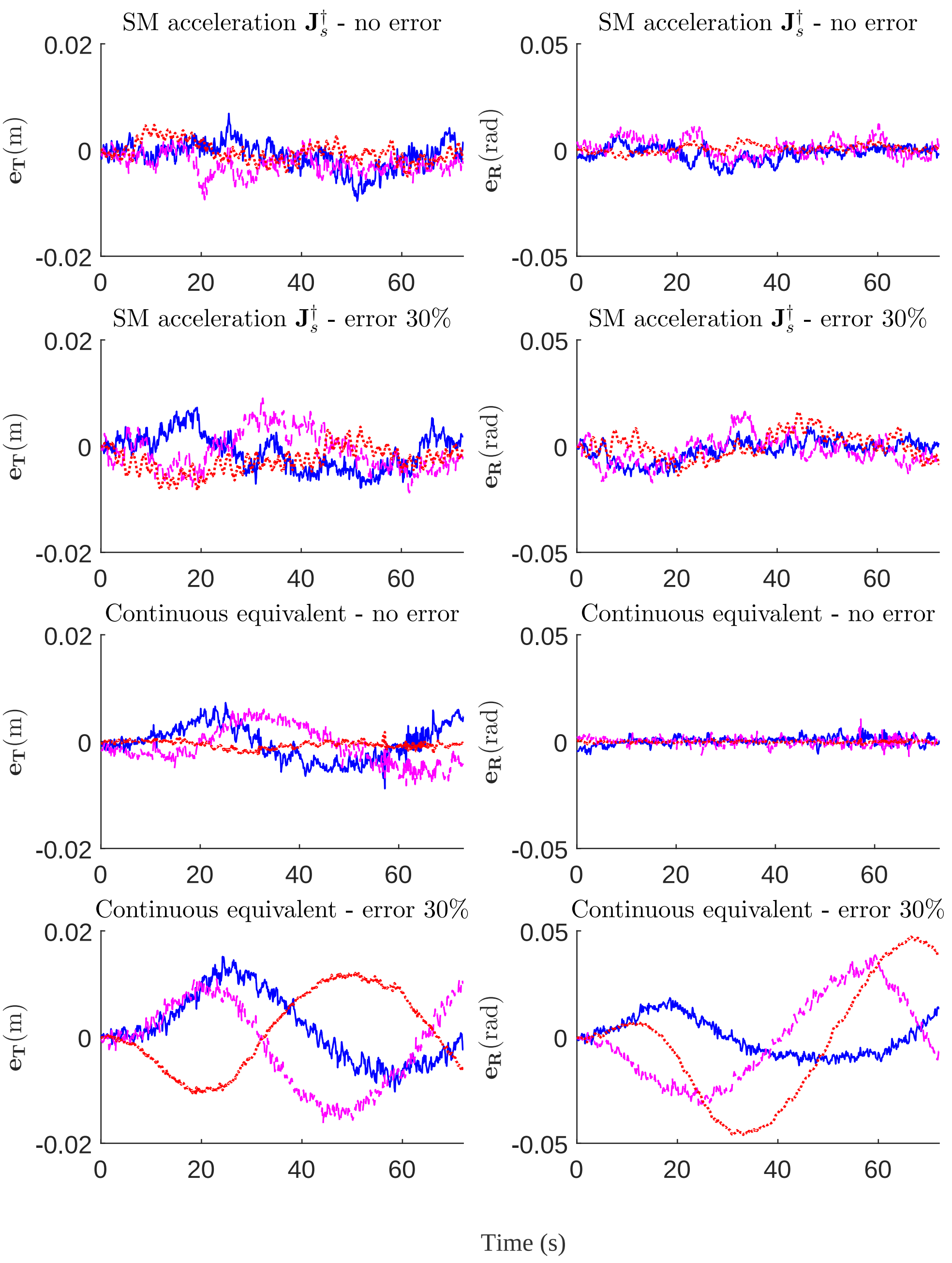


Figure 5.20: Real circular tracking experiment with 30% signed error in the Jacobian Matrix. Comparison of SM control using joint accelerations and the continuous equivalent.

5.8 Conclusions

An approach for reference tracking in VS has been presented using a sliding mode strategy. In particular, two SM controls have been obtained depending on whether the joint accelerations or the joint jerks are considered as the discontinuous control action. Both SM controls have been compared theoretically and in simulation to their equivalent continuous counterparts.

Advantages of the proposed approach:

– Valid both for PBVS and IBVS

In contrast to some SM controllers proposed in literature for VS (Kim et al. (2006); Parsapour et al. (2015); Burger et al. (2015); Zhao et al. (2017); Becerra and Sagüés (2011); Becerra et al. (2011); Parsapour and Taghirad (2015); Xin et al. (2016)), the proposed method is valid either for PVBS and IBVS. In fact, the proposed SM control given by (5.19) or (5.24) is valid either if the visual feature vector $\mathbf{s}$ is defined in PBVS or IBVS domain. Obviously, each case yields a specific Jacobian matrix $\mathbf{J}_s$ to be used in (5.19) and (5.24) .

– Smoothness

In contrast to the SM controllers proposed in literature for VS (Zanne et al. (2000); Kim et al. (2006); Oliveira et al. (2009, 2014); Li and Xie (2010); Parsapour et al. (2015); Burger et al. (2015); Becerra and Sagüés (2011); Becerra et al. (2011); Parsapour and Taghirad (2015); Zhao et al. (2017); Xin et al. (2016)), see Section 5.1, the proposed method yields *continuous joint velocities* given that the SM discontinuous control action are joint accelerations or joint jerks.

– Robustness

Instead of using a SM discontinuous control action to enforce $\dot{\boldsymbol{\phi}}_a = \mathbf{0}$ or $\dot{\boldsymbol{\phi}}_j = \mathbf{0}$, in order to keep the system on the sliding surface, the *analytic computation* of $\ddot{\mathbf{q}}_c$ or $\dddot{\mathbf{q}}_c$, respectively, could obtained solving (5.16) or (5.21), respectively. However, the accurate computation of these continuous control actions requires a *perfect knowledge* of the system model: Jacobian matrix $\mathbf{J}_s$ and its derivatives, partial derivative of the error vector $\partial\mathbf{e}/\partial t$ and its derivatives, joint velocities $\dot{\mathbf{q}}$, inaccuracies $\mathbf{d}_c$ of the low-level control loop, etc. The same applies to the classical continuous control given by (5.27) and (5.30).

For instance, if the disturbance $\mathbf{d}_c$ and/or the partial derivatives $\{\partial\mathbf{e}/\partial t,\partial\dot{\mathbf{e}}/\partial t\}$ given by a moving target are not known a priori, as common in practice, the accurate computation of the mentioned continuous control actions is not possible.

In contrast, the proposed SM approach is *robust* (Edwards and Spurgeon (1998)) against $\mathbf{d}_c$ and $\{\partial\mathbf{e}/\partial t,\partial\dot{\mathbf{e}}/\partial t\}$ since they are collinear with the discontinuous control action, see (5.2) and (5.3). The same applies to the remaining terms collinear with the discontinuous control action: time-derivative of the Jacobian matrix, etc.

Even more, although the pseudoinverse Jacobian matrix $\mathbf{J}_s^\dagger$ used in the proposed SM approach is not collinear with the discontinuous control action, see (5.19) and (5.24), a non-accurate value of this matrix can be used as long as it provides a component perpendicular to the sliding surface given by $\boldsymbol{\phi} = \mathbf{0}$ in order for the SM control action to be able to switch the value of the constraint functions ϕ_i from positive to negative or vice versa. In fact, it has been proven in this work that the transpose of the Jacobian matrix $\mathbf{J}_s^{\mathrm{T}}$ can be used, see (3.5), instead of its pseudoinverse, see (5.9), and the SM algorithm works also fine, only changing the lower bound for the discontinuous action magnitude u^+, see (3.5) and (3.7).

The robustness feature of the proposed SM approach is illustrated in the simulation of Section 5.6.3 and in the experimental results of Section 5.7.3.

– Low computational cost

The proposed SM approach only requires to compute the Jacobian matrix $\mathbf{J}_s$ and the constraint function vector $\boldsymbol{\phi}$. In contrast, the classical continuous control given by (5.27) and (5.30) require to compute: the partial derivative $\partial\mathbf{e}/\partial t$ due to a moving target and its derivatives; the inaccuracies $\mathbf{d}_c$ of the low-level control loop; the time derivative of the Jacobian matrix; etc. Therefore, the computational cost is reduced.

It is interesting to remark that, the partial derivative $\partial\mathbf{e}/\partial t$ due to a moving target is also used as a feedforward term by some SM controllers proposed in literature for VS (Becerra and Sagüés (2011); Becerra et al. (2011); Burger et al. (2015); Kim et al. (2006); Parsapour et al. (2015); Li and Xie (2010); Zanne et al. (2000)), while the proposed SM approach does not require any kind of feedforward, as commented above.

Note also that, in contrast to the classical continuous control, the inversion of the Jacobian matrix is not mandatory for the proposed SM approach, since the transpose of the Jacobian matrix $\mathbf{J}_s^{\mathrm{T}}$ can also be used, as mentioned above. Therefore, the computational cost can be further reduced.

– SM using joint jerks

  If the joint jerks are used for the proposed SM control instead of the joint accelerations, i.e., Eq. (5.24) instead of Eq. (5.19), two main advantages are obtained: the joint velocities are smoother, i.e., they are C^1 instead of C^0; and the error differential equation has one more degree-of-freedom, i.e., there are two poles to be assigned instead of one. However, practical implementations for this case may be affected if the sampling time is not small enough or significant measurement noise is present.

Main limitations of the method:

– Like other SM applications, the proposed method has the *chattering* drawback, see Section 5.3.1.5. Nevertheless, the chattering problem becomes negligible for reasonable fast sampling rates, see (5.25).

The applicability and feasibility of the proposed approach is substantiated by experimental results using a conventional 6R industrial manipulator for positioning and tracking tasks. In particular, the robustness of the method compared to the continuous equivalent has been successfully verified in the experiments by introducing an error in the Jacobian matrix. System stability has been demonstrated with a theoretical proof.

In the next chapter, visual servoing techniques based on Pulse Width Modulation (PWM) and Pulse Frequency Modulation (PFM) for fully decoupled approaches are presented .

Chapter 6

PWM and PFM for visual servoing in fully decoupled approaches

6.1 Introduction

Visual servo controllers have been typically designed to get an exponential decoupled decrease of the error under ideal conditions. Each component of the error vector has an individual convergence speed, and thus an individual convergence time, that is directly related to their respective initial errors. In the absence of constraints, the desired control strategy would be to even all the components out of the error vector to have the same convergence time, avoiding any part of the system to be overstretched. This can be accomplished introducing a diagonal weighting matrix for modifying error. Weighting matrices have been proposed in Hafez and Jawahar (2006a), Hafez and Jawahar (2006b) and Kermorgant and Chaumette (2011) to combine classical PBVS and IBVS approaches, in 5D VS (VS) (Hafez and Jawahar (2007)) for balancing the PBVS and the IBVS schemes and in Comport et al. (2006) to propose a robust control scheme. Similarly to weighting matrices for modifying the error, gain matrices could be used to tune the gain associated to each of the camera velocities, but effects of motion coupling in the interaction matrix arise. That is, each factor in the weighting matrix is associated with one and only one error, but this uniqueness is lost with camera velocities if coupling in the interaction matrix exists, which makes the velocities evolution unpredictable. Researchers in the field have proposed different VS schemes trying to decouple the camera degrees of freedom (DOFs) by selecting adequate visual features. The goal is to find six features, such that each one is related only to one degree of freedom, forcing the interaction matrix to become diagonal. As stated in Chaumette and Hutchinson (2008), the Grail would be to find out a diagonal interaction matrix whose elements are constant, that is as near as possible to the identity matrix, leading to a pure, direct and simple linear control problem. This goal is still pending, but steps have been done in the direction of partially decoupling camera DOFs: partitioned approach for IBVS to isolate motion related to the optic axis (Corke and Hutchinson (2001)); image moments for IBVS (Tahri et al. (2004)); homography (Deguchi (1998), (Chaumette et al. (1997)) and epipolar geometry to decouple rotation from translation in IBVS (Deguchi (1998)); hybrid approaches with block-triangular interaction matrix to decouple the rotational control loop from the translational one (Malis et al. (1999)) or vice versa (Chaumette and Malis (2000)). In this regard, it is noteworthy that all PBVS methods are at least partially decoupled, since the camera rotation is decoupled from camera translation, but furthermore one of the PBVS methods (Martinet (1999)) is fully decoupled, allowing to control independently translational and rotational motions. In this case, weighting matrix and gain matrix are equivalent if they are designed as block-diagonal matrices: one block dedicated to the control of the translation motion and another one to the rotational motion. This work proposes to transfer to the VS domain the core of the well-known Pulse Width Modulation (PWM) and Pulse Frequency Modulation (PFM) techniques: obtaining an average value from signals at high frequency. PWM and PFM have been widely used for improving efficiency in power electronics devices, against noise in data transmission, and to obtain analog outputs from digital control devices. Underlying benefits can be somehow transferred to the VS domain.

The main contents of this chapter are represented using a mind map in Figure 6.1.

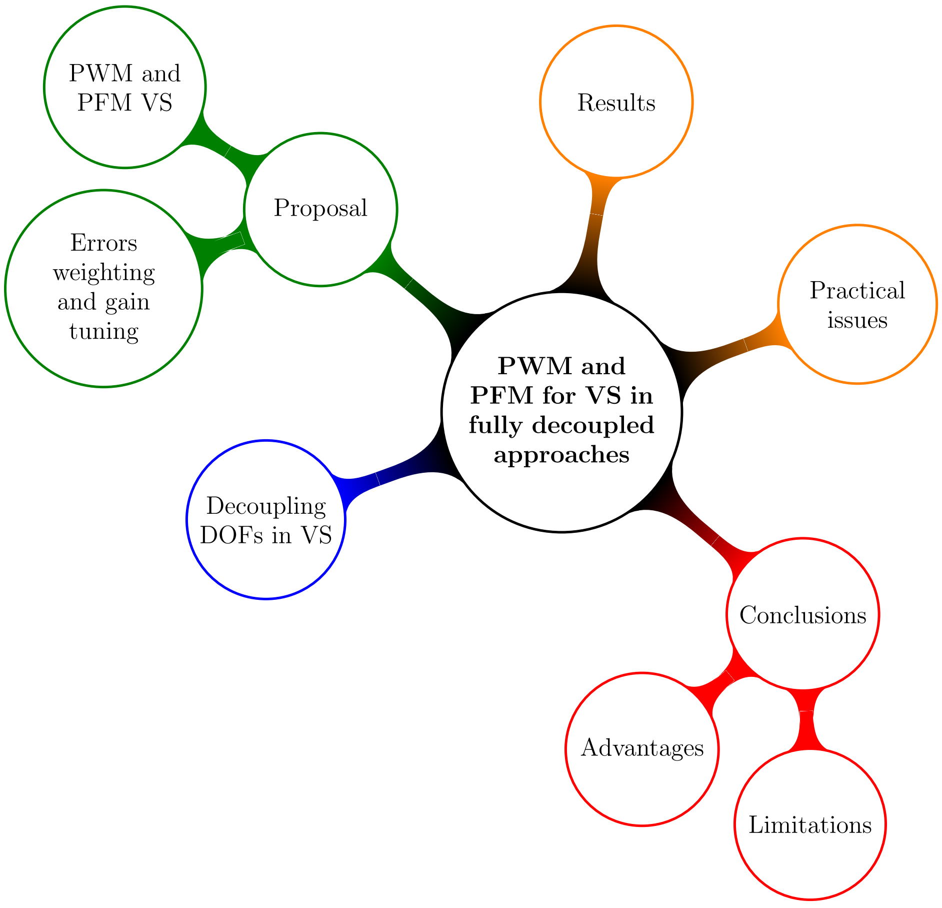


Figure 6.1: Mind map representing the main contents of this chapter.

6.2 Preliminaries: Decoupling DOFs in Visual Servoing Approaches

Some different VS schemes have been proposed in order to decouple DOFs, some of which are shortly described next. The problem is restricted to the classical positioning task of a free-flying camera with six DOFs with respect to a motionless target, which corresponding control law was introduced in Chapter 2:

$$\dot{\mathbf{s}} = \mathbf{L}_s \,\boldsymbol{\tau}, \tag{6.1}$$

where $\mathbf{L}_s$ is the interaction matrix and $\boldsymbol{\tau} = [\mathbf{v}, \boldsymbol{\omega}]$ is the camera kinematic screw.

Approaches with block-triangular interaction matrix of the form:

$$\mathbf{L}_s = \begin{bmatrix} \mathbf{L}_{s1} & 0 \\ \mathbf{L}_{s3} & \mathbf{L}_{s2} \end{bmatrix}, \; \mathbf{L}_s = \begin{bmatrix} \mathbf{L}_{s1} & \mathbf{L}_{s3} \\ 0 & \mathbf{L}_{s2} \end{bmatrix}, \tag{6.2}$$

allow a decoupled behavior of the translation or rotation camera motion respectively (partially decoupled approaches), while approaches with block-diagonal interaction matrix of the form:

$$\mathbf{L}_s = \begin{bmatrix} \mathbf{L}_{s1} & 0 \\ 0 & \mathbf{L}_{s2} \end{bmatrix}, \tag{6.3}$$

induce completely decoupled translational and rotational motions (fully decoupled approaches).

6.2.1 Partitioned approach to IBVS control

This approach is proposed in Corke and Hutchinson (2001) to overcome the problem of undesired camera trajectories in Cartesian space produced in IBVS. Visual features s are defined to isolate motion related to the optical axis.

$$\dot{\mathbf{s}} = \mathbf{L}_s \,\boldsymbol{\tau} = \mathbf{L}_{XY} \,\boldsymbol{\tau}_{XY} + \mathbf{L}_Z \,\boldsymbol{\tau}_Z = \dot{\mathbf{s}}_{XY} + \dot{\mathbf{s}}_Z \tag{6.4}$$

$\dot{\mathbf{s}}_{\mathbf{Z}}$ gives the component of $\mathbf{s}$ due to the camera motion along and rotation about the optical axis, while $\dot{\mathbf{s}} = \mathbf{L}_{XY} \,\boldsymbol{\tau}_{XY}$ gives the component of $\mathbf{s}$ due to velocity along and rotation about the camera X and Y axes.

Control actions associated to the optical axis are defined as:

$$\begin{cases} \mathbf{v}_Z = -\lambda_{\mathbf{v}_Z} \log(\frac{\sigma^*}{\sigma}) \\ \boldsymbol{\omega}_Z = -\lambda_{\boldsymbol{\omega}_Z} \log(\frac{\alpha^*}{\alpha}) \end{cases} \tag{6.5}$$

where σ and α are two new image features defined to determine $\boldsymbol{\tau}_Z$. For X and Y axes, the resulting control action can be seen as a common IBVS control action, but with a modified error to take into account the error induced by $\boldsymbol{\tau}_Z$.

$$\boldsymbol{\tau}_{XY} = \mathbf{L}_{XY}^{\dagger} \left(\lambda \mathbf{e}_{XY} + \mathbf{L}_Z \boldsymbol{\tau}_Z\right) \tag{6.6}$$

where $\mathbf{e}_{XY}$ is the error vector associated to the common visual features in IBVS.

6.2.2 Image moments for partially decoupled IBVS

New visual features (image moments) are presented in Tahri et al. (2004) to decouple DOFs under IBVS. This approach leads to a block-triangular interaction matrix for all camera poses, such that the image plane is parallel to the object:

$$\mathbf{L}_s^{||} = \begin{bmatrix} \mathbf{L}_{s1}^{||} & \mathbf{L}_{s3}^{||} \\ 0 & \mathbf{L}_{s2}^{||} \end{bmatrix} \tag{6.7}$$

Then, a generalization based on a virtual camera rotation is proposed to extend the decoupling properties for any desired camera orientation with respect to the considered object.

6.2.3 Homography and epipolar geometry to decouple rotation from translation in IBVS

Homography is used in Chaumette et al. (1997) for planar objects to decouple rotation from translation. The proposed control law results in the following block-triangular interaction matrix:

$$\tau = -\lambda \begin{bmatrix} d^* \, \mathbf{M}_v^{-1} & -d^* \, \mathbf{M}_v^{-1} \, \mathbf{M}_\omega \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} s - s^* \\ r - r^* \\ {}^{c^*}\boldsymbol{\theta}\mathbf{u}_c \end{bmatrix} \tag{6.8}$$

with $r = d/d^*$ being the ratio between the current distance d and desired distance d^* to the object and ${}^{c^*}\boldsymbol{\theta}\mathbf{u}_c$ the rotation matrix in angle-axis representation that gives the orientation of the current camera frame relative to the desired frame. The same approach is used in Deguchi (1998), but with a different control action: they generate a straight optimal trajectory by constraining the translation direction using the homography matrix. They assign camera translation to take the shortest path to the goal and camera rotation to keep the object in the field of view, and control them separately. Another algorithm proposed in Deguchi (1998) applies for general 3D object and uses the epipolar condition held between the goal image and the current image to generate the optimal trajectory of the robot motion to reach the goal straightforwardly.

6.2.4 Hybrid visual servoing

In Malis et al. (1999) the coordinates of the central point in the image plane and the logarithm of its depth in the camera frame are used as features related to camera translation $\mathbf{s}_t = [x, y, \log(Z)]$, $\mathbf{s}_t^* = [x^*, y^*, \log(Z^*)]$, $\mathbf{e}_t = [x - x^*, y - y^*, \log(Z/Z^*)]$. ${}^{c^*}\boldsymbol{\theta}\mathbf{u}_c$ is used as feature related to rotation.

The resulting interaction matrix is block-triangular, allowing the decoupling of rotational from translational camera motions:

$$\dot{\mathbf{s}} = \begin{bmatrix} \mathbf{L}_v & \mathbf{L}_\omega \\ \mathbf{0} & \mathbf{L}_{\theta u} \end{bmatrix} \tag{6.9}$$

The control law takes the form:

$$\begin{cases} \mathbf{v}_c = -\mathbf{L}_v^{\dagger}(\lambda \, \mathbf{e}_t + \mathbf{L}_\omega \, \boldsymbol{\omega}_c) \\ \boldsymbol{\omega}_c = -\lambda \, {}^{c^*}\boldsymbol{\theta}\mathbf{u}_c \end{cases} \tag{6.10}$$

A similar approach is presented in Chaumette and Malis (2000), but in this case to decouple translation from rotation. Visual features vector is defined with the translational vector of the desired camera frame with respect to the current camera frame ${}^{c}\mathbf{t}_{c^*}$, the coordinates of the central point in the image plane (x, y), and the third component of the rotation matrix in axis-angle representation $\boldsymbol{\theta}\mathbf{u}_z$. Thus, $\mathbf{s} = [{}^{c}\mathbf{t}_{c^*}, x, y, \boldsymbol{\theta}\mathbf{u}_z]$, $\mathbf{s}^* = [0, x^*, y^*, 0]$, and $\mathbf{e} = [{}^{c}\mathbf{t}_{c^*}, x - x^*, y - y^*, \boldsymbol{\theta}\mathbf{u}_z]$. The resulting interaction matrix is block-triangular:

$$\dot{\mathbf{s}} = \begin{bmatrix} {}^{c^*}\mathbf{R}_c & \mathbf{0} \\ \mathbf{L}_v^{\mathrm{T}} & \mathbf{L}_\omega^{\mathrm{T}} \end{bmatrix} \tag{6.11}$$

with ${}^{c^*}\mathbf{R}_c$ being the rotation matrix that gives the orientation of the current camera frame relative to the desired frame.

6.2.5 Position Based Visual Servoing

Camera rotational motion control is decoupled from the translational one by definition in all common implementations of Position Based Visual Servoing (PBVS). The resulting interaction matrix is either block-triangular or block-diagonal. We focus here on the fully decoupled PBVS approach (Martinet (1999)), in which current visual features are set as $\mathbf{s} = [{}^{c^*}\mathbf{t}_c, {}^{c^*}\boldsymbol{\theta}\mathbf{u}_c]$, desired visual features as $\mathbf{s}^* = 0$ and error as $\mathbf{e} = [{}^{c^*}\mathbf{t}_c, {}^{c^*}\boldsymbol{\theta}\mathbf{u}_c]$. Thus, the relationship between the camera motion and the features is:

$$\dot{\mathbf{s}} = \mathbf{L}_v\, \boldsymbol{\tau} = \begin{bmatrix} {}^{c^*}\mathbf{R}_c & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta u} \end{bmatrix} \tag{6.12}$$

The expected behavior is described in Chaumette and Hutchinson (2006): In this case the corresponding desired feature is a null vector, and the interaction matrix is known to be block-diagonal, inducing decoupled translational and rotational motions. Moreover, each of the rotation DOFs could be controlled independently since they are directly related to one of the rotation camera motions. The corresponding camera trajectory is a straight 3D line. However, no control at all is done in the image and the visual features used for the pose estimation may be lost. This image boundary constraint is not considered in the present work.

6.3 Proposed approach

In this section, a procedure to adjust the convergence time of all the components of the error vector in VS schemes is presented. The goal of this approach is to avoid overstretching on any part of the VS system during the control motion task and for that, a modification of the error vector by adding a weighting matrix in the control law is proposed. Afterwards, the same procedure is applied to tune the control gains in fully decoupled VS approaches. The resulting gains can be seen as two different levels of a continuous signal, what led us to explore the possibility of applying signal modulation techniques. It turned finally on what we have coined PWM and PFM visual servoing, which allows controlling independently translational and rotational camera motions with signals at high frequency.

6.3.1 Errors weighting and gain tuning

VS schemes use normally the same scalar gain λ for all the components of the error vector $\mathbf{e}$. In that case, each component has its own convergence speed and thus, its own settling time that differs from each other according to their respective initial errors. If the error vector has dimension m, a diagonal weighting matrix $\mathbf{W}$ with dimension m x m can be inserted in the control law in order to tune the convergence velocity of each component:

$$\boldsymbol{\tau} = -\lambda\, \widehat{\mathbf{L}}_s^{\dagger}\, \mathbf{W}\, \mathbf{e} \tag{6.13}$$

The goal is to achieve an equal settling time for all the components of the error vector, which determines the convergence time of the entire process. Since the control is implemented in discrete domain, the j component of the error vector at iteration i, e_j^i, is expressed as:

$$e_j^i = e_j^0\,(1 - \lambda w_j T_s)^i \tag{6.14}$$

where e_j^0 is the j component of the initial error vector, λ the control gain, w_j the weight for the j component and T_s the sampling time.

The number of iterations needed for convergence N is,

$$N = \frac{\log(e_j^N / e_j^0)}{\log(1 - \lambda\, w_j\, T_s)} \tag{6.15}$$

where e_j^N is the desired final error threshold.

The weight w_j needed to have a given convergence time (i.e. a given number of iterations) is:

$$w_j = \frac{1 - (e_j^N / e_j^0)^{1/N}}{\lambda \, T_s} \tag{6.16}$$

Once the problem is stated, i.e. the coordinates of the initial and desired features are known, the weights w_j for each component of the error vector e_j, are computed as follows: the weight for the highest initial error is set to a maximum fixed level, the number of iterations to convergence N is calculated with equation (6.15), and then the remaining weights are computed with (6.16). Note that this is a generic procedure, valid independently of the dimension of the error vector. Let us focus now on fully decoupled VS approaches, such as the fully decoupled PBVS presented in section 6.2.5. In this case translational and rotational camera motions can be controlled independently, and their respective convergence times can be adjusted with scalar coefficients, w_v and w_ω, of a translation-rotation weighting matrix $\mathbf{W}$. Due to the fact that the interaction matrix is block-diagonal, the translation-rotation weighting matrix $\mathbf{W}$ can also be expressed as a translation-rotation control gain matrix $\mathbf{K}$, positioned on the left of the interaction matrix in the control expression, equation (6.17). The tuning of the scalar coefficients k_v and k_ω is then equivalent to the errors weighting.

$$\boldsymbol{\tau} = -\lambda \, \mathbf{K} \widehat{\mathbf{L}}_s^{\dagger} \mathbf{e} = -\lambda \, \widehat{\mathbf{L}}_s^{\dagger} \mathbf{W} \mathbf{e} \tag{6.17}$$

$$\mathbf{K} = \begin{bmatrix} k_v \, \mathbf{I_{3x3}} & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & k_\omega \, \mathbf{I_{3x3}} \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_v \, \mathbf{I_{3x3}} & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & w_\omega \, \mathbf{I_{3x3}} \end{bmatrix} \tag{6.18}$$

with $k_v = w_V$ and $k_\omega = w_\omega$.

6.3.2 PWM and PFM visual servoing

Well-known PWM and PFM techniques can be also applied to the VS control of many systems. In Fig. 6.2 the parameters of a modulated signal are shown, A being the pulse amplitude, τ_0 the pulse width, and τ_c pulse period (or $f_c = 1/\tau_c$ pulse frequency). If the signal is supplied as input to a device or

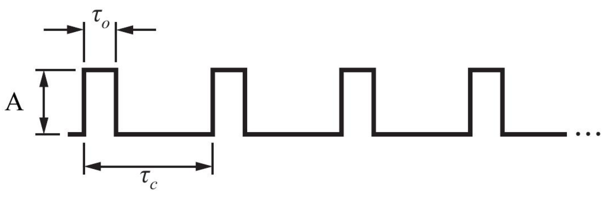


Figure 6.2: Signal modulation.

system, which has a response time much larger than τ_c , it experiences the signal as an average value:

$$V_{av} = A\frac{\tau_0}{\tau_c} \tag{6.19}$$

The ratio $d = \tau_0/\tau_c$ is called the duty cycle of the square wave pulses. The average value is controlled by adjusting the duty cycle. If amplitude A and pulse period τ_c are constant, the average value can be modified with the pulse width τ_0 (PWM). Otherwise, if amplitude A and pulse width τ_0 are constant, the average value can be modified with the pulse period τ_c or pulse frequency f_c (PFM). The existence of fully decoupled VS approaches, with independent control of translational and rotational camera motions, allows us the possibility of using PWM and PFM signals for each of the independent motions. The same procedure used for errors weighting and gain tuning presented in Section6.3.1 can be applied to tune the duty cycles. Equivalence between gains and PWM-PFM duty cycles is straightforward: $k_j = d_j$. Fig. 6.3 shows the equivalence between continuous levels of control gains and the modulated signals.

6.4 Results

In Gans et al. (2003) a comparison of the performance of different VS techniques based on quantitative metrics is described. They are meaningful to compare different approaches, but what we present here are actually different implementations of the same approach (PBVS), so no one of that metrics is considered. They have also categorized work conditions that visual servo systems often experience difficulty to handle. Given that, we focus on task 1 and

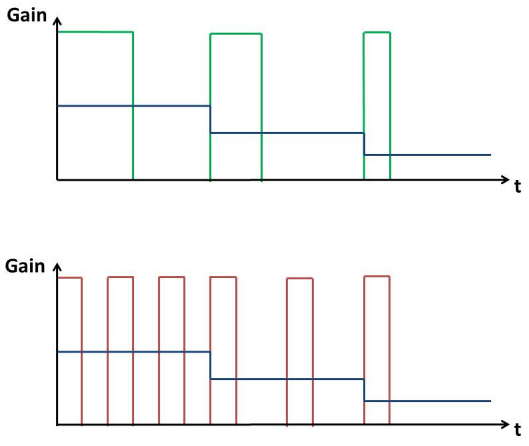


Figure 6.3: Gain modulation in PWM and PFM visual servoing.

task 4 from (Gans et al. (2003)), as we are interested in having a task involving all the 6 degrees of a free camera. Task 1 proposes to modify the initial pose through rotations in the camera Z-axis (φ_Z) between 30° and 210°. Task 4 defines axes (X_F, Y_F) lying in the feature points plane and perpendicular to the optical axis Z, and proposes initial poses resulting from rotations about X_F axis (φ_{X_F}) and Y_F axis (φ_{Y_F}) between 10° and 80°. Rotations about X_F axis imply camera translations in X and Z axes and camera rotation about Y axis, while rotations about Y_F axis imply camera translations in Y and Z axes and camera rotation about X axis. A combination of the former two tasks is proposed: rotation of the features plane, about X_F and Y_F axes, together with a rotation around the camera optical axis Z, involving simultaneously the six DOFs of the camera. The task is evaluated under the fully decoupled PBVS approach described in section 6.2.5. Euclidean norm of the initial translation and rotation errors is used to compute the desired gains and PWM-PFM duty cycles according to the procedure explained in section 6.3:

$$\|\mathbf{e}_0\|_2 = \left(\sqrt{{}^{c*}\mathbf{t}_{c,x}^2 +^{c*} \mathbf{t}_{c,y}^2 +^{c*} \mathbf{t}_{c,x}^z}\ ,\ \sqrt{{}^{c*}\boldsymbol{\theta u}_{c,x}^2 +^{c*} \boldsymbol{\theta u}_{c,y}^2 +^{c*} \boldsymbol{\theta u}_{c,z}^2}\right)^{\mathrm{T}} \tag{6.20}$$

To characterize the behavior of the proposed PWM and PFM implementations, in particular to quantify the expected appearance of ripple, new error indexes are needed. The ripple can be observed in these cases: 1) in the trajectory of the visual features in the normalized image plane (2D); 2) in the time variation of the errors (3D). However, for simplicity only the first case is considered to propose the following error indexes: 1) summation of Euclidean distances, measured in the normalized image plane, between the centers of the visual features in the different approaches with respect to the number of iterations and 2) maximum value among the Euclidean distances. That is:

$$\varepsilon = \frac{\sum_{i=1}^{N} \|\mathbf{p}_i - \mathbf{p}_{i,gain}\|_2}{N} \tag{6.21}$$

$$\delta = \max_i \{\|\mathbf{p}_i - \mathbf{p}_{i,gain}\|_2\}_1^N \tag{6.22}$$

p_i being the coordinates of the center of the visual features under PWM and PFM implementations, N the number of iterations, and $p_{i,gain}$ the coordinates of the center of the visual features in the gain tuning case, namely the non-rippled continuous case.

The experimentation is carried out in simulation under a common VS scenario: positioning task of a camera with six DOFs with respect to a motionless target, object pose estimation updated with a period $\overline{T} = 40$ ms and the sampling period $T_s = 2$ ms (fast enough to avoid discretization errors). The resolution of the duty cycles (d_v, d_ω), and hence the available d_v/d_ω and λ_v/λ_ω relations are determined by T_s:

– i) Under the PWM implementation, the gain pulse periods are equal to the frame period $\tau_{c,v} = \tau_{c,\omega} = \overline{T} = 40$ ms and the gain pulse widths are $\tau_{0,v} = n_v T_s$ and $\tau_{c,\omega} = n_\omega T_s$, in the range between T_s and $\overline{T}$, (with n_v and n_ω integers).

– ii) Under the PFM implementation, the gain pulse width is equal to the sampling period $\tau_{0,v} = \tau_{0,\omega} = T_s$ with gain pulse periods $\tau_{c,v} = T/n_v$ and $\tau c, \omega = T/n_\omega$.

Rotations φ_{X_F} and φ_{Y_F} are evaluated between 10° and 60° (a combination of rotations higher than 60° in both axes leads the object to a pose non-visible from the camera) and rotation φ_Z between 30° and 180° (rotations $\varphi_Z > 180°$ are equivalent to rotations $\varphi_Z - 360°$ and rotations $\varphi_Z = [-30, -180]°$ have similar behaviors to $\varphi_Z = [30, 180]°$).

$(\varphi_{X_F}, \varphi_{Y_F}(^\circ)$	$\varphi_Z(^\circ)$	$(\lambda_v/\lambda_\omega) = (d_v/d_\omega)$	ε_{PWM} (x10^{-3})	Δ_{PWM} (x10^{-3})
$(10, 10)$	30	0.75	4.1	1.8
	100	0.65	7.3	2.2
	180	0.60	12.9	4.3
$(60, 10) - (10, 60)$	30	0.90	6.2	3.3
	100	0.80	18.6	5.9
	180	0.75	35.5	8.8
$(60, 60)$	30	0.90	9.7	4.7
	100	0.85	29.1	7.2
	180	0.80	48.8	14.9

Table 6.1: PWM performance for representative cases.

$(\varphi_{X_F}, \varphi_{Y_F}(^\circ)$	$\varphi_Z(^\circ)$	$(\lambda_v/\lambda_\omega) = (d_v/d_\omega)$	ε_{PFM} (x10^{-5})	Δ_{PFM} (x10^{-5})
$(10, 10)$	30	0.75	5.9	1.03
	100	0.65	23.3	5.20
	180	0.60	45.4	11.33
$(60, 10) - (10, 60)$	30	0.90	10.4	1.83
	100	0.80	62.8	13.92
	180	0.75	150	39.84
$(60, 60)$	30	0.90	19.3	3.38
	100	0.85	75.6	17.66
	180	0.80	200	67.48

Table 6.2: PFM performance for representative cases.

Tables 6.1 and 6.2 summarize the performances for representative cases: limits in the rotation ranges and an extra case for intermediate Z axis rotation. The normalized image plane is considered infinite in order to overpass the image boundary constraint. Therefore, the effects derived from gain modulation and consequently control action modulation will be analyzed separately of other effects. Figs. 6.4 and 6.5 show graphically the trajectory of the center of the visual features in the normalized image plane (a), the time variation of its coordinates (b), the camera translational velocities (c), and the Euclidean distance of the center with respect to the gain tuning case (d), in both PWM and PFM implementations, for one of the cases $(\varphi_{X_F}, \varphi_{Y_F}, \varphi_{Z_F}) = (10, 60, 180)^\circ$.

By observing one of the VS task $(\varphi_{X_F}, \varphi_{Y_F}, \varphi_{Z_F}) = (10, 10, 30)^\circ$, the following conclusions arise:

- i) Error indices increase with the initial errors in both implementations, PWM and PFM, with the same $(\varphi_{X_F}, \varphi_{Y_F})$ values:

$$\varepsilon(\varphi_{X_F}, \varphi_{Y_F}, 30) < \varepsilon(\varphi_{X_F}, \varphi_{Y_F}, 100) < \varepsilon(\varphi_{X_F}, \varphi_{Y_F}, 180)$$
$$\Delta(\varphi_{X_F}, \varphi_{Y_F}, 30) < \Delta(\varphi_{X_F}, \varphi_{Y_F}, 100) < \Delta(\varphi_{X_F}, \varphi_{Y_F}, 180)$$

  and also with the same φ_Z value:

$$\varepsilon(10, 10, \varphi_Z) < \varepsilon(10, 60, \varphi_Z) < \varepsilon(60, 60, \varphi_Z)$$
$$\Delta(10, 10, \varphi_Z) < \Delta(10, 60, \varphi_Z) < \Delta(60, 60, \varphi_Z).$$

- ii) Error indices in PWM are greater than in the PFM implementation for the same $(\varphi_{X_F}, \varphi_{Y_F}, \varphi_Z)$ case:

$$\varepsilon_{PWM}((\varphi_{X_F}, \varphi_{Y_F}, \varphi_Z)) > \varepsilon_{PWM}((\varphi_{X_F}, \varphi_{Y_F}, \varphi_Z))$$
$$\Delta_{PWM}((\varphi_{X_F}, \varphi_{Y_F}, \varphi_Z)) > \Delta_{PFM}((\varphi_{X_F}, \varphi_{Y_F}, \varphi_Z)).$$

  The comparison between Δ_{PWM} and Δ_{PFM} can also be done by observing the ripple of the trajectory in the normalized image plane (Figs. 6.4(a) and 6.5(a)) and the maximum of the Euclidean distances (Figs. 6.4(d) and 6.5(d)).

- iii) The resulting ripper in the normalized image plane is not significant compared to the dimensions of the trajectory, as can be seen in Fig.6.6.

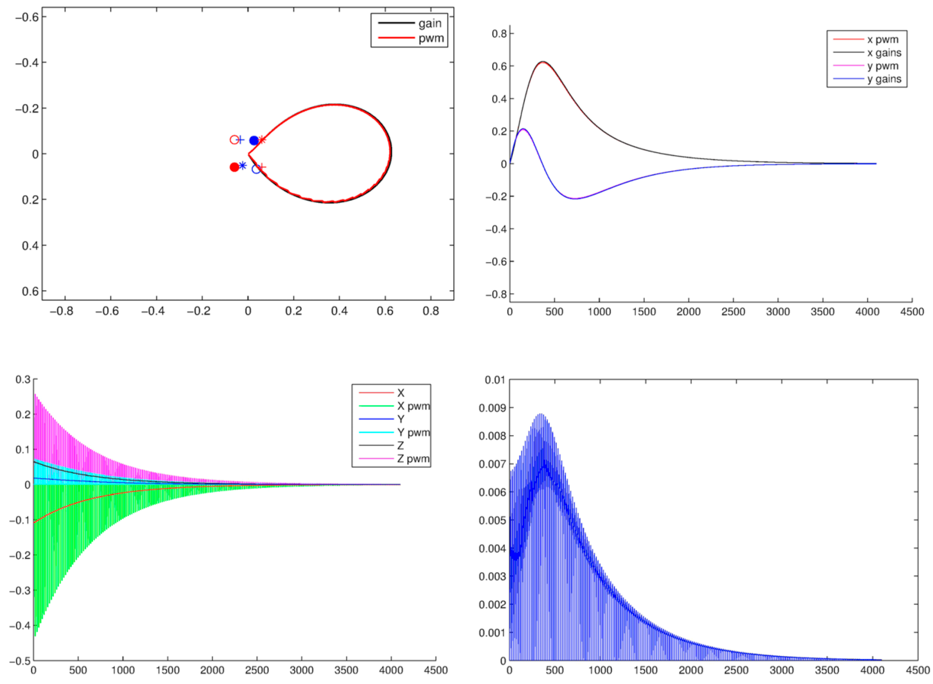


Figure 6.4: Gain tuning vs. PWM implementation for $(\varphi_{X_F}, \varphi_{Y_F}, \varphi_{Z_F}) = (10, 60, 180)^\circ$. (a) Trajectory of the center of the visual features in the normalized image plane (m). (b) Coordinates of the center of the visual features in the normalized image plane (m). (c) Camera translational velocities (m/s). (d) Euclidean distances - Coordinates of the center of the visual features in the normalized image plane (m).

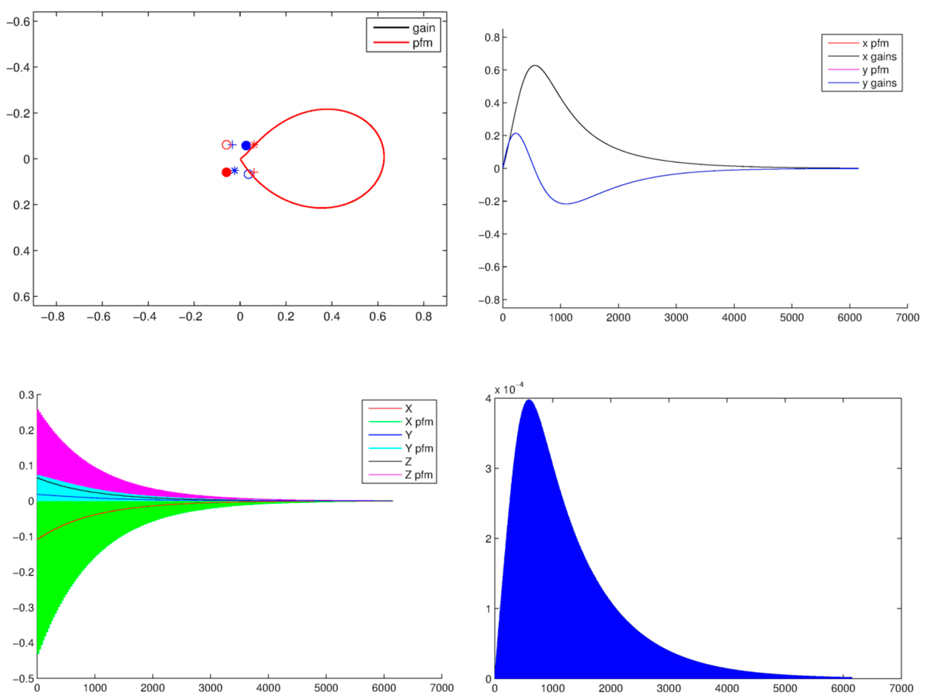


Figure 6.5: Gain tuning vs. PFM implementation for $(\varphi_{X_F}, \varphi_{Y_F}, \varphi_{Z_F}) = (10, 60, 180)°$. (a) Trajectory of the center of the visual features in the normalized image plane (m). (b) Coordinates of the center of the visual features in the normalized image plane (m). (c) Camera translational velocities (m/s). (d) Euclidean distances - Coordinates of the center of the visual features in the normalized image plane (m).

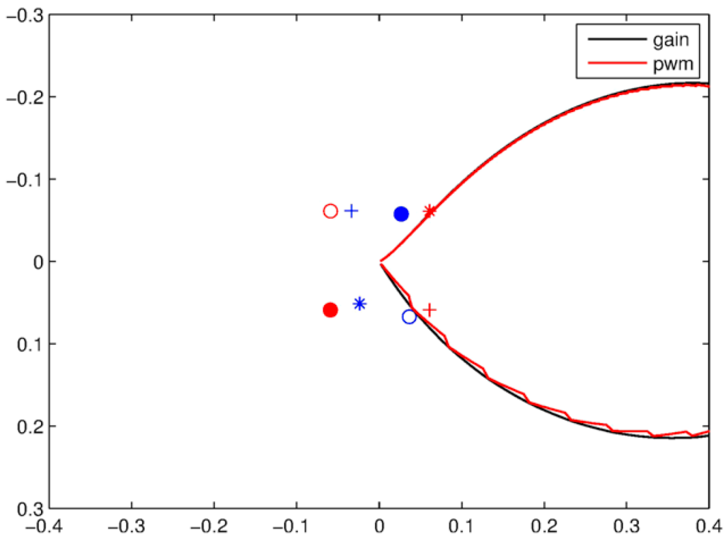


Figure 6.6: Zoom of the trajectory of the center of the visual features in the normalized image plane (m).

6.5 Practical issues

In this section we discuss about the practical implementation and how the presence of a real robot manipulator affects the proposed method. When a robot is taken into account, the control law is modified according to the following expression:

$$\dot{\mathbf{q}} = -\lambda(\widehat{\mathbf{L}}_s\, {}^c\mathbf{V}_e\, {}^e\mathbf{J}_e)^{\dagger}\, \mathbf{e} \tag{6.23}$$

where $\mathbf{q}$ is the robot joints velocity vector, ${}^c\mathbf{V}_e$ is the twist transformation matrix between camera and end-effector and ${}^e\mathbf{J}_e$ is the robot Jacobian. The presence of these two new matrices would introduce coupling between errors and robot joints if they are not block-diagonal. The twist transformation matrix ${}^c\mathbf{V}_e$ is a fixed matrix in the case of eye-in-hand systems and is defined as follows:

$${}^c\mathbf{V}_e = \begin{bmatrix} {}^c\mathbf{R}_e & [{}^c\mathbf{t}_e]_x\, {}^c\mathbf{R}_e \\ \mathbf{0}_{3x3} & {}^c\mathbf{R}_e \end{bmatrix} \tag{6.24}$$

Since ${}^c\mathbf{V}_e$ depends on how the camera is mounted, it can be forced to be block-diagonal if $[{}^c\mathbf{t}_e]_x\, {}^c\mathbf{R}_e = \mathbf{0}$.

The robot Jacobian ${}^e\mathbf{J}_e$ depends on the type of robot. Under robots with decoupled joints to end-effector transformation (such as Cartesian robots with wrist rotations), the Jacobian matrix is block-diagonal and the PWM-PFM implementation can also be applied to obtain the corresponding joint velocities $\dot{\mathbf{q}}$. However, if any coupling in the Jacobian matrix exists, the desired camera velocity would be transformed to joints velocity by means of the robot Jacobian. Discontinuities would appear in the joints velocities, but they would be filtered by the system dynamics, resulting thus in an equivalent continuous signal. Multi-rate VS architectures would be ideal to deal with different rates resulting from the inclusion of a robot. Information regarding the robot (that is, robot Jacobian) could be updated with higher rate than the visual information. Moreover, a parameterized image processing could be performed if selective information is needed (for example if only the translation error is needed). In conclusion, the proposed approach could be directly applied under Cartesian robots with wrist rotations to obtain modulated joints velocities, and multi-rate techniques could be applied to address the sensor latency.

6.6 Conclusions

Firstly, the proposal copes with the idea of having the same convergence time for all components of the error vector by inserting an error weighting matrix in the control law without overstretching any part of the VS system. The procedure was then transferred to fully-decoupled VS approaches, i.e. those with block-diagonal interaction matrix. In particular, in the fully decoupled PBVS approach, the coefficients of the gain matrix were tuned to get the same convergence time for camera translation and rotation. Next, novel PWM and PFM visual servoing techniques were presented, consisting in modulating, in pulse width (PWM) and pulse frequency (PFM) with high-frequency signals. This opens the possibility of transferring some of the advantage of PWM and PFM to the VS problem. The expected appearance of ripple due to the concentration of the control action in pulses was analyzed under a common VS scenario: a positioning task of a 6-DOFs camera with respect to a motionless object. Three main conclusions were extracted: 1) the higher the initial errors are, the higher the ripper is in both implementations; 2) the appearance of ripple is more evident in the PWM implementation; 3) the order of magnitude of the ripple is low compared with the dimensions of the signals. This high frequency ripple does not affect to the performance since it is filtered by the dynamics of the system. Moreover, the proposed control could be used to minimize the impact of friction since it can be seen as a dither signal, a high frequency component added to the control signal to keep the system at a non-zero velocity and avoiding stick-slip friction.

Advantages of the proposed approach:

– Same convergence time in VS using weighting matrices. Convergence time of all the components of the error vector in VS schemes is adjusted to avoid overstretching on any part of the system.

– Same convergence time in fully decoupled VS using gain matrices. The same procedure as the one explained in the previous item can be applied to fully decoupled VS schemes using gain matrices.

Traditional PWM and PFM benefits, which could apply to the VS problem, are:

– Discrete equivalent to analog controllers. PWM and PFM duty cycles are adjusted to obtain an equivalent average continuous value.

- Signal modulation in telecommunications. PWM and PFM are forms of signal modulation: data values are encoded at one end and decoded at the other end of the transmission. In robotics, this concept can also be useful in rough industrial environments: transmitting a modulated signal instead of an analog value to avoid effects of noise.

- Power efficiency. PWM is used to control the amount of power delivered to a load in a more efficient way than with power delivery by resistive means. This concept of power efficiency can be transferred to the control of joint motors in robot arms. Power can be directly delivered to the power electronics stage without needing A/D conversion.

Main limitations of the method:

- The presence of ripper in the signal due to the concentration of the control action in pulses is an expected drawback of this approach.

In the next chapter, the development of other visual servoing techniques and applications, coauthored with members of the same robotics and automation research group, are presented.

Chapter 7

Other results in visual servoing applications

During the PhD, the author of this thesis has also contributed to the development of other visual servoing (VS) techniques and applications, together with members of the same robotics and automation research group. Three of these co-works are detailed below.

The main contents of this chapter are represented using a mind map in Figure 7.1.

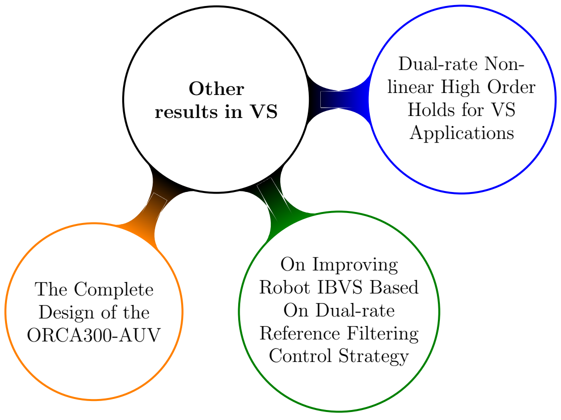

Other results in VS
Dual-rate Non-linear High Order Holds for VS Applications
The Complete Design of the ORCA300-AUV
On Improving Robot IBVS Based On Dual-rate Reference Filtering Control Strategy

Figure 7.1: Mind map representing the main contents of this chapter.

7.1 Dual-rate Non-linear High Order Holds for Visual Servoing Applications

This first work presents novels dual-rate non-linear high order holds (DR-NLHOH), where inputs of the process are updated at every frame-period $\bar{T}$, meanwhile outputs are updated at a base-period T, where $\bar{T} = NT$, being $N \in \mathbb{Z}^{+}$, (Armesto et al. (2008)). It is interesting to remark its aim is to generate an inter-sampling signal at higher frequency by including the knowledge of the closed loop system behavior. In this sense, conventional dual-rate high order holds (DR-HOHs) (Armesto and Tornero (2003)) do not take such information into account, frequently failing in predicting the estimation of the signal.

It also introduces a methodology for training such dual-rate non-linear holds based on artificial neural networks (ANN) with synthetic data. The key idea is to provide a sequence of values of the signal to be estimated, obtained from a closed loop simulated environment at base period, T. From collected data, inputs of the ANN are considered to be $\bar{T}$ time-spaced, while target outputs are considered to be T time-spaced. As a consequence, outputs of DR-NLHOH are, indeed, a lifted signal (Huang and Xu (2011)), which means that they are packed into a single signal whose elements must be distributed over time every base period.

Fig. 7.2 shows the comparative between the estimations provided by DR-FOH and its equivalent DR-NLHOH, trained with the ANN-based proposal. It can be seen that estimations provided by DR-NLHOH are more similar to ideal signal than the produced by DR-FOH.

DR-NLHOH have been implemented in a Kuka KR5 sixx R650 with an IBVS application in eye-in-hand configuration. Two different types of experiments are proposed: i) positioning task and ii) tracking task of an object describing a square trajectory.

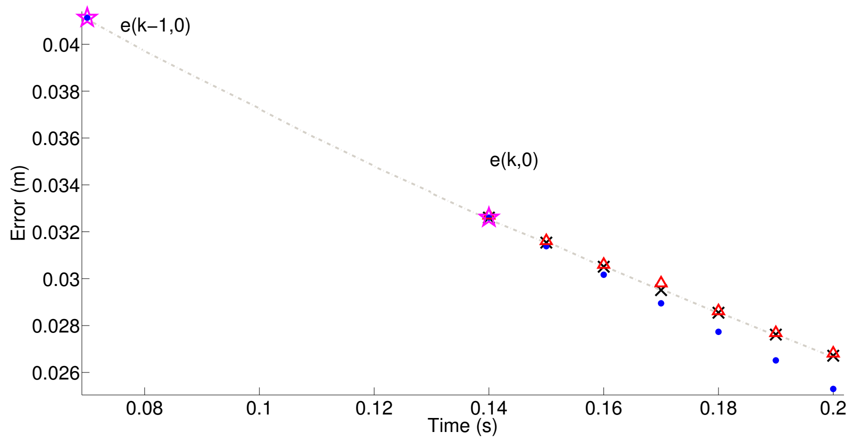
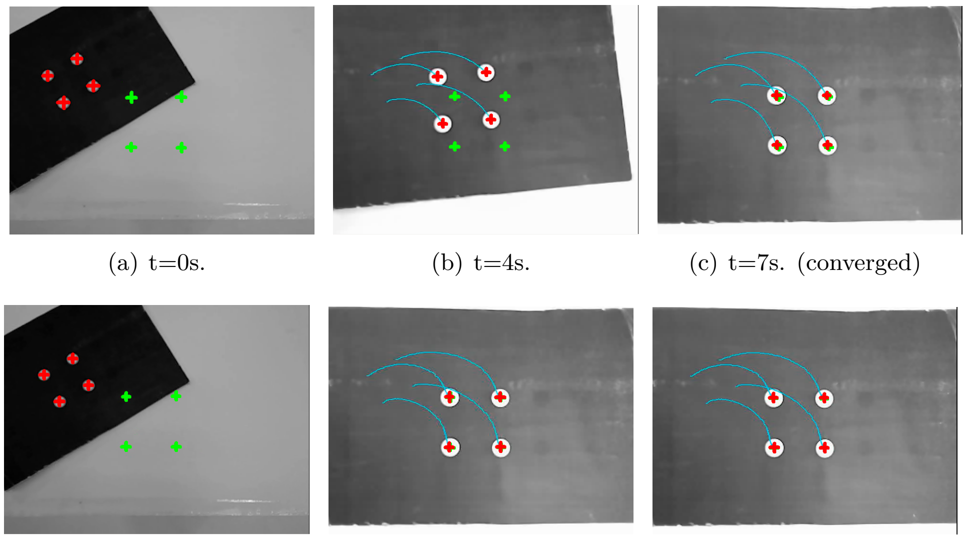


Figure 7.2: Comparison between dual-rate estimations: ideal (black crosses), dual-rate first order holds (blue dots), dual-rate non-linear first order hold (red triangles), $\mathbf{I}(k) = \{\mathbf{e}(k-1,0), \mathbf{e}(k,0)\}$ (magenta diamonds).

Positioning task. Fig. 7.3 shows a sequence of frames at different time instants in order to compare single-rate at low frequency and dual-rate using DR-NLHOHs. The dual-rate approach using DR-NLHOHs provides a good performance of features trajectories and the algorithm convergence is around two times faster than conventional single-rate approach and nearly the same convergence time than DR-FOH, but with higher stability margin.

Tracking task. Fig. 7.4 shows the quadratic error evolution. Once again, DR-NLHOHs presents less quadratic error than the system with conventional dual-rate holds due to better predictions, especially when the object changes its trajectory. In any case, dual-rate system has an important improvement in the object tracking with respect to single-rate cases at low sampling frequencies.

The proposed approach increases the robustness and the stability margin, without increasing the convergence time with respect to the conventional standard dual-rate high order holds.



(a) t=0s. (b) t=4s. (c) t=7s. (converged)

(d) t=0s. (e) t=4s. (converged) (f) t=7s. (converged)

Figure 7.3: Image plane trajectories: (a-c) single-rate; (d-f) DR-NLHOHs. Initial features (red crosses) and desired features (green crosses).

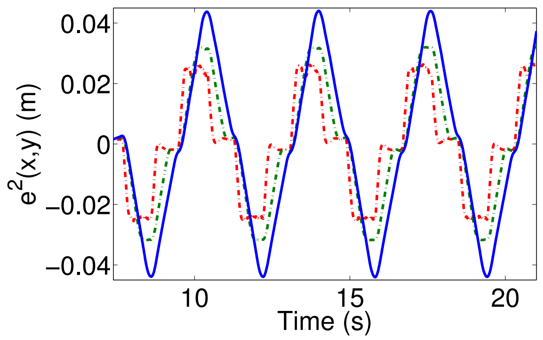


Figure 7.4: $e^2(x, y)$ with optimal gain (λ): single-rate at low frequency (blue solid line), dual-rate first order hold (green dashed line), dual-rate non-linear hold (red dot-dashed line).

7.2 On Improving Robot Image Based Visual Servoing Based On Dual-rate Reference Filtering Control Strategy

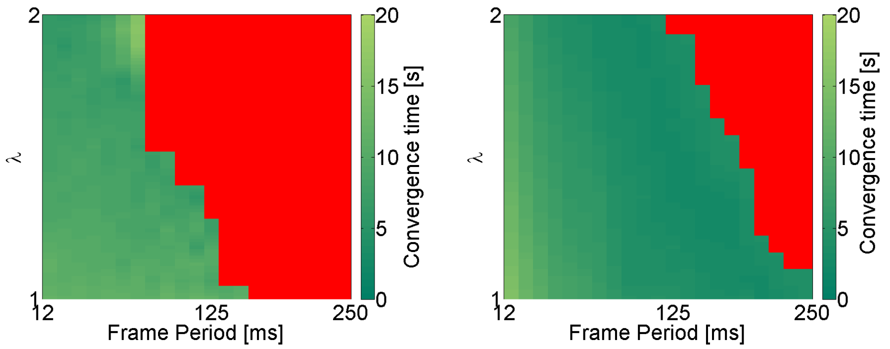
It is well known that the use of multi-rate control techniques have improved the performance of many systems in general, and robotic systems, in particular. The main contribution of this second work is the generalization of the Reference Filtering Control Strategy, presented in Section 7.1 and Solanes et al. (2012), from a dual-rate point of view. The proposal exploits the dual-rate nature of many VS applications, resulting in the dual-rate reference filtering controller, which improves the control properties by overcoming the problem of sensor latency.

The proposed controller has been implemented using industrial and embedded systems with significant hardware limitations to prove that the proposed dual-rate reference filtering controller performs better (in terms of convergence time, reachability and robustness), not only than the classic single-rate IBVS controller but even better than the single-rate reference filtering controller. For this purpose, the experimental requirements were addressed using a 6 DOF industrial manipulator (KUKA KR5 sixx R650) with a smart camera (VC6212 nano) in eye-in-hand configuration, controlled by an IGEPv2 board. The entire system is subject to hard real-time requirements.

Simulation results. Comparison between DR-EKFS-IBVS, SR-HF-EKFS-IBVS and SR-LF-EKFS-IBVS controllers is carried out throughout a positioning tasks, which consists in a 150° pure rotation error around the camera optical axis.

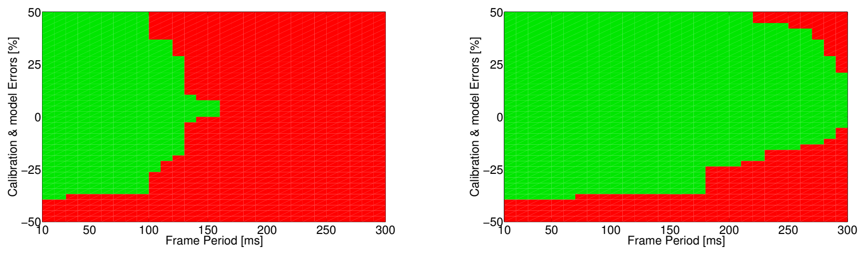
Figure 7.5 shows the reachability and the convergence time performed by using SR-LF-IBVS and DR-EKFS-IBVS controllers, and varying the controller gain parameter λ and the camera's frame rate. In those sub-figures, the zone colored in red indicates that the algorithm has failed to solve the task, while degraded green shows its convergence time. The test shows that the DR-EKFS-IBVS controller is not only more robust against the delay introduced by the camera's frame rate, but also converges faster than its equivalent single-rate one working at frame rate.

Moreover, results in Figure 7.6 demonstrate that the dual-rate control strategy is not only able to reach out the solution when high frame periods are required by the vision system but also is much more robust against camera calibration and object model errors than its single-rate counterpart.



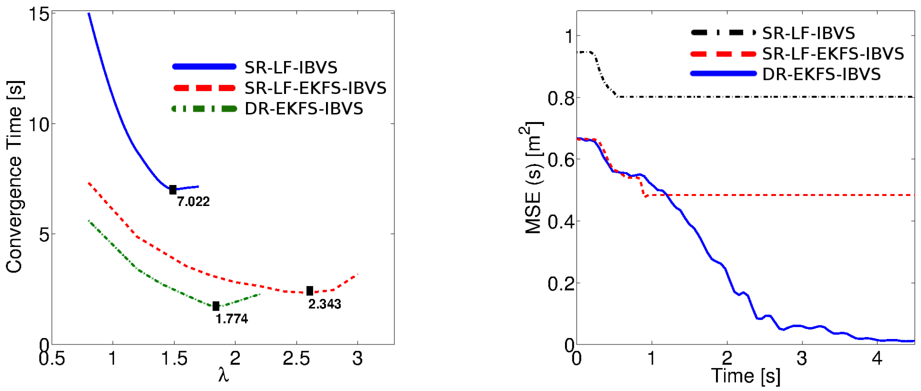
(a) SR-LF-EKFS-IBVS performance. (b) DR-EKFS-IBVS performance.

Figure 7.5: Task reachability and convergence time in function of the controller gain and the frame period used.



(a) SR-LF-EKFS-IBVS. (b) DR-EKFS-IBVS.

Figure 7.6: Analysis of the resference filtering control strategy robustness against calibration and model errors.



(a) Comparison of convergence time for positioning task 1. The parameters used are the optimal ones in each case.

(b) Comparison of success for positioning task 2. The case shown corresponds to $N = 8$ and $h = 8$ and the rest of parameters used are the optimal ones in each case.

Figure 7.7: Analysis of the improvements reached by using the proposed dual-rate reference filtering control strategy.

Experimental results. The set-up includes a 6 DOF industrial manipulator (KUKA KR5 sixx R650) with an internal robot controller running at $\delta = 10ms$, an IGEPv2 embedded board implementing the control algorithms and an industrial smart camera VC6212, which implements the computer vision algorithms at updates the visual features each $80ms$ (which means $N = 8$).

The comparison between DR-EKFS-IBVS, SR-LF-IBVS and classic SR-HF-IBVS controllers has been carried out for two positioning tasks: the first one allows the convergence for all those controllers, while for the second task the object is placed in a configuration in which the classic IBVS fails (see Solanes et al. (2013)), which allows us to study the benefits of the proposed approach with respect to our previous single-rate method.

Figure 7.7(a) shows the performance obtained using each control algorithm by solving the first positioning task. Choosing the optimal covariance matrices values for each controller and tuning the controller gain, the figure shows the minimum convergence time that is possible for each controller. It can be seen that, the best performance obtained is by using the DR-EKFS-IBVS controller while the worst is by using the classic SR-IBVS one.

The dual-rate filtering controller shows a much better performance in terms of convergence time and robustness. Moreover, it has been proved that, in some scenarios where the single-rate controllers fail because of sensor latency, the dual-rate reference filtering controller succeeds.

7.3 The Complete Design of the ORCA300-AUV

In this third work, the design, manufacture and control of an Autonomous Underwater Vehicle (AUV) called ORCA300-AUV was fully described. ORCA300-AUV was developed under the DIVISAMOS ("DIseño de un Vehículo de Inspección Submarina Autónoma para Misiones OceanográficaS") research project, as a first stage to have an underwater auto-guided and partially teleoperated vehicle that can be fitted with sensors and instrumentation to perform a variety of missions, e.g. the analyses of habitat degradation and biodiversity reduction caused by toxic substances as well as of seaboard erosion caused by construction at harbor areas. Preserving marine and continental waters requires the development of new techniques, technologies and devices able to explore different habitats with the goal of protecting and managing them. In this regard, hardware architecture and sensor/control software are developed in addition to an autonomous navigation system for submarine vehicles along with a water quality monitoring system as well as other bathymetry mapping applications for aquatic ecosystems with high resolution geo-referenced data and cartographic projection of the sea bed.

ORCA300-AUV was equipped with thrusters that allow maneuvering and control of 4 degrees of freedom, a control unit and navigation system composed of a computer, an inertial navigation sensor and a power supply unit made of lithium batteries, in addition to sensors such as two cameras for stereoscopic vision, a sonar, and inertial sensors, among others. The experience of the research group in auto-guided terrain vehicles could be applied to maritime technology, by means of different control applications such as: kinematic and dynamic modeling, opti-acoustic 3D scene reconstruction, sensorial fusion for autonomous navigation, visual line tracking and VS.

In the underwater robotics field few attempts have been made to use vision sensors for control (Marks et al. (1994), Rives and Borrelly (1997), Negahdaripour et al. (1999), Lane et al. (2000), Lots et al. (2000) and Lots et al. (2001)). However, vision sensors provide some interesting features compared to classical positioning sensors. For example, magnetic compasses suffer from

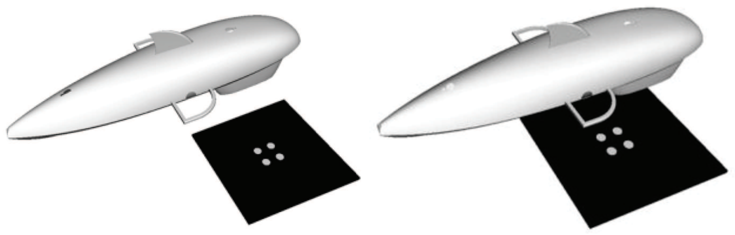


Figure 7.8: IBVS positioning task with ORCA300-AUV. Initial and desired pose.

a slow update rate and cannot be used in the vicinity of man-made metallic structures. With the exception of depth sensors, which are both accurate and fast, on-board translational motion sensors are integrating sensors (i.e. accelerometers, Doppler velocity logs) hence subject to drift, and therefore unsuitable for station keeping. On the contrary, a camera is not subject to magnetic influences and can also be used as a local absolute positioning sensor. Despite its short range (typically 3-10 meters) and the need for heavy computing power, VS or visual control allows very diverse tasks in underwater robotics, such as for example station keeping or pipe-following to be carried out.

Simulation results of a positioning task in IBVS with the ORCA300-AUV are depicted in the following figures. The goal is to achieve the desired pose, with respect to a mark in the terrain formed by four points. Fig. 7.8 shows the initial and desired poses. Fig. 7.9 and 7.10 shows the evolution of the errors and the control action, whereas Fig. 7.11 shows the image features trajectory and Fig. 7.12 the 3D camera trajectory.

In the next chapter, the conclusions of this thesis are exposed.

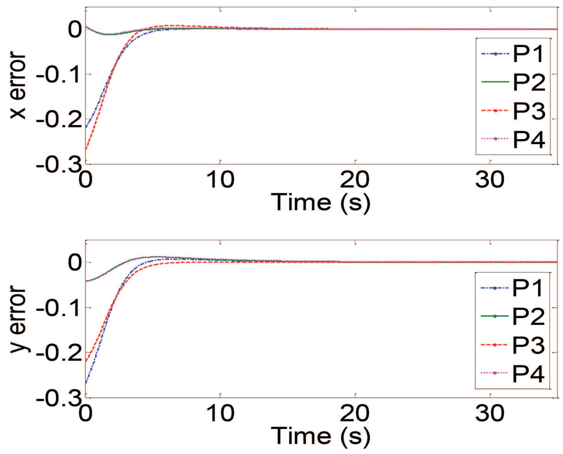


Figure 7.9: IBVS positioning task with ORCA300-AUV. Errors evolution.

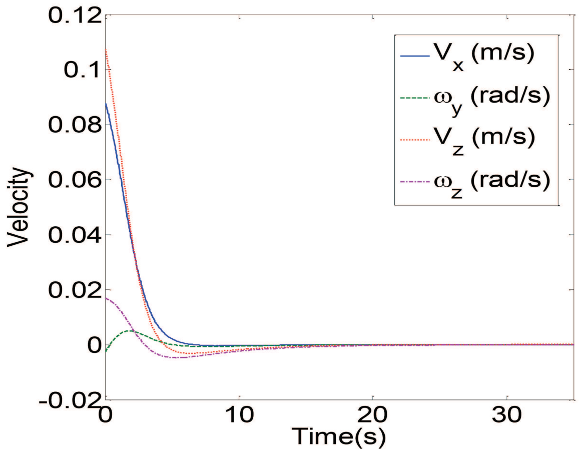


Figure 7.10: IBVS positioning task with ORCA300-AUV. Control action evolution.

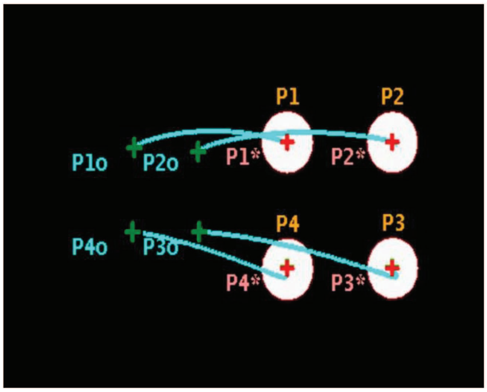


Figure 7.11: IBVS positioning task with ORCA300-AUV. Image features trajectory.

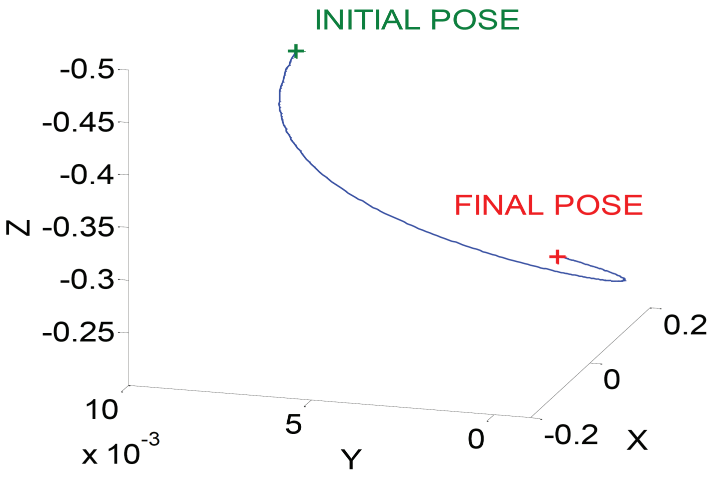


Figure 7.12: IBVS positioning task with ORCA300-AUV. 3D camera trajectory.

Chapter 8

Conclusions

The main contents of this chapter are represented using a mind map in Figure 8.1.

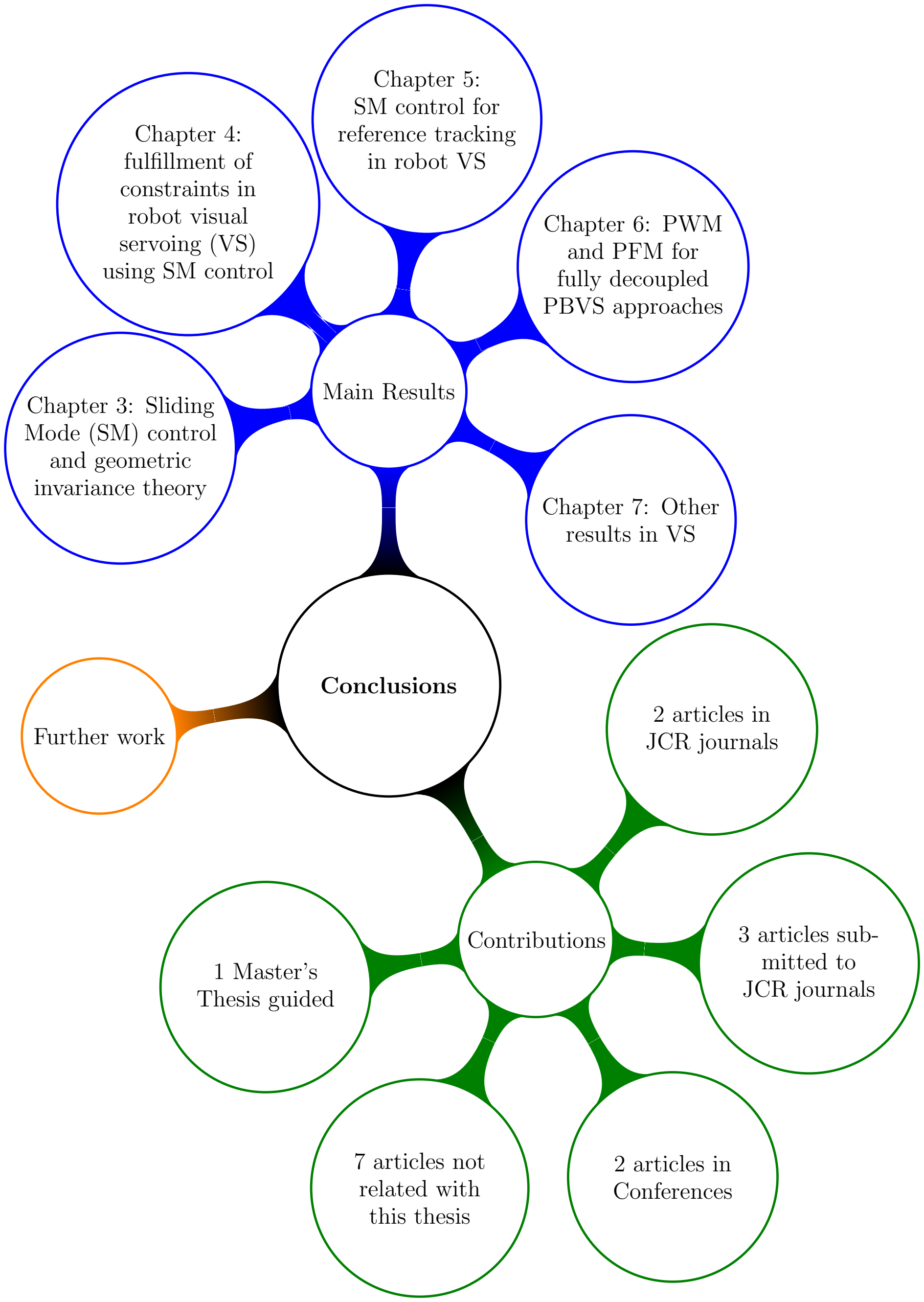

Chapter 5: SM control for reference tracking in robot VS
Chapter 4: fulfillment of constraints in robot visual servoing (VS) using SM control
Chapter 6: PWM and PFM for fully decoupled PBVS approaches
Main Results
Chapter 3: Sliding Mode (SM) control and geometric invariance theory
Chapter 7: Other results in VS
Conclusions
Further work
2 articles in JCR journals
Contributions
3 articles submitted to JCR journals
1 Master's Thesis guided
7 articles not related with this thesis
2 articles in Conferences

Figure 8.1: Mind map representing the main contents of this chapter.

8.1 Main Results

Different proposals to deal with common problems in robot visual servoing (VS) based on the application of discontinuous control methods have been proposed and substantiated by simulation results and real experiments.

- In Chapter 3 the principles of sliding mode (SM) control and geometric invariance theory (Garelli et al. (2011)), that are used by the proposed approach to tackle problems in VS have been described.
- In Chapter 4 a proposal for fulfillment of constraints in VS using SM control has been presented. In particular, the proposal uses SM methods to satisfy motion constraints (joint limits, speed limits, forbidden area to avoid collisions, task space limits and robot workspace limits) and visibility constraints (camera field-of-view and occlusions) of VS applications. Moreover, another task with low-priority is considered to properly track the target object. The feasibility and effectiveness of the proposed approach have been illustrated in simulation and with real experiments. It is interesting to remark that, despite that the sampling time of the real platform used for experimentation was not small, 0.1 s, the performance of the proposed SM algorithm was satisfactory for both Position Based Visual Servoing (PBVS) and Image Based Visual Servoing (IBVS) experiments, even for a gradient vector error of 30%. The main advantages of the proposed approach are: low computational cost (see Appendix at the end of that chapter), robustness and fully utilization of the allowed space for the constraints. On the other hand, like other sliding-mode applications, the proposed method has the chattering drawback, but becomes negligible for reasonable fast sampling rates. Furthermore, an automated approach for tool changing in industrial robots using VS and the proposed method has been presented.
- In Chapter 5, an approach based on SM control is proposed for reference tracking in robot VS using industrial robot manipulators. In particular, two sliding-mode controls have been obtained depending on whether the joint accelerations or the joint jerks are considered as the discontinuous control action. Both sliding-mode controls have been compared theoretically and in simulation to their equivalent continuous counterparts. The main advantages of the proposed sliding-mode approach are smoothness, robustness and low computational cost, while its main limitation is the

chattering drawback. The robustness of the method compared to the continuous equivalent has been successfully verified in experiments by introducing an error in the Jacobian matrix. System stability has been demonstrated with a theoretical proof.

- In Chapter 6 discontinuous control based on pulse width and pulse frequency modulation is proposed for fully decoupled PBVS approaches, in order to get the same convergence time for camera translation and rotation. The expected appearance of ripple due to the concentration of the control action in pulses was analyzed. However, this high frequency ripple does not affect to the performance since it is filtered by the dynamics of the system.
- In Chapter 7 other results in VS approaches, in which the author of thesis has collaborated, are described.

8.2 Contributions

The work done for this PhD thesis has led to several publications in specialized international journals and conferences, which are summarized in this section.

8.2.1 Articles published in journals

From Chapter 6:

- **Muñoz-Benavent, P.**, Solanes, J. E., Gracia, L., & Tornero, J. (2015). PWM and PFM for visual servoing in fully decoupled approaches. *Robotics and Autonomous Systems*, 65, 57-64. DOI 10.1016/j.robot.2014.11.011.

From Section 7.2:

- Solanes, J. E., **Muñoz-Benavent, P.**, Girbés, V., Armesto, L., & Tornero, J. (2016). On improving robot image-based visual servoing based on dual-rate reference filtering control strategy. *Robotica*, 34, 2842-2859. DOI 10.1017/S0263574715000454

8.2.2 Articles submitted to journals

From Chapter 4:

- **Muñoz-Benavent, P.**, Solanes, J. E., Gracia, L., & Tornero, J. (2017). Robust Auto Tool Change for Industrial Robots Using Visual Servoing. *Robotics and Computer-Integrated Manufacturing.* Under review.
- **Muñoz-Benavent, P.**, Solanes, J. E., Gracia, L., Esparza, A., & Tornero, J. (2017). One-Side Sliding-Mode Method to Satisfy Limited Field of View and Object Occlusion in Visual Servoing. *IEEE Transactions on Industrial Electronics.* Under review.

From Chapter 5:

- **Muñoz-Benavent, P.**, Solanes, J. E., Gracia, L., Esparza, A., & Tornero, J. (2017). Sliding Mode Control for Robust and Smooth Reference Tracking in Robot Visual Servoing. *International Journal of Robust and Nonlinear Control.* Under review.

8.2.3 Articles published in conferences

From Section 7.1:

– Solanes, J. E., Armesto, L., Tornero, J., **Muñoz-Benavent, P.**, & Girbés, V. (2012). Dual-Rate Non-Linear High Order Holds for Visual Servoing Applications. In: Herrmann G. et al. (eds) *Advances in Autonomous Robotics. TAROS 2012. Lecture Notes in Computer Science*, 7429, 152-163. Springer, Berlin, Heidelberg. DOI 10.1007/978-3-642-32527-4_14.

From Section 7.3:

– F. Aguirre, A. Muñoz, **P. Muñoz-Benavent**, J.E. Solanes, V. Girbés, V. Colomer, L. Armesto, J. Tornero. (2012). The complete design of the ORCA300-AUV. *In World Conference on Maritime Technology.*

Other published works

– **Muñoz-Benavent, P.**, Andreu-García, G., Valiente-González, José M., Atienza-Vanacloig, V., Puig-Pons, V., Espinosa, V. (2017) Automatic Bluefin Tuna Sizing using a Stereoscopic Vision System. *ICES Journal of Marine Science.* In press. DOI: 10.1093/icesjms/fsx151

– Muñoz M., **Muñoz-Benavent P.**, Munera E., Blanes J.F., Simó J. (2014) Limited Resources Management in a RoboCup Team Vision System. In: Armada M., Sanfeliu A., Ferre M. (eds) *ROBOT2013: First Iberian Robotics Conference. Advances in Intelligent Systems and Computing*, 253, 27-39. DOI 10.1007/978-3-319-03653-3_3.

– **Muñoz-Benavent, P.**, Armesto, L., Girbés, V., Solanes, J. E., Dols, J. F., Muñoz, A., & Tornero, J. (2012). Advanced Driving Assistance Systems for an Electric Vehicle. *International Journal of Automation and Smart Technology*, 2(4), 329-338. DOI 10.5875/ausmt.v2i4.169.

– **Muñoz-Benavent, P.**, Armesto, L., Soria, D., Pasieka, M., & Tornero, J. (2012). Advanced Driving Assistance Systems for an Electric Vehicle. *In The 43rd International Symposium on Robotics, ISR*, 940-945.

– Armesto, L., Girbés, V., Vincze, M., Olufs, S., & **Muñoz-Benavent, P.** (2012). Mobile robot obstacle avoidance based on quasi-holonomic smooth paths. In: Herrmann G. et al. (eds) *Advances in Autonomous Robotics. TAROS 2012. Lecture Notes in Computer Science*, 7429, 244-255. Springer, Berlin, Heidelberg. DOI 10.1007/978-3-642-32527-4_22.

- Blanes, F., **Muñoz-Benavent, P.**, Muñoz, M., Simó, J. E., Coronel, J. O., & Albero, M. (2011). Embedded distributed vision system for humanoid soccer robot. *Journal of Physical Agents (Jopha)*, 5, 55-62.

- **Muñoz-Benavent, P.**, Blanes Noguera, F., Simó Ten, J. E., Coronel Parada, J. O., & Albero, M. (2010). Sistema de visión empotrado en arquitectura de control distribuida para robot humanoide. *In Congreso Español De Informática (CEDI). XI Workshop of Physical Agents.*

Master's Thesis Guided

- Arnal Benedicto, L. (2012). *Detección de personas con visión artificial y sensores de rango.* Master's Degree in Automation and Industrial Computing. Universitat Politècnica de València. Guided by Leopoldo Armesto Ángel and **Pau Muñoz Benavent**.

8.3 Further work

The proposed SM algorithm for fulfillment of constraints in VS 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. (2012a)). Some of these situations could be avoided using a high-level planner with the complete geometric data of the problem to perform long-term planning. As stated in Chapter 4, path planning algorithms have been studied for VS applications, so the integration of online corrective terms based on the proposed SM control in a global path planner would be of great interest.

With regard to the proposed SM control for reference tracking, implementation of faster vision algorithms, together with the technological progress, would allow to update the visual information with faster sampling times. In this case, the chattering band would be reduced and the speed of the target object, in tracking tasks, and the convergence time, in position tasks, would be improved.

VS and the proposed approaches could be implemented for real industrial applications to strengthen the autonomy of robotic cells and to perform more complex tasks.

Moreover, other applications derived from VS, such as object grasping, would also be studied, as well as different robotic configurations (collaborative robots, multiple cameras) and other robotic platforms (mobile robots, unmanned aerial vehicles and autonomous underwater vehicles).

Appendix A

Experimental Platform

The applicability and feasibility of the proposed approaches are substantiated by experimental results using a conventional 6R industrial manipulator. However, the process to set up the software and hardware architectures needed to have this test platform has been very time consuming. The positive point is the experience and knowledge accumulated.

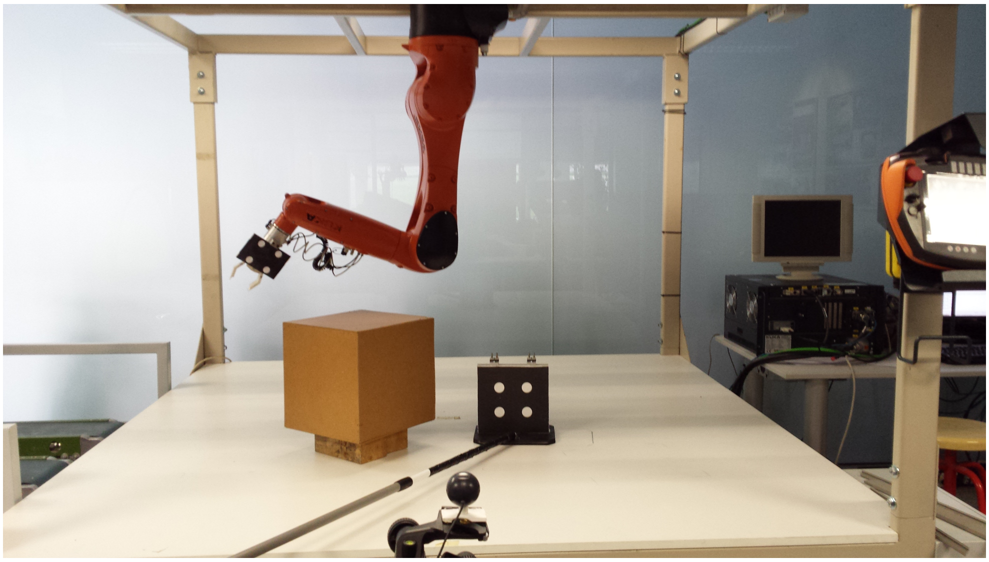


Figure A.1: Industrial robot manipulator cell.

The platform consists of a Kuka Agilus KR6 R900 sixx robot manipulator in ceiling-mounted position, equipped with the Kuka.RobotSensorInterface (RSI) technology that allows external real-time communication using the Ethernet UDP protocol; a general purpose web cam used for image acquisition; 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 software architecture implemented on the external PC consists of three main modules:

– Computer vision module. Implements the image processing and computer vision algorithms needed to update the visual feedback information. It is based on ViSP (Visual Servoing Platform) (Marchand et al. (2005)).

– Control module. Implements the different approaches proposed along this thesis.

– Communication module. Guarantees a real-time communication with the robot.

These modules are implemented as three periodic threads and scheduled following a fixed priority scheme, from highest to lowest: server, control and vision threads. The server period must be set to 4 milliseconds due to robot specification. Both the vision period and the control period T_s are set to 100 milliseconds to guarantee an appropriate scheduling. This real-time constraints are fulfilled thanks to the real time operative system and the Orocos Toolchain (Soetens and Bruyninckx (2005)).

Table A.1 shows the main technical specification of the Kuka Agilus KR6 R900 sixx robot manipulator. It is equipped with the KR C4 compact controller, which offers high performance and reliability in a compact design. Its flexible configuration and expansion capability make it a real all-rounder. The number of hardware components, cables and connectors has been significantly reduced and replaced by software-based solutions. The robust, high-quality controller is designed for low maintenance; the temperature-controlled technology only switches on briefly when needed, and is barely audible. The KUKA smartPAD is used to communicate with the KR C4 compact controller and manage the robot. Interactive dialogues provide the user with those operator control elements that are currently required.

Model	Agilus R900 sixx
Payload	6 kg
Max. reachability	901 mm
Max. speed	13 m/s
Controller	KR C4sr
Number of axes	6
Repeatability	$<\pm0.03$ mm
Weight	52 kg
Mounting positions	Floor, ceiling, wall

Table A.1: Kuka Agilus KR6 R900 sixx specifications.

On Externally Control of Kuka Robots

Industrial robot manipulators are usually controlled with classical DIO cards. However, that is because the automatic program is developed offline and repeated once and again, the most of the cases in open loop. It is possible to control Kuka robots from external devices in two ways: using "KUKA.OPC" technology, or Kuka "*KUKA.Ethernet RSI XML*" technology.

"KUKA.OPC" technology is commonly used within the industry. The problem is that real-time cannot be assured since the KCP layer has higher priority rather than this technology. On the contrary, "*KUKA.Ethernet RSI XML*" technology assures real-time since no-task has a higher priority than this one. The reason because the former one is more used, even though real-time requirements are not assured, is because the majority of industrial applications do not need to be control at real-time.

However, in order to validate our approaches, real-time has to be guaranteed, so "*KUKA.Ethernet RSI XML*" technology is used to perform all the validations presented along this work.

Real-time Control of Kuka Robots

The *KUKA.Ethernet RSI XML* is an add-on technology package with the following functions:

– Cyclical data transmission from the robot controller to an external system in the interpolation cycle of $3-12$ms (e.g. position data, axis angle, operating mode, etc.)

- Cyclical data transmission from an external system to the robot controller in the interpolation cycle of $3 - 12$ms (e.g. sensor data)
- Influencing the robot in the interpolation cycle of $3 - 12$ ms
- Direct intervention in the path planning of the robot

The characteristics of the package are the following:

- Reloadable RSI-object for communication with an external system, in conformity with *KUKA.RobotSensorInterface* (RSI)
- Communications module with access to standard Ethernet.
- Freely definable inputs and outputs of the communication object.
- Data exchange time-out monitoring.
- Expandable data frame that is sent to the external system. The data frame consists of a fixed section that is always sent and a freely definable section.

The *KUKA.Ethernet RSI XML* enables the robot controller to communicate with the external system via a real-time-capable point-to-point network link. This technology has suffered several changes between the version 2.1 used by the robot Kr5 and the version 3.1 used by robot Agilus. On e of the changes is that, meanwhile in version 2.8 the exchanged data could be transmitted via the Ethernet TCP/IP or UDP/IP protocol as XML strings, in the new version 3.1 this XML strings transmission is only allowed via UDP/IP protocol. In the case of the Kr5, the Ethernet TCP/IP was used, the UDP/IP protocol is the one already working on Agilus.

Programming of the *KUKA.Ethernet RSI XML* package is based on creating and linking RSI-objects. RSI-objects are small pieces of pre-programmed code that can be executed and has additional functionalities than the normal KRL-code. To be able to communicate externally through Ethernet, a specific standard object (ST_ETHERNET or ST_COROB) needs to be created. The code line for creating for example the ST_ETHERNET is typically: ***err=ST_ETHERNET(A,B,config_file.xml)***, where err is a type of string used by the RSI XML (called RSIERR) containing the error code produced when creating the object (normally #RSIOK when it works), A is

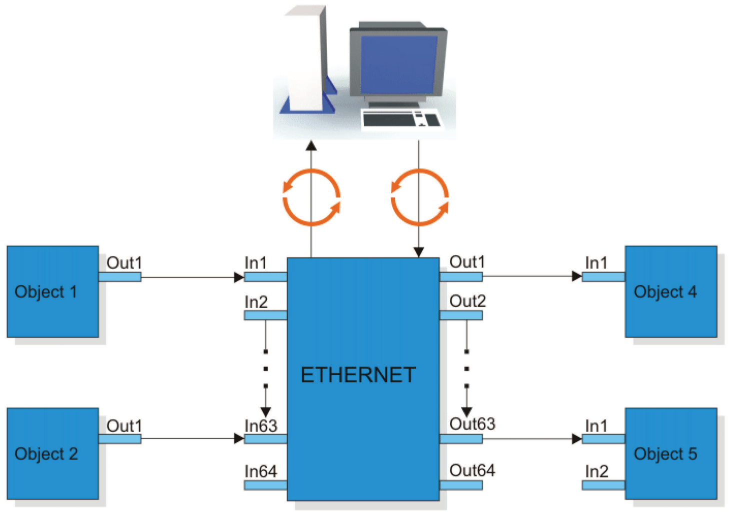


Figure A.2: Functional principle of data exchange.

an integer value that contains the specific RSI-object ID so that it is possible to locate and refer to, B is an integer value for the container to which you want the RSI-object to belong in order to create a group of different objects containing to the same container, config_file.xml is a configuration file located in the INIT folder (path *C:/KRC/ROBOTER/INIT*) on the robot controller that specifies what should be sent and received by the robot controller. The content of this file will be explained further down.

ST_ETHERNET and ST_COROB are objects that can be influenced by external signals, and also send data back to the external system in form of XML files, containing different tags with data. The data can be for example information about the robot's actual axis positions, Cartesian actual positions etc. This data shall send to the server and back within each interpolation cycle of 12ms. ST_ETHERNET has the same functionality as ST_COROB but with additional functionalities; one of these object always need to be created and correctly linked in the KRL code in order to establish communication with the external system. For this thesis, the communication object ST_ETHERNET was used. When one of these objects is created and linked correctly, the robot controller connects to the external system as a client.

There are different types of RSI-objects and depending on what you want to do, you have to create and link the correct objects to each other. Besides the standard Ethernet card, an additional card (3COM) was needed, to be able to handle the speed of the transferred data.

The robot controller initiates the cyclical data exchange with a KRC data packet and transfers further KRC data packets to the external system in the interpolation cycle of 12ms. This communication cycle is called an IPO-cycle (Input Process Output), and can be seen in Figure A.2 above. The external system must respond to the KRC data packets with a data packet of its own.

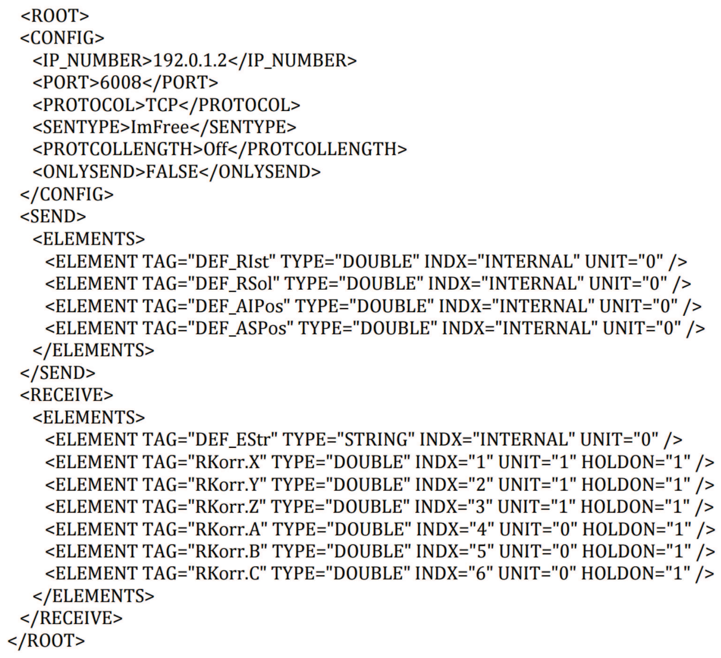
To be able to influence the robot, one needs to initiate an RSI-object for the movements. There are mainly two objects used for this: First, an object called ST_AXISCORR(A,B) for specific movements in axis A1 to A6, where A is the specific ID of the created object, and B is the container that the object belongs to; The second object is called ST_PATHCORR(A,B) for movements in Cartesian coordinates, where A, B are the same as for the ST_AXISCORR object.

A coordinate system is also needed (normally BASE, TCP, WORLD) as a reference for the movements. This is done by creating a RSI_object called ST_ON(A,B), where the parameter A is a string containing the coordinate system that is supposed to be used (expressed as #BASE, #TCP or #WORLD), and B is an integer value, 0 if the correction values sent to the robot shall be absolute, or 1 if they shall be relative.

Though it was discovered that when starting the WORLD coordinate system, the origin where set on where the robot were standing when starting the system. This only applies for the communication to the robot controller, so that the starting position shall always be set as 0 before starting to move the robot. The response sent back to the external system was in real coordinates with the WORLD coordinate system starting at its standard position on the base of the robot.

When the robot controller communicates with the external system it interchanges XML Strings. The content in the XML strings for the demo program provided by KUKA, is decided and defined in a file called ERX_config.xml (the configuration file, mentioned above), which is located in the robot controller, inside the INIT folder.

The IP address and port to which the robot controller will connect, when establishing the connection, are set in this file. The sub-tags under <SEND> are what the robot controller sends to the external system. The most important tags for sending to the external system for in this thesis are the tags called



```
 <ROOT>
 <CONFIG>
   <IP_NUMBER>192.0.1.2</IP_NUMBER>
   <PORT>6008</PORT>
   <PROTOCOL>TCP</PROTOCOL>
   <SENTYPE>ImFree</SENTYPE>
   <PROTCOLLENGTH>Off</PROTCOLLENGTH>
   <ONLYSEND>FALSE</ONLYSEND>
 </CONFIG>
 <SEND>
   <ELEMENTS>
     <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
     <ELEMENT TAG="DEF_RSol" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
     <ELEMENT TAG="DEF_AIPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
     <ELEMENT TAG="DEF_ASPos" TYPE="DOUBLE" INDX="INTERNAL" UNIT="0" />
   </ELEMENTS>
 </SEND>
 <RECEIVE>
   <ELEMENTS>
     <ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" UNIT="0" />
     <ELEMENT TAG="RKorr.X" TYPE="DOUBLE" INDX="1" UNIT="1" HOLDON="1" />
     <ELEMENT TAG="RKorr.Y" TYPE="DOUBLE" INDX="2" UNIT="1" HOLDON="1" />
     <ELEMENT TAG="RKorr.Z" TYPE="DOUBLE" INDX="3" UNIT="1" HOLDON="1" />
     <ELEMENT TAG="RKorr.A" TYPE="DOUBLE" INDX="4" UNIT="0" HOLDON="1" />
     <ELEMENT TAG="RKorr.B" TYPE="DOUBLE" INDX="5" UNIT="0" HOLDON="1" />
     <ELEMENT TAG="RKorr.C" TYPE="DOUBLE" INDX="6" UNIT="0" HOLDON="1" />
   </ELEMENTS>
 </RECEIVE>
</ROOT>
```

Figure A.3: The structure of *ERXconfig.xml*.

DEF_RIst, which is the real position of the robot's end-effector, and DEF_RSol that is the set position received from the external system. A typical example of the content in this file is shown in Figure A.3.

The DEF_AIPos and DEF_ASPos are real axis position and set axis position respectively. Under the tag RECEIVE is described what the robot controller expects to receive from the external system. In this example only corrections in six values (X, Y, Z, A, B and C) are included, tagged as RKorr.

Bibliography

Allibert, G., Courtial, E., and Chaumette, F. (2010a). Predictive Control for Constrained Image-Based Visual Servoing. *IEEE Trans. on Robotics*, 26(5):933–939.

Allibert, G., Courtial, E., and Chaumette, F. (2010b). Visual servoing via Nonlinear Predictive control. In Chesi, G. and Hashimoto, K., editors, *Visual Servoing via Advanced Numerical Methods*, pages 375–394. LNCIS 401, Springer-Verlag.

Armesto, L., Ippoliti, G., Longhi, S., and Tornero, J. (2008). Probabilistic Self-Localization and Mapping - An Asynchronous Multirate Approach. *IEEE Robotics & Automation Magazine*, 15(2):77–88.

Armesto, L. and Tornero, J. (2003). Dual-rate high order holds based on primitive functions. In *American Control Conf.*, pages 1140–1145.

Bakthavatchalam, M., Tahri, O., and Chaumette, F. (2014). Improving moments-based visual servoing with tunable visual features. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6186–6191.

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

Becerra, H. M., López-Nicolás, G., and Sagüés, C. (2011). A sliding-mode-control law for mobile robots based on epipolar visual servoing from three views. *IEEE Transactions on Robotics*, 27(1):175–183.

Becerra, H. M. and Sagüés, C. (2011). Sliding Mode Control for Visual Servoing of Mobile Robots using a Generic Camera. In Prof. Andrzej Bartoszewicz, editor, *Sliding Mode Control*, pages 221–236.

Bonfe, M., Mainardi, E., and Fantuzzi, C. (2002). Variable structure pid based visual servoing for robotic tracking and manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 396–401 vol.1.

Burger, W., Dean-Leon, E., and Cheng, G. (2015). Robust second order sliding mode control for 6D position based visual servoing with a redundant mobile manipulator. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1127–1132. IEEE.

Cazy, N., Wieber, P., Giordano, P., and Chaumette, F. (2015). Visual servoing when visual information is missing: Experimental comparison of visual feature prediction schemes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6031–6036, Seattle WA, USA.

Chan, A., Leonard, S., Croft, E. A., and Little, J. J. (2011). Collision-free visual servoing of an eye-in-hand manipulator via constraint-aware planning and control. In *Proceedings of the 2011 American Control Conference*, pages 4642–4648.

Chaumette, F. (1998). Potential problems of stability and convergence in image-based and position-based visual servoing. *The confluence of vision and control*, pages 66–78.

Chaumette, F. (2004). Image Moments: A General and Useful Set of Features for Visual Servoing. *IEEE Transactions on Robotics*, 20(4):713–723.

Chaumette, F. and Hutchinson, S. (2006). Visual servo control. i. basic approaches. *IEEE Robotics Automation Magazine*, 13(4):82–90.

Chaumette, F. and Hutchinson, S. (2007). Visual servo control. ii. advanced approaches [tutorial]. *IEEE Robotics Automation Magazine*, 14(1):109–118.

Chaumette, F. and Hutchinson, S. (2008). Visual servoing and visual tracking. In *Springer Handbook of robotics*, chapter 24, pages 563–583. Springer-Verlag, London, UK.

Chaumette, F. and Malis, E. (2000). 2 1/2 d visual servoing: a possible solution to improve image-based and position-based visual servoings. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 630–635. IEEE.

Chaumette, F., Malis, E., and Boudet, S. (1997). 2D 1/2 visual servoing with respect to a planar object. In *IEEE International Conference on Robotics*, pages 45–52.

Chaumette, F. and Marchand, T. (2001). A redundancy-based iterative approach for avoiding joint limits: application to visual servoing. *IEEE Transactions on Robotics and Automation*, 17(5):719–730.

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

Chesi, G. (2009). Visual Servoing Path Planning via Homogeneous Forms and LMI Optimizations. *IEEE Transactions on Robotics*, 25(2):281–291.

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

Chesi, G. and Hung, Y. S. (2007). Global path-planning for constrained and optimal visual servoing. *IEEE Transactions on Robotics*, 23(5):1050–1060.

Chesi, G. and Vicino, A. (2004). Visual servoing for large camera displacements. *Robotics, IEEE Transactions on*, 20(4):724–735.

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

Comport, A. I., Marchand, E., and Chaumette, F. (2006). Statistically robust 2D visual servoing. *IEEE Transactions on Robotics*, 22(April):415–420.

Corke, P. (2011). *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer-Verlag, Berlin, Germany.

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

Cowan, N. J., Weingarten, J. D., and Koditschek, D. E. (2002). Visual servoing via navigation functions. *IEEE Transactions on Robotics and Automation*, 18(4):521–533.

Deguchi, K. (1998). Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, volume 2, pages 705–711 vol.2.

Deng, L. and Janabi-Sharifi, F. (2005). Hybrid motion control and planning strategies for visual servoing. *IEEE Transactions on Industrial Electronics*, 52(4):1024–1040.

Deo, A. and Walker, I. (1995). Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent & Robotic Systems*, 14(1):43–68.

Du, G. and Zhang, P. (2013). Online robot calibration based on vision measurement. *Robotics and Computer-Integrated Manufacturing*, 29(6):484–492.

Edwards, C. and Spurgeon, S. (1998). *Sliding Mode Control: Theory and Applications.* Taylor & Francis, UK, 1st edition.

Elena, M., Cristiano, M., Damiano, F., and Bonfe, M. (2003). Variable structure pid controller for cooperative eye-in-hand/eye-to-hand visual servoing. In *Proceedings of 2003 IEEE Conference on Control Applications, 2003. CCA 2003*, volume 2, pages 989–994 vol.2.

Espiau, B., Chaumette, F., and Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326.

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

Feddema, J. and Mitchell, O. (1989). Vision-guided servoing with feature-based trajectory generation (for robots). *IEEE Transactions on Robotics and Automation*, 5(5):691–700.

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

Gans, N. R., Hutchinson, S. a., and Corke, P. I. (2003). Performance Tests for Visual Servo Control Systems, with Application to Partitioned Approaches to Visual Servo Control. *The International Journal of Robotics Research*, 22(10):955–981.

Garcia, G., Pomares, J., and Torres, F. (2009). Automatic robotic tasks in unstructured environments using an image path tracker. *Control Engineering Practice*, 17(5):597–608.

Garcia, G., Pomares, J., Torres, F., and Gil, P. (2014). Event-based visual servoing with features' prediction. In *ROBOT2013: First Iberian Robotics Conference, Advances in Robotics*, pages 679–691, Madrid, Spain. Springer International Publishing.

Garcia-Aracil, N., Malis, E., Aracil-Santonja, R., and Perez-Vidal, C. (2005). Continuous visual servoing despite the changes of visibility in image features. *IEEE Transactions on Robotics*, 21(6):1214–1220.

Garelli, F., Mantz, R., and De Battista, H. (2011). *Advanced Control for Constrained Processes and Systems*. IET, Control Engineering Series, London, UK.

Golub, G. and Van Loan, C. (1996). *Matrix Computations*. The Johns Hopkins University InPress, Baltimore, MD, 3rd edition.

Gordic, Z. and 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., and Sala, A. (2013). Integrated sliding-mode algorithms in robot tracking applications. *Robotics and Computer-Integrated Manufacturing*, 29(1):53–62.

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

Gracia, L., Sala, A., and Garelli, F. (2012b). A supervisory loop approach to fulfill workspace constraints in redundant robots. *Robotics and Autonomous Systems*, 60(1):1–15.

Gracia, L., Sala, A., and Garelli, F. (2014). Robot coordination using task-priority and sliding-mode techniques. *Robotics and Computer-Integrated Manufacturing*, 30(1):74–89.

Hafez, A. H. A., Cervera, E., and Jawahar, C. V. (2008). Hybrid visual servoing by boosting ibvs and pbvs. In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, pages 1–6.

Hafez, A. H. A. and Jawahar, C. (2007). Visual Servoing by Optimization of a 2D/3D Hybrid Objective Function. In *Proceedings 2007 IEEE Int. Conference on Robotics and Automation*, pages 1691–1696. IEEE.

Hafez, A. H. A. and Jawahar, C. V. (2006a). Integration Framework for Improved Visual Servoing in Image and Cartesian Spaces. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2320–2325. Ieee.

Hafez, A. H. A. and Jawahar, C. V. (2006b). Probabilistic Integration of 2D and 3D Cues for Visual Servoing. In *International Conference on Control, Automation, Robotics and Vision*, pages 1–6. IEEE.

Hajiloo, A., Keshmiri, M., Xie, W. F., and Wang, T. T. (2016). Robust Online Model Predictive Control for a Constrained Image-Based Visual Servoing. *IEEE Transactions on Industrial Electronics*, 63(4):2242–2250.

Hammouda, L., Kaaniche, K., Mekki, H., and Chtourou, M. (2015). Robust visual servoing using global features based on random process. *Int. J. Comput. Vision Robot.*, 5(2):138–154.

Hashimoto, K., Ebine, T., and Kimura, H. (1996). Visual servoing with hand-eye manipulator-optimal control approach. *IEEE Transactions on Robotics and Automation*, 12(5):766–774.

Hashimoto, K., Kimoto, T., Ebine, T., and Kimura, H. (1991). Manipulator Control with Image-Based Visual Servo. In *IEEE International Conference on Robotics and Automation*, number April, pages 2267–2272.

Heshmati-alamdari, S., Karavas, G. K., Eqtami, A., Drossakis, M., and Kyriakopoulos, K. J. (2014). Robustness analysis of model predictive control for constrained Image-Based Visual Servoing. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4469–4474. IEEE.

Huang, D. and Xu, J.-X. (2011). Discrete-time adaptive control for nonlinear systems with periodic parameters: A lifting approach. *Asian Journal of Control*.

Huang, Y., Zhang, X., and 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., and Corke, P. I. (1996). A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670.

Janabi-Sharifi, F., Deng, L., and Wilson, W. J. (2011). Comparison of Basic Visual Servoing Methods. *IEEE/ASME Transactions on Mechatronics*, 16(5):967–983.

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

Kermorgant, O. and Chaumette, F. (2011). Combining IBVS and PBVS to ensure the visibility constraint. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2849–2854. IEEE.

Keshmiri, M., Xie, W., and 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. and Dombre, E. (2002). *Modeling, Identification and Control of Robots*. Taylor & Francis Inc., Bristol, PA.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98.

Kim, D., Lovelett, R., Wang, Z., and 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*.

Kim, J., Kim, D., Choi, S., and Won, S. (2006). Image-based visual servoing using sliding mode control. In *Proceedings of the SICE-ICASE International Joint Conference*, pages 4996–5001.

Kragic, D. and Christensen, H. I. (2003). Robust visual servoing. *The International Journal of Robotics Research*, 22(10-11):923–939.

Kragic, D., Christensen, H. I., et al. (2002). Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, 15.

Kunusch, C., Puleston, P., and Mayosky, M. (2012). Fundamentals of sliding-mode control design. In *Sliding-Mode Control of PEM Fuel Cells*, pages 35–71. Springer.

Kyrki, V., Kragic, D., and Christensen, H. I. (2004). New shortest-path approaches to visual servoing.

Lane, D., Trucco, E., et al. (2000). Embedded sonar & video processing for auv applications. In *Offshore Technology Conference*. Offshore Technology Conference.

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer, Boston.

Lee, A. X., Levine, S., and Abbeel, P. (2017). Learning Visual Servoing with Deep Features and Fitted Q-Iteration. *ArXiv e-prints*.

Li, F. and Xie, H.-L. (2010). Sliding mode variable structure control for visual servoing system. *International Journal of Automation and Computing*, 7(3):317–323.

Lots, J.-F., Lane, D., and Trucco, E. (2000). Application of 2 1/2 d visual servoing to underwater vehicle station-keeping. In *Oceans 2000 MTS/IEEE Conference and Exhibition*, volume 2, pages 1257–1264. IEEE.

Lots, J.-F., Lane, D. M., Trucco, E., and Chaumette, F. (2001). A 2d visual servoing for underwater vehicle station keeping. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2767–2772. IEEE.

Malis, E., Chaumette, F., and Boudet, S. (1999). 2 1/2 D visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250.

Mansard, N. and Chaumette, F. (2007). Task sequencing for sensor-based control. *IEEE Trans. on Robotics*, 23(1):60–72.

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

Marks, R. L., Wang, H. H., Lee, M. J., and Rock, S. M. (1994). Automatic visual station keeping of an underwater robot. In *OCEANS'94.'Oceans Engineering for Today's Technology and Tomorrow's Preservation.'Proceedings*, volume 2, pages II–137. IEEE.

Martinet, P. (1999). Comparison of visual servoing techniques: experimental results. In *Proc. of the European Control Conference.*

Mejias, L., Saripalli, S., Campoy, P., and Sukhatme, G. S. (2006). Visual servoing of an autonomous helicopter in urban areas using feature tracking. *Journal of Field Robotics*, 23(3-4):185–199.

Mezouar, Y. and Chaumette, F. (2000). Path planning in image space for robust visual servoing. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 3, pages 2759–2764 vol.3.

Mezouar, Y. and Chaumette, F. (2002). Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549.

Morel, G., Liebezeit, T., Szewczyk, J., Boudet, S., and Pot, J. (2000). *Explicit incorporation of 2D constraints in vision based control of robot manipulators*, pages 99–108. Springer London, London.

Morel, G., Zanne, P., and Plestan, F. (2005). Robust visual servoing: bounding the task function tracking errors. *IEEE Transactions on Control Systems Technology*, 13(6):998–1009.

Motta, J. M. S., de Carvalho, G. C., and 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.

Nakamura, Y. and Hanafusa, H. (1986). Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME Journal of Dynamic Systems, Measurement, and Control*, 108(3):163–171.

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

Negahdaripour, S., Xu, X., and Jin, L. (1999). Direct estimation of motion from sea floor images for automatic station-keeping of submersible platforms. *IEEE Journal of Oceanic Engineering*, 24(3):370–382.

Nelson, B. J. and Khosla, P. K. (1995). Strategies for Increasing the Tracking Region of an Eye-in-Hand System by Singularity and Joint Limit Avoidance. *The International Journal of Robotics Research*, 14(3):255–269.

Nematollahi, E. and Janabi-Sharifi, F. (2009). Generalizations to Control Laws of Image-Based Visual Servoing. *International Journal of Optomechatronics*, 3(3):167–186.

Oliveira, T. R., Leite, A. C., Peixoto, A. J., and Hsu, L. (2014). Overcoming limitations of uncalibrated robotics visual servoing by means of sliding mode control and switching monitoring scheme. *Asian Journal of Control*, 16(3):752–764.

Oliveira, T. R., Peixoto, A. J., Leite, A. C., and Hsu, L. (2009). Sliding mode control of uncertain multivariable nonlinear systems applied to uncalibrated robotics visual servoing. *2009 American Control Conference*, (April 2017):71–76.

Parra-Vega, V., Arimoto, S., Liu, Y.-H., Hirzinger, G., and Akella, P. (2003). Dynamic sliding pid control for tracking of robot manipulators: theory and experiments. *IEEE Transactions on Robotics and Automation*, 19(6):967–976.

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

Parsapour, M. and Taghirad, H. (2015). Kernel-based sliding mode control for visual servoing system. *IET Computer Vision*, 9(3):309–320.

Rimon, E. and Koditschek, D. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518.

Rives, P. and Borrelly, J.-J. (1997). Visual servoing techniques applied to an underwater vehicle. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 1851–1856. IEEE.

Ryan, E. P. and Corless, M. (1984). Ultimate boundedness and asymptotic stability of a class of uncertain dynamical systems via continuous and discontinuous feedback control. *IMA Journal of Mathematical Control and Information*, 1(3):223.

Sadeghzadeh, M., Calvert, D., and Abdullah, H. A. (2015). Self-learning visual servoing of robot manipulator using explanation-based fuzzy neural networks and q-learning. *Journal of Intelligent & Robotic Systems*, 78(1):83–104.

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

Siciliano, B. and 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)*, pages 1211–1216, 1991, Pisa, Italy.

Soetens, P. and Bruyninckx, H. (2005). Realtime hybrid task-based control for robots and machine tools. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 259–264. IEEE.

Solanes, J. E., Armesto, L., Tornero, J., and Girbes, V. (2013). Improving image-based visual servoing with reference features filtering. In *To be published on Int. Conference on Robotics and Automation (ICRA).*

Solanes, J. E., Armesto, L., Tornero, J., Muñoz-Benavent, P., and Girbés, V. (2012). Dual-rate non-linear high order holds for visual servoing applications. In *Conference Towards Autonomous Robotic Systems*, pages 152–163. Springer.

Solanes, J. E., Muñoz-Benavent, P., Girbés, V., Armesto, L., and Tornero, J. (2016). On improving robot image-based visual servoing based on dual-rate reference filtering control strategy. *Robotica*, 34:2842–2859.

Song, X. and Miaomiao, F. (2017). CLFs-based optimization control for a class of constrained visual servoing systems. *ISA Transactions*, 67:507–514.

Tahri, O., Rennes, I. I., and Beaulieu, C. D. (2004). Image Moments : Generic Descriptors for Decoupled Image-based Visual Servo. In *IEEE International Conference on Robotics and Automation*, number April, pages 1185–1190.

Utkin, V., Guldner, J., and Shi, J. (2009). *Sliding Mode Control in Electro-Mechanical Systems*. Taylor & Francis, London, 2nd edition.

Wampler, C. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):93–101.

Weiss, L., Sanderson, A., and Neuman, C. (1987). Dynamic sensor-based control of robots with visual feedback. *IEEE Journal on Robotics and Automation*, 3(5):404–417.

Wilson, W., Williams Hulls, C., and Bell, G. (1996). Relative end-effector control using Cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696.

Xin, J., Ran, B.-J., and Ma, X.-M. (2016). Robot visual sliding mode servoing using SIFT features. In *2016 35th Chinese Control Conference (CCC)*, pages 4723–4729. IEEE.

Yang, Y.-X., Liu, D., and Liu, H. (2002). Robot-self-learning visual servoing algorithm using neural networks. In *Proceedings. International Conference on Machine Learning and Cybernetics*, volume 2, pages 739–742 vol.2.

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

Yu (2013). Stability Analysis of Visual Servoing with Sliding-mode Estimation and Neural Compensation. *International Journal of Control, Automation, and Systems*, 4(5):545–558.

Zanne, P., Morel, G., and Piestan, F. (2000). Robust vision based 3D trajectory tracking using sliding mode control. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and*

Automation. Symposia Proceedings (Cat. No.00CH37065), volume 3, pages 2088–2093. IEEE.

Zhao, Y. M., Lin, Y., Xi, F., Guo, S., and Ouyang, P. (2017). Switch-Based Sliding Mode Control for Position-Based Visual Servoing of Robotic Riveting System. *Journal of Manufacturing Science and Engineering*, 139(4):041010–041010–11.

Zhao, Y.-M., Xie, W.-F., Liu, S., and Wang, T. (2015). Neural network-based image moments for robotic visual servoing. *Journal of Intelligent & Robotic Systems*, 78(2):239–256.

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