Ph.D. Thesis in
Systems Engineering
XX Ciclo

Visual Estimation and Control of Robot Manipulating Systems

Paolo Robuffo Giordano

Supervisor
Prof. Alessandro De Luca

Coordinator
Prof. Carlo Bruni

December 2007

All men by nature desire to know. An indication of this is the delight we take in our senses; for even apart from their usefulness they are loved for themselves; and above all others the sense of sight. For not only with a view to action, but even when we are not going to do anything, we prefer seeing (one might say) to everything else. The reason is that this, most of all the senses, makes us know and brings to light many differences between things.

– Aristotle

Abstract

WITH THIS SENTENCE FROM HIS *Metaphysica*, Aristotle perfectly introduces us to the importance of eyesight for humans, as well as for any advanced living being. Since, to a large extent, robotics is concerned with the emulation of human skills in an artificial context, a natural requirement is to cope with vision for a full interaction with the world. In this respect, this Thesis explores the problem of exploiting visual information to control the motion of robotic systems equipped with onboard cameras. We build our proposals upon the Visual Servoing paradigm, which bridges Computer Vision (image processing, scene interpretation, feature extraction, etc.) with topics proper to the Control Theory field. Indeed, within Visual Servoing, a camera is modeled as a nonlinear function of the scene, i.e., of 3D states subject to rigid body kinematics. Therefore, visual pose control reduces to a problem of *output regulation*, or *task realization* if we conform to the robotic nomenclature. Once this view is adopted, any task realization algorithm can be used to fulfill a visual task, and, more in general, the problem can be tackled with the tools of Control Theory.

In order to fully exploit this formulation, however, a suitable task-oriented modeling of robot manipulators is required. Therefore, in the first part of the Thesis, we develop a theoretical framework for kinematic modeling and control of such systems. While keeping the treatment at the most general level, we place some emphasis on the cases of fixed and nonholonomic mobile manipulators. Indeed, the former class is ubiquitous in robotics, and the latter conveniently merges dexterity with extended ($\sim$ unlimited) workspace capabilities. A special attention is also devoted to the exploitation of possible redundancy with respect to a given task, in terms of both improvement of overall performance, and satisfaction of secondary constraints.

Such modeling framework is then combined with the schemes proper to Visual Servoing to obtain a unique formulation for visual task control. Again, the two cases of fixed-base and nonholonomic mobile manipulators are explicitly considered, and any inherent peculiarity is pointed out. We also show how a suitable use of redundancy can effectively improve the fulfillment of Visual Servoing tasks. For instance, it can allow realization of tasks that would be close to singularity when addressed altogether. Another contribution to this topic is the possibility to estimate online several unmeasurable 3D quantities that are lost through the projective mapping performed by a camera. To this end, a set of estimation tools based on the *nonlinear observation theory* is proposed, so as to recover at runtime any 3D information needed by Visual Servoing schemes. With respect to other possible approximations, this solution takes advantage of the observation convergence in order to improve the overall closed-loop stability.

Finally, the theoretical claims and simulation results presented in the Thesis are further validated by a number of experiments run on real robots equipped with cameras. The proposed methodology is also exploited within an industrial application involving pick-and-assemble tasks. A fixed-base manipulator carrying a camera on the gripper must *(i)* autonomously locate a set of planar parts on a table, *(ii)* pick them up, *(iii)* locate the corresponding holes on a movable plate, and *(iv)* insert the parts. Robot motion during the approaching phases to parts/holes is governed by Visual Servoing techniques, so as to obtain the needed degree of robustness and reactivity with respect to external 'disturbances', such as unexpected displacements of the target items.

Keywords: Vision, Robotics, Kinematic Modeling, Kinematic Control, Visual Servoing, Nonlinear Estimation.

Acknowledgments

By completing this dissertation, my Doctorate takes the final step towards its conclusion. The last three years have been so intense that, sometimes, they seem to have passed in a trice. While conceiving these few words, I realized how many people I owe for their support, so that acknowledging all of them within this limited space is not possible. Therefore, I wish to express my sincere gratitude to all those who have helped me in attaining such a goal.

Besides that, it is my pleasure to thank my supervisor, Prof. Alessandro De Luca, for all that I could learn from his talent and experience, and for all the (quite many) possibilities he gave me during this time. I particularly appreciated the freedom he accorded to my research, and his trust in my capabilities that always encouraged my steps from the very beginning. I am aware that spending these years under his supervision was an invaluable opportunity in my professional and scientific career.

In addition, I also greatly benefited from the constant support of Prof. Giuseppe Oriolo who advised me throughout the Doctorate as if he were my 'second' supervisor. Among the many skills I have gained by working with him, I believe rigorousness and care for the details were the most important for my education. Therefore I wish to thank him for his helpfulness and patience in front of my (sometimes clumsy) research attempts.

The same, of course, goes with Dr. Marilena Vendittelli and Dr. Raffaella Mattone who contributed to 'steer' my footsteps towards the art of doing research in a systematic fashion.

Hearty thanks also go to my colleagues who accompanied me along these years: Dr. Massimo Cefalo, Dr. Luigi Freda, Andrea Pranzo Cherubini, and Antonio Franchi just to cite a few (for the missing ones: ok, I owe you *one* coffee).

During the last year of my Doctorate, I had the opportunity to spend a fruitful stay at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). It is, therefore, my wish to express my full gratitude to Prof. Dr. Gerhard Hirzinger who kindly agreed to host me at his Institute.

A special thank is also meant for Dr. Alin Albu-Schäffer who friendly welcomed and supported me to a great extent during this period, both scientifically and practically.

And I cannot forget the many colleagues met at DLR: Andreas Stemmer, Klaus H. Strobl, Wolfgang Sepp, Roberto Lampariello, Sami Haddadin, Thomas Wimböck, Sebastian Wolf, and many more. The nice atmosphere they shared with me was very enjoyable from all points of view.

Finally, a special mention for Dr. Klaus Arbter who dedicated considerable amounts of his time to teach me the knowledge and expertise gained from his wide experience. He followed me step-by-step and constantly supported me whenever I ran in trouble.

Last, but not least, my gratitude goes to my parents for their 'being always there' regardless of my choices, and for having allowed, with their efforts, to get to where I am now.

Sharing life with someone involved in research activities may be an hard task from time to time. Luckily, if the other one is a researcher too, things can go smoother thanks to the understanding of the mutual needs. For this, and for many other reasons, I am very delighted of having met my wife Lilia and express my gratitude for her patience and love.

Contents

Notations

General rules

Throughout the text, the following conventions hold

- scalar quantities are denoted by plain lower-case symbols (e.g., m, n, t)
- vector quantities, intended as *column vectors* unless otherwise stated, are denoted by bold lower-case symbols (e.g., $\mathbf{u}$, $\mathbf{v}_C$, $\boldsymbol{\omega}_C$)
- matrix quantities are denoted by bold upper-case symbols (e.g., $\mathbf{A}$, $\mathbf{J}$, $\boldsymbol{\Omega}$). The symbol $\mathbf{I}_n$ represents the identity matrix of dimension n
- notation $\widehat{\xi}$ stands for estimation/approximation of quantity ξ
- notation $\bar{g}$ denotes homogeneous representation of g
- a leading superscript (e.g., ${}^{C}T$) specifies the reference frame where a quantity is expressed
- subscript r_d specifies desired value of quantity r
- superscript $r^*(t)$ denotes a planned trajectory for r

Acronyms

w.r.t.	$\longmapsto$	With respect to
lhs (rhs)	$\longmapsto$	Left (right) hand side
iff	$\longmapsto$	If and only if
dof	$\longmapsto$	Degree of freedom
FBM	$\longmapsto$	Fixed-base manipulator
NNM	$\longmapsto$	Nonholonomic mobile manipulator
LWR	$\longmapsto$	Light-weight robot
VS	$\longmapsto$	Visual Servoing
PBVS	$\longmapsto$	Position-based Visual Servoing
IBVS	$\longmapsto$	Image-based Visual Servoing
HVS	$\longmapsto$	Hybrid Visual Servoing

Sets

$\mathbb{N}$	$\longmapsto$	The set of nonnegative integers
$\mathbb{R}$	$\longmapsto$	The set of real numbers
$\mathbb{C}$	$\longmapsto$	The set of complex numbers
$\mathbb{E}^n$	$\longmapsto$	The n-dimensional Euclidean space
$\mathbb{R}^n$	$\longmapsto$	The n-dimensional real linear space
$\mathbb{S}^n$	$\longmapsto$	The unit n-sphere
$\mathcal{E}$	$\longmapsto$	The essential space
$T_p\mathcal{S}$	$\longmapsto$	Tangent space of $\mathcal{S}$ at p
$SO(n)$	$\longmapsto$	Special orthogonal group in $\mathbb{R}^n$
$so(n)$	$\longmapsto$	Lie algebra (tangent space at the identity) of group $SO(n)$
$SE(n)$	$\longmapsto$	Special Euclidean group in $\mathbb{R}^n$
$se(n)$	$\longmapsto$	Lie algebra (tangent space at the identity) of group $SE(n)$
$\mathcal{Q}$	$\longmapsto$	Configuration space of a manipulator locally diffeomorphic to $\mathbb{R}^n$

Vector space operations

$\langle \mathbf{u}, \mathbf{v} \rangle$, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$	$\longmapsto$	Inner (scalar) product of vectors $\mathbf{u}$ and $\mathbf{v}$
$[\mathbf{u}]_\times$, $\mathbf{u} \in \mathbb{R}^3$	$\longmapsto$	The 3×3 skew-symmetric matrix associated to vector $\mathbf{u}$
$\mathbf{u} \times \mathbf{v}$, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$	$\longmapsto$	Cross product of vectors $\mathbf{u}$ and $\mathbf{v}$: $\mathbf{u} \times \mathbf{v} = [\mathbf{u}]_\times \mathbf{v}$
$\nabla_{\mathbf{q}} H(\mathbf{q})$	$\longmapsto$	Gradient of a scalar function $H(\mathbf{q})$ at $\mathbf{q}$
$\mathbf{u} \sim \mathbf{v}$, $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$	$\longmapsto$	Equivalence up to a scalar factor
$\mathbf{A}^T \in \mathbb{R}^{m \times n}$	$\longmapsto$	Transpose of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$
$\mathbf{A}^{-1} \in \mathbb{R}^{n \times n}$	$\longmapsto$	Inverse of a square nonsingular matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
$\mathbf{A}^\dagger \in \mathbb{R}^{m \times n}$	$\longmapsto$	Moore Penrose pseudoinverse of a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$
$\sigma(\mathbf{A})$	$\longmapsto$	Smallest singular value of a matrix $\mathbf{A}$
$\mathbf{A} \lesseqqgtr 0$	$\longmapsto$	(semi-)positive/negative definiteness of matrix $\mathbf{A}$
$\det \mathbf{A} \in \mathbb{R}^{n \times n}$	$\longmapsto$	Determinant of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$
$\operatorname{rank} \mathbf{A}$	$\longmapsto$	Rank of a matrix A
$\ker \mathbf{A}$	$\longmapsto$	Null space of a matrix $\mathbf{A}$
$\operatorname{Im} \mathbf{A}$	$\longmapsto$	Range space of a matrix $\mathbf{A}$
$\operatorname{diag} \mathbf{a}$, $\mathbf{a} \in \mathbb{R}^n$	$\longmapsto$	Square diagonal $n \times n$ matrix having the components of $\mathbf{a}$ on the main diagonal
$\Vert\mathbf{u}\Vert$	$\longmapsto$	The standard vector 2-norm $\sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$
$\Vert\mathbf{A}\Vert$	$\longmapsto$	The standard matrix 2-norm $\max_{\Vert\mathbf{u}\Vert=1} \Vert\mathbf{A}\mathbf{u}\Vert$

Robotics

$\mathbf{q} \in \mathcal{Q}$	$\longmapsto$	Configuration vector of a manipulator
$h(\mathbf{q}) = 0$	$\longmapsto$	Scalar holonomic constraint
$\mathbf{a}_i^T(\mathbf{q})\dot{\mathbf{q}} = 0$	$\longmapsto$	Scalar nonholonomic constraint
$\mathbf{A}^T(\mathbf{q}) \in \mathbb{R}^{k\times n}$	$\longmapsto$	Matrix having vectors $\mathbf{a}_i^T$, $i = 1 \dots k$, as rows
$\mathbf{N}(\mathbf{q}) \in \mathbb{R}^{n\times(n-k)}$	$\longmapsto$	Matrix spanning the null space of $\mathbf{A}^T(\mathbf{q})$
$T(\mathbf{q}, \dot{\mathbf{q}})$	$\longmapsto$	Kinetic energy of a manipulator
$U(\mathbf{q})$	$\longmapsto$	Potential energy of a manipulator
$\mathbf{B}(\mathbf{q}) \in \mathbb{R}^{n\times n}$	$\longmapsto$	The $n \times n$ symmetric and positive definite inertia matrix of the manipulator
$\mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^n$	$\longmapsto$	Vector of Coriolis and centrifugal terms
$\boldsymbol{\lambda} \in \mathbb{R}^n$	$\longmapsto$	Vector of Lagrange multipliers
$\boldsymbol{\tau} \in \mathbb{R}^m$	$\longmapsto$	Vector of external forces/torque performing work on $\mathbf{q}$
$\mathbf{v} \in \mathbb{R}^{n-k}$	$\longmapsto$	Pseudo-velocity vector
$\mathbf{a} \in \mathbb{R}^{n-k}$	$\longmapsto$	Pseudo-acceleration vector
$\mathbf{q}_p \in \mathbb{R}^{n_p}$	$\longmapsto$	Platform configuration variables
$\mathbf{q}_m \in \mathbb{R}^{n_m}$	$\longmapsto$	Manipulator configuration variables
$\mathbf{u}_p \in \mathbb{R}^{n_p-k}$	$\longmapsto$	Platform pseudo-velocity inputs
$\mathbf{u}_m \in \mathbb{R}^{n_m}$	$\longmapsto$	Manipulator velocity inputs
$\mathbf{u} \in \mathbb{R}^{s\times(n_m+n_p-k))}$	$\longmapsto$	Vector of velocity input
$\mathbf{G}(\mathbf{q}_p) \in \mathbb{R}^{n_p\times(n_p-k)}$	$\longmapsto$	Platform kinematic model
$\mathbf{r} \in \mathbb{R}^s$	$\longmapsto$	Task variables
$\mathbf{J}_p(\mathbf{q}) \in \mathbb{R}^{s\times n_p}$	$\longmapsto$	Platform task Jacobian
$\mathbf{J}_m(\mathbf{q}) \in \mathbb{R}^{s\times n_m}$	$\longmapsto$	Manipulator task Jacobian
$\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{s\times(n_m+n_p-k))}$	$\longmapsto$	Robot task Jacobian
$H(\mathbf{q})$	$\longmapsto$	Scalar optimization criterion

3D vision

$\mathcal{F}_O \in SE(3)$	$\longmapsto$	World reference frame
$\mathcal{F}_C \in SE(3)$	$\longmapsto$	Camera reference frame
${}^O\mathbf{T}_{OC} \in \mathbb{R}^3$	$\longmapsto$	Translation vector from the origin of $\mathcal{F}_O$ to the origin of $\mathcal{F}_C$ expressed in $\mathcal{F}_O$
$R_{OC} \in SO(3)$	$\longmapsto$	Rotation matrix representing the orientation of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$
$g_{OC} \in SE(3)$	$\longmapsto$	Rigid displacement from $\mathcal{F}_O$ to $\mathcal{F}_C$
${}^OV_{OC} \in se(3)$	$\longmapsto$	Velocity twist of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$ expressed in $\mathcal{F}_O$
$\mathbf{v}_C \in \mathbb{R}^3$	$\longmapsto$	Linear velocity of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$ expressed in $\mathcal{F}_C$ (camera linear velocity)

$\boldsymbol{\omega}_C \in \mathbb{R}^3$	$\longmapsto$	Angular velocity of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$ expressed in $\mathcal{F}_C$ (camera angular velocity)
p	$\longmapsto$	Image of a 3D point $P \in \mathbb{E}^3$
λ	$\longmapsto$	Distance of the image plane from the camera optical center (focal length)
$\boldsymbol{\Pi}_0 \in \mathbb{R}^{3\times 4}$	$\longmapsto$	Standard projection matrix
$\mathbf{K}_C \in \mathbb{R}^{3\times 3}$	$\longmapsto$	Camera calibration matrix
$\mathbf{E} \in \mathcal{E}$	$\longmapsto$	Essential matrix
π	$\longmapsto$	Target plane in the scene
$\mathbf{n} \in \mathbb{S}^2$	$\longmapsto$	Unit vector normal to the target plane
d	$\longmapsto$	Distance of the target plane from $\mathcal{F}_C$
$\mathbf{H} \in \mathbb{R}^{3\times 3}$	$\longmapsto$	Homography matrix

Visual servoing

$\mathbf{m}(t)$	$\longmapsto$	Vector of image quantities
$\boldsymbol{\chi}(t)$	$\longmapsto$	Vector of 3D quantities
$\mathbf{r} \in \mathbb{R}^s$	$\longmapsto$	Feature vector
$\mathbf{J}_r \in \mathbb{R}^{s\times 6}$	$\longmapsto$	Interaction matrix of $\mathbf{r}$
$\mathcal{F}_{C_d} \in SE(3)$	$\longmapsto$	Camera frame at the desired pose
$\mathbf{J}_p \in \mathbb{R}^{2\times 6}$	$\longmapsto$	Interaction matrix of a feature point p
$\mathbf{f} \in \mathbb{R}^{2k}$	$\longmapsto$	Vector of k feature points p_i
$\mathbf{J}_f \in \mathbb{R}^{2k\times 6}$	$\longmapsto$	Interaction matrix of $\mathbf{f}$
m_{ij}	$\longmapsto$	Image moment of order $i+j$
μ_{ij}	$\longmapsto$	Centered image moment of order $i+j$
n_{ij}	$\longmapsto$	Normalized centered moment of order $i+j$
a	$\longmapsto$	Area of an image region
$(x_g,\, y_g)$	$\longmapsto$	Barycenter of an image region
α	$\longmapsto$	Main orientation of an image region
$\mathcal{M}$	$\longmapsto$	Set of unrealizable image motions
$\mathbf{J}_{\text{img}}$	$\longmapsto$	Image Jacobian
$\mathbf{J}_M$	$\longmapsto$	Robot geometric Jacobian relating velocity inputs $\mathbf{u}$ to the pair $(\mathbf{v}_C,\, \boldsymbol{\omega}_C)$

Nonlinear observation

$\mathbf{x}_m \in \mathbb{R}^n$	$\longmapsto$	Vector of measurable quantities
$\mathbf{x}_u \in \mathbb{R}^l$	$\longmapsto$	Vector of unmeasurable quantities

$\boldsymbol{\xi} \in \mathbb{R}^n$	$\longmapsto$	Error vector $\mathbf{x}_m - \widehat{\mathbf{x}}_m$
$\mathbf{z} \in \mathbb{R}^l$	$\longmapsto$	Error vector $\mathbf{x}_u - \widehat{\mathbf{x}}_u$
$\mathbf{e} \in \mathbb{R}^{n+l}$	$\longmapsto$	Total error vector $\mathbf{e} = [\boldsymbol{\xi}^T \ \mathbf{z}^T]^T$
$\boldsymbol{\Omega}(t) \in \mathbb{R}^{l \times n}$	$\longmapsto$	Persistency of excitation matrix
$\mathbf{A}(t) \in \mathbb{R}^{(n+l)\times(n+l)}$	$\longmapsto$	State matrix of the error dynamics
$\mathbf{g}(\mathbf{e}, t) \in \mathbb{R}^{n+l}$	$\longmapsto$	Perturbation term of the error dynamics

Introduction

EMPOWERING ROBOTS WITH THE ABILITY TO SEE makes it possible to realize advanced tasks like grasping, assembling, exploring, and, more in general, to actively interact with the surrounding environment. In some sense, eyesight can actually shift robot behavior towards higher levels of complexity, thus contributing to fill the gap between machines and human beings. Such a substantial improvement, however, comes at a price: vision provides rich but *unstructured* information, and the extraction of the relevant data hardly ever results in a trivial task. As an order of magnitude of the difficulties behind scene interpretation, it is worth noting that about half of the primate cerebral cortex is devoted to processing visual information [Felleman & Essen 1991]. Hence, it should not be surprising that making a robot see, i.e., soundly interpret a 3D scene, proves to be a major issue in many practical situations. In addition, once the relevant information is available, the problem of exploiting it so as to reach a location, or grasp an item, still remains. Indeed, cameras (and human eyes) perform a nonlinear and non-injective projection from 3D world objects to 2D image quantities. The 'missing dimension' does not allow to instantaneously invert this mapping and to recover full knowledge of the original 3D scene. As a consequence, two objects with the same shape, but different size and depth, may project onto the *very same* 2D image on a camera/eye system — an effect known as *scale ambiguity* [Faugeras 1993].

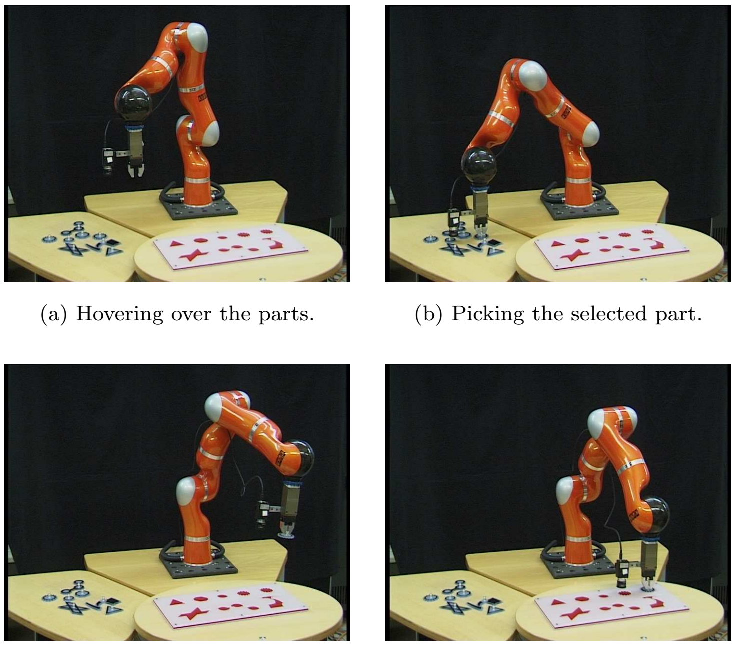
Usually, living beings overcome this problem by exploiting both *stereovision*, and the *context* where images are taken thanks to the experience ripened from daily life. The first solution allows to recover *depth* by comparing two views (from the two eyes) of the same scene, and works well in a close-range setting. More generally, the same principle can be extended to a stream of images taken from a moving viewpoint/target, so that, by exploiting the two dimensional image motion, one can infer information about the 3D structure of the scene as well as its motion relative to the viewer. This possibility, for instance, endows predators with the ability of localizing a moving prey even in presence of big occlusions, cluttered environments, or poor lighting conditions. Besides such 'low-level' strategies, however, the *context* still plays a central role in scene interpretation. Consider two objects of interest, say an apple and a car. Humans are able to *(i)* recognize them as belonging to different classes, and, by relying on internal geometric models built upon prior experience, to *(ii)* conjecture *in advance* the approximate size of apples and cars, and to infer their pose regardless of projective ambiguities.

Unfortunately, such an understanding of the 'external world' is still out of reach for artificial machines. However, the significant improvement experienced over the last decades in the vision hardware is encouraging. Nowadays, commercial cameras can acquire images with a resolution close enough to the one of human eyes at high frame rates (about 30 [Hz]). Furthermore, the ever-growing computational power of standard PCs allows to process the image stream in real-time, and to implement sophisticated algorithms for scene interpretation. Therefore, to some extent, reasonable solutions are at hand, though typically tailored to the specific applications. It is then possible to consider the problem in a more general perspective, i.e., how an artificial machine (a robot) can effectively take advantage of the available (structured) visual information in order to fulfill a given task, such as positioning itself w.r.t. a target object, or grasping items on a table.

In this respect, this dissertation proposes a general framework for exploiting robotic systems equipped with cameras, with the emphasis placed on Control Theory: we assume that the problem of scene interpretation is solved separately, and discuss the algorithms needed to control the motion of a robot by means of visual information. The fundamental idea is to model cameras as sensors yielding a nonlinear output function of 3D states subjected to Euclidean rigid body dynamics, so that visual pose control reduces to an output regulation problem. Taking a parallel with standard robotic nomenclature, image quantities can be regarded as *task variables*, and regulation of their value becomes a *task realization* problem. A natural consequence of this conceptual step is the possibility to adopt any existing task realization algorithm to fulfill a visual task, giving rise to the so-called *Visual Servoing* paradigm, upon which the methods proposed in this Thesis are based. In particular, we address visual pose control from two sides.

First, the proposed 'robotic task' interpretation requires suitable *task-oriented models* for robot manipulators that can act as building blocks for any subsequent visual feedback law. To this end, we develop a general methodology for task-oriented modeling and control of robotic systems, with a particular attention to the cases of fixed-base manipulators and nonholonomic mobile manipulators. Indeed, the latter class of robots is particularly suited for visual tasks such as navigation/exploration or pick and place operations, thanks to its extended ($\sim$ unlimited) workspace capabilities. Special attention is also placed on the exploitation of (possible) redundancy w.r.t. the given task in terms of both execution performance and satisfaction of secondary constraints.

Subsequently, we study how this general framework for task modeling of manipulators can be tailored to the special case of visual tasks. We analyze the merging between cameras, seen as nonlinear output functions, and classical task realization schemes, by pointing out the relevant peculiarities of their coupling, and the possible benefits of a suitable exploitation of redundancy. Considerable effort is also devoted to developing tools for the online estimation of unmeasurable 3D quantities, with the aim of improving the stability and overall performance of Visual Servoing schemes. Indeed, because of the aforementioned projective ambiguities, a direct measurement of, e.g., depth or other related 3D quantities is not possible by processing just one image at a time, unless special hypotheses on the scene are assumed. In this respect, we propose a set of techniques borrowed from the *nonlinear observation theory* to estimate at runtime the missing 3D information, so as to make it available to any Visual Servoing feedback law.



(a) Hovering over the parts.

(b) Picking the selected part.

(c) Hovering over the plate.

(d) Inserting the part.

Figure 1: Screenshots of the industrial application developed during this Thesis at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). The 3-rd generation DLR light-weight robot, equipped with an onboard camera, must locate complex planar parts (Fig. (a)), pick them (Fig. (b)), locate the corresponding hole on a movable plate (Fig. (c)), and insert them (Fig. (d)). Motion during the approaching phases to parts/holes is governed by Visual Servoing feedback laws.

Finally, we present a thorough experimental campaign, run on fixed and mobile manipulators equipped with cameras, whose goal is to validate both the theoretical claims and simulation results given throughout the Thesis. These results are then further exploited within a potential industrial application involving a robot manipulator committed with the task of looking for given parts on a table, picking them, and assembling them inside the corresponding holes on a target movable plate (Figs. 1(a–d)). In this context, robot pose control during the approaching phases to parts/holes is realized via Visual Servoing techniques. This choice, indeed, guarantees the needed robustness and reactivity for a sound task fulfillment in realistic conditions, such as, for instance, possible displacements of parts/plate during the execution of the task.

Organization of the Thesis

This Thesis is composed of three main parts according to the conceptual subdivision mentioned before, i.e., *(i)* modeling of fixed/mobile manipulators, *(ii)* development of theoretical tools for robot visual control, and *(iii)* discussion of experimental results — Fig. 2 shows a logical scheme. Within each part, Chapters represent self-consistent modules with their own introductive sections and overviews. Any dependency among units is explicitly reported throughout the text to the best of the author's possibilities.

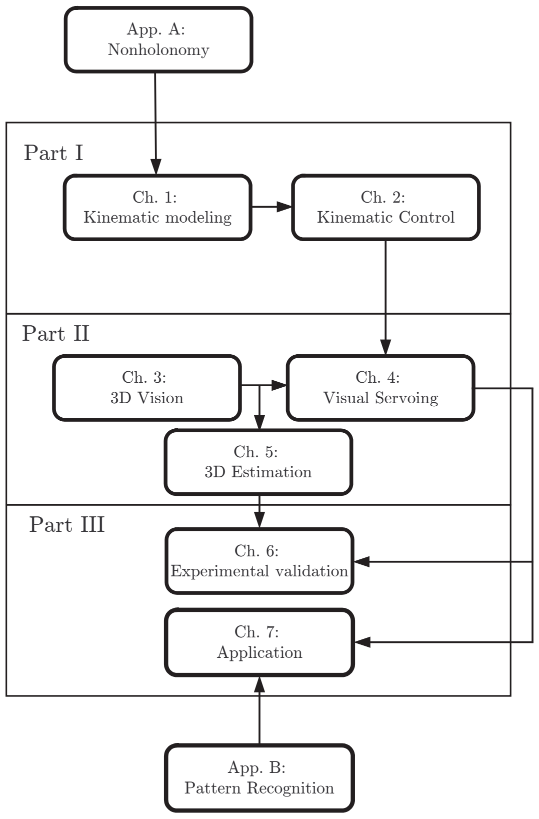


Figure 2: Logical scheme of parts and Chapters of the Thesis.

Outline of part I

In the first part, a general framework for task-oriented modeling of robot manipulators is proposed.

Chapter 1 introduces the so-called *kinematic modeling* approach, which consists in separating, from a control point of view, the kinematic properties of a manipulator from its dynamical behavior. As a result, one can solve motion control of a manipulator directly at the kinematic (first-order) level by assuming velocities as actual inputs, while a lower-level controller is in charge of realizing the desired velocity profiles by means of a suitable torque-level feedback. By exploiting this separation, it is then possible to obtain a *task-oriented kinematic model* for any given task that can be expressed as a function of the state of the manipulator. This methodology, originally proposed for standard fixed-base manipulators, is here extended to also cover the case of nonholonomic mobile manipulators. The key idea is to implicitly take into account the nonholonomic constraints due to the presence of wheels into the resulting kinematic model, so that a formulation formally equivalent to the fixed-base case can be obtained. For the interested reader, Appendix A further delves into the theoretical background of nonholonomy.

Chapter 2 builds upon the proposed task-oriented kinematic modeling to illustrate several *motion generation* and *redundancy resolution* techniques proper to the *kinematic control* approach, and at the core of any subsequent visual feedback design. The formulation is intentionally kept at the most general level, but any relevant discrepancy between fixed-base and nonholonomic cases is thoroughly discussed. The methods are both presented from a theoretical point of view, and assessed by simulations.

Part of this material has been published as author's original work in [De Luca *et al.* 2006, 2007a].

Outline of part II

Part II takes the reader to the heart of the Thesis, i.e., the development of suitable tools for visual control of robot manipulators.

Chapter 3 consists in an introductory step where basic notions of rigid body kinematics and geometry of 3D vision are recalled in a compact way. The aim is to provide the theoretical background needed in the following Chapters.

Chapter 4 introduces the Visual Servoing paradigm upon which all the proposed visual feedback laws are based. Visual Servoing is first presented through an overview of both its historical perspective and current state of the art. The three main existing approaches, namely *Position-based*, *Image-based*, and *Hybrid*, are illustrated and compared in terms of stability, final achievable accuracy and overall performance. A special emphasis is set on the Image-based case because of its easiness of implementation and robustness w.r.t. uncertainties/calibration errors. Then, the merging of Visual Servoing schemes within the framework of task-oriented modeling proposed in the first Chapters is considered. Again, the special classes of fixed-base and nonholonomic mobile manipulators are explicitly addressed, and the peculiarities arising from the coupling of Visual Servoing/task-oriented modeling are thoroughly analyzed. In the last sections, the proposed theoretical framework is validated through extensive simulations.

Chapter 5 tackles the problem of 3D structure identification from motion. As explained before, the processing of single images does not allow to recover certain 3D information lost during the perspective projection mapping. However, some of these quantities are required by Visual Servoing schemes and must, then, be approximated/recovered in actual implementations. To this end, the Chapter proposes an estimation framework based on the theory of *nonlinear observation*, and in particular built upon the *persistency of excitation* Lemma. The outcome of this analysis is a set of tools conceived for an online converging estimation of the unmeasurable 3D data needed by Visual Servoing algorithms. As a byproduct, the same estimation methodology is also tailored to the online observation of the camera focal length. Numerical simulations are finally discussed in order to evaluate the performance of the estimation tools presented in the Chapter.

The ideas and methods presented in these Chapters have been published as author's original work in [De Luca *et al.* 2007a,b; Robuffo Giordano *et al.* 2008].

Outline of part III

The last part of the Thesis is devoted to the experimental validation of the theoretical tools and simulation results presented throughout the previous Chapters.

Chapter 6 provides an extensive experimental campaign addressing: *(i)* the merging of task-oriented modeling and Visual Servoing schemes, *(ii)* the exploitation of redundancy to improve the overall performance of task execution as well as the satisfaction of secondary constraints, and *(iii)* the benefits, in terms of stability, of plugging the proposed 3D observation tools into a Visual Servoing loop. The experiments are run on nonholonomic mobile robots equipped with onboard actuated cameras, so that the overall design falls into the class of nonholonomic mobile manipulators.

Chapter 7 illustrates a potential industrial application which takes advantage of the methods developed in this Thesis. A pick-and-assemble sequence of operations, involving small parts with complex geometry, is executed by the 3-rd generation DLR light-weight robot, a fixed-base manipulator equipped with a gripper and a camera attached to it. The robot must autonomously locate the given parts on a table, pick them up, locate the corresponding hole on a movable table, and insert them. In order to achieve the needed robustness w.r.t. disturbances, and reactivity w.r.t. intentional displacements of the target items, motion of the robot is governed by Visual Servoing laws when approaching parts/holes. The experimental results reported at the end of the Chapter show the effectiveness of the adopted approach in terms of motion performance and reliability over extended operation. Additional details are also given in Appendix B, where the algorithm for pattern classification chosen for this application is illustrated.

The experimental results presented in these Chapters have been submitted as author's original work in [De Luca *et al.* 2008a,b; Robuffo Giordano *et al.* 2008].

Part I

Modeling of Robot Manipulators

1

Kinematic Modeling of Robot Manipulators

In this Chapter, we consider the derivation of suitable kinematic models for fixed-base manipulators (FBMs) and nonholonomic mobile manipulators (NMMs). The analysis is carried out from a general point of view with the aim of obtaining a common background of theoretical tools for both classes of robots, so that any subsequent development can be tailored to the specific case with minimum effort. The proposed methodology is at the core of all the kinematic inversion, redundancy resolution, and kinematic control techniques presented and discussed in the following Chapters.

The chapter is structured as follows: first, an overview of kinematic modeling for FBMs and NMMs is given in Sect. 1.1. Next, Sect. 1.2 details the steps needed to obtain the separation between kinematic and dynamic levels entailed by kinematic control approaches. Finally, Sect. 1.3 introduces the general methodology for task-oriented kinematic model design which is used throughout the rest of this work.

1.1 Introduction

1.1.1 Behind kinematic modeling

A robot manipulator, either with fixed or mobile base, consists in a mechanical structure with actuated degrees of freedom. This implies that, from the control point of view, the available manipulator inputs (actuator forces/torques) always act on the system behavior at a second-order (acceleration) level. The state-space model representing the differential link between actuator efforts and system motion is given by the *dynamics* of a robot manipulator [Sciavicco & Siciliano 2000; Murray *et al.* 1994]. Therefore, whatever the task assigned to the robot, any control algorithm must eventually cope with the differential structure entailed by the robot dynamics, and yield a suitable force/torque signal to be sent to the joint actuators. In many cases, however, it is possible to decompose the control design in two distinct phases, one dealing with the manipulator (first-order) kinematics, and the other accounting for the remaining dynamical issues, so that the overall design is considerably facilitated. The core idea is that, for a fully actuated mechanical system, any differentiable velocity profile can be realized by an appropriate force/torque controller. It is, therefore, possible to proceed as explained: first, system velocities are assumed as inputs and a first-order control law is designed on the basis of a suitable differential kinematic model, i.e., by neglecting the manipulator dynamic properties. Then, realization of these reference velocities is obtained by means of a separate controller in charge of computing the needed dynamic level inputs. Under the assumption that the momentum of the manipulator is limited (i.e., its velocity and/or inertia are low), this choice will typically result in reasonable actuator demands.

This design approach, commonly known as *kinematic control*, motivates the study of suitable kinematic models that can fully represent the key first-order (velocity) properties of the robot. When considering a FBM, this derivation has a trivial solution. Indeed, being fixed-base manipulators kinematically unconstrained systems, any arbitrary joint velocity can be imposed at each configuration, and the kinematic model reduces to a single integrator for each actuated joint. In the mobile manipulator case, on the other hand, things can get more complicated because of the possible motion restrictions (nonholonomy) that can affect the mobile platform carrying the manipulator arm. Therefore, special care is required when deriving kinematic models for these systems, and many solutions have been proposed to deal with different types of mobile platforms. The next section gives an overview of the existing literature with a special attention to the NMM case.

1.1.2 Mobile manipulators

A mobile manipulator consists of an articulated arm mounted on a mobile platform. Since this mechanical arrangement combines the dexterity of the former with the workspace extension of the latter, it is clearly appealing for many applications, and in particular for service robotics [Arai 1996; Nenchev *et al.* 1992]. In general, the platform may have different modes of locomotion, e.g., wheeled bases, walking robots, free-flying space robots. Whenever the platform kinematics falls into the class of holonomic systems, like in [Seraji 1993a,b, 1998; Khatib 1991], kinematic modeling and control of the whole robot is conceptually equivalent to the FBM case: the platform actuated dofs act as additional manipulator joints, and standard techniques can be directly applied. In the NMM case, on the other hand, because of the platform nonholonomic nature, some care is required when dealing with modeling and control issues, and most existing techniques need a suitable and nontrivial extension.

A large class of wheeled vehicles are subject to nonholonomic constraints arising from the rolling without slipping of the wheels on the ground [Campion *et al.* 1996]. These constraints restrict the admissible generalized velocities and thus the instantaneous mobility of the platform, but do not affect the global accessibility of its configuration space, which is still guaranteed through suitable maneuvering with a reduced set of independent velocity commands (the so-called pseudovelocities). Several methods have been proposed in the literature for planning and controlling the motion of these robotic systems, see, e.g., [Li & Canny 1993; Laumond 1998]. An NMM is obtained when the platform providing mobility to the manipulator base is a nonholonomic vehicle. The resulting structure clearly inherits the constraint on the admissible platform velocities – instantaneous mobility in the configuration space is still limited. However, the number of degrees of freedom (dofs) is also increased, typically yielding kinematic redundancy with respect to standard tasks, such as end-effector placement. This means that in general the instantaneous mobility is not constrained at the task level, in spite of the nonholonomic nature of the system. Moreover, the extra dofs can be used to optimize performance criteria and/or to perform auxiliary operations in addition to the primary task [Nakamura 1991].

Kinematic modeling and motion generation for NMMs has been addressed in the literature following two basic approaches. Some authors add the nonholonomic constraints in the description of the differential kinematics [Pin *et al.* 1996; Seraji 1998]. A more efficient formulation, adopted also in this work, explicitly entails the differential motions that are feasible w.r.t. the nonholonomic constraints [Gardner & Velinsky 2000; Fourquet *et al.* 2003]. The problem is usually tackled at a first-order kinematic level (i.e., with (pseudo-)velocities as command inputs) since planning the motion of NMMs is essentially a kinematic problem, but inclusion of dynamic aspects has also been considered [Yamamoto & Yun 1999]. When an NMM is redundant for a given task (this concept requires some caution, see Sect. 2.2), redundancy can be exploited by extending schemes already available for standard manipulators, e.g., task space augmentation [Seraji 1998; Lamiraux *et al.* 2002] or pseudoinversion of the Jacobian and use of its null-space motion [Bayle *et al.* 2001]. Chpater 2 illustrates in detail the exploitation of such techniques for FBMs and NMMs.

1.2 From dynamics to kinematics

Let $\mathbf{q} = [q_1 \dots q_n]^T \in \mathcal{Q}$ be the vector of generalized coordinates representing the configuration of a mechanical system, where the configuration space $\mathcal{Q}$ is assumed to be a n-dimensional smooth manifold locally diffeomorphic to $\mathbb{R}^n$. As an example, the configuration space of a manipulator made of n rotating joints is the n torus $\mathbb{T}^n = \underbrace{\mathbb{S}^1 \times \dots \times \mathbb{S}^1}_{n}$, with $\mathbb{S}^1$ being the unit circle. Furthermore, let $T_\mathbf{q}\mathcal{Q}$ be the tangent space of $\mathcal{Q}$ at $\mathbf{q}$. The generalized velocity of a mechanical system at a generic point of a trajectory $\mathbf{q}(t)$ is a vector $\dot{\mathbf{q}} = [\dot{q}_1 \dots \dot{q}_n]^T \in T_\mathbf{q}\mathcal{Q}$.

A mechanical system may be subjected to k_1 scalar geometric constraints $h_i(\mathbf{q}) = 0$, $i = 1 \dots k_1$, that define a suitable $(n - k_1)$-dimensional submanifold of $\mathcal{Q}$ where the trajectories of $\mathbf{q}(t)$ are confined. This kind of constraints is called 'holonomic'. In addition, a set of k_2 *Pfaffian*[1] first-order constraints (linear in $\dot{\mathbf{q}}$ and time-invariant)

$$\mathbf{a}_i^T(\mathbf{q})\dot{\mathbf{q}} = 0, \quad i = 1 \dots k_2, \tag{1.1}$$

may also exist. If constraints (1.1) are not integrable, i.e., cannot be reduced to equivalent holonomic constraints $h_i(\mathbf{q})$ such that $\partial h_i(\mathbf{q})/\partial \mathbf{q} = \mathbf{a}_i^T(\mathbf{q})$, they are denoted as 'nonholonomic' [Murray *et al.* 1994]. In contrast to the former case, these constraints do not affect the possible motion of $\mathbf{q}(t)$, but restrict the instantaneous velocity of the system to a suitable $(n - k_2)$-dimensional subspace of $T_\mathbf{q}\mathcal{Q}$. As a consequence, the number of system inputs is also reduced by the same amount k_2. Nonholonomic constraints arise in a variety of situations, like floating systems in which the angular momentum is conserved, or bodies that are in rolling contact without slipping. The latter is the case of many wheeled mobile robotic systems like, e.g., the NMMs considered in this work. A more detailed treatment of nonholonomy, including tools to determine when constrains (1.1) are integrable, is given in Appendix A.

Let $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q})$ be the *Lagrangian* of the considered mechanical system, with $T(\mathbf{q}, \dot{\mathbf{q}})$ and $U(\mathbf{q})$ being the scalar kinetic and potential energy functions, respectively. For a robot manipulator, the Lagrangian takes the specific form

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{B}(\mathbf{q})\dot{\mathbf{q}} - U(\mathbf{q}), \tag{1.2}$$

where $\mathbf{B}(\mathbf{q})$ is the positive definite $n \times n$ manipulator *inertia matrix*. Assume that a set of k

[1]In principle, a generic constraint $\mathbf{a}_i^T(\mathbf{q}, \dot{\mathbf{q}}, t) = 0$ is possible, but in this work only the Pfaffian case is considered.

nonholonomic constraints of the form (1.1) exist, and collect them in the compact form

$$\mathbf{A}^T(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{a}_1^T(\mathbf{q}) \\ \vdots \\ \mathbf{a}_k^T(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{0}, \quad \mathbf{A}(\mathbf{q}) \in \mathbb{R}^{n\times k}. \tag{1.3}$$

The Euler-Lagrange equations of the constrained systems are

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \mathbf{q}}\right)^T = \mathbf{A}(\mathbf{q})\boldsymbol{\lambda} + \mathbf{S}(\mathbf{q})\boldsymbol{\tau}, \tag{1.4}$$

where $\mathbf{S}(\mathbf{q})$ is a $n \times m$, $m = n - k$, matrix mapping the m independent inputs $\boldsymbol{\tau}$ into forces/torques performing work on the generalized coordinates $\mathbf{q}$, and $\boldsymbol{\lambda} \in \mathbb{R}^k$ is the vector of *Lagrange multipliers* associated to the k constraints in (1.3). By plugging (1.2) into (1.4) and using (1.3), we obtain the *dynamic model* of the manipulator subject to nonholonomic constraints

$$\begin{aligned} \mathbf{A}^T(\mathbf{q})\dot{\mathbf{q}} &= \mathbf{0} \\ \mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) &= \mathbf{A}(\mathbf{q})\boldsymbol{\lambda} + \mathbf{S}(\mathbf{q})\boldsymbol{\tau} \end{aligned} \tag{1.5}$$

with

$$\mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) = \dot{\mathbf{B}}(\mathbf{q})\dot{\mathbf{q}} - \frac{1}{2}\frac{\partial}{\partial \mathbf{q}}(\dot{\mathbf{q}}^T\mathbf{B}(\mathbf{q})\dot{\mathbf{q}}) + \frac{\partial U(\mathbf{q})}{\partial \mathbf{q}}$$

representing the Coriolis, centrifugal and gravity terms.

Formulation (1.5) describes the system dynamics by explicitly incorporating the nonholonomic constraints. For control design purposes, however, it is often convenient to rewrite (1.5) in an equivalent reduced state-space form where constraints (1.3) are implicitly included into the system dynamics. To this end, let $\mathbf{N}(\mathbf{q})$ be a $n \times m$ matrix spanning the null-space of $\mathbf{A}^T(\mathbf{q})$, i.e.,

$$\mathbf{A}^T(\mathbf{q})\mathbf{N}(\mathbf{q}) = \mathbf{0}. \tag{1.6}$$

Since the k constraints (1.3) are assumed to be independent, $\mathbf{N}(\mathbf{q})$ has full rank m for every $\mathbf{q}$. It is, then, possible to rewrite the first row of (1.5) as

$$\dot{\mathbf{q}} = \mathbf{N}(\mathbf{q})\mathbf{v}, \quad \mathbf{v} \in \mathbb{R}^m, \tag{1.7}$$

i.e., in terms of the (feasible) system velocities $\dot{\mathbf{q}}$ that meet constraint (1.3). This set of n first-order differential equations can be considered as the kinematic model of the manipulator, and vector $\mathbf{v}$ plays the role of (pseudo-)velocity input. Note that, as expected, $\mathbf{v}$ has dimension $m < n$. By multiplying the second row of (1.5) by $\mathbf{N}^T(\mathbf{q})$, the Lagrange multipliers are eliminated and a reduced set of m differential equations is obtained

$$\mathbf{N}^T(\mathbf{q})(\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}})) = \mathbf{N}^T(\mathbf{q})\mathbf{S}(\mathbf{q})\boldsymbol{\tau}. \tag{1.8}$$

Equations (1.7) and (1.8) can be rearranged as

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{N}(\mathbf{q})\mathbf{v} \\ \mathbf{M}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{m}(\mathbf{q},\, \mathbf{v}) &= \mathbf{N}^T(\mathbf{q})\mathbf{S}(\mathbf{q})\boldsymbol{\tau}, \end{aligned} \tag{1.9}$$

with $\mathbf{M}(\mathbf{q}) = \mathbf{N}^T(\mathbf{q})\mathbf{B}(\mathbf{q})\mathbf{N}(\mathbf{q}) > 0$, and $\mathbf{m}(\mathbf{q},\, \mathbf{v}) = \mathbf{N}^T(\mathbf{q})\mathbf{B}(\mathbf{q})\dot{\mathbf{N}}(\mathbf{q})\mathbf{v} + \mathbf{N}^T(\mathbf{q})\mathbf{n}(\mathbf{q},\, \mathbf{N}(\mathbf{q})\mathbf{v})$. System (1.9), of dimension $n + m = 2n - k$, is the reduced state-space dynamic model of the constrained manipulator. Note that, if no constraints are present, $\mathbf{A}^T(\mathbf{q}) = \mathbf{0}$, $\mathbf{N}(\mathbf{q}) = \mathbf{I}_n$, and the standard $2n$ state-space dynamic model of an unconstrained manipulator is recovered. Moreover, the structure of (1.9) clearly exhibits the aforementioned separation between kinematics and dynamics upon which the kinematic control approach is built. Indeed, the first row of (1.9) (n first-order equations) is only representative of the manipulator kinematic properties, including nonholonomy, while all the dynamical effects are shifted to the second row (m second-order equations). Under the hypothesis that $\det \mathbf{N}^T(\mathbf{q})\mathbf{S}(\mathbf{q}) \neq 0$, it is possible to completely remove any dynamic parameter from (1.9) via the nonlinear state feedback law $\boldsymbol{\tau} = (\mathbf{N}^T(\mathbf{q})\mathbf{S}(\mathbf{q}))^{-1}(\mathbf{M}(\mathbf{q})\mathbf{a} + \mathbf{m}(\mathbf{q},\, \mathbf{v}))$, where $\mathbf{a} \in \mathbb{R}^m$ is a vector of (pseudo-)accelerations. This choice yields the equivalent system

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{N}(\mathbf{q})\mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{a}, \end{aligned} \tag{1.10}$$

consisting of the first-order manipulator kinematic model coupled with a dynamic extension on the (pseudo)-velocity inputs. Note that, for an unconstrained manipulator, this result would be equivalent to what is obtained with the *computed torque* approach [Murray *et al.* 1994; Sciavicco & Siciliano 2000]. Indeed, in the absence of nonholonomic constraints, (1.10) further simplifies to a set of n double integrators $\ddot{\mathbf{q}} = \mathbf{a}$ as in that case.

Formulation (1.10) can be fully exploited for kinematic control purposes. Assume, for instance, that a (differentiable) reference velocity command $\mathbf{v}_d(t)$ is able to realize a given task at the first-order kinematic level, i.e., by only considering the first row of (1.10). Asymptotic tracking of $\mathbf{v}_d(t)$ can be easily obtained by means of the acceleration command $\mathbf{a} = \dot{\mathbf{v}}_d + \mathbf{K}(\mathbf{v}_d - \mathbf{v})$, $\mathbf{K} > 0$, with convergence rate tuned by $\mathbf{K}$. If $\mathbf{K}$ is big enough, any transient error between $\mathbf{v}$ and $\mathbf{v}_d$ can be neglected, so that, with an arbitrary degree of approximation, $\mathbf{v}$ can be considered as the 'actual' control input of the system as required by the kinematic control framework.

1.3 Task-oriented kinematic modeling

Consider a robot manipulator with configuration vector $\mathbf{q} \in \mathcal{Q}$ subjected to k nonholonomic constraints (1.3). From (1.10), the manipulator kinematic model takes the general expression

$$\dot{\mathbf{q}} = \mathbf{N}(\mathbf{q})\mathbf{u}, \tag{1.11}$$

where[2] $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{N}(\mathbf{q}) \in \mathbb{R}^{n\times m}$, and $m = n - k$. If $k = 0$, as in the FBM case, $\mathbf{N}(\mathbf{q}) = \mathbf{I}_n$ and (1.11) trivially simplifies to

$$\dot{\mathbf{q}} = \mathbf{u}, \tag{1.12}$$

i.e., any joint velocity can be specified at each arm configuration. In the NMM case, on the other hand, $k > 0$, and a more specific expression for $\mathbf{N}(\mathbf{q})$ can be obtained by exploiting the particular robotic structure considered. To this end, reorder vector $\mathbf{q}$ as $\mathbf{q} = [\mathbf{q}_p\ \mathbf{q}_m]^T$, where $\mathbf{q}_p \in \mathbb{R}^{n_p}$ and $\mathbf{q}_m \in \mathbb{R}^{n_m}$ are the generalized coordinates of the platform and the manipulator, respectively, and let $\mathbf{u} = [\mathbf{u}_p\ \mathbf{u}_m]^T$ represent the (pseudo)-velocity inputs of both components. Since only the platform is affected by the nonholonomic constraints, (1.3) can be rearranged as

$$\mathbf{A}^T(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{a}^T_{1_p}(\mathbf{q}_p) & 0 \\ \vdots & \vdots \\ \mathbf{a}^T_{k_p}(\mathbf{q}_p) & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_m \end{bmatrix} = \mathbf{0},$$

so that the feasible NMM velocities result in

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\mathbf{q}}_p \\ \dot{\mathbf{q}}_m \end{bmatrix} = \begin{bmatrix} \mathbf{G}(\mathbf{q}_p) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n_m} \end{bmatrix} \begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_m \end{bmatrix} = \mathbf{N}(\mathbf{q}_p) \begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_m \end{bmatrix}, \tag{1.13}$$

where $\mathbf{u}_p \in \mathbb{R}^p$, $p = n_p - k$, $\mathbf{u}_m \in \mathbb{R}^{n_m}$, and the $n_p \times p$ matrix $\mathbf{G}(\mathbf{q}_p)$ spans the null-space of $[\mathbf{a}_{1_p}(\mathbf{q}_p) \ldots \mathbf{a}_{k_p}(\mathbf{q}_p)]^T$. Note again that, in this case, the number of velocity inputs $p + n_m = n - k = m$ is less than the number of generalized coordinates n because of the presence of k nonholonomic constraints on the platform kinematics.

Assume that the task is described in terms of a vector $\mathbf{r}$ of s scalar variables. Possible choices for $\mathbf{r}$ include the position/orientation of the manipulator end-effector in a positioning task, or the location of image features in a visual task. The task variables $\mathbf{r}$ are related to the configuration variables $\mathbf{q}$ of the robot by the kinematic map

$$\mathbf{r} = \mathbf{f}(\mathbf{q}),\ \mathbf{r} \in \mathbb{R}^s. \tag{1.14}$$

By differentiating (1.14) w.r.t. time, and using (1.11), we get

$$\dot{\mathbf{r}} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}\dot{\mathbf{q}} = \mathbf{J}_t(\mathbf{q})\mathbf{N}(\mathbf{q})\mathbf{u} = \mathbf{J}(\mathbf{q})\mathbf{u}. \tag{1.15}$$

Equation (1.15) is the task-oriented kinematic model of the manipulator, relating the instantaneous mobility of whole robot to the velocity of the task variables. The $s \times m$ matrix $\mathbf{J}$ in (1.15) will be simply referred as the *FBM/NMM task Jacobian* for the given task also if, strictly speaking, some elements of $\mathbf{J}$ may not be partial derivatives, due to the possible

[2]Here, symbol $\mathbf{u}$ is used in place of $\mathbf{v}$ to stress that (pseudo-)velocities are regarded as control inputs.

nonholonomic constraint entailed by (1.11). By plugging (1.12) or (1.13) in (1.15), it follows that, for a FBM, it is $\mathbf{J}(\mathbf{q}) = \mathbf{J}_t(\mathbf{q})$, while, for an NMM, (1.15) becomes

$$\dot{\mathbf{r}} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}_p}\mathbf{G}(\mathbf{q}_p)\mathbf{u}_p + \frac{\partial \mathbf{f}}{\partial \mathbf{q}_m}\mathbf{u}_m = \mathbf{J}_p(\mathbf{q})\mathbf{G}(\mathbf{q}_p)\mathbf{u}_p + \mathbf{J}_m(\mathbf{q})\mathbf{u}_m. \tag{1.16}$$

Note that, with this formulation all classical problems addressed for standard redundant manipulators (study of singularities and their avoidance, augmentation of tasks and their priority [Chiacchio *et al.* 1991], optimization of performance criteria [Nakamura 1991], cyclicity of configuration motion [Shamir & Yomdin 1988; De Luca *et al.* 1992], etc.) can be directly reformulated also for NMMs in terms of the Jacobian $\mathbf{J}$ in (1.15) with $\mathbf{N}(\mathbf{q})$ chosen as in (1.13). For example, a configuration $\bar{\mathbf{q}}$ is singular iff rank $J(\bar{\mathbf{q}}) < s$, and so on. The following Chapter illustrates how kinematic inversion, redundancy resolution and kinematic control techniques can be adapted to cope with the framework proposed in the previous sections.

2
Kinematic Control of Robot Manipulators

There are two conceptually different problems that can be formulated on the basis of the task differential kinematics (1.15). The first is kinematic inversion, i.e., generating velocity commands that realize an assigned task trajectory, provided that the initial task error is zero. If the manipulator is kinematically redundant with respect to a given task, a redundancy resolution problem has to be addressed. Many solutions have been proposed in the literature to deal with redundancy resolution issues for FBMs, see [Nakamura 1991] for an overview. More recently, suitable extensions to to the NMM case have also been considered like task space augmentation [Seraji 1998; Lamiraux *et al.* 2002], or local optimization of cost functions [Bayle *et al.* 2001; De Luca *et al.* 2006, 2007a]. The second use of the task differential kinematics is to devise kinematic control schemes. The velocity commands, actually considered as control inputs, are designed so as to achieve trajectory tracking or set-point regulation at the task level by combining a feedforward and a feedback action. The presence of the latter allows to recover nonzero initial task errors as well as counteract model perturbations, including numerical drifts [Chiacchio *et al.* 1991].

The rest of the Chapter is organized as follows: Sect. 2.1 and Sect. 2.2 illustrate task realization techniques for the non-redundant and redundant case, respectively, and Sect. 2.4 presents a collection of simulation results for two case studies.

2.1 The non-redundant case

Consider first the case $m = s$, in which the manipulator Jacobian is square. Outside singularities, we can realize any task velocity by setting

$$\mathbf{u} = \mathbf{J}^{-1}(\mathbf{q})\dot{\mathbf{r}}. \tag{2.1}$$

The problem reduces then to choosing $\dot{\mathbf{r}}$ in such a way that the control problem is solved.

For a tracking problem, one simply sets in (2.1)

$$\dot{\mathbf{r}} = \dot{\mathbf{r}}_d + \mathbf{K}(\mathbf{r}_d - \mathbf{r}) = \dot{\mathbf{r}}_d + \mathbf{K}\mathbf{e}, \tag{2.2}$$

where $\mathbf{K} > 0$ is a (diagonal) gain matrix and $\mathbf{e} \in \mathbb{R}^s$ is the task error. For a regulation problem, where $\dot{\mathbf{r}}_d = 0$, we let

$$\dot{\mathbf{r}} = \mathbf{K}\mathbf{e}. \tag{2.3}$$

In both cases, the closed-loop system is described by

$$\dot{\mathbf{e}} = -\mathbf{K}\mathbf{e}, \tag{2.4}$$

so that exponential convergence to zero is achieved for each component of the task error.

For regulation tasks it is also possible to use, in place of (2.1) and (2.3), the computationally lighter expression

$$\mathbf{u} = \mathbf{J}^T(\mathbf{q})\mathbf{K}\mathbf{e}, \tag{2.5}$$

which still guarantees asymptotic (but no longer exponential) convergence of the task error to zero [Sciavicco & Siciliano 2000].

2.2 The redundant case

Assume now that $m > s$, so that the manipulator Jacobian matrix has more columns than rows. Since the number of velocity commands (i.e., of dofs) exceeds the dimension of the task, we say that the manipulator is *kinematically redundant* with respect to the given task. Note that, one may also define a *static* redundancy property, which occurs when the number of generalized coordinates of the manipulator exceeds the dimension of the task ($n > s$). The two redundancy concepts collapse for FBMs since, in this case, $n = m$, but the same equivalence does not hold for NMMs. Indeed, because of the nonholonomy of these systems, kinematic redundancy implies static redundancy but the converse is not true. Static redundancy in NMMs is relevant, e.g., when searching for collision-free configurations in motion planning problems with obstacles [Oriolo & Mongillo 2005]. In this work, however, only kinematic redundancy will be considered for both FBMs and NMMs.

In the following, several classical redundancy resolution methods are presented and discussed, assuming a manipulator and task kinematic model of the form (1.11) and (1.15), respectively.

2.2.1 Extended Jacobian (EJ)

Assume that an additional constraint $\mathbf{y} = \mathbf{h}(\mathbf{q})$ of dimension $k = m - s$ is attached to (1.14) in order to specify some desirable aspect of the solution. Differentiation yields

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_f(\mathbf{q}) \\ \mathbf{J}_h(\mathbf{q}) \end{bmatrix} \mathbf{N}(\mathbf{q}) \begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_m \end{bmatrix} = \mathbf{J}_e(\mathbf{q})\mathbf{u},$$

being $\mathbf{J}_e$ the square $m \times m$ *extended Jacobian* of the NMM. Whenever $\mathbf{J}_e$ is nonsingular, motion commands can be generated as

$$\mathbf{u} = \mathbf{J}_e^{-1}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{y}} \end{bmatrix}.$$

In choosing $\mathbf{h}$, special attention should be paid to *algorithmic singularities*, i.e., configurations where $\mathbf{J}_e$ is singular [Nakamura 1991] in spite of the fact that rank $\mathbf{J}_f\mathbf{N} = s$ and the last k rows of $\mathbf{J}_e$ are linearly independent. When an NMM is considered, another version of the EJ method has been proposed in [Seraji 1998]. Since the solution realizing the given (square) extended task is unique, the two formulations are equivalent.

2.2.2 Projected Gradient (PG)

In the classical *Projected Gradient* (PG) method, all solutions of (1.15) are expressed as

$$\mathbf{u} = \mathbf{J}^\dagger(\mathbf{q})\dot{\mathbf{r}} + (\mathbf{I}_m - \mathbf{J}^\dagger(\mathbf{q})\mathbf{J}(\mathbf{q}))\mathbf{u}_0 \tag{2.6}$$

where $\mathbf{J}^\dagger$ is the unique pseudoinverse of the manipulator Jacobian, $\mathbf{I}_m - \mathbf{J}^\dagger\mathbf{J}$ is the orthogonal (and symmetric) projection operator into the null-space of $\mathbf{J}$, $\ker \mathbf{J}$, and $\mathbf{u}_0 \in \mathbb{R}^m$ is an arbitrary vector usually chosen so as to optimize a given criterion $H(\mathbf{q})$. In general, $\mathbf{J}^\dagger$ can be computed from the Singular Value Decomposition (SVD) of $\mathbf{J}$. If rank $(\mathbf{J}) = s$, it is $\mathbf{J}^\dagger = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}$ [Maciejewski & Klein 1989].

In (2.6), we shall use again (2.2) or, respectively, (2.3), depending on the tracking or regulation nature of the problem. The closed-loop system will be described as before by the task error dynamics $\dot{\mathbf{e}} = -\mathbf{K}\mathbf{e}$, with the additional presence of an internal dynamics which is unobservable at the output level, i.e., in the task space.

For FBMs, $\dot{\mathbf{q}}$ and $\mathbf{u}$ coincide, and one can directly set $\mathbf{u}_0 = \pm\alpha\nabla_{\mathbf{q}}H(\mathbf{q})$ in (2.6), where $\alpha > 0$ is a stepsize in the direction of the gradient of H. A suitable value $\alpha > 0$ can be found by line search techniques [Luenberger 1984]. Care is, however, required when devising a similar scheme in the NMM case, because the available command vector $\mathbf{u}$ has a lower dimension than $\dot{\mathbf{q}}$. By differentiating $H(\mathbf{q})$ w.r.t. time and using (1.13)–(1.15), we have

$$\dot{H} = \frac{\partial H(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} = \nabla_{\mathbf{q}}^T H(\mathbf{q}) \mathbf{N}(\mathbf{q}_p) \mathbf{u}. \tag{2.7}$$

Therefore, the choice of $\mathbf{u}_0$ that locally realizes the best improvement of $H(\mathbf{q})$ is

$$\mathbf{u}_H(\mathbf{q}) = \pm\, \alpha \mathbf{N}^T(\mathbf{q}_p) \nabla_{\mathbf{q}} H(\mathbf{q}). \tag{2.8}$$

Motion commands are then generated by setting $\mathbf{u}_0 = \mathbf{u}_H(\mathbf{q})$ in (2.6). The corresponding generalized velocity of the platform

$$\dot{\mathbf{q}}_{p,H} = \pm\, \alpha\, \mathbf{G}(\mathbf{q}_p) \mathbf{G}^T(\mathbf{q}_p) \nabla_{\mathbf{q}_p} H(\mathbf{q})$$

represents a projection of $\pm\, \alpha \nabla_{\mathbf{q}_p} H$ onto the subspace of generalized velocities that are admissible with respect to the nonholonomic constraints.

2.2.3 Reduced Gradient (RG)

Both the PG and EJ resolution schemes require a large number of operations, either due to the computation of $\mathbf{J}^\dagger$ or to the inversion of the extended Jacobian $\mathbf{J}_e$. An alternative strategy is to perform optimization of the objective function (or satisfaction of a secondary task) by directly working in the reduced $(m - s)$-dimensional space of velocity inputs that satisfy the s-dimensional task (1.14). The Reduced Gradient method, originally introduced for manipulators in [De Luca & Oriolo 1991], implements this idea.

Assume that the Jacobian matrix $\mathbf{J}(\mathbf{q})$ in (1.15) has full rank at the current configuration $\mathbf{q}$. Then, it is always possible to find a permutation matrix T such that

$$\mathbf{J}(\mathbf{q})\mathbf{T} = \begin{bmatrix} \mathbf{J}_a(\mathbf{q}) & \mathbf{J}_b(\mathbf{q}) \end{bmatrix},$$

with a nonsingular $s \times s$ matrix $\mathbf{J}_a$. This induces a reordering of the velocity input vector since

$$\mathbf{u} = \mathbf{T} \begin{bmatrix} \mathbf{u}_a \\ \mathbf{u}_b \end{bmatrix} = \begin{bmatrix} \mathbf{T}_a & \mathbf{T}_b \end{bmatrix} \begin{bmatrix} \mathbf{u}_a \\ \mathbf{u}_b \end{bmatrix}, \tag{2.9}$$

where $\mathbf{u}_a \in \mathbb{R}^s$ and $\mathbf{u}_b \in \mathbb{R}^{m-s}$.The differential kinematics (1.15) becomes accordingly (dropping dependencies)

$$\dot{\mathbf{r}} = \mathbf{J}\mathbf{u} = \mathbf{J}\mathbf{T} \begin{bmatrix} \mathbf{u}_a \\ \mathbf{u}_b \end{bmatrix} = \begin{bmatrix} \mathbf{J}_a & \mathbf{J}_b \end{bmatrix} \begin{bmatrix} \mathbf{u}_a \\ \mathbf{u}_b \end{bmatrix}.$$

The task motion constraint is automatically satisfied by letting

$$\mathbf{u}_a = \mathbf{J}_a^{-1} \left(\dot{\mathbf{r}} - \mathbf{J}_b \mathbf{u}_b\right). \tag{2.10}$$

The remaining command $\mathbf{u}_b$ is chosen so as to locally optimize an objective function $H(\mathbf{q})$ as follows. Using (2.9) and (2.10) in (2.7) leads to

$$\begin{aligned}\dot{H}(\mathbf{q}) &= \nabla_{\mathbf{q}}^T(H)\mathbf{N}\mathbf{T}\begin{bmatrix}\mathbf{u}_a \\ \mathbf{u}_b\end{bmatrix} = \nabla_{\mathbf{q}}^T(H)\mathbf{N}\mathbf{T}_a\mathbf{J}_a^{-1}(\dot{\mathbf{r}} - \mathbf{J}_b\mathbf{u}_b) + \nabla_{\mathbf{q}}^T(H)\mathbf{N}\mathbf{T}_b\mathbf{u}_b = \\ &= \nabla_{\mathbf{q}}^T(H)\mathbf{N}\mathbf{T}_a\mathbf{J}_a^{-1}\dot{\mathbf{r}} + \nabla_{\mathbf{q}}^T(H)\mathbf{N}\mathbf{T}\begin{bmatrix}-\mathbf{J}_a^{-1}\mathbf{J}_b \\ \mathbf{I}_{m-s}\end{bmatrix}\mathbf{u}_b.\end{aligned}$$

Therefore, the command vector $\mathbf{u}_b$ that locally realizes the maximum improvement of the objective function H is

$$\mathbf{u}_b = \pm\alpha\begin{bmatrix}-(\mathbf{J}_a^{-1}\mathbf{J}_b)^T & \mathbf{I}_{m-s}\end{bmatrix}\mathbf{T}^T\mathbf{N}^T\nabla_{\mathbf{q}}H \tag{2.11}$$

By comparing (2.11) with (2.8), we note that $\mathbf{u}_b$ is the *reduction* (up to a permutation of components) of the velocity input $\mathbf{u}_H$ to the subspace of commands that automatically satisfy the task constraint.

When $\dot{\mathbf{r}} = 0$ (self-motion) and the degree of kinematic redundancy of the manipulator is 1, the RG and PG methods generate velocity commands in the same direction, although the former method allows longer steps to be taken and results in a faster optimization [De Luca & Oriolo 1991]. In any other case, the two methods generate different directions in the command space, and thus in the manipulator configuration space. Note that, although the computational load of the RG method is typically lower, it may still be necessary to change the permutation matrix $\mathbf{T} = \mathbf{T}(\mathbf{q})$ in (2.9) along the motion so as to extract a (different) nonsingular matrix $\mathbf{J}_a(\mathbf{q})$.

In the special case of an NMM with $n_m = s$ and the onboard manipulator not in a singular configuration, one can choose $\mathbf{J}_a = \mathbf{J}_m$ and $\mathbf{J}_b = \mathbf{J}_p\mathbf{G}$ so that (2.10) and (2.11) simplify to

$$\begin{aligned}\mathbf{u}_m &= \mathbf{J}_m^{-1}(\dot{\mathbf{r}} - \mathbf{J}_p\mathbf{G}\mathbf{u}_p) \\ \mathbf{u}_p &= \pm\alpha\,\mathbf{G}^{\mathbf{T}}\begin{bmatrix}-(\mathbf{J}_a^{-1}\mathbf{J}_b)^{\mathbf{T}} & \mathbf{I}_{n_p}\end{bmatrix}\nabla_{\mathbf{q}}H.\end{aligned}$$

2.2.4 Task Priority (TP)

In the neighborhood of singular points of the task Jacobian, the use of the PG method (2.6) may result in very high velocity commands which cannot be realized by the low-level actuator controllers. One way to deal with this problem is to use the *Task Priority* (TP) technique [Chiacchio *et al.* 1991]. The idea is to reorder the task vector $\mathbf{r}$ into μ subtasks $(\mathbf{r}_1, \ldots, \mathbf{r}_\mu)$, each of dimension s_i ($i = 1, \ldots, \mu$, with $\sum s_i = s$), and to consider $\mathbf{r}_i$ as a task with higher priority than $\mathbf{r}_j$ if $i < j$. This is associated to a task-oriented partition of the task

Jacobian in (1.15) in the form

$$\begin{bmatrix} \dot{\mathbf{r}}_1 \\ \vdots \\ \dot{\mathbf{r}}_\mu \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1(\mathbf{q}) \\ \vdots \\ \mathbf{J}_\mu(\mathbf{q}) \end{bmatrix} \mathbf{u}.$$

The TP method is formulated in such a way that, whenever $\mathbf{J}$ is rank deficient, the lowest priority tasks are relaxed while correctly executing those with highest priority.

For illustration, consider a regulation problem in which the task $\mathbf{r}$ is partitioned in $\mu = 2$ subtasks:

$$\begin{bmatrix} \dot{\mathbf{r}}_1 \\ \dot{\mathbf{r}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1(\mathbf{q}) \\ \mathbf{J}_2(\mathbf{q}) \end{bmatrix} \mathbf{u}. \tag{2.12}$$

By solving the first set of s_1 equations, we get

$$\mathbf{u} = \mathbf{J}_1^\dagger \dot{\mathbf{r}}_1 + (\mathbf{I}_m - \mathbf{J}_1^\dagger \mathbf{J}_1)\mathbf{u}_0 \tag{2.13}$$

where $\dot{\mathbf{r}}_1 = \mathbf{K}_1(\mathbf{r}_{1d} - \mathbf{r}_1) = \mathbf{K}_1\mathbf{e}_1$, with $\mathbf{K}_1 > 0$. If rank $\mathbf{J}_1 = s_1$ throughout the motion, (2.13) ensures that the primary task variables $\mathbf{r}_1$ converge exponentially to their desired values. The auxiliary command $\mathbf{u}_0$ can be chosen so as to minimize the weighted norm of the secondary task error, i.e., the scalar function

$$H_2(\mathbf{q}) = \frac{1}{2}\mathbf{e}_2^T\mathbf{K}_2\mathbf{e}_2 = \frac{1}{2}(\mathbf{r}_{2d} - \mathbf{r}_2)^T\mathbf{K}_2(\mathbf{r}_{2d} - \mathbf{r}_2), \qquad \mathbf{K}_2 > 0. \tag{2.14}$$

The time derivative of (2.14) is

$$\dot{H}_2 = -\mathbf{e}_2^T\mathbf{K}_2\dot{\mathbf{r}}_2 = -\mathbf{e}_2^T\mathbf{K}_2\mathbf{J}_2\mathbf{u} = -\mathbf{e}_2^T\mathbf{K}_2\mathbf{J}_2\mathbf{J}_1^\dagger\mathbf{K}_1\mathbf{e}_1 - \mathbf{e}_2^T\mathbf{K}_2\mathbf{J}_2(\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1)\mathbf{u}_0.$$

By using the symmetry of $\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1$, we conclude that the choice

$$\mathbf{u}_0 = (\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1)\mathbf{J}_2^T\mathbf{K}_2\mathbf{e}_2 \tag{2.15}$$

provides the maximum reduction of H_2, subject to the satisfaction of the primary task. By plugging (2.15) into (2.13), and using the idempotency of $\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1$, we can write the TP kinematic control law as

$$\mathbf{u} = \mathbf{J}_1^\dagger\mathbf{K}_1\mathbf{e}_1 + (\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1)\mathbf{J}_2^T\mathbf{K}_2\mathbf{e}_2. \tag{2.16}$$

The generalization of (2.16) to tracking problems is straightforward but computationally more cumbersome [Nakamura 1991]. Different options for interchanging transpose and pseudoinverse of the involved Jacobians and the convergence properties of the associated schemes have been considered in [Chiacchio *et al.* 1991].

2.2.5 Task Sequencing (TS)

For regulation problems, an alternative way to drive all the task variables to their set-points is to follow a *Task Sequencing* (TS) approach. The idea is to process the μ subtasks one at a time. The s_1 subtask variables $\mathbf{r}_1$ are regulated first, then the s_2 subtask variables $\mathbf{r}_2$ are driven toward their desired value without changing $\mathbf{r}_1$, then the s_3 subtask variables $\mathbf{r}_3$ are brought to their desired value without changing $\mathbf{r}_1$ and $\mathbf{r}_2$, and so on. With this approach, an 'artificial' redundancy is introduced in the kinematic control process: in particular, the degree of redundancy during the execution of the sequence will be $p + n_m - s_1$ in the first phase, $p+n_m-(s_1+s_2)$ in the second phase, and so on. In the last phase, the redundancy degree will be back to its (possibly zero) original value $p+n_m-s \geq 0$. This method applies to manipulators that are either redundant or non-redundant w.r.t. the task.

Consider for simplicity the case of $\mu = 2$ subtasks, and assume $p + n_m = s = s_1 + s_2$, i.e., the non-redundant case for the global task. Define the two-phase task sequence

$$\begin{cases} \dot{\mathbf{r}}_I = \dot{\mathbf{r}}_1 = \mathbf{K}_1\mathbf{e}_1, & t \in [0,\, T_1] \\ \dot{\mathbf{r}}_{II} = \begin{bmatrix} \mathbf{0} \\ \dot{\mathbf{r}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{K}_2\mathbf{e}_2 \end{bmatrix}, & t \in [T_1,\, T_2], \end{cases}$$

where T_i $(i = 1, 2)$ denotes the time of completion of task i, i.e., such that a suitable termination condition is reached[1]. This leads to the kinematic control scheme

$$\begin{cases} \mathbf{u}_I = \mathbf{J}_1^\dagger\mathbf{K}_1\mathbf{e}_1 + (\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1)\mathbf{u}_0, & t \in [0,\, T_1] \\ \mathbf{u}_{II} = (\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1)\mathbf{J}_2^T\mathbf{K}_2\mathbf{e}_2, & t \in [T_1,\, T_2], \end{cases} \tag{2.17}$$

where $\mathbf{u}_0$ can be chosen as in (2.8) for optimization purposes.

In this basic version, the first task variable $\mathbf{r}_1$ is no longer controlled during the second phase, so that its value may drift, e.g., due to linearization errors (recall that we are using the Jacobian matrix which is a mapping between tangent spaces). One way to counteract this effect is to replace the second-phase control law in (2.17) with

$$\mathbf{u}_{II} = (\mathbf{I}_m - \mathbf{J}_1^\dagger\mathbf{J}_1)\mathbf{J}_2^T\mathbf{K}_2\mathbf{e}_2 + \mathbf{J}_1^\dagger\mathbf{K}_1\mathbf{e}_1, \qquad t \in [T_1,\, T_2],$$

i.e., by adding a feedback term aimed at rejecting perturbations on $\mathbf{e}_1$. This second phase of the TS approach becomes then identical to the TP method (2.16), with the notable difference that $\mathbf{e}_1$ is already very small as a result of the first phase.

The extension of the above formulas to the case $\mu > 2$ and/or $p+n_m > s$ (redundant case) is straightforward. It is also easy to prove that the TS strategy guarantees convergence of the task

[1]For instance, the termination condition $\|\mathbf{r}_{id} - \mathbf{r}_i(T_i)\| < \epsilon$, for a given $\epsilon > 0$, will result in an arbitrarily small final error. Finite-time convergence to the set-point can be easily obtained by using a *terminal controller* [Zak 1989] to define each phase of the task sequence.

variables to their set-point (within an arbitrarily small tolerance), provided that the Jacobian of stacked subtasks $\bar{\mathbf{J}}_i = \left[\mathbf{J}_1^T\,\mathbf{J}_2^T \ldots \mathbf{J}_i^T\right]^T$ has full rank during the i-th phase of the sequence.

The potential advantage of the TS kinematic control strategy over the classical approach which tries to drive all task variables *simultaneously* to their set-point rests on the artificial redundancy introduced in all but the last phase of the sequence. This can be used to optimize performance indices, and in particular to stay away from singular configurations. It is worth citing [Mansard & Chaumette 2007] where a different TS concept is proposed to deal with visual servoing tasks. In that work, the authors develop a layered controller architecture able to reactively build and execute a stack of subtasks part of a given main task. During the motion, this stack is updated according to several criteria (interaction with the environment, avoidance of joint limits, etc.), while an higher-level controller ensures global convergence of the whole task by avoiding local minima or dead locks with a suitable path planning phase. Our TS method can be seen as a simplified version of that work, with a time-based switching criterium in place of the complex reactive-based strategy developed in [Mansard & Chaumette 2007].

2.3 Case studies

In this section two modeling examples of task-oriented kinematics for NMMs are presented. FBMs are not explicitly considered here since there already exists a vast literature covering their case. Moreover, the following derivation also shows how the combination of nonholonomic platform/manipulator can often reduce the set of singular points that would affect the manipulator taken alone. Some of the following developments are also preliminary to the use of the RG method in Sect. 2.4.

2.3.1 Unicycle platform with 2R planar manipulator

Consider a 2R manipulator in the horizontal plane, with link lengths l_1 and l_2, mounted on a two-wheel differentially driven mobile platform with unicycle kinematics, as shown in Fig. 2.1. The base of the manipulator is placed along the main axis of the platform, at a distance d from the wheels' axis. The configuration vector of this NMM is $\mathbf{q} = [\mathbf{q}_p^T\,\mathbf{q}_m^T]^T \in \mathbb{R}^5$, with $\mathbf{q}_p = [x\,y\,\theta]^T \in \mathbb{R}^3$ and $\mathbf{q}_m = [q_1\,q_2]^T \in \mathbb{R}^2$.

The task of positioning the NMM end-effector in the plane has dimension $s = 2$, so that the degree of static redundancy is 3. The associated kinematic map is

$$\begin{bmatrix} r_x \\ r_y \end{bmatrix} = \begin{bmatrix} x + dc_\theta + l_1 c_{\theta q_1} + l_2 c_{\theta q_1 q_2} \\ y + ds_\theta + l_1 s_{\theta q_1} + l_2 s_{\theta q_1 q_2} \end{bmatrix}, \tag{2.18}$$

where c_{ijk} and s_{ijk} stand for $\cos(i+j+k)$ and $\sin(i+j+k)$. The Jacobians $\mathbf{J}_p(\mathbf{q})$ and $\mathbf{J}_m(\mathbf{q})$

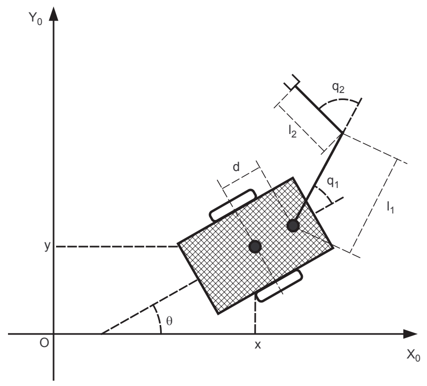


Figure 2.1: A planar NMM with a 2R polar manipulator (top view).

are

$$
\begin{aligned}
\mathbf{J}_p &= \begin{bmatrix} 1 & 0 & -ds_\theta - l_1 s_{\theta q_1} - l_2 s_{\theta q_1 q_2} \\ 0 & 1 & dc_\theta + l_1 c_{\theta q_1} + l_2 c_{\theta q_1 q_2} \end{bmatrix} \\
\mathbf{J}_m &= \begin{bmatrix} -l_1 s_{\theta q_1} - l_2 s_{\theta q_1 q_2} & -l_2 s_{\theta q_1 q_2} \\ l_1 c_{\theta q_1} + l_2 c_{\theta q_1 q_2} & l_2 c_{\theta q_1 q_2} \end{bmatrix}.
\end{aligned}
$$

As for the mobile platform, the unicycle kinematics is subjected to the single nonholonomic constraint ($k = 1$)

$$\mathbf{a}_1^T(\mathbf{q}_p)\dot{\mathbf{q}}_p = [-\sin\theta \;\; \cos\theta \;\; 0]\dot{\mathbf{q}}_p = 0$$

so that the feasible platform velocities are

$$\dot{\mathbf{q}}_p = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \mathbf{G}(\mathbf{q}_p)\mathbf{u}_p, \tag{2.19}$$

with input $\mathbf{u}_p$ of dimension $p = n_p - k = 2$. Being $\mathbf{u}_m = [\dot{\mathbf{q}}_1 \, \dot{\mathbf{q}}_2]^T \in \mathbb{R}^2$, the degree of kinematic redundancy is 2 for this motion task. The 2×4 Jacobian $\mathbf{J}(\mathbf{q})$ in the differential kinematics (1.16) is

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} c_\theta & -ds_\theta - l_1 s_{\theta q_1} - l_2 s_{\theta q_1 q_2} & -l_1 s_{\theta q_1} - l_2 s_{\theta q_1 q_2} & -l_2 s_{\theta q_1 q_2} \\ s_\theta & dc_\theta + l_1 c_{\theta q_1} + l_2 c_{\theta q_1 q_2} & l_1 c_{\theta q_1} + l_2 c_{\theta q_1 q_2} & l_2 c_{\theta q_1 q_2} \end{bmatrix}. \tag{2.20}$$

Note that, in order to obtain dimensionally homogeneous velocity inputs for the NMM (all angular quantities), the unicycle commands $\mathbf{u}_p = [v \, \omega]^T$ may be replaced by the angular velocities $[\dot{\phi}_R \, \dot{\phi}_L]^T$ of the right and left wheel of the platform. Since this is achieved by means of a (constant) invertible matrix $\mathbf{M}$, the analysis of singularities is unaffected. However, working with homogeneous commands is convenient when dealing with vectors of minimum norm in inverse kinematics solutions, as will be shown in Sect. 4.6. To identify its singularities, consider the 2×2 minors Δ_{ij} obtained by selecting the i-th and j-th column from (2.20):

$$\begin{aligned} \Delta_{12} &= d + l_1 c_{q_1} + l_2 c_{q_1 q_2} & \Delta_{23} &= -d(l_1 s_{q_1} + l_2 s_{q_1 q_2}) \\ \Delta_{13} &= -l_1 c_{q_1} - l_2 c_{q_1 q_2} & \Delta_{24} &= -l_2(l_1 s_{q_2} + d s_{q_1 q_2}) \\ \Delta_{14} &= -l_2 c_{q_1 q_2} & \Delta_{34} &= l_1 l_2 s_{q_2}. \end{aligned}$$

Clearly, Δ_{12} and Δ_{13} cannot simultaneously vanish if $d \neq 0$. In this case, $\operatorname{rank}\mathbf{J}(\mathbf{q}) = 2$ everywhere and the NMM has no singular configurations (as opposed to the manipulator taken alone). Moreover, if $d > l_1 + l_2$, Δ_{12} is always nonzero and the RG method of Sect. 2.2.3 can be used with the globally defined inverse of $\mathbf{J}_a = \mathbf{J}_p\mathbf{G}$ (first two columns in (2.20)) without changing the permutation matrix $\mathbf{T}$ in (2.9). In general, one may need to switch between the matrices corresponding to Δ_{12} and Δ_{13}.

Suppose now that the task is to follow a given cartesian trajectory while pointing at a fixed object with a camera mounted on the end-effector. To this end, we add the absolute orientation $\delta = \theta + q_1 + q_2$ of the second link to the end-effector positioning task (2.18), obtaining an extended task of dimension $s = 3$. The associated 3×4 Jacobian $\mathbf{J}(\mathbf{q})$ becomes

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} c_\theta & -d s_\theta - l_1 s_{\theta q_1} - l_2 s_{\theta q_1 q_2} & -l_1 s_{\theta q_1} - l_2 s_{\theta q_1 q_2} & -l_2 s_{\theta q_1 q_2} \\ s_\theta & d c_\theta + l_1 c_{\theta q_1} + l_2 c_{\theta q_1 q_2} & l_1 c_{\theta q_1} + l_2 c_{\theta q_1 q_2} & l_2 c_{\theta q_1 q_2} \\ 0 & 1 & 1 & 1 \end{bmatrix}. \tag{2.21}$$

As before, by computing the minors Δ_{ijk}

$$\begin{aligned} \Delta_{123} &= d & \Delta_{134} &= l_1 c_{q_1} \\ \Delta_{124} &= d + l_1 c_{q_1} & \Delta_{234} &= l_1 d s_{q_1}, \end{aligned}$$

we conclude that, as long as $d \neq 0$, Δ_{123} never vanishes and the Jacobian has full rank everywhere.

2.3.2 Unicycle platform with 3R elbow-type manipulator

Consider a 3R elbow-type manipulator (with link lengths l_i, $i = 1, 2, 3$) mounted on the same previous platform of height h, with an offset $d \neq 0$ with respect to its center (see Fig. 2.2). Let $\mathbf{q}_p = [x\, y\, \theta]^T \in \mathbb{R}^3$ and $\mathbf{q}_m = [q_1\, q_2\, q_3]^T \in \mathbb{R}^3$ be the configuration vectors of the platform and manipulator, respectively. For the task of positioning the end-effector in 3D-space ($s = 3$), the kinematic map is

$$\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} x + d c_\theta + l_2 c_{\theta q_1} s_{q_2} + l_3 c_{\theta q_1} s_{q_2 q_3} \\ y + d s_\theta + l_2 s_{\theta q_1} s_{q_2} + l_3 s_{\theta q_1} s_{q_2 q_3} \\ h + l_1 + l_2 c_{q_2} + l_3 c_{q_2 q_3} \end{bmatrix},$$

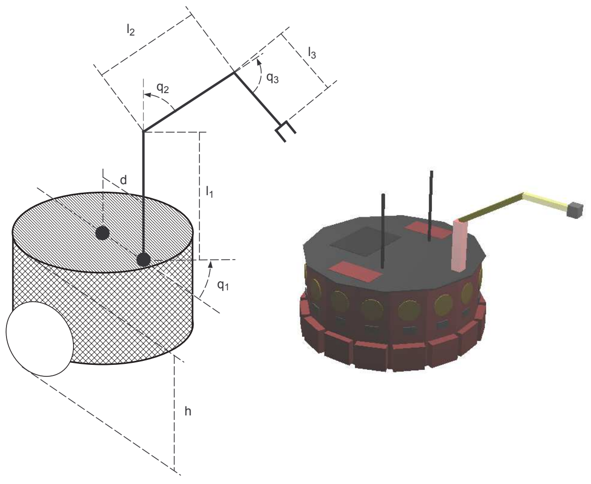


Figure 2.2: A spatial NMM with a 3R elbow-type manipulator.

and the associated Jacobian $\mathbf{J}(\mathbf{q})$ of the NMM is

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} c_\theta & -ds_\theta - l_2 s_{\theta q_1} s_{q_2} - l_3 s_{\theta q_1} s_{q_2 q_3} & -l_2 s_{\theta q_1} s_{q_2} - l_3 s_{\theta q_1} s_{q_2 q_3} & l_2 c_{\theta q_1} c_{q_2} + l_3 c_{\theta q_1} c_{q_2 q_3} & l_3 c_{\theta q_1} c_{q_2 q_3} \\ s_\theta & dc_\theta + l_2 c_{\theta q_1} s_{q_2} + l_3 c_{\theta q_1} s_{q_2 q_3} & l_2 c_{\theta q_1} s_{q_2} + l_3 c_{\theta q_1} s_{q_2 q_3} & l_2 s_{\theta q_1} c_{q_2} + l_3 s_{\theta q_1} c_{q_2 q_3} & l_3 s_{\theta q_1} c_{q_2 q_3} \\ 0 & 0 & 0 & -l_2 s_{q_2} - l_3 s_{q_2 q_3} & -l_3 s_{q_2 q_3} \end{bmatrix}.$$

Since $\mathbf{u} = [u_p^T \, u_m^T]^T = [v \, \omega \, \dot{q}_1 \, \dot{q}_2 \, \dot{q}_3]^T \in \mathbb{R}^5$, the degree of kinematic redundancy is 2. In order to study the rank of $\mathbf{J}(\mathbf{q})$, we compute the $\binom{5}{3} = 10$ possible minors:

$$\begin{array}{ll} \Delta_{123} = -l_3 s_{q_2 q_3}(\Lambda c_{q_1} + d) & \Delta_{145} = l_2 l_3 \Lambda s_{q_3} \\ \Delta_{124} = -l_3 s_{q_2 q_3} c_{q_1} \Lambda & \Delta_{234} = 0 \\ \Delta_{125} = -l_2 l_3 s_{q_1} s_{q_3} & \Delta_{235} = -\Lambda(\Lambda c_{q_1} + d) \\ \Delta_{134} = -d l_3 s_{q_1} s_{q_2 q_3} \Lambda & \Delta_{245} = -\Lambda^2 c_{q_1} \\ \Delta_{135} = l_2 l_3 s_{q_3}(d c_{q_1} + \Lambda) & \Delta_{345} = -d\Lambda^2 s_{q_1}, \end{array}$$

where $\Lambda = l_2 s_{q_2} + l_3 s_{q_2 q_3} = 0$ corresponds to the shoulder singularity for the elbow-type manipulator. No minor is always nonzero in this case. However, it is easy to verify that if $s_{q_2} \neq 0$ or $s_{q_3} \neq 0$ there is always a nonzero minor. Hence, rank $\mathbf{J}(\mathbf{q}) = 3$ for this NMM, except when the manipulator is stretched or folded along the vertical direction.

2.4 Simulation results

We present here two simulations of kinematic control schemes for the NMM in Fig. 2.1, and one for the NMM in Fig. 2.2. As redundancy resolution scheme, we adopt the RG

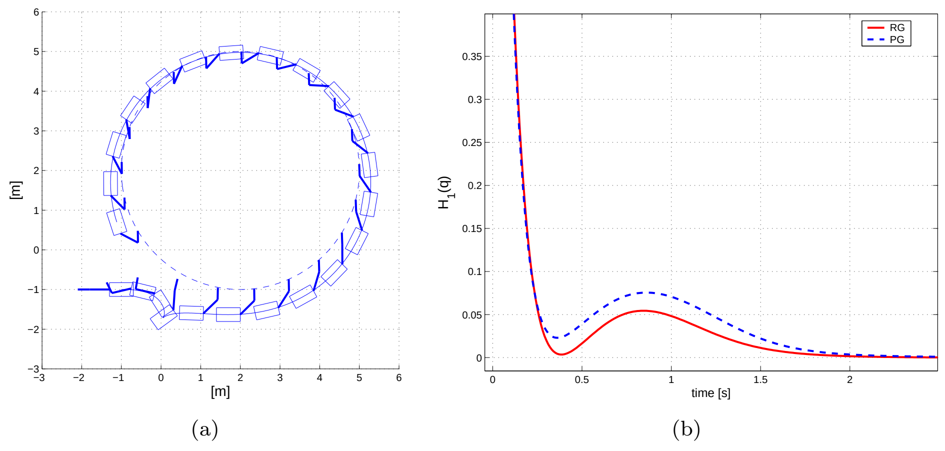


Figure 2.3: **First simulation**. Left: trajectory tracking for the planar NMM with RG method. Right: comparison between RG and PG methods in minimizing the objective function $H_1(\mathbf{q})$ (zoom on the initial transient).

method, whose optimization performance is compared with the PG method. Further results involving the use of PG, TP and TS methods are presented in Chapter 6 and Chapter 7. The task control matrix gain $\mathbf{K}$ is set equal to the identity, while a constant step size $\alpha = 10$ has been used for simplicity in gradient computations. Movie clips of additional simulations, including also the case of manipulability optimization, can be found at the website `http://www.dis.uniroma1.it/labrob/research/NMM.html`.

2.4.1 Position task for the NMM with planar manipulator

Consider the robot in Fig. 2.1 with $d = 0.3$, $l_1 = 0.5$, and $l_2 = 0.3$ [m]. Since $d < l_1 + l_2$, there is no minor guaranteed to be always nonzero. However, a simple switching strategy can be adopted for the RG method. Let $\mathbf{j}_i$ be the i-th column of the NMM Jacobian $\mathbf{J}(\mathbf{q})$ in (2.20):

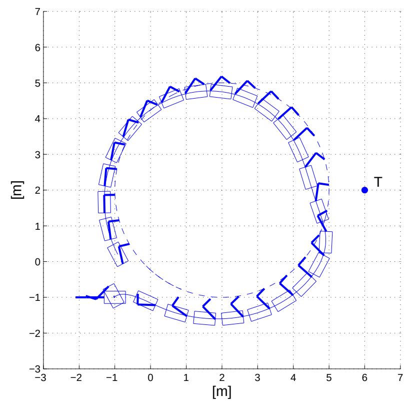
1. If $|\Delta_{13}| \geq |\Delta_{12}|$, start with the inversion of $\mathbf{J}_a = [\mathbf{j}_1\,\mathbf{j}_3]$, otherwise start with $\mathbf{J}_a = [\mathbf{j}_1\,\mathbf{j}_2]$;
2. if $|\Delta_{13}| < \Theta$ (or $|\Delta_{12}| < \Theta$) switch to the other minor.

In the simulations, a fixed threshold $\Theta = 10^{-2}$ has been used. The end-effector should follow the circular trajectory

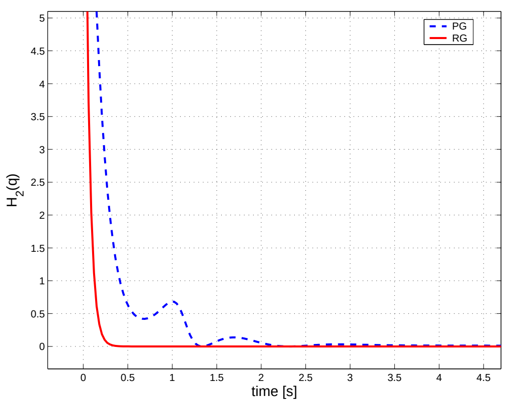
$$\mathbf{r}_d(t) = \begin{bmatrix} r_{dx}(t) \\ r_{dy}(t) \end{bmatrix} = \begin{bmatrix} 2 + 3\cos(0.08\pi t + \frac{5}{4}\pi) \\ 2 + 3\sin(0.08\pi t + \frac{5}{4}\pi) \end{bmatrix}, \tag{2.22}$$

for $T_s = 25$ s, while minimizing the objective function

$$H_1(\mathbf{q}) = \frac{1}{2}\left(\theta + q_1 + q_2 - \frac{\pi}{2}\right)^2 + \frac{1}{2}\left(q_1 - \frac{\pi}{4}\right)^2,$$



(a)



(b)

Figure 2.4: **Second simulation**. Left: trajectory tracking and target pointing for the spatial NMM with RG method. Right: comparison between RG and PG methods in minimizing the objective function $H_2(\mathbf{q})$ (zoom on the initial transient).

i.e., keeping the first link at $q_1 = \pi/4$ with the end-effector pointing in the (absolute) upward direction. The start configuration at $t = 0$ is $\mathbf{q}_0 = [-1\ -1\ \pi\ 0\ 0]^T$ and corresponds to the end-effector being out of the desired path.

The stroboscopic motion of the NMM in Fig. 2.3(a) shows the good tracking behavior, despite the initial backup of the platform. The performance comparison of the RG and PG methods in Fig. 2.3(b) indicates that RG is faster in approaching the minimum value of H_1 (and also better in keeping a lower ripple during steady-state motion).

2.4.2 Position/orientation task for the NMM with planar manipulator

The former task is extended to include the absolute orientation of the second link of the manipulator as in (2.21). In particular, this should point towards a fixed target point T located at $(x_T, y_T) = (6, 2)$ [m]. Therefore, we append to the positioning task (2.22) a third component

$$\mathbf{r}_{d\delta}(t) = \text{atan2}(y_T - r_{dy}(t), x_T - r_{dx}(t)).$$

The single degree of kinematic redundancy left is used to minimize the objective function

$$H_2(\mathbf{q}) = \frac{1}{2}\left(q_2 + \frac{\pi}{2}\right)^2.$$

Figure 2.4(a) shows the stroboscopic motion of the NMM generated with the RG method. A waving behavior around the nominal end-effector trajectory is now realized by the platform in order to satisfy the additional task constraint. No switching strategy is needed here since submatrix $\mathbf{J}_a = [\mathbf{j}_1\, \mathbf{j}_2\, \mathbf{j}_3]$ is always nonsingular, as shown in Sect. 2.3.1. Also in this case, the

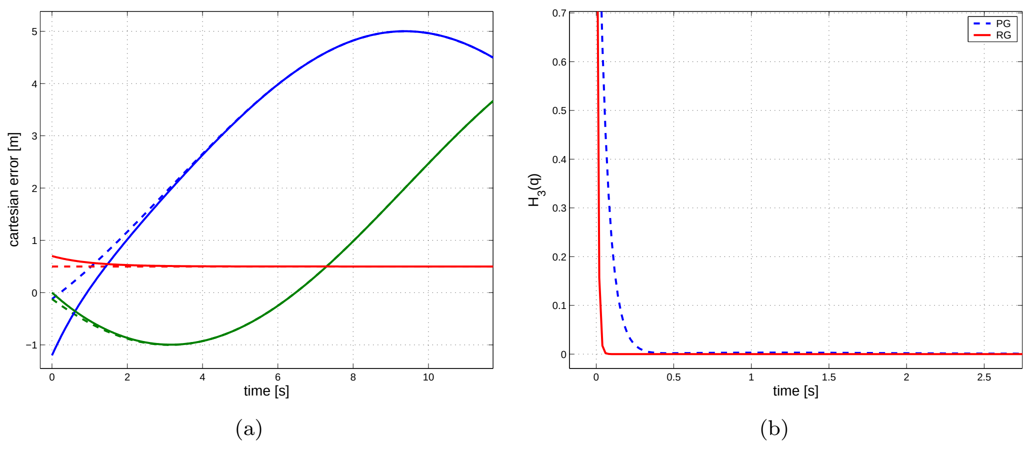


Figure 2.5: **Third simulation**. Left: cartesian trajectory tracking error for the NMM in Fig. 2.2 with RG method (dashed lines represent the reference motion). Right: comparison between RG and PG methods in minimizing the objective function $H_3(\mathbf{q})$ (zoom on the initial transient).

optimization with the RG method is more efficient than with the PG method (see Fig. 2.4(b)), reaching faster and firmly keeping the absolute minimum of H_2.

2.4.3 Position task for the NMM with elbow-type manipulator

The geometric data of the robot in Fig. 2.2 are $d = 0.3$, $h = 0.3$, $l_1 = 0.4$, $l_2 = 0.5$, and $l_3 = 0.4$ [m]. For the 3D task of end-effector spatial positioning, the two degrees of kinematic redundancy are used for minimizing the function

$$H_3(\mathbf{q}) = \frac{1}{2}\,q_1^2 + \frac{1}{2}\left(q_3 + \frac{\pi}{2}\right)^2.$$

The end-effector should follow the circular trajectory (2.22) taking place at a constant height $r_{dz} = 0.5$ [m]. The initial configuration is $\mathbf{q}_0 = [0\ 0\ \pi\ 0\ \pi/2\ 0]^T$, which corresponds again to the end-effector being out of path. Since there is no minor guaranteed to be always nonzero in this case, a switching strategy similar to the one discussed in case 1 has been implemented for the RG method. The results in Figs. 2.5(a) and 2.5(b) show a fast recovery of the initial cartesian error and of the optimal value for H_3 with both RG and PG methods.

Part II

Visual Control: Theory

3 Elements of 3D Vision

A camera can be considered as a sensor yielding 2D outputs (the 2D camera images) from 3D input data (the 3D external scene) through a process called *image formation*. A thorough illustration of camera image formation involves topics from many different areas: description of the camera pose in the world (*rigid body kinematics*), mathematical modeling of lenses and light through them (*photometry*), derivation of perspective projections models (*perspective geometry*), representation of the luminosity information (*quantization and digitization*), are just a few. In this Thesis, only the geometric aspects of image formation are discussed, while a detailed treatment of the photometric and optical effects, not addressed here, can be found in [Horn 1986; Lavest *et al.* 1993; Born & Wolf 1999; Stroebel 1999].

Even within the reduced scope of geometric description, however, several camera models are still possible. For instance, the nature of the imaging surface (planar, spherical, etc.), and the approximations introduced in the 3D projection modeling give rise to different possibilities, such as paraperspective projection [Aloimonos 1990; Basri 1996], orthographic projection [Tomasi & Kanade 1992], affine projection [Koenderink & van Doorn 1991; Mundy & Zisserman 1992], and catadioptric projection [Geyer & Daniilidis 2001; Mariottini & Prattichizzo 2007]. In most cases, however, a convenient (and widely adopted) choice is to restrict the analysis to the so-called *pin-hole camera model* which captures, in a simple but sufficiently accurate way, the relevant perspective geometry transformations common to many real cameras with planar imaging surfaces. Therefore, by pursuing this idea, the next sections are devoted to the geometric and kinematic equations describing a pin-hole camera moving through the world. Section 3.1 briefly recalls some relevant rigid body kinematics concepts at the basis of most subsequent developments. Next, Sect. 3.2 describes the main properties of the pin-hole camera model and Sect. 3.3 sheds some light onto the geometric constraints that bind two images of the same 3D scene. Most of the concepts presented hereafter are borrowed from [Ma *et al.* 2004] for what concerns camera modeling, and [Murray *et al.* 1994] for the rigid body kinematics part.

3.1 Rigid body kinematics

Let $\mathbb{E}^3$ represent the standard three-dimensional Euclidean space. A 3D point $P \in \mathbb{E}^3$ can be identified with a point in $\mathbb{R}^3$ with coordinates[1]

$$P = [P_x \; P_y \; P_z]^T \in \mathbb{R}^3.$$

A *bound vector* $\mathbf{v} \in \mathbb{R}^3$ is determined by a pair of points $P_1, P_2 \in \mathbb{E}^3$ as

$$\mathbf{v} = P_2 - P_1 \in \mathbb{R}^3.$$

One can also introduce the concept of *free vector*, if the vector definition does not depend on the base point P_1. In this case, every two pairs of points (P_1, P_2) and (P_1', P_2') such that $P_2 - P_1 = P_2' - P_1'$ define the same free vector $\mathbf{v}$. The set of all free vector forms a *linear vector space*. The *inner product*, or *dot product*, among vectors is defined as

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} = \mathbf{v}^T \mathbf{u} = u_x v_x + u_y v_y + u_z v_z \in \mathbb{R}.$$

Whenever $\langle \mathbf{u}, \mathbf{v} \rangle = 0$, the two vectors are said to be *orthogonal*. The norm (or length) of a vector $\mathbf{u}$ is defined as $\|\mathbf{u}\| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}$. Given two vectors $\mathbf{u}$, $\mathbf{v}$, their *cross product*, or *outer product*, is a third vector

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{bmatrix} \in \mathbb{R}^3.$$

The cross product is linear in each of its arguments: $\mathbf{u}\times(\alpha\mathbf{v}+\beta\mathbf{w}) = \alpha\mathbf{u}\times\mathbf{v}+\beta\mathbf{u}\times\mathbf{w}$, $\forall\alpha, \beta \in \mathbb{R}$, and satisfies

$$< \mathbf{u} \times \mathbf{v}, \mathbf{u} >=< \mathbf{u} \times \mathbf{v}, \mathbf{v} >= 0, \quad \mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}. \tag{3.1}$$

[1]Here, with a slight abuse of notation, the same symbol P is used to denote the coordinates of a point in $\mathbb{R}^3$ and the point itself in $\mathbb{E}^3$.

Therefore, the cross product of two vectors is orthogonal to both of its factors, and does not commute with the order of multiplication. A convenient matrix expression of the cross product is given by

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

where $[\mathbf{u}]_\times \in so(3)$ is the 3×3 skew-symmetric matrix associated to a vector $\mathbf{u} \in \mathbb{R}^3$. Note that, the map $[\cdot]_\times : \mathbf{u} \to [\mathbf{u}]_\times$ is a linear bijection: $[\alpha\mathbf{u} + \beta\mathbf{v}]_\times = \alpha[\mathbf{u}]_\times + \beta[\mathbf{v}]_\times$. Furthermore, from (3.1), it follows $\mathbf{u}^T[\mathbf{u}]_\times = [\mathbf{u}]_\times \mathbf{u} = 0$, and $[\mathbf{u}]_\times \mathbf{v} = -[\mathbf{v}]_\times \mathbf{u}$.

Consider an inertial 'world' reference frame $\mathcal{F}_O : \{O;\, \mathbf{X}_O,\, \mathbf{Y}_O,\, \mathbf{Z}_O\}$ and a (possibly) moving frame $\mathcal{F}_C : \{O_C;\, \mathbf{X}_C,\, \mathbf{Y}_C,\, \mathbf{Z}_C\}$. When needed, a leading superscript will be used to specify the reference frame where quantities are expressed. Without loss of generality, we assume $\mathcal{F}_C$ to be rigidly attached to a camera that moves through the scene (see Sect. 3.2). Let

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} |\; \mathbf{R}^T\mathbf{R} = \mathbf{I}_3,\; \det \mathbf{R} = 1\} \tag{3.2}$$

be the set of 3×3 orthogonal matrices with determinant 1. The orientation of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$ can be univocally specified in terms of an orthogonal matrix $\mathbf{R}_{OC} \in SO(3)$. Due to definition (3.2), the 'inverse' rotation matrix of $\mathcal{F}_O$ w.r.t. $\mathcal{F}_C$ is given by $\mathbf{R}_{OC}^{-1} = \mathbf{R}_{OC}^T = \mathbf{R}_{CO}$. The full rigid body motion (rotation + translation) of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$ is an element g_{OC} of the group

$$SE(3) = \{(\mathbf{R}_{OC},\, {}^O\mathbf{T}_{OC}) |\; \mathbf{R}_{OC} \in SO(3),\; {}^O\mathbf{T}_{OC} \in \mathbb{R}^3\},$$

where ${}^O\mathbf{T}_{OC} \in \mathbb{R}^3$ is the vector from O to O_C expressed in $\mathcal{F}_O$. It is worth noting that g_{OC} acts differently on points and vectors. Indeed, in the former case, it is

$${}^OP = g_{OC}({}^CP) = \mathbf{R}_{OC}\,{}^CP + {}^O\mathbf{T}_{OC} \tag{3.3}$$

while, in the latter case,

$${}^O\mathbf{v} = g_{OC}({}^C\mathbf{v}) = \mathbf{R}_{OC}\,{}^C\mathbf{v}. \tag{3.4}$$

If points and vectors are represented in *homogeneous coordinates*, it is possible to express the action of g_{OC} in a compact matrix form. To this end, let

$$\bar{P} = \begin{bmatrix} P \\ 1 \end{bmatrix} \in \mathbb{R}^4, \quad \bar{\mathbf{v}} = \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \in \mathbb{R}^4$$

be the homogeneous representations of points and vectors, respectively. The homogeneous matrix representation of g_{OC},

$$\bar{g}_{OC} = \begin{bmatrix} \mathbf{R}_{OC} & {}^O\mathbf{T}_{OC} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4},$$

allows to express (3.3) and (3.4) as ${}^{O}\bar{P} = \bar{g}_{OC}{}^{C}\bar{P}$ and ${}^{O}\bar{\mathbf{v}} = \bar{g}_{OC}{}^{C}\bar{\mathbf{v}}$. The inverse element of g_{OC}, denoted as $g_{OC}^{-1} = g_{CO}$, takes the matrix form

$$\bar{g}_{CO} = \begin{bmatrix} \mathbf{R}_{OC}^{T} & -\mathbf{R}_{OC}^{T}{}^{O}\mathbf{T}_{OC} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{CO} & -\mathbf{R}_{CO}{}^{O}\mathbf{T}_{OC} \\ \mathbf{0} & 1 \end{bmatrix},$$

so that

$${}^{C}\bar{P} = \bar{g}_{CO}{}^{O}\bar{P}. \tag{3.5}$$

In general, both point P and $\mathcal{F}_C$ may be in motion w.r.t. the inertial frame $\mathcal{F}_O$. It is then relevant, also in view of the next developments, to determine the (apparent) velocity of a point P in $\mathcal{F}_C$, due to point P own motion and to the ego-motion of $\mathcal{F}_C$. Given a trajectory $\mathbf{R}_{CO}(t) : \mathbb{R} \to SO(3)$ that describes a smooth rotational motion, the following relationship links the rate of change of $\mathbf{R}_{CO}$ to the angular velocity ${}^{C}\boldsymbol{\omega}_{CO} \in \mathbb{R}^3$ of $\mathcal{F}_O$ w.r.t $\mathcal{F}_C$ expressed in $\mathcal{F}_C$:

$$\dot{\mathbf{R}}_{CO}\mathbf{R}_{CO}^{T} = [{}^{C}\boldsymbol{\omega}_{CO}]_{\times}. \tag{3.6}$$

It is easy to verify that the opposite quantity $-[{}^{C}\boldsymbol{\omega}_{CO}]_{\times} = [-{}^{C}\boldsymbol{\omega}_{CO}]_{\times} = [{}^{C}\boldsymbol{\omega}_{OC}]_{\times}$ represents the angular velocity of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$ expressed in $\mathcal{F}_C$. By differentiating w.r.t. time (3.5), one gets

$${}^{C}\dot{\bar{P}} = \dot{\bar{g}}_{CO}{}^{O}\bar{P} + \bar{g}_{CO}{}^{O}\dot{\bar{\mathbf{P}}} = \dot{\bar{g}}_{CO}\bar{g}_{OC}{}^{C}\bar{P} + \bar{g}_{CO}{}^{O}\dot{\bar{\mathbf{P}}} = {}^{C}\bar{V}_{CO}{}^{C}\bar{P} + \bar{g}_{CO}{}^{O}\dot{\bar{\mathbf{T}}}_{P}. \tag{3.7}$$

In this equation, ${}^{O}\dot{\bar{\mathbf{T}}}_{P} \in \mathbb{R}^4$ is the homogeneous representation of point P absolute own velocity in $\mathcal{F}_O$, and ${}^{C}V_{CO} = \dot{g}_{CO}g_{OC}$, called *twist*, is an element of the Lie Algebra $se(3)$ of the matrix group $SE(3)$. The homogeneous matrix representation of ${}^{C}V_{CO}$ is

$${}^{C}\bar{V}_{CO} = \begin{bmatrix} \dot{\mathbf{R}}_{CO}\mathbf{R}_{OC} & -\mathbf{R}_{CO}{}^{O}\dot{\mathbf{T}}_{OC} \\ \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} [{}^{C}\boldsymbol{\omega}_{CO}]_{\times} & -{}^{C}\dot{\mathbf{T}}_{OC} \\ \mathbf{0} & 0 \end{bmatrix} = \begin{bmatrix} -[{}^{C}\boldsymbol{\omega}_{OC}]_{\times} & -{}^{C}\dot{\mathbf{T}}_{OC} \\ \mathbf{0} & 0 \end{bmatrix},$$

where the pair $\left({}^{C}\dot{\mathbf{T}}_{OC}, {}^{C}\boldsymbol{\omega}_{OC}\right)$ stands for the linear/angular velocity of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$ expressed in $\mathcal{F}_C$. By expanding (3.7), one obtains the expression

$${}^{C}\dot{\bar{P}} = \begin{bmatrix} \mathbf{R}_{CO}{}^{O}\dot{\mathbf{T}}_{P} - {}^{C}\dot{\mathbf{T}}_{OC} - [{}^{C}\boldsymbol{\omega}_{OC}]_{\times}{}^{C}P \\ 0 \end{bmatrix} = \begin{bmatrix} {}^{C}\dot{\mathbf{T}}_{P} - {}^{C}\dot{\mathbf{T}}_{OC} - [{}^{C}\boldsymbol{\omega}_{OC}]_{\times}{}^{C}P \\ 0 \end{bmatrix} \tag{3.8}$$

with all quantities expressed in $\mathcal{F}_C$.

This fundamental kinematic relationship, describing how 3D points move in the camera frame as a consequence of camera and points own velocities, is at the core of many following developments. In order to simplify the notation, from now on we drop the dependency on $\mathcal{F}_C$ assuming that all quantities are expressed in the camera frame, unless otherwise stated. Hence, by letting ${}^{C}\dot{\mathbf{T}}_{P} = \mathbf{v}_P$, ${}^{C}\dot{\mathbf{T}}_{OC} = \mathbf{v}_C$ and ${}^{C}\boldsymbol{\omega}_{OC} = \boldsymbol{\omega}_C$, (3.8) can be rearranged in a more

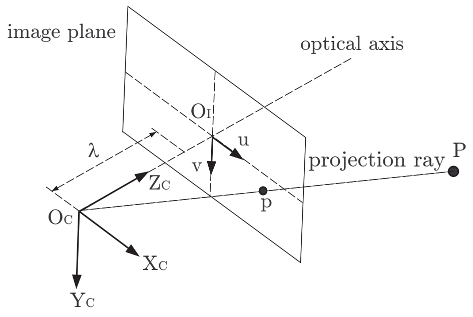


Figure 3.1: Pin-hole camera model. The 2D image p of a 3D point P is at the intersection of the ray going through the optical center O_C and the image plane at a distance λ in front of the optical center.

convenient matrix form

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_C - \mathbf{v}_P \\ \boldsymbol{\omega}_C \end{bmatrix}. \tag{3.9}$$

Note that the pair $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ (the camera linear/angular velocity) can be seen as an input of the first-order differential system defined by (3.9), while $\mathbf{v}_P$ (the absolute velocity of point P) represents an exogenous (and usually unmeasurable) quantity. Throughout the next Chapters, we always assume that $\mathbf{v}_P = \mathbf{0}$, i.e., that point P is fixed in the scene.

3.2 Pin-hole camera model

A camera, or in general an optical system, is composed by a set of lenses used to control direction of light towards a 2D surface sensitive to luminosity intensity like, e.g., photographic media or CCD arrays. As stated in the introduction, a thorough treatment of lenses and optics is far beyond the scope of this Thesis. The interested reader is therefore referred to the classical works of [Born & Wolf 1999; Stroebel 1999] for more details.

For our goals, a reasonable approximation consists in considering the camera optical system as a *thin lens* with almost zero aperture where all rays are forces to pass through. This conceptual model, said *pin-hole* camera, is represented in Fig. 3.1. The camera, associated to the moving frame $\mathcal{F}_C : \{O_C; \mathbf{X}_C, \mathbf{Y}_C, \mathbf{Z}_C\}$ with $\mathbf{Z}_C$ coincident with the camera optical axis, projects a 3D point P into a 2D point p onto the *image plane*. This plane, perpendicular to the optical axis, lies at a distance λ (the focal length) from O_C, and is endowed with a 2D reference frame $\mathcal{F}_I : \{O_I; \mathbf{u}, \mathbf{v}\}$ with axes parallel to $\mathbf{X}_C$ and $\mathbf{Y}_C$, respectively. According to this model, a 3D point $P = [X\ Y\ Z]^T$ is projected at the image point

$$p = \begin{bmatrix} p_u \\ p_v \end{bmatrix} = \frac{\lambda}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}.$$

In homogeneous coordinates, this relationship can be written as

$$Z\bar{p} = Z \begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \bar{P} = \mathbf{K}_\lambda \mathbf{\Pi}_0 \bar{P}, \tag{3.10}$$

where $\mathbf{K}_\lambda \in \mathbb{R}^{3\times3}$ is a nonsingular matrix, and $\mathbf{\Pi}_0 \in \mathbb{R}^{3\times4}$ is often referred to as the *standard projection matrix.*

The ideal model (3.10) represents a very particular choice of the camera frame $\mathcal{F}_C$, the 'canonical retinal frame', centered at the camera optical center and with one axis aligned with the camera optical axis. In practice, the images captured with, e.g., a digital camera are obtained in terms of pixels (i, j), with the origin of the image coordinate frame $\mathcal{F}_I$ typically in the upper-left corner of the image, with axis $\mathbf{u}$ and $\mathbf{v}$ not perfectly perpendicular, and so on. In order to render model (3.10) compatible with the actual pixel measurements yielded by a real camera, the relationship between the retinal frame coordinates and the pixel array coordinates must be specified. It turns out that the pixel coordinates $\tilde{p}$ of an image point p can be obtained through the linear transformation[2]

$$\bar{\tilde{p}} = \begin{bmatrix} k_u & k_u/\tan\delta & u_0 \\ 0 & k_v/\sin\delta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \bar{p} = \mathbf{K}_s \bar{p}, \tag{3.11}$$

where $[u_0\ v_0]^T$ are the pixel coordinates of the principal point (the intersection of $\mathbf{Z}_c$ with the image plane), k_u and k_v the magnifications in the $\mathbf{u}$ and $\mathbf{v}$ directions (in pixel/meters), and δ the angle between these axes. Therefore, from (3.10) and (3.11), it follows

$$Z\bar{\tilde{p}} = \mathbf{K}_s \mathbf{K}_\lambda \mathbf{\Pi}_0 \bar{P} = \mathbf{K}_C \mathbf{\Pi}_0 \bar{P},$$

with

$$\mathbf{K}_C = \begin{bmatrix} \lambda k_u & \lambda k_u/\tan\delta & u_0 \\ 0 & \lambda k_v/\sin\delta & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

[2]Actually, a more realistic image formation model should also include the nonlinear effects of radial distortion due to camera optics. However, in the rest of this work, we assume that radial distortion has been compensated for (see [Tsai 1987; Zhang 2000] for more details) so that (3.11) holds.

In the above equation the effect of a real camera is split into two distinct steps: first, the standard projection matrix $\mathbf{\Pi}_0$ projects point P w.r.t. a *normalized coordinate system*, i.e., as if $\lambda = 1$, $k_u = 1$, $k_v = 1$, $(u_0, v_0) = (0, 0)$, and $\delta = \pi/2$. Next, matrix $\mathbf{K}_C$ performs an additional transformation that depends on 'intrinsic' parameters of a particular camera, such as the focal length λ. Therefore, because of this role, matrix $\mathbf{K}_C$ is usually called the *intrinsic parameter matrix*, or *calibration matrix* of a given camera. Note that $\mathbf{K}_C$ is always nonsingular and, assuming that $\delta = \pi/2$ and $k_u = k_v$ (as in almost all cases), its expression reduces to

$$\mathbf{K}_C = \begin{bmatrix} \lambda & 0 & u_0 \\ 0 & \lambda & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

where now λ represents the focal length in pixels. When the calibration matrix is known, i.e., the camera is calibrated, the *normalized* coordinates p can always be recovered from their measured pixel counterparts as

$$Z\bar{p} = Z\mathbf{K}_C^{-1}\bar{\tilde{p}} = \mathbf{\Pi}_0\bar{P}, \tag{3.13}$$

thus allowing to work directly in the normalized space. Many (typically off-line) techniques yield an estimation of $\mathbf{K}_C$ from a sequence of images taken at different points of view w.r.t. a given (and known) target object. The classical works of [Tsai 1987; Weng *et al.* 1992] and the more recent [Sturm & Maybank 1999; Zhang 2000] provide a theoretical analysis and solution to the problem, while in [Strobl & Paredes 2005; Sepp *et al.* 2005; Bouguet 2007] actual implementations of these algorithms can be found. From now on, the calibration matrix $\mathbf{K}_C$ is supposed known and every subsequent development is carried out in the normalized space. The only exception takes place in Sect. 5.3 where the focal length λ is assumed unknown and estimated online during the camera motion via a suitable observation scheme.

The perspective pin-hole model described by (3.13) maps points $P \in \mathbb{R}^3$ to image points $p \in \mathbb{R}^2$ up to a scale factor (the depth Z of P). Since one dimension is lost through this mapping, one would intuitively expect that the information carried by an image cannot be fully representative of the original 3D object. Indeed, expression of (3.13) implies that any point $P' = \alpha P \in \mathbb{R}^3$, $\alpha \neq 0$, is projected onto the same image point $p \in \mathbb{R}^2$ with scale factor αZ. In other words, p is not the image of a single point in $\mathbb{R}^3$, but of all the points $P' \in \mathbb{R}^3$ lying on the line passing through P, i.e., it is the image of the direction of P. As a consequence, the image of any object $\mathcal{O}$ is undistinguishable from the image of an object $\mathcal{O}'$ with same shape but scaled size and distance — see Fig. 3.2. Due to this scale ambiguity, it is often convenient to rewrite (3.13) as

$$\bar{p} \sim \mathbf{\Pi}_0\bar{P} \tag{3.14}$$

where $\sim$ denotes equivalence up to an arbitrary scalar factor. Scale ambiguity plays a central role in computer vision. For instance, consider the following problem: from a set of input

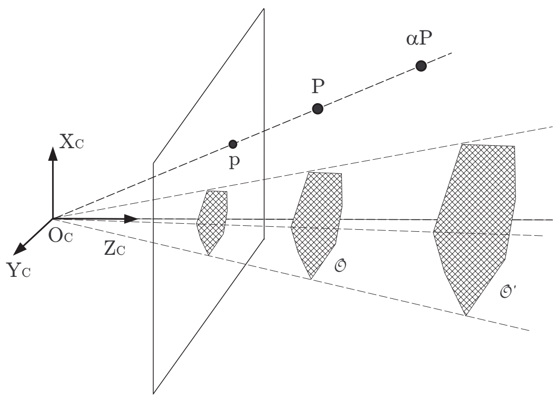


Figure 3.2: The pin-hole camera projection loses any scale information. Therefore, any point on the direction of P has the same image p, and any two objects $\mathcal{O}$ and $\mathcal{O}'$ with same shape but scaled size and depth are indistinguishable on the image plane.

images of a object $\mathcal{O}$ taken at different camera poses, recover the translations/rotations among the views. In absence of additional information, any multiple view reconstruction algorithm will yield a solution modulo the equivalence relation $\sim$. The actual amounts of translation can be recovered only if some (external) scaling information is introduced in the process, such as the size of $\mathcal{O}$ [Faugeras 1993]. This limitation will be also evident in Chapter 5 where the problem of 3D structure identification is formulated in the framework of nonlinear observers. In this context, as a consequence of scale ambiguity, 3D identification proves to be possible iff a suitable *persistency of excitation* condition is met, condition that in turn will require knowledge of some external 'scale' information: the magnitude of the camera linear velocity $\mathbf{v}_C$.

3.3 Geometry of two views

In this section we investigate how two images of the same 3D points, taken at different vantage points, are intrinsically related to their 3D positions and to the rigid camera displacements among the two poses. Goal of this geometric analysis is to obtain an algorithm that almost completely reconstructs the camera pose given two views of the same object. Such information will be then exploited by a class of visual servoing techniques in Chapter 4, and within the estimation algorithms proposed in Chapter 5. Apart from [Ma *et al.* 2004], the interested reader can refer to [Maybank 1992; Faugeras & Luong 2001] for a full theoretical treatment of the geometry of multiple views.

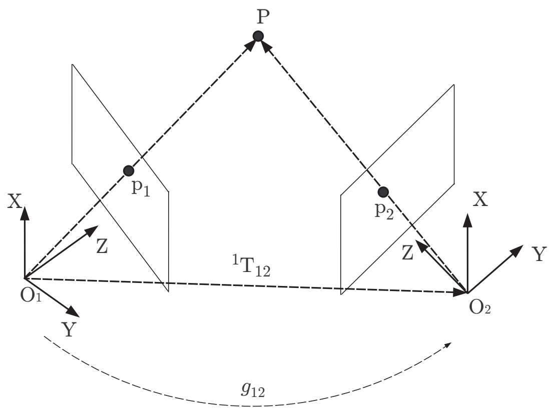


Figure 3.3: Two projections of a 3D point P from two camera poses with rigid displacement g_{12}. The two projection rays towards P and the translation vector from O_1 to O_2 lie on the same plane, giving rise to the so-called *epipolar constraint*. Because of the equivalence (3.14), the same constraint holds also if P is replaced by its two images p_1, p_2, since they retain the same direction in space.

3.3.1 Epipolar constraint

Consider a point P in space and a camera moving through the scene. Fix two camera poses and let $g_{12} = (\mathbf{R}_{12}, {}^1\mathbf{T}_{12})$ represent the rigid body transformation among them. The coordinates of P in the two camera frames and the translation vector ${}^1\mathbf{T}_{12}$ form a triangle, i.e., the three vectors $({}^1\mathbf{P}, {}^2\mathbf{P}, {}^1\mathbf{T}_{12})$ lie on the same plane (see Fig. 3.3). Such constraint, known as *epipolar constraint*, holds also if the pair $({}^1\mathbf{P}, {}^2\mathbf{P})$ is replaced by its image (p_1, p_2), since p and P identify the same direction in space. Hence, given enough image points of the same object, the epipolar constraint can be used to solve for the camera relative pose g_{12}.

An explicit expression of the epipolar constraint can be found by combining ${}^2\bar{\mathbf{P}} = g_{21}{}^1\bar{\mathbf{P}}$ and (3.13) as

$$Z_2\bar{p}_2 = {}^2\mathbf{T}_{21} + \mathbf{R}_{21}Z_1\bar{p}_1.$$

In order to eliminate the (unknown) depths Z_i, premultiply both sides by $\left[{}^2\mathbf{T}_{21}\right]_\times$ to obtain

$$Z_2\left[{}^2\mathbf{T}_{21}\right]_\times \bar{p}_2 = \left[{}^2\mathbf{T}_{21}\right]_\times \mathbf{R}_{21}Z_1\bar{p}_1. \tag{3.15}$$

Taking the inner product of both sides with $\bar{p}_2$ and dividing by $Z_1 > 0$ yields the searched depth-free epipolar constraint

$$\bar{p}_2^T\left[{}^2\mathbf{T}_{21}\right]_\times \mathbf{R}_{21}\bar{p}_1 = \bar{p}_2^T\mathbf{E}\bar{p}_1 = 0. \tag{3.16}$$

Matrix $\mathbf{E} = \left[{}^2\mathbf{T}_{21}\right]_\times \mathbf{R}_{21} \in \mathbb{R}^{3\times3}$, called the *essential matrix*, encodes the relative pose between the two views and belongs to a special set of matrices in $\mathbb{R}^{3\times 3}$, the *essential space*, defined as

$$\mathcal{E} = \left\{ [\mathbf{T}]_\times \mathbf{R} |\ \mathbf{R} \in SO(3),\ \mathbf{T} \in \mathbb{R}^3 \right\}. \tag{3.17}$$

It can be shown that a nonzero matrix $\mathbf{E}$ is an essential matrix iff the SVD of $\mathbf{E}$, $\mathbf{E} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, has $\mathbf{\Sigma} = \mathrm{diag}\{\sigma, \sigma, 0\}$, $\sigma > 0$ [Huang & Faugeras 1989]. Notice that the epipolar constraint (3.16) is linear in $\mathbf{E}$. Therefore, given a sufficient number of matched points, linear algebra tools allow to solve (3.16) for $\mathbf{E}$. In its original version, the *eight-point algorithm* [Longuet-Higgins 1981] recovers $\mathbf{E}$ from eight pairs of matched points, but further developments showed that one needs only six correspondent points in *general position* solve the problem. A set of 3D points is said to be in general position if it does not belong to some degenerate configurations, called critical quadric surfaces, which prevent an unique solution for $\mathbf{E}$. Many of these critical surfaces occur rarely in practice, so that their importance is limited. However, 2D planes, which are ubiquitous in most environments, happen to be a special subclass of critical surfaces, and therefore their case must be dealt with separately (see next section). Once $\mathbf{E}$ is known, it is possible to decompose it into the four solutions

$$([\mathbf{T}]_\times, \mathbf{R}) = \left(\mathbf{U}\mathbf{R}_Z\left(\pm\frac{\pi}{2}\right)\mathbf{\Sigma}\mathbf{U}^T,\ \mathbf{U}\mathbf{R}_Z^T\left(\pm\frac{\pi}{2}\right)\mathbf{V}^T \right), \tag{3.18}$$

where $\mathbf{R}_Z(\theta)$ stands for the rotation matrix about $\mathbf{Z}$ axis of angle θ. By imposing the positive depth constraint, i.e., that depths Z_i in (3.15) are both positive, one can eventually select the correct solution out of the four ones in (3.18).

It is worth noting that the epipolar constraint has also an homogeneous structure. Therefore, if $\mathbf{E}$ is a solution, $\alpha\mathbf{E}$ is a solution as well, with $\alpha \in \mathbb{R}$ being an arbitrary scale factor. As a consequence, the translation $\mathbf{T}$ in (3.18) is found up an arbitrary scale or, in other words, only the direction of $\mathbf{T}$ can be recovered from the decomposition of $\mathbf{E}$. Indeed, as stated at the end of Sect. 3.2, this is a consequence of the scale ambiguity inherently present in perspective systems that only allows the recovering of 3D information modulo the scale equivalence $\sim$ defined in (3.14).

3.3.2 Planar homography

In many applications like, e.g., man-made environments or aerial imaging, the scene is (approximately) planar, so that the matched points selected for 3D reconstruction lie on the same 2D plane. Being the plane a particular critical surface, the epipolar constraint may not yield a unique solution to the reconstruction problem. However, an additional constraint besides the epipolar one is also shared by the points on the plane. This constraint can then be used to obtain a meaningful solution also in this specific case.

With reference to Fig. 3.4, let $\mathbf{n} \in \mathbb{S}^2$ be the unit normal vector to plane π w.r.t. the first

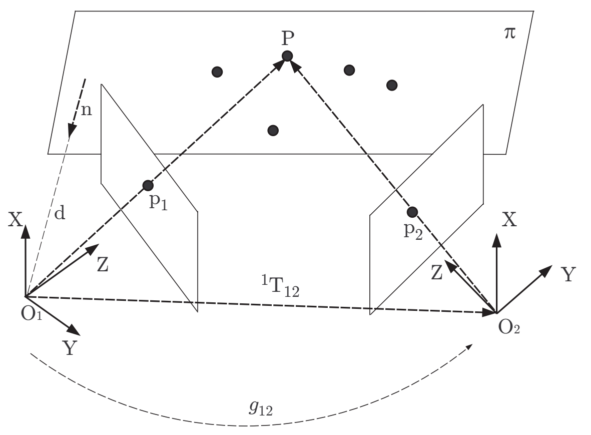


Figure 3.4: Two images p_1, p_2 of a 3D points P on a plane π. They are related by a homography $\mathbf{H}$ that is induced by the plane and by the rigid transformation g_{12} between the two camera poses.

camera frame, and $d > 0$ denote the distance of π from O_1. Then it is

$$\frac{1}{d}\mathbf{n}^T\, {}^1P = 1, \quad \forall P \in \pi. \tag{3.19}$$

By plugging (3.19) into ${}^2\bar{P} = g_{21}{}^1\bar{P}$, it follows

$$ {}^2P = \mathbf{R}_{21}{}^1P + {}^2\mathbf{T}_{21}\frac{1}{d}\mathbf{n}^T\, {}^1P = \left(\mathbf{R}_{21} + \frac{1}{d}\, {}^2\mathbf{T}_{21}\mathbf{n}^T\right) {}^1P = \mathbf{H}\, {}^1P. \tag{3.20}$$

Matrix $\mathbf{H} \in \mathbb{R}^{3\times3}$ is called *(planar) homography matrix* and linearly maps 3D points between the two camera poses. From $p_1 \sim {}^1P$ and $p_2 \sim {}^2P$ we can express (3.20) in terms of image points as

$$p_2 \sim \mathbf{H}p_1.$$

The unknown scale factor can then be eliminated by multiplying both sides by matrix $[p_2]_\times$, yielding the constraint

$$[p_2]_\times \mathbf{H}p_1 = 0,$$

which, similarly to (3.16), is homogeneous and linear in $\mathbf{H}$. Hence, again, if a sufficient number of matched points is available, $\mathbf{H}$ can be recovered up to an arbitrary scalar factor. It turns out that only four pairs are necessary in this case, provided that no three of them are collinear, i.e., that they are in a general configuration in the plane. Knowing $\mathbf{H}$, the next step consists in decomposing it into g_{21} and $(\mathbf{n}, d)$. As one could expect, such decomposition cannot be exact due to the scale ambiguity effect. Indeed, while $\mathbf{R}_{21}$ and $\mathbf{n}$ can be fully reconstructed, only the ratio ${}^2\mathbf{T}_{21}/d$, i.e., the translation vector scaled by the distance to the plane, is recoverable.

Hence, as with the epipolar constraint, the actual magnitude of ${}^{2}\mathbf{T}_{21}$ is out of reach without some external scale information. Furthermore, as with the essential matrix $\mathbf{E}$, the decomposition of $\mathbf{H}$ yields four possible solutions, but only two of them can be discarded thanks to the positive depth constraint. The remaining two solutions are both physically valid and apriori undistinguishable. As will be explained in Chapter 4, the 'true' solution can be isolated via some additional hypotheses on the scene structure, such as (independent) knowledge of the plane normal direction $\mathbf{n}$ in one of the two frames.

Finally, it is worth citing the recent works of [Chesi *et al.* 2000; Benhimane & Malis 2004] in which the authors propose an algorithm able to estimate $\mathbf{H}$ directly on two input images of a dense unstructured object, i.e., without the explicit need of extracting and matching pairs of points. A free implementation of this method can be found at [Malis *et al.* 2004].

4 Visual Servoing

Visual servoing (VS) refers to the use of cameras (seen as sensors) to control the motion and the pose of a robot, thus endowing it with the ability to perceive, actually to ‘see’, the external world. In a wide sense, visual servoing represents more a general paradigm than a specific set of techniques: it encompasses and combines fields ranging from image processing, computer vision, pattern classification, and control theory. The vision data may be acquired from a camera rigidly mounted on the robot (*eye-in-hand* configuration), or the camera may be fixed in the workspace so that it can externally observe both the robot and the scene (*eye-to-hand* configuration). Configurations with multiple on/offboard cameras are also possible as in, e.g., stereovision applications. The mathematical developments of all these possibilities are conceptually equivalent, and in the following we will consider only the case of single eye-in-hand camera.

Among the many VS facets, the focus of this Thesis is set on the Control Theory point of view. Therefore, a detailed analysis of all the image processing and computer vision issues common to most applications is not considered here. As stated in the introduction, we assume that scene interpretation is solved by means of an external module able to provide the control part with all the needed information, and we shift the attention to the stability and convergence properties of VS feedback laws. Note that, however, image analysis and interpretation is seldom a trivial task and considerable efforts are usually devoted to design, implementation and tuning of suitable algorithms for the specific case. Indeed, there exists a vast literature on these topics so that they can be considered as a stand-alone engineering area with its own methodologies and solutions. The reader can refer to [Russ 1998; Young *et al.* 1998; Duda *et al.* 2000; Gonzalez & Woods 2002] for a presentation of the most popular methods.

In the following, Sect. 4.1 provides a general overview of the VS control framework, while Sects. 4.2–4.4 illustrate the state of the art of most existing VS techniques from a theoretical point of view. Next, Sect. 4.5 discusses how VS laws can be actually implemented on a robot manipulator within the kinematic control framework developed in Chapters 1–2, and Sect. 4.6 presents some simulation results of VS schemes implemented on NMMs carrying a camera on the end-effector.

4.1 Overview

Roots of Visual Servoing trace back to the end of '70s when the first seminal attempts to close a feedback loop on visual information were investigated [Hill & Park 1979; Sanderson & Weiss 1980]. These pioneering works were mainly affected by the slow sample rate due to poor performance of vision hardware, so that a 'first look, then move blindly, then look again' result was almost unavoidable. The increase of computing power, and thus of better computer vision algorithms, experienced during the following decades, permitted a more 'natural' behavior during camera motion. The emphasis could slowly shift from ad-hoc software/hardware solutions to more general VS concepts [Sanderson & Weiss 1983; Weiss *et al.* 1987; Wijesoma *et al.* 1993]. In these years, a significant contribution to the the VS mathematical modeling was given by [Espiau *et al.* 1992], where the authors formulated in a rigorous way the kinematic relationships between a moving pin-hole camera and the motion induced on image plane quantities, such as projections of points, lines, planes, circles, spheres, etc. These ideas, further investigated in [Corke 1994], were organically reviewed and merged with topics from computer vision and robot control in the famous Corke's book [Corke 1997] that represented for many years a milestone for VS researchers. Another notable contribution was given by the Hutchinson-Corke-Hager tutorial [Hutchinson *et al.* 1996] meant to present, in a compact and didactic fashion, the basic VS modeling concepts. In the last years, VS solutions bloomed thanks to the work of many researchers in the field. Schemes ranging smoothly from full 3D reconstructions to pure 2D image-based feedbacks were proposed, and a big effort was devoted to determine local (or global) stability conditions w.r.t. noise, calibration accuracy, modeling errors, and so on. A nice picture of the current VS state of the art has been recently proposed in [Chaumette & Hutchinson 2006a,b].

Despite the big variety of the proposed schemes, however, an analysis based on the Control

Theory point of view yields a unique formulation for all existing vision-based feedbacks. This is obtained by interpreting any visual quantity as a task variable, with a form close to (1.14), and visual control as a task regulation problem. The visual task variables may represent either direct 2D image measurements or 3D parameters estimated online, or a combination of both. A convenient and unified representation of all possibilities is then given by the expression

$$\mathbf{r} = \mathbf{f}(\mathbf{m}(t),\, \boldsymbol{\chi}(t)) \in \mathbb{R}^s. \tag{4.1}$$

Here, $\mathbf{m}(t)$ stands for a set of image measurements taken from a target object (the so-called *image features*) and $\boldsymbol{\chi}(t)$ is a vector representing additional (and not directly measurable) 3D information associated to the object. Note that, choice of the task variables is not trivial: one must always guarantee that there exists an isomorphism between $\mathbf{r}$ and the camera pose w.r.t. the target or, in other words, that regulation of $\mathbf{r}(t)$ to a desired value $\mathbf{r}_d$ strictly implies regulation of the camera pose to the spatial configuration associated to $\mathbf{r}_d$. In any case, if a stationary target is assumed, time variation of $\mathbf{r}(t)$ results from the apparent motion induced by the camera velocity $(\mathbf{v}_C,\, \boldsymbol{\omega}_C)$, and can be generically expressed as

$$\dot{\mathbf{r}} = \mathbf{J}_r(\mathbf{m}(t),\, \boldsymbol{\chi}(t)) \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} \tag{4.2}$$

where the $s \times 6$ matrix $\mathbf{J}_r$ is named the *interaction matrix* related to $\mathbf{r}$. Once such task-oriented formulation is adopted, any VS task can be executed by applying one of the existing task realization algorithm as, e.g., the schemes presented in Chapter 2, with the interaction matrix $\mathbf{J}_r$ playing the role of a task Jacobian. Of course, some care is required in actual implementations since both $\mathbf{r}$ and $\mathbf{J}_r$ may depend on vector $\boldsymbol{\chi}(t)$ which, in general, cannot be directly obtained from instantaneous measurements on the image plane. Moreover, in eye-in-hand configurations, where camera motion is generated by the supporting robotic structure, robot kinematics must also be taken into account in the rhs of (4.2) as will be discussed in Sect. 4.5.

Most of existing VS approaches stem from formulation (4.1–4.2) by differing in the number and kind of visual quantities included in the task vector. When $\mathbf{r}$ is made of a set of 3D parameters describing the pose of the target w.r.t. the camera (the relative translation/rotation), one speaks of *Position-Based Visual Servoing* (PBVS) [Wilson *et al.* 1996; Taylor *et al.* 2000]. In PBVS control, vector $\mathbf{r}$ is estimated from image measurements through a pose reconstruction algorithm, and the obtained cartesian error is used to move the camera/robot system towards its goal. Usually, PBVS methods take advantage of this cartesian formulation by yielding nice camera trajectories in space, but, on the other hand: *(i)* an a priori 3D model of the target is needed in order to fully reconstruct the relative camera pose; *(ii)* any uncertainty in the camera calibration parameters leads to 3D reconstruction errors and, as a consequence, to inaccurate task execution; *(iii)* there is no direct control on the motion of the features on the image plane, so that an object of interest (e.g., the target itself) may leave the field of view during the motion, causing the failure of the servoing task. Section 4.2 presents an illustrative example where pros/cons of PBVS schemes are examined.

If, on the contrary, $\mathbf{r}$ only consists of image features, the VS scheme is said Image-Based Visual Servoing (IBVS) [Weiss *et al.* 1987; Feddema & Mitchell 1989; De Luca *et al.* 2007a]. In IBVS methods, the task error can be directly evaluated on the image plane — a pose reconstruction step is not needed. However, the interaction matrix $\mathbf{J}_r$ in (4.2) still depends on certain 3D information $\boldsymbol{\chi}(t)$ which must be known, or approximated, in real implementations. In any case, since the error is built on image measurements, IBVS presents some advantages compared to pure PBVS schemes: convergence is generally more robust w.r.t. disturbances and uncertainties in the camera/robot model [Espiau 1993], and direct control of the feature motion on the image plane allows the implementation of strategies aimed at keeping the target always in the field of view of the camera [Corke & Hutchinson 2001]. Since in the following Chapters most of the developments are based on this framework, a detailed analysis of IBVS schemes is given in Sect. 4.3.

As a final case, vector $\mathbf{r}$ can also be a mixture of 2D/3D quantities. In this case an Hybrid Visual Servoing (HVS) is obtained, with the end of combining the advantages of both previous methods [Malis *et al.* 1999; Malis & Chaumette 2002]. Typically, HVS settings decouple rotation control from the translation part by exploiting the partial pose estimation algorithms presented in Sect. 3.3. Indeed, knowledge of the rotation matrix between current and desired views allows to regulate the camera rotational error independently in cartesian space. Control of the translation can then be achieved by considering suitable image quantities such as the coordinates of a reference point on the target object. One of the main advantages of HVS is that, in some cases, necessary and sufficient conditions for global stability can be found, while IBVS methods have a local convergence. Furthermore, cartesian control of rotation typically implies 'nicer' (i.e., more predictable) camera trajectories in space during the servoing w.r.t. the pure IBVS case. A brief review of basic HVS concepts is given in Sect. 4.4.

4.2 Position-based visual servoing

According to the notation introduced in (4.1), PBVS methods are characterized by a task vector made of sole 3D quantities

$$\mathbf{r} = \mathbf{f}(\boldsymbol{\chi}(t)) \in \mathbb{R}^s,$$

making it possible to control the motion directly in cartesian space. Consider the current camera frame $\mathcal{F}_C$, and the desired camera frame $\mathcal{F}_{C_d}$ fixed w.r.t. the target object. By comparing the current and desired views, and by exploiting the knowledge of a 3D model of the target and of the camera calibration matrix K_C, the rigid displacement $g_{C_dC} = ({}^{C_d}\mathbf{T}_{C_dC}, \mathbf{R}_{C_dC})$ between the two frames can be recovered [Lowe 1987; Dementhon & Davis 1995]. In this case, one

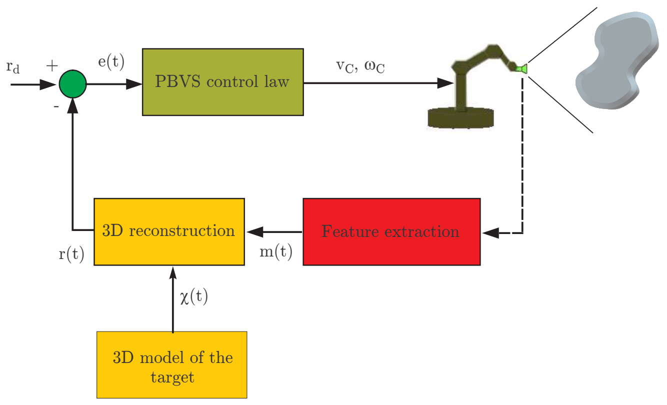


Figure 4.1: Conceptual scheme of PBVS. The image features $\mathbf{m}(t)$ are used in conjunction with a 3D model of the target $\boldsymbol{\chi}(t)$ to recover the current camera pose $\mathbf{r}(t)$. The cartesian error $\mathbf{r}_d - \mathbf{r}(t)$ is then exploited to control motion of the camera towards the desired pose.

can set $\mathbf{r}(t) \simeq \boldsymbol{\chi}(t) = g_{C_dC}(t)$ and drive the camera towards $\mathcal{F}_{C_d}$ by regulating the value of $\mathbf{r}(t)$ in cartesian space, see Fig. 4.1 for a conceptual scheme. As an example, consider a minimal parameterization of $\mathbf{R}_{C_dC}$, such as the angle/axis vector $\theta\mathbf{u} \in \mathbb{R}^3$. One can choose $\mathbf{r} = ({}^{C_d}\mathbf{T}_{C_dC},\, \theta\mathbf{u}) \in \mathbb{R}^s$, $s = 6$, thus obtaining the 6×6 square interaction matrix $\mathbf{J}_r$

$$\mathbf{J}_r = \begin{bmatrix} \mathbf{R}_{C_dC} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix} \tag{4.3}$$

where $\mathbf{L}_{\theta\mathbf{u}}$ is given by [Malis *et al.* 1999]

$$\mathbf{L}_{\theta\mathbf{u}} = \mathbf{I}_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\operatorname{sinc}\theta}{\operatorname{sinc}^2\dfrac{\theta}{2}}\right)[\mathbf{u}]_\times^2. \tag{4.4}$$

Regulation of $\mathbf{r}$ to $\mathbf{r}_d = (\mathbf{0},\, \mathbf{0})$ can then be easily achieved by inverting (4.2) according to the algorithm detailed in Sect. 2.1.

This PBVS scheme guarantees decoupled and exponential convergence of translational and rotational motions ($\mathbf{J}_r$ is block diagonal), in particular the camera moves along a straight line and rotates by following a geodesic on $SE(3)$. Moreover, matrix $\mathbf{J}_r$ is nonsingular as long as $\theta \neq 2k\pi$, $k \neq 0$ (see (4.4)), practically ensuring global convergence since the object of interest is always supposed to lie in front of the camera at the beginning of the servoing, i.e., with $|\theta| < 2\pi$. Note that, however, g_{C_dC} is an estimated quantity, typically an approximation of the actual displacement because of, e.g., inaccuracies in the camera calibration or in the target 3D model. Therefore, a coarse estimation will affect at the same time both the transient behavior and the final pose accuracy. Indeed, since g_{C_dC} appears both in the task definition and in the

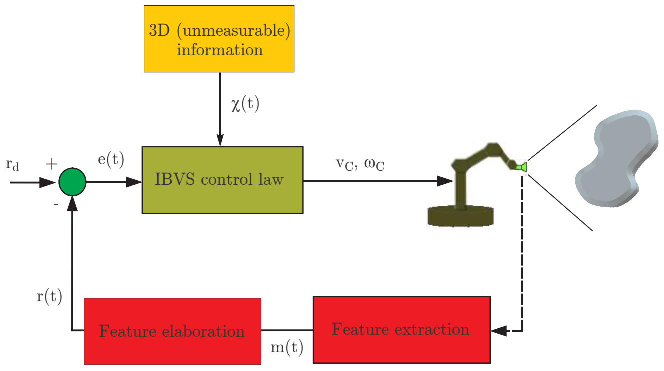


Figure 4.2: Conceptual scheme of IBVS. The current image features $\mathbf{m}(t)$ are elaborated into the image task vector $\mathbf{r}(t)$ which is then compared to the desired $\mathbf{r}_d$ in order to obtain an error $\mathbf{e}(t)$ expressed on the image plane. By using $\mathbf{e}(t)$ and some additional 3D information $\boldsymbol{\chi}(t)$, the IBVS controller computes the motion commands driving the camera towards its desired pose.

interaction matrix, stability may be lost because of coarse computation of $\mathbf{J}_r$, and a zero task error may not guarantee complete reaching of the final pose because of coarse evaluation of $\mathbf{r}$. In addition, no direct control over the image plane motion is possible, so that a temporary loss of the target in the camera field of view may be experienced during the servoing. As will be explained in the next section, many of these drawbacks can be alleviated by adopting an IBVS scheme, especially for what concerns final accuracy and control over the image plane motion. The price to pay mainly consists in weaker (local) stability conditions, less predictable camera motion in cartesian space, and existence of singularities and local minima for the feedback law.

4.3 Image-based visual servoing

IBVS methods represent, in some sense, the antipodal case w.r.t. PBVS approach. The idea is to minimize as much as possible the need of 3D quantities, and to work directly on the image plane by defining visual task (4.1) as a function of sole image features

$$\mathbf{r} = \mathbf{f}(\mathbf{m}(t)) \in \mathbb{R}^s. \tag{4.5}$$

Possible choices include point coordinates, line parameters, image moments, and so on. In principle, any image feature could be used as visual task, provided that its interaction matrix is known in closed form[1], see [Espiau *et al.* 1992] and [Chaumette 2004; Tahri & Chaumette 2005] for many examples involving 2D/3D geometric primitives and 2D image moments. If

[1]There also exist numerical techniques which yield an estimation of $\mathbf{J}_r$ [Lapresté *et al.* 2004; Piepmeier *et al.* 2004]. The drawbacks of these methods is that no theoretical stability and robustness analysis can be obtained.

the task is properly chosen so that there exists an isomorphism between $\mathbf{r}(t)$ and the current camera pose w.r.t. the target, one can again invert (4.2) and compute the motion commands that realize the visual task without the need of an explicit 3D pose recovering step. As depicted in Fig. 4.2, however, some 'external' 3D information $\boldsymbol{\chi}(t)$ is still required by the control. Indeed, it turns out that the interaction matrix of any image feature included in $\mathbf{r}$ inevitably depends also on a suitable $\boldsymbol{\chi}(t)$ associated to the selected features. For instance, for an image point p projection of a 3D point P, $\boldsymbol{\chi}(t)$ becomes the (unknown) depth Z of P. All implementations of IBVS schemes must then face this problem, and estimate/approximate $\boldsymbol{\chi}(t)$ during the servoing. However, in contrast with the PBVS case, coarse estimation of $\boldsymbol{\chi}(t)$ affects $\mathbf{J}_r$ but not $\mathbf{r}$ which is directly measured on the image plane. As a consequence, IBVS may show a poor transient behavior (stability can be an issue) but, if the scheme converges, final accuracy is unaffected by calibration/modeling errors. On the other hand, while matrix $\mathbf{J}_r$ in (4.3) is square (six dofs cartesian task for a fully actuated system) and practically free of singularities, IBVS interaction matrices may:

1. be more prone to ill-conditioning;
2. have more rows than columns if $s > 6$ features are chosen in (4.5). In such cases, the structural existence of a nontrivial null-space gives rise to local minima configurations that may prevent convergence of the servoing [Chaumette 1998].

The next section presents explicit expressions of the interaction matrices associated to some relevant image features, while Sect. 4.3.2 further delves into IBVS stability issues.

4.3.1 The interaction matrix

Point features

A point feature p (projection of a 3D point P) is the simplest shape one can consider when processing camera images with the aim of extracting structured information. Point features are easily detectable in a large variety of situations (corners, bright spots, etc.) and considerable efforts have been devoted for design and implementation of reliable tracking/matching softwares. Among the vast literature on this topic, the popular Lucas–Kanade algorithm [Lucas & Kanade 1981; Lucas 1984] deserves a special mention because of its handiness and robustness. Free implementations can be found in [Birchfield 2007; Intel 2007]. Because of their simplicity, point features were the first items considered for visual servoing purposes, and, nowadays, still continue to have a major role in most applications.

An explicit derivation of the point feature interaction matrix ($\mathbf{J}_p$ from now on) can be found by differentiating w.r.t. time (3.13) to express motion of $\dot{\mathbf{p}}$ in terms of $\dot{\mathbf{P}}$, and by using the kinematic relationship (3.9) to take into account camera velocity. These computations lead to the well-known expression

$$\dot{\mathbf{p}} = \begin{bmatrix} -\dfrac{1}{Z} & 0 & \dfrac{p_u}{Z} & p_u p_v & -(1+p_u^2) & p_v \\ 0 & -\dfrac{1}{Z} & \dfrac{p_v}{Z} & 1+p_v^2 & -p_u p_v & -p_u \end{bmatrix} \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \mathbf{J}_p(p,\, Z) \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix}. \tag{4.6}$$

As expect, the 2×6 matrix $\mathbf{J}_p$ depends on 2D image quantities, the point feature itself, and on additional 3D information, the depth Z of point P. Therefore, in this case it is $\mathbf{m}(t) = p(t)$ and $\boldsymbol{\chi}(t) = Z(t)$.

If a vector $\mathbf{f} = [p_1^T \dots p_n^T]^T \in \mathbb{R}^{2n}$ of n point features is considered, the total interaction matrix $\mathbf{J}_f$ can be obtained as the stack of n matrices $\mathbf{J}_p$ of the form (4.6), each one accounting for one point feature

$$\begin{bmatrix} \dot{p}_1 \\ \vdots \\ \dot{p}_n \end{bmatrix} = \mathbf{J}_f(\mathbf{f},\, Z) \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{p_1}(p_1,\, Z_1) \\ \vdots \\ \mathbf{J}_{p_n}(p_n,\, Z_n) \end{bmatrix} \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix}, \tag{4.7}$$

being $\mathbf{Z} = [Z_1 \dots Z_n]^T \in \mathbb{R}^n$ the vector of the depths associated to the n feature points. Of course, in this case we have $\mathbf{m}(t) = [p_1(t) \dots p_n(t)]^T$ and $\boldsymbol{\chi}(t) = [Z_1(t) \dots Z_n(t)]^T$.

The kinematic relationship (4.6) can be further exploited to obtain other closed-form interaction matrices. Indeed, since all shapes are basically made of points, matrix $\mathbf{J}_p$ acts as a building block for virtually all the possible structures one can consider as, e.g., length and orientation of a segment, surface and mass center of a polygon, plane orientations, geometric parameters of 2D/3D primitives (circles, ellipses, spheres, cylinders), and so on [Chaumette 1990; Espiau *et al.* 1992]. In all cases, however, dependence on additional 3D information is always present. Finally, as discussed in the next section, it is also possible to consider the interaction matrix of 'global' (integral) features like image moments, instead of local descriptors like feature points.

Moments

In the computer vision community, image moments have represented for a long time a valuable tool to solve pattern-recognition problems. Indeed, moments provide a global/integral representation of any shape that can be segmented in the image plane, and suitable combinations of them can be used to capture, in some sense, the fingerprint of an object of interest [Hu 1962; Mukundan & Ramakrishane 1998]. Moreover, evaluation of moments is typically free of the so-called correspondence problem, i.e., tracking and matching of individual structures during the camera motion. This step may not be always easy or convenient — think to dense objects as spheres, ellipsoids, etc. All these nice properties naturally motivated the study of possible applications for IBVS schemes and, after several attempts, a main contribution was given by

Chaumette in [Chaumette 2004] where the analytical form of the interaction matrix related to any 2D moment was derived. These results, valid for a single dense region on the image plane, were also extended to the case of moments computed on a set of distinct points [Tahri & Chaumette 2005]. Since, in this Thesis, both region-based and point-based moments are considered for IBVS, some basic definitions and relationships relevant for the next developments are presented hereafter.

Let $\mathcal{R}$ be a dense and closed image plane region resulting from the projection of a 3D object $\mathcal{O}$. The 2D *region-based moments* m_{ij} of order $i + j$ are defined as

$$m_{ij} = \iint_{\mathcal{R}} x^i y^j \mathrm{d}x\mathrm{d}y$$

where $(x,\, y)$ represent a 2D point on the image plane. The zeroth-order moment m_{00} is simply the area a of the region $\mathcal{R}$. The *centered moments* μ_{ij} are moments computed w.r.t. the barycenter $(x_g,\, y_g)$ of $\mathcal{R}$ as

$$\mu_{ij} = \iint_{\mathcal{R}} (x - x_g)^i (y - y_g)^j \mathrm{d}x\mathrm{d}y$$

with $x_g = m_{10}/a$ and $y_g = m_{01}/a$. Finally, one can also define the *normalized centered moments*

$$n_{ij} = \mu_{ij}/a. \tag{4.8}$$

Similarly, given a discrete set of n image points, the *point-based moments* are defined by

$$m_{ij} = \sum_{l=1}^{n} x_l^i y_l^j,$$

while the centered moments become

$$\mu_{ij} = \sum_{l=1}^{n} (x_l - x_g)^i (y_l - y_g)^j$$

with $x_g = m_{10}/n$ and $y_g = m_{01}/n$ ($m_{00} = n$ in this case), and, as before $n_{ij} = \mu_{ij}/m_{00} = \mu_{ij}/n$. In order to use moments within the IBVS framework, one needs an analytical expression of the interaction matrix relating the rate of change of a moment to the camera velocity $(\mathbf{v}_C,\, \boldsymbol{\omega}_C)$, similarly to what done in the point features case (4.6). If the object $\mathcal{O}$ can be approximated as being planar, or as having a planar limb surface [Espiau *et al.* 1992], a closed-form expression can be easily obtained. To this end, let

$$\mathbf{n} \cdot P + d = 0 \tag{4.9}$$

be the limb plane equation in the camera frame, where $\mathbf{n} = [n_x\ n_y\ n_z]^T \in \mathbb{S}^2$ is the plane unit normal and d the plane distance to the origin of $\mathcal{F}_C$. The depth Z of any 3D point P lying on this plane can be expressed in terms of its image coordinates p as

$$\frac{1}{Z} = Ap_u + Bp_v + C, \tag{4.10}$$

where

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = -\mathbf{n}/d. \tag{4.11}$$

The interaction matrix $\mathbf{J}_{m_{ij}}$ of moment m_{ij}, either region-based or point-based, can then be written as

$$\dot{m}_{ij} = \mathbf{J}_{m_{ij}}(m_{kl}, \boldsymbol{\chi}) \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix}, \tag{4.12}$$

where m_{kl} stands for generic (k, l)-th moments of order up to $i + j + 1$, and $\boldsymbol{\chi} = [A\,B\,C]^T$. In particular, it is worth noting that (4.12) can be rearranged linearly in (A, B, C) as

$$\begin{aligned} \dot{m}_{ij} =& A\lambda_A(m_{kl}, \mathbf{v}_C) + B\lambda_B(m_{kl}, \mathbf{v}_C) + C\lambda_C(m_{kl}, \mathbf{v}_C) + \\ & \lambda_D(m_{kl}, \boldsymbol{\omega}_C), \end{aligned} \tag{4.13}$$

where $\lambda_i(\cdot)$ are suitable scalar functions. From (4.12), one can also obtain the interaction matrix of centered and normalized moments, respectively. For instance, in the region-based case, (4.8) and the relation

$$\mu_{ij} = \sum_{k=0}^{i} \sum_{l=0}^{j} \binom{i}{k} \binom{j}{l} (-x_g)^{i-k} (-y_g)^{j-l} m_{kl}$$

allow a direct computation. The same holds also for the point-based case. It is worth noting that, when considering moments, the 3D information represented by $\boldsymbol{\chi}$ always reduces to the plane normal $\mathbf{n}$ scaled by the plane distance d, i.e., the 'depth' of the plane, despite the shape or number of points considered. On the other hand, compared to the pure point features case, some additional scene structure information must be known besides depth, i.e., the plane current orientation in the camera frame.

4.3.2 Stability analysis of IBVS

As discussed in the previous section, IBVS schemes possess many practical advantages w.r.t. PBVS in terms of final accuracy, robustness against calibration errors, and control over image trajectories. As in all engineering solutions, however, such benefits come at a price. First of all, the interaction matrix can become singular during the servoing, or false equilibria may be reached due to the presence of unrealizable feature motions [Chaumette 1998]. Additionally, the actual value of the 3D information $\boldsymbol{\chi}(t)$ is commonly unknown, and some estimate must be used (for example, the constant value at the desired pose). Thus, the convergence of such schemes can be guaranteed only locally [Malis & Rives 2003]. Local convergence may also result from a rough approximation of the camera intrinsic parameters. If the camera is not accurately calibrated, or if its intrinsic parameters are changing over time, it is still possible to plan a path for the features in a suitable *invariant space* [Malis 2004], but any global convergence property for the servoing scheme is lost.

Many of these stability issues can be conveniently tackled thanks to task-oriented point of view introduced in Chapter 2. To this end, let $\mathbf{e}(t) = \mathbf{r}_d - \mathbf{r}(t) \in \mathbb{R}^s$ be the task error on the image plane, and

$$\dot{\mathbf{e}} = -\mathbf{J}_r(\mathbf{m}(t),\, \boldsymbol{\chi}(t)) \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix}$$

the corresponding error dynamics, with $\mathbf{J}_r$ being the interaction matrix associated to $\mathbf{r}$. As with PBVS, any task realization algorithm can be adopted to fulfill task (4.5). For instance, the simple feedback

$$\begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \mathbf{J}_r^{\dagger}(\mathbf{m}(t),\, \boldsymbol{\chi}(t))\mathbf{K}\mathbf{e}, \quad \mathbf{K} > 0, \tag{4.14}$$

would guarantee a linear, decoupled, and exponentially stable closed-loop behavior

$$\dot{\mathbf{e}} = -\mathbf{K}\mathbf{e}, \tag{4.15}$$

for task error $\mathbf{e}$. Clearly, this represents just an ideal case since in practice one does not have full knowledge of matrix $\mathbf{J}_r$, typically because of poor knowledge of the camera intrinsic parameters in (3.12), and because of uncertainties on the value of $\boldsymbol{\chi}(t)$. Therefore, only an approximation $\widehat{\mathbf{J}}_r$ of the real interaction matrix is commonly available when implementing IBVS schemes. Closed-loop stability of this 'approximated' feedback can be assessed by considering the Lyapunov candidate function $V(\mathbf{e}) = \frac{1}{2}\mathbf{e}^T\mathbf{e}$, whose time derivative is given by

$$\dot{V} = -\mathbf{e}^T\mathbf{J}_r\widehat{\mathbf{J}}_r^{\dagger}\mathbf{e},$$

where, for simplicity, we set $\mathbf{K} = \mathbf{I}_s$. From standard Lyapunov theory [Khalil 2002], global asymptotic stability is ensured if the sufficient condition

$$\mathbf{J}_r\widehat{\mathbf{J}}_r^{\dagger} = \mathbf{P} > 0 \tag{4.16}$$

is met $\forall\, \mathbf{e} \in \mathbb{R}^s$.

In order to further proceed with the analysis, consider first the case of $s \leq 6$ features used for IBVS, so that the $s \times 6$ matrix $\mathbf{J}_r$ does not have more rows than columns. If $\mathbf{J}_r$ and $\widehat{\mathbf{J}}_r^{\dagger}$ have full rank s, matrix $\mathbf{P}$ is nonsingular and mild approximations in $\widehat{\mathbf{J}}_r^{\dagger}$ do not usually violate condition (4.16) in some neighborhood of the origin. This motivates the popular choice of letting $\widehat{\mathbf{J}}_r = \mathbf{J}_r(\mathbf{m}(t),\, \boldsymbol{\chi}_d)$ where $\boldsymbol{\chi}_d$ represents the constant value of $\boldsymbol{\chi}(t)$ at the desired pose. Indeed, in a neighborhood of the goal, one has $\boldsymbol{\chi}(t) \simeq \boldsymbol{\chi}_d$ so that $\mathbf{J}_r \simeq \widehat{\mathbf{J}}_r$, and (4.16) is trivially satisfied. However note that, besides the intrinsic local nature of this choice, one must still face the problem of obtaining $\boldsymbol{\chi}_d$. In most applications, an off-line estimation can be performed during the preliminary teaching of the desired image, but there also exist cases when such a solution is not feasible. For instance, in a navigation/exploration task, a mobile robot could store several images of interesting locations while exploring the environment, but may not have the possibility to recover at the same time the corresponding needed 3D information. In this respect, Chapter 5 is fully devoted to the development of a set of general tools aimed at obtaining an online converging observation of $\boldsymbol{\chi}(t)$ to be used in place of other approximations such as $\boldsymbol{\chi}_d$.

Things can also get worse if $s > 6$ features are selected for IBVS. In general, focusing on many features is a sound choice since it smooths the problem of possible loss of tracking during the camera motion. Besides that, there also exist some cases in which more than six features are required to control the full six dofs camera pose. For instance, relying on three noncollinear points (with $s = 6$) is theoretically sufficient for pose control. Unfortunately, in this case the associated 6×6 interaction matrix $\mathbf{J}_f$ can become singular, and four distinct camera poses for which $\mathbf{e} = \mathbf{0}$ (four undistinguishable global minima) necessarily exist [Michel & Rives 1993]. Here, a global (local) minimum represents a configuration where a nonzero error $\mathbf{e}$ results in a null camera velocity command, and the system stops away from its desired pose. Because of these limitations, four or more points are typically used, resulting in a task with dimension $s > 6$. However, in all such cases, being matrix $\mathbf{P}$ at most of rank 6, a structural nontrivial null-space of dimension $\Delta = \dim \ker \mathbf{P} \geq s - 6$ is always present. As a consequence, whenever $\mathbf{e} \in \ker \mathbf{P}$, additional *local minima* configurations arise. One can only prove that, in some neighborhood of $\mathbf{e} = \mathbf{0}$, no local minima can be encountered. Therefore, only local stability is guaranteed, and, in most cases, an analytical characterization of the stability domain is hardly possible.

Existence of local minima has also an interesting geometric interpretation in terms of realization of image motions. Indeed feedback (4.14) and other equivalent laws can be seen as imposing an image plane velocity $\dot{\mathbf{r}} = \mathbf{K}(\mathbf{r}_d - \mathbf{r}) = \mathbf{K}\mathbf{e}$ to the set of selected features, velocity that can be realized if, and only if, $\dot{\mathbf{r}} \in \operatorname{Im} \mathbf{J}_r$. When $s > 6$, the set $\mathcal{M} = \{\dot{\mathbf{r}} \in \mathbb{R}^s | \ \dot{\mathbf{r}} \notin \operatorname{Im} \mathbf{J}_r\}$ has dimension Δ and coincides with $\ker \mathbf{P}$ if no approximations are involved in $\widehat{\mathbf{J}}_r$. Therefore, given more than 6 features, $\mathcal{M}$ represents the set of $s - 6$ independent feature motions unrealizable by any rigid motion of the camera. Whenever the imposed feature velocity $\mathbf{K}(\mathbf{r}_d - \mathbf{r}) = \mathbf{K}\mathbf{e}$ happens to belong to $\mathcal{M}$, the visual task cannot be fulfilled and the system gets stuck in a local minimum.

The presence of unrealizable image motions can be a relevant issue for IBVS schemes. It should, then, not be surprising that many ideas have been explored to conceive possible solutions. Loosely speaking, the source of the problem lies in the very strength of most IBVS schemes, i.e., the simplicity of feedback (4.14). Requiring, for instance, that all features move in a straight line towards their final positions (as in (4.15)) can be too demanding, and possibly go against the camera rigid motion constraint. Such considerations led to the exploration of more sophisticated techniques in which a consistent image plane motion is imposed to the features. Among the various solutions it is worth citing [Malis 2004] where IBVS control is interpreted as a nonlinear least squares minimization problem, and the proposed second-order minimization scheme proves to be quite effective in improving the overall convergence. Furthermore, other researchers focused on the possibility to plan an path $\mathbf{r}^*(t)$ on the image plane, joining the initial feature value $\mathbf{r}(t_0)$ to the desired value $\mathbf{r}_d$. The idea is that, if one is able to design $\mathbf{r}^*(t)$ consistently with the rigid camera motion constraint, standard IBVS schemes can then be used to track $\mathbf{r}^*(t)$ over time. In addition, other constraints can also be considered when choosing $\mathbf{r}^*(t)$, such as ensuring visibility, avoiding joint limits of the supporting manipulator, and, in general, minimizing any given cost function. One of the first solutions was proposed in [Mezouar & Chaumette 2002] where a potential field approach was used in combination with the partial pose estimation algorithm discussed in Sect. 3.3. As a result, a suitable 3D camera motion could be planned yielding, as a byproduct, the corresponding feature trajectories on the image plane $\mathbf{r}^*(t)$. This basic idea was subsequently refined with some improvements concerning other possible cost functions, or the avoidance of the pose estimation step — see [Allotta & Fioravanti 2005; Chesi & Hung 2007; Schramm *et al.* 2007] for some proposals.

4.4 Hybrid approaches

As final case, we consider the HVS approach which defines task $\mathbf{r}$ as a function of both on image and 3D quantities

$$\mathbf{r} = \mathbf{f}(\mathbf{m}(t),\, \boldsymbol{\chi}(t)) \in \mathbb{R}^s.$$

As stated in Sect. 4.1, the goal is to combine the advantages of the two previous methods, in particular for what concerns robustness (IBVS) and stability (PBVS). This is usually obtained by exploiting some knowledge about the relative orientation between $\mathcal{F}_C$ and $\mathcal{F}_{C_d}$ (e.g., matrix $\mathbf{R}_{C_dC}$) so that rotation control can be addressed directly in cartesian space. The remaining translational dofs are then regulated by means of a suitable set of image quantities $\mathbf{m}(t)$ and, possibly, additional 3D information $\boldsymbol{\chi}(t)$. Many solutions have been proposed in the last years, exploring different possibilities in the choice of $\mathbf{m}(t)$ and $\boldsymbol{\chi}(t)$, see [Malis *et al.* 1999; Malis & Chaumette 2000; Malis *et al.* 2003; Cervera *et al.* 2003] for some examples, and [Malis & Chaumette 2002] for a general and thorough stability analysis of this VS class.

For the sake of illustration, consider the following case: assume a planar scene, so that the homography matrix $\mathbf{H}$ can be computed from desired and current views (see Sect. 3.3.2), and let

$$\mathbf{r} = \begin{bmatrix} p \\ \log Z \\ \theta\mathbf{u} \end{bmatrix} \in \mathbb{R}^6$$

be the task vector, where $\theta\mathbf{u}$ is a minimal parameterization of $\mathbf{R}_{C_dC}$, p an image point belonging to the target object, and Z its associated depth. In this case it is $\mathbf{m}(t) = p$ and $\boldsymbol{\chi}(t) = [Z\ \theta\mathbf{u}]^T$. By defining

$$\mathbf{L}_v = \frac{1}{Z_d \rho_Z} \begin{bmatrix} -1 & 0 & p_u \\ 0 & -1 & p_v \\ 0 & 0 & -1 \end{bmatrix}, \quad \rho_Z = \frac{Z}{Z_d} = \det \mathbf{H},$$

and

$$\mathbf{L}_\omega = \begin{bmatrix} p_u p_v & -(1+p_u^2) & p_v \\ 1+p_v^2 & -p_u p_v & -p_u \\ -p_v & p_u & 0 \end{bmatrix},$$

the 6×6 interaction matrix of task $\mathbf{r}$ takes the triangular form

$$\mathbf{J}_r = \begin{bmatrix} \mathbf{L}_v & \mathbf{L}_\omega \\ \mathbf{0} & \mathbf{L}_{\theta\mathbf{u}} \end{bmatrix}, \tag{4.17}$$

where $\mathbf{L}_{\theta\mathbf{u}}$ is given by (4.4). Hence, regulation of $\mathbf{r}$ to $\mathbf{r}_d$ can be achieved though inversion of (4.17), as in the previous cases. Note that, computation of $\mathbf{J}_r$ still requires knowledge of Z_d. This value can be either estimated online via adaptive techniques [Chen *et al.* 2005], or directly measured during the learning stage of the desired image. As for task error $\mathbf{e}$, it has the measurable expression

$$\mathbf{e} = \mathbf{r}_d - \mathbf{r} = \begin{bmatrix} p_d - p \\ -\log \rho_Z \\ -\theta\mathbf{u} \end{bmatrix}.$$

Being $\mathbf{J}_r$ an upper triangular matrix, it is easy to prove global closed-loop stability of this scheme. Moreover, definition of task $\mathbf{r}$ implies that the image trajectory of p is a straight line. Therefore, as opposite to pure PBVS cases, direct control over image plane motion is possible, at least for what concerns the point selected for the servoing. Of course, the same considerations of Sect. 4.2 about the achievable final accuracy hold also for this solution, i.e., any uncertainty in the 3D components of $\mathbf{r}$ (Z and $\theta\mathbf{u}$) will unavoidably affect the final pose of the camera. An application based on the HVS approach is presented in Chapter 7 for pose control of a FBM.

4.5 Velocity-level control schemes

Throughout the previous developments we preliminarily assumed that the camera linear/angular velocity pair $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ can be considered as a control input for any VS scheme. This is usually a reasonable choice for analysis purposes, since it allows to focus on the intrinsic properties of VS algorithms without being affected by issues related to the particular kinematic structure of the considered manipulator. Many VS works follow this approach and assume a camera with full mobility, so that its velocity twist may be arbitrarily specified at any configuration (see, e.g., [Chaumette & Marchand 2001; Corke & Hutchinson 2001]). However, in eye-in-hand settings the camera is carried by a robot manipulator, often rigidly fixed on the end-effector, so that $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ is actually a consequence of the manipulator velocity input $\mathbf{u}$. The differential link (4.2) must then be further expanded in order to include robot kinematics, and yield a new mapping

$$\dot{\mathbf{r}} = \mathbf{J}_{\text{img}}\mathbf{u} \tag{4.18}$$

equivalent to the task-oriented kinematic modeling formulation introduced in (1.15). In the same spirit, all the feedback schemes presented so far must be revised accordingly, i.e., they must comply with the new differential mapping (4.18) and yield as output the manipulator velocity command $\mathbf{u}$.

Note that, avoiding this step and neglecting robot kinematics is not convenient, in particular, for two conceptually opposite cases. The first is when the dimension of the visual task exceeds the motion dofs of the camera, e.g., when a six-dimensional task is specified for a camera/robot system having less than six dofs. In this case, a solution that directly relates feature velocities to the actual dofs of the robot and solves the visual servoing problem at that level is computationally more advantageous. For instance, this is what was done for nonholonomic mobile robots with an onboard fixed camera in [Mariottini *et al.* 2007]. The second case occurs when the visual task is under-dimensioned with respect to the camera dofs, so that there are an infinity of camera motions which achieve the task. In addition, the robot manipulator may be itself redundant for the camera positioning task. Therefore, this distributed redundancy is better exploited in an integrated way at the level of the robot dofs, where it can be effectively used for optimizing configuration-dependent criteria, such as singularity indices, distance from obstacles, or dynamic cost functions.

Being motivated by these considerations, we analyze, in the next sections, the structure of matrix $\mathbf{J}_{\text{img}}$ in (4.18) for the FBM and NMM cases, respectively.

4.5.1 The FBM case

Consider a FBM with configuration vector $\mathbf{q} \in \mathcal{Q} \simeq \mathbb{R}^n$ as illustrated in Chapter 1. We assume that the camera is rigidly attached to one of the manipulator links, and let $g_{OC}(\mathbf{q}) = (\mathbf{R}_{OC}(\mathbf{q}), {}^B\mathbf{T}_{OC}(\mathbf{q}))$ represent the camera pose w.r.t. the manipulator base frame $\mathcal{F}_O$ coincident with the world frame. For a fixed-base manipulator arm the pair $({}^O\dot{\mathbf{T}}_{OC}, {}^O\boldsymbol{\omega}_{OC})$ is related to $\mathbf{u} = \dot{\mathbf{q}}$ through the $6 \times n$ geometric (basic) Jacobian $\mathbf{J}_G$ [Sciavicco & Siciliano 2000]

$$\begin{bmatrix} {}^O\dot{\mathbf{T}}_{OC} \\ {}^O\boldsymbol{\omega}_{OC} \end{bmatrix} = \mathbf{J}_G(\mathbf{q})\mathbf{u},$$

from which one can easily obtain the expression of $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ as

$$\begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{OC}^T(\mathbf{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{OC}^T(\mathbf{q}) \end{bmatrix} \mathbf{J}_G(\mathbf{q})\mathbf{u} = \mathbf{J}_M(\mathbf{q})\mathbf{u}. \tag{4.19}$$

By combining (4.19) with (4.2), it follows

$$\dot{\mathbf{r}} = \mathbf{J}_r(\mathbf{m}(t), \boldsymbol{\chi}(t))\mathbf{J}_M(\mathbf{q})\mathbf{u} = \mathbf{J}_{\text{img}}(\mathbf{m}(t), \boldsymbol{\chi}(t), \mathbf{q})\mathbf{u}. \tag{4.20}$$

The $s \times n$ matrix $\mathbf{J}_{\text{img}}$ is the *image Jacobian* of visual task (4.1) and replaces the interaction matrix $\mathbf{J}_r$ in all the relevant cases, e.g., whenever control action computation or closed-loop stability analysis are involved. Note that, a correct evaluation of matrix $\mathbf{J}_M$ requires perfect knowledge of the so-called *hand-eye* transformation, i.e., the rigid displacement between the camera frame $\mathcal{F}_C$ and some reference frame on the link where the camera is mounted. Typically, the camera is rigidly attached to the robot end-effector, so that the hand-eye transformation reduces to the relative translation/rotation between $\mathcal{F}_C$ and the end-effector (tool) frame. Note that, poor knowledge of the camera hand-eye transformation affects the computation of $\mathbf{J}_{\text{img}}$ to the same extent of any uncertainty on camera calibration $\mathbf{K}_C$ or 3D quantities $\boldsymbol{\chi}(t)$. However, since both the camera hand-eye and intrinsic parameters are unknown but constant, they can be preliminary estimated off-line for any given camera/manipulator setting [Strobl & Hirzinger 2006].

The image Jacobian can actually lose rank when $g = \dim \operatorname{Im} \mathbf{J}_r - \dim(\operatorname{Im} \mathbf{J}_M \cap \ker \mathbf{J}_r) < s$. When this condition is encountered, it is again not possible to impose an arbitrary (controlled) motion to *all* features on the image plane, but only to a subset of dimension g. In turn, any choice for this subset of g features will constrain the motion of the remaining $s - g$ ones. This problem cannot be avoided even if we reach a condition where $\ker \mathbf{J}_r = \mathbf{0}$, e.g., by selecting a number $s > 6$ of features so that $\operatorname{Im} \mathbf{J}_r = 6$, while keeping $n > s$ (redundancy w.r.t. the visual task). Indeed, since in any case $\operatorname{rank} \mathbf{J}_{\text{img}} \leq 6$, the set $\mathcal{M} = \{\dot{\mathbf{r}} \in \mathbb{R}^s | \, \dot{\mathbf{r}} \notin \operatorname{Im} \mathbf{J}_{\text{img}}\}$ will still have dimension $\Delta \geq s - 6$, and thus, as before, there will still be Δ independent feature motions unrealizable by any motion of the camera — see Sect. 4.3.2.

4.5.2 The NMM case

Derivation of $\mathbf{J}_{\text{img}}$ for a NMM formally retraces the developments of the previous case with, in addition, the presence of the 'special' NMM kinematic model (1.13). In particular, by plugging (1.16) into (4.19), it is

$$\begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{OC}^T(\mathbf{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{OC}^T(\mathbf{q}) \end{bmatrix} \mathbf{J}_G(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{R}_{OC}^T(\mathbf{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{OC}^T(\mathbf{q}) \end{bmatrix} \mathbf{J}_G(\mathbf{q}) \begin{bmatrix} \mathbf{G}(\mathbf{q}_p) & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n_m} \end{bmatrix} \begin{bmatrix} \mathbf{u}_p \\ \mathbf{u}_m \end{bmatrix} = \\ = \mathbf{J}_M(\mathbf{q})\mathbf{u}. \tag{4.21}$$

However, for such systems it is possible to prove that

$$\begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \mathbf{J}_M(\mathbf{q}_m)\mathbf{u},$$

i.e., that the NMM Jacobian $\mathbf{J}_M$ does not depend on the platform absolute position/orientation $\mathbf{q}_p$, but only on the manipulator variables $\mathbf{q}_m$. The proof of this notable property exploits the fact that velocity twist $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ of the camera is expressed in the moving *camera frame*, and it would not hold if other reference frames were selected (e.g., world or platform frames). As a result, the computation of $\mathbf{J}_M(\mathbf{q}_m)$ is independent of the mobile base absolute localization, which is typically obtained from noisy and possibly unreliable data processing algorithms (such as dead-reckoning). Some examples of $\mathbf{J}_M$ are given in Sect. 4.6 and Chapter 6.

Proposition 4.1. *The NMM Jacobian* $\mathbf{J}_M$ *in (4.21), relating the NMM commands* $\mathbf{u}$ *to the linear/angular camera velocity* $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ *expressed in the camera frame, is structurally independent of the platform generalized coordinates* $\mathbf{q}_p$.

Proof. In order to show that $\mathbf{J}_M$ is independent of $\mathbf{q}_p$, assume without loss of generality a camera mounted on the end-effector, and let $\mathbf{R}_{\mathbf{Z}_i}(\alpha)$ be the 3×3 rotation matrix of angle α about the absolute $\mathbf{Z}_O$ axis or about the joint axes $(\mathbf{Z}_1, \ldots, \mathbf{Z}_{n_m})$ of the manipulator arm, and $[x\ y\ h]^T$ be the position of the platform reference point expressed in $\mathcal{F}_O$ (h is the constant platform height). We have

$$^{O}\mathbf{T}_{OC} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} x \\ y \\ h \end{bmatrix} + \mathbf{R}_{\mathbf{Z}_O}(\theta)\mathbf{E}(\mathbf{q}_m) \tag{4.22}$$

where θ is the platform orientation and $\mathbf{E}(\mathbf{q}_m)$ is the vector pointing from the platform reference point to the camera optical center. Assume that the nonholonomic mobile base is a rigid body that can move with a linear velocity v only along the direction of its orientation θ (this is the case of most wheeled mobile platforms, like those with unicycle or car-like kinematics).

Differentiating (4.22), we get

$$\begin{bmatrix} \dot{T}_x \\ \dot{T}_y \\ \dot{T}_z \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \mathbf{R}_{\mathbf{Z}_O}(\theta)\dot{\mathbf{E}} + [\mathbf{Z}_O\dot{\theta}]_\times \mathbf{R}_{\mathbf{Z}}(\theta)\mathbf{E}.$$

The orientation of frame $\mathcal{F}_C$ w.r.t. frame $\mathcal{F}_O$ is given by the rotation matrix

$$\mathbf{R}_{OC} = \mathbf{R}_{\mathbf{Z}_O}(\theta)\mathbf{R}_0\mathbf{R}_{\mathbf{Z}_1}(q_1)\ldots\mathbf{R}_{\mathbf{Z}_{n_m}}(q_{n_m})\mathbf{R}_E,$$

where $\mathbf{R}_{\mathbf{Z}_k}(q_k)$, $k \in \{1, \ldots, n_m\}$ are the rotations associated to the n_m joints of the manipulators arm, $\mathbf{R}_0$ represents the constant orientation of the manipulator first joint axis w.r.t. the platform frame, and $\mathbf{R}_E$ is the constant orientation between the camera frame $\mathcal{F}_C$ and the frame located at the arm end-effector, i.e., the hand-eye transformation. Therefore, the expression of the camera linear velocity in the *camera frame*

$$\mathbf{v}_C = \mathbf{R}_{OC}^T \, {}^O\dot{\mathbf{T}}_{OC} = \mathbf{R}_E^T \mathbf{R}_{\mathbf{Z}_{n_m}}^T(q_{n_m}) \dots \mathbf{R}_{\mathbf{Z}_1}^T(q_1) \mathbf{R}_0^T \left(\begin{bmatrix} v \\ 0 \\ 0 \end{bmatrix} + \dot{\mathbf{E}} + [\mathbf{Z}\dot{\theta}]_\times \mathbf{E} \right)$$

is independent from $\mathbf{q}_p = [x \; y \; \theta]^T$, where we used the property $\mathbf{R}_{\mathbf{Z}_O}^T(\theta)[\mathbf{Z}_O\dot{\theta}]_\times \mathbf{R}_{\mathbf{Z}_O}(\theta) = [\mathbf{Z}_O\dot{\theta}]_\times$. Similar arguments can be used to prove independence of the angular velocity $\boldsymbol{\omega}_C$ from $\mathbf{q}_p$. $\square$

4.6 Simulations

In the following, we propose some numerical examples of the velocity-level control schemes illustrated in the previous section. To this end, we consider a NMM carrying a camera on the end-effector, and exploit the kinematic modeling/control techniques developed in Chapters 1–2 to design the needed feedback laws. An experimental validation of these simulation results will be provided in Chapter 6.

The task-oriented kinematic modeling of NMMs, and the proposed design of associated IBVS kinematic control laws, are illustrated by means of two case studies in which the mobile platform is a two-wheeled differentially-driven vehicle with unicycle kinematics. In the first case, the platform carries a 3R spatial manipulator with an eye-in-hand camera and the NMM is redundant for the chosen visual task. In the second case study, a simpler 2R polar manipulator is considered and the NMM is non-redundant w.r.t. the given task. We shall see how the TP and TS strategies are effective in avoiding singularities during visual servoing. The feature depths needed to compute the interaction matrix are supposed known. The next Chapter will deal with the problem of online estimation of such quantities through a suitable nonlinear observation scheme. Numerical simulations are performed in the Webots[2] environment [Cyberbotics 2007]. Video clips showing the 3D motion of the NMMs in the different cases are available at the website `www.dis.uniroma1.it/~labrob/research/NMM_Visual_Servoing.html`.

4.6.1 Unicycle platform with 3R elbow-type manipulator

Kinematic modeling

Consider the NMM made of a unicycle platform and a 3R elbow-type manipulator analyzed in Sect. 2.3.2 (Fig. 2.2). We recall that, for this robot, the configuration vector is $\mathbf{q}$ =

[2]Webots is a commercial robot simulation software developed by Cyberbotics Ltd. Its simulation engine automatically takes into account the presence of image noise as well as motion disturbances.

$[\mathbf{q}_p^T\ \mathbf{q}_m^T]^T \in \mathbb{R}^6$ with $\mathbf{q}_p = [x\ y\ \theta]^T \in \mathbb{R}^3$ and $\mathbf{q}_m = [q_1\ q_2\ q_3]^T \in \mathbb{R}^3$, and the pseudo-velocity command vector is $\mathbf{u} = [v\ \ \omega\ \ \dot{q}_1\ \ \dot{q}_2\ \ \dot{q}_3]^T = [\mathbf{u}_p^T\ \ \mathbf{u}_m^T]^T \in \mathbb{R}^5$. In order to obtain an homogeneous representation for the NMM commands, it is useful to rewrite the unicycle kinematic model (2.19) in terms of the actuated angular velocities $\widetilde{\mathbf{u}}_p = [\dot{\phi}_R\ \dot{\phi}_L]^T$ of the right and left wheel of the platform. This is done by means of the invertible transformation

$$\mathbf{u}_p = \mathbf{M}\widetilde{\mathbf{u}}_p = \begin{bmatrix} \dfrac{\rho}{2} & \dfrac{\rho}{2} \\ \dfrac{\rho}{w} & -\dfrac{\rho}{w} \end{bmatrix} \widetilde{\mathbf{u}}_p,$$

where ρ is the wheel radius and w is the axle length, leading to

$$\dot{\mathbf{q}}_p = \mathbf{G}(\mathbf{q}_p)\mathbf{M}\widetilde{\mathbf{u}}_p = \widetilde{\mathbf{G}}(\mathbf{q}_p)\widetilde{\mathbf{u}}_p.$$

With these settings, we obtain the following 6×5 matrix $\mathbf{J}_M$

$$\mathbf{J}_M = \begin{bmatrix} \dfrac{\rho(\frac{1}{2}ws_1 - dc_1 - l_3c_{23} - l_2c_2)}{w} & \dfrac{\rho(l_3c_{23} + \frac{1}{2}ws_1 + dc_1 + l_2c_2)}{w} & -l_2c_2 - l_3c_{23} & 0 & 0 \\ \dfrac{\rho(2d\,\overline{c}_{123} - 2dc_{123} + ws_{123} - w\overline{s}_{123})}{4w} & \dfrac{\rho(ws_{123} - w\overline{s}_{123} + 2dc_{123} - 2d\,\overline{c}_{123})}{4w} & 0 & -l_2c_3 - l_3 & -l_3 \\ \dfrac{\rho(wc_{123} + w\overline{c}_{123} + 2ds_{123} + 2d\,\overline{s}_{123})}{4w} & \dfrac{\rho(wc_{123} + w\overline{c}_{123} - 2d\,\overline{s}_{123} - 2ds_{123})}{4w} & 0 & l_2s_3 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ -\dfrac{\rho c_{23}}{w} & \dfrac{\rho c_{23}}{w} & -c_{23} & 0 & 0 \\ \dfrac{\rho s_{23}}{w} & -\dfrac{\rho s_{23}}{w} & s_{23} & 0 & 0 \end{bmatrix},$$

where s_{ijk} and c_{ijk} stand for $\sin(q_i + q_j + q_k)$ and $\cos(q_i + q_j + q_k)$, and $\overline{s}_{123}$, $\overline{c}_{123}$ stand for $\sin(q_1 - q_2 - q_3)$, $\cos(q_1 - q_2 - q_3)$, respectively. Note that, as expected, $\mathbf{J}_M$ does not depend on $\mathbf{q}_p$. By following the analysis developed in Sect. 2.3, it can be shown that this Jacobian is always full rank except when the second link is aligned with the vertical direction, i.e., $q_2 = \pm\pi/2$, where the rank of $\mathbf{J}_M$ drops to 4. Note that the presence of an offset $d \neq 0$ between the platform center and the manipulator base (see Fig. 2.2) is again crucial for deleting some of the singularities that affect the manipulator taken alone. For the NMM geometric data, we set $h = 0.13$, $d = 0.15$, $l_1 = 0.1$, $l_2 = 0.15$, $l_3 = 0.1$, $\rho = 0.1$ and $w = 0.25$ (all units are [m]).

Task description

As visual task, we consider the problem of regulating the position on the image plane of $k = 2$ point features $\mathbf{f} = [p_1^T\ p_2^T]^T \in \mathbb{R}^4$ taken from a fixed target. The features are extracted from two (red) markers placed on a cubic target, as shown in Fig. 4.3. The cube has sides of 0.1 [m], and the two markers are positioned at a distance of 0.03 [m] from their nearest edges.

The cube location in the scene can be described as follows. Consider a frame attached to the target $\mathcal{F}_T$, and a frame $\mathcal{F}_R : \{O_R;\ \mathbf{X}_R,\ \mathbf{Y}_R,\ \mathbf{Z}_R\}$ centered on the platform and aligned with

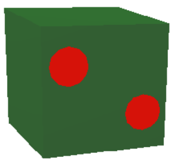


Figure 4.3: A view of the target with the two markers.

the platform heading, i.e., with

$$\begin{cases} O_R &= [x\ y\ 0]^T \\ \mathbf{X}_R &= [\cos\theta\ \ \sin\theta\ 0]^T \\ \mathbf{Y}_R &= [-\sin\theta\ \ \cos\theta\ 0]^T \\ \mathbf{Z}_R &= [0\ 0\ 1]^T. \end{cases}$$

The pose of the target w.r.t. the robot $({}^R\mathbf{T}_{RT}, \mathbf{R}_{\mathbf{Z}_O}(\alpha_t)) \in SE(3)$ can be then expressed as follows. Vector ${}^R\mathbf{T}_{RT}$ points to the center of the cube, while $\mathbf{R}_{\mathbf{Z}_O}(\alpha_t)$ is the rotation matrix of an angle α_t around the world $\mathbf{Z}_O$ axis. When $\alpha_t = 0$, the cube face with the markers is parallel to the plane $(\mathbf{Y}_R, \mathbf{Z}_R)$ and looks toward $-\mathbf{X}_R$. As initial conditions, we have chosen

$$\begin{cases} {}^R\mathbf{T}_{RT}(t_0) &= [1.149\ \ -0.392\ 0.314]^T\ [\text{m}] \\ \alpha_t(t_0) &= 0.2473\ [\text{rad}] \end{cases}$$

for the NMM (with respect to the fixed cube target) and

$$\begin{cases} q_1(t_0) &= 0 \\ q_2(t_0) &= 0.57\ [\text{rad}] \\ q_3(t_0) &= -0.57\ [\text{rad}] \end{cases}$$

for the manipulator joint angles. As a result, the initial position of the point features on the image plane is

$$\mathbf{f}(t_0) = [\ 0.5829 \quad -0.0791 \quad 0.5162 \quad -0.0273\ ]^T,$$

while their desired reference position is

$$\mathbf{f}_d = [\ 0.082 \quad -0.2932 \quad -0.1480 \quad -0.1371\ ]^T.$$

Being $s = 2k = 4$, the complete 4×5 NMM image Jacobian is computed as

$$\dot{\mathbf{f}} = \begin{bmatrix} \mathbf{J}_{p_1}(p_1,\ Z_1) \\ \mathbf{J}_{p_2}(p_2,\ Z_2) \end{bmatrix} \mathbf{J}_M(\mathbf{q}_m)\widetilde{\mathbf{u}} = \mathbf{J}_{\text{img}}(\mathbf{f},\ Z,\ \mathbf{q}_m)\widetilde{\mathbf{u}},$$

with $\widetilde{\mathbf{u}} = [\widetilde{\mathbf{u}}_p^T\ \mathbf{u}_m]^T$. The single degree of kinematic redundancy is used to avoid the singularities of $\mathbf{J}_M$. This is obtained by minimizing the cost function

$$H_4(\mathbf{q}) = \frac{1}{\cos^2 q_2},$$

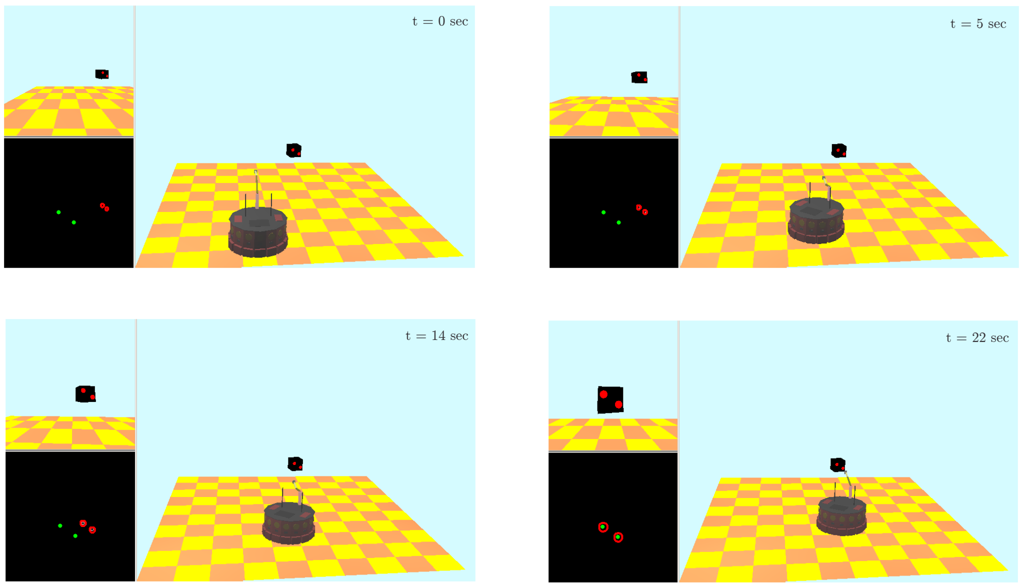


Figure 4.4: Snapshots of the visual servoing task with the PG method.

which goes to infinity at the singularities $q_2 = \pm\pi/2$. For redundancy resolution, we adopted the PG method (2.6) with the control gain matrix $\mathbf{K} = 0.2\, I_2$ in (2.3) and a stepsize $\alpha = 1$ in (2.8), evaluated for $H = H_4(\mathbf{q})$ and using the negative sign.

Simulation results

In Fig. 4.4, four snapshots taken during the execution of the visual servoing task are shown. In each picture, the small upper-left window shows the current image acquired by the camera, while the lower-left window shows the same image after the segmentation process. In the same window, two (green) dots represent the desired positions of the features.

The left plot in Fig. 4.5 shows the path followed by the two feature points on the image plane, starting from the circular markers and ending at the triangle markers. As expected, due to the task decoupling properties of the PG method (with diagonal $\mathbf{K}$), and since no singularities were encountered during visual servoing, the features move on the straight line connecting their initial positions to the desired ones. The time evolution of $H_4(\mathbf{q})$ in the right part of Fig. 4.5 confirms that the the NMM has been kept away from singular configurations. Finally, the velocity commands of the NMM during the servoing are shown in Fig. 4.6. For the platform, we reported the $\mathbf{u}_p$ commands instead of the $\widetilde{\mathbf{u}}_p$, since linear and angular velocities have a more direct interpretation. We observe that the requested task motion is executed by distributing effort among both the platform and the manipulator.

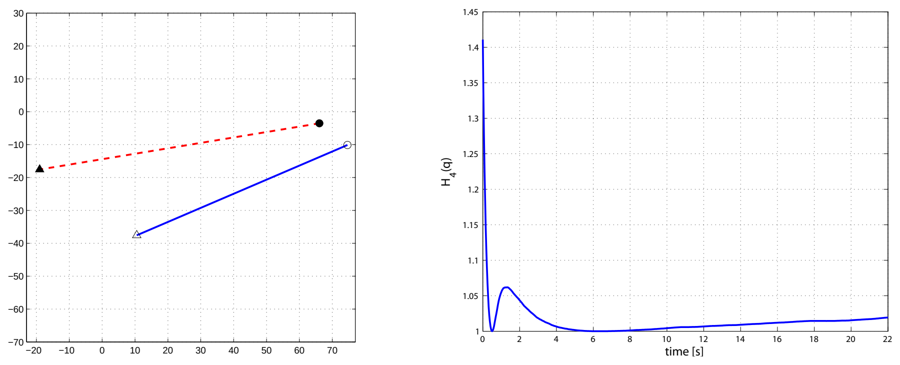


Figure 4.5: PG method. Left: motion of point features p_1 (solid blue line) and p_2 (dashed red line). Right: constrained minimization of index $H_4(\mathbf{q})$.

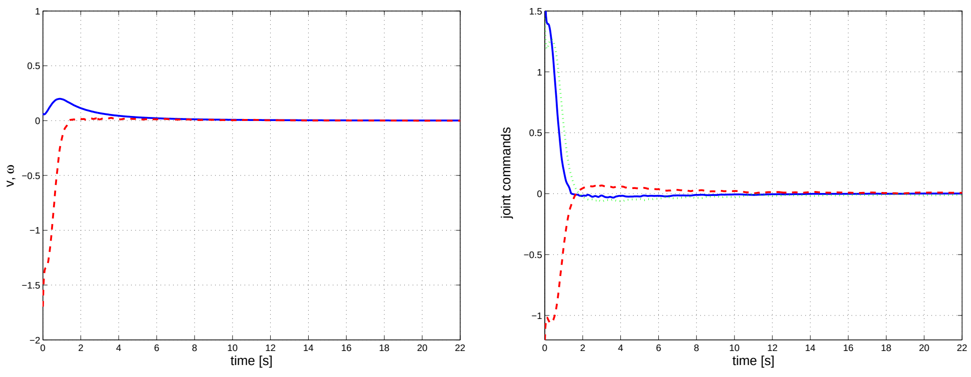


Figure 4.6: NMM control commands with the PG method. Left: linear velocity v (solid blue line) and angular velocity ω (dashed red line) of the platform. Right: manipulator joint velocities $\dot{q}_1$ (solid blue line), $\dot{q}_2$ (dashed red line) and $\dot{q}_3$ (dotted green line).

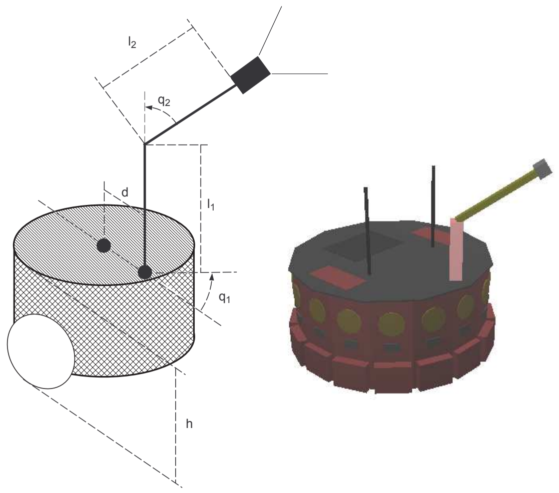


Figure 4.7: Unicycle platform with a 2R polar manipulator.

4.6.2 Unicycle platform with 2R polar manipulator

Kinematic modeling

In this second case study, the NMM has the same previous platform but uses a 2R polar manipulator on board for carrying the camera, see Fig. 4.7. Note that this arrangement may cover the interesting situation of a wheeled mobile robot equipped with a pan-tilt camera (whose two dofs can be modeled as manipulator joints). An experiment involving this setup is reported in Sect. 6.1. In this case, we have $\mathbf{q}_p = [x\, y\, \theta]^T$, $\mathbf{q}_m = [q_1\, q_2]^T$, and thus $\mathbf{q} \in \mathbb{R}^5$, while the actual commands are $\tilde{\mathbf{u}} = [\dot{\phi}_R\, \dot{\phi}_L\, \dot{q}_1\, \dot{q}_2]^T \in \mathbb{R}^4$. The geometric data of this NMM are the same as in Sect. 4.6.1 (eliminating the third link).

The associated 6×4 matrix $\mathbf{J}_M$ is

$$\mathbf{J}_M = \begin{bmatrix} \dfrac{-\rho(dc_1 + \frac{w}{2}s_1 - l_2c_2)}{w} & \dfrac{\rho(dc_1 + \frac{w}{2}s_1 + l_2c_2)}{w} & -l_2c_2 & 0 \\ \dfrac{\rho(2d\bar{c}_{12} - 2dc_{12} + ws_{12} - w\bar{s}_{12})}{4w} & \dfrac{\rho(-2d\bar{c}_{12} + 2dc_{12} + ws_{12} - w\bar{s}_{12})}{4w} & 0 & -l_2 \\ \dfrac{\rho(ds_{12} + d\bar{s}_{12} + \frac{1}{2}w\bar{c}_{12} + \frac{1}{2}wc_{12})}{2w} & \dfrac{\rho(-ds_{12} - d\bar{s}_{12} + \frac{1}{2}w\bar{c}_{12} + \frac{1}{2}wc_{12})}{2w} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{\rho c_2}{w} & \dfrac{\rho c_2}{w} & -c_2 & 0 \\ \dfrac{\rho s_2}{w} & -\dfrac{\rho s_2}{w} & s_2 & 0 \end{bmatrix},$$

with $s_{ij} = \sin(q_i + q_j)$, $c_{ij} = \cos(q_i + q_j)$, $\bar{s}_{ij} = \sin(q_i - q_j)$ and $\bar{c}_{ij} = \cos(q_i - q_j)$. It can be shown that this matrix has *always* full rank.

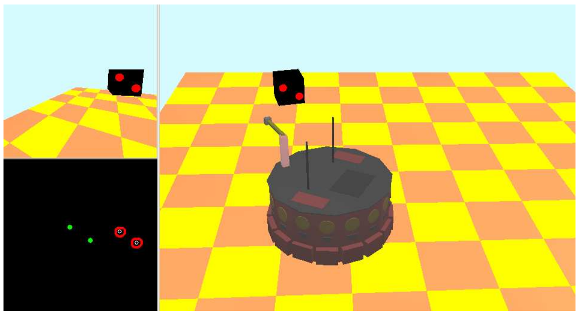


Figure 4.8: A situation where the NMM image Jacobian is close to a singularity.

Task description

As in the previous case study, we choose to regulate the position of $k = 2$ point features, obtaining a 4×4 NMM image Jacobian. This NMM is no longer redundant for the given task, and visual servoing can be obtained using the scheme (2.1) with (2.3). However, inversion of the square NMM image Jacobian does not yield in general good results, since this matrix easily becomes ill-conditioned during the servoing task as discussed in Sect. 4.3.2. Consider for example the situation shown in Fig. 4.8, corresponding to the following initial conditions and desired position of the two point features:

$$\begin{cases} {}^{R}\mathbf{T}_{RT}(t_0)(t_0) & = \begin{bmatrix} 0.3956 & -0.453 & 0.2334 \end{bmatrix}^T \text{ [m]} \\ \alpha_t(t_0) & = -0.8571 \text{ [rad]} \\ q_1(t_0) & = -0.7025 \text{ [rad]} \\ q_2(t_0) & = -0.1368 \text{ [rad]} \\ \mathbf{f}(t_0) & = \begin{bmatrix} 0.7273 & -0.0939 & 0.5123 & 0.0482 \end{bmatrix}^T \\ \mathbf{f}_d & = \begin{bmatrix} 0.1274 & -0.0691 & -0.1354 & 0.1065 \end{bmatrix}^T . \end{cases}$$

Matrix $\mathbf{J}_{\text{img}}$ is very close to a singularity and its inversion will generate very large velocity commands for the NMM. However, the two 2×4 blocks $\mathbf{J}_{\text{img}_1}$ and $\mathbf{J}_{\text{img}_2}$ in

$$\mathbf{J}_{\text{img}} = \begin{bmatrix} \mathbf{J}_{\text{img}_1} \\ \mathbf{J}_{\text{img}_2} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{p_1}\mathbf{J}_M \\ \mathbf{J}_{p_2}\mathbf{J}_M \end{bmatrix},$$

which are associated to the positioning task of each single point feature, are well-conditioned in this configuration; in fact, the smallest singular values are $\sigma(\mathbf{J}_{\text{img}}) = 0.16$, $\sigma(\mathbf{J}_{\text{img}_1}) = 198.75$, and $\sigma(\mathbf{J}_{\text{img}_2}) = 196.01$, respectively. This implies that the robotic system is too constrained in order to impose an arbitrary velocity to both features, although it is still possible to move them individually. Therefore, it is interesting to test the performance of the TP and TS methods

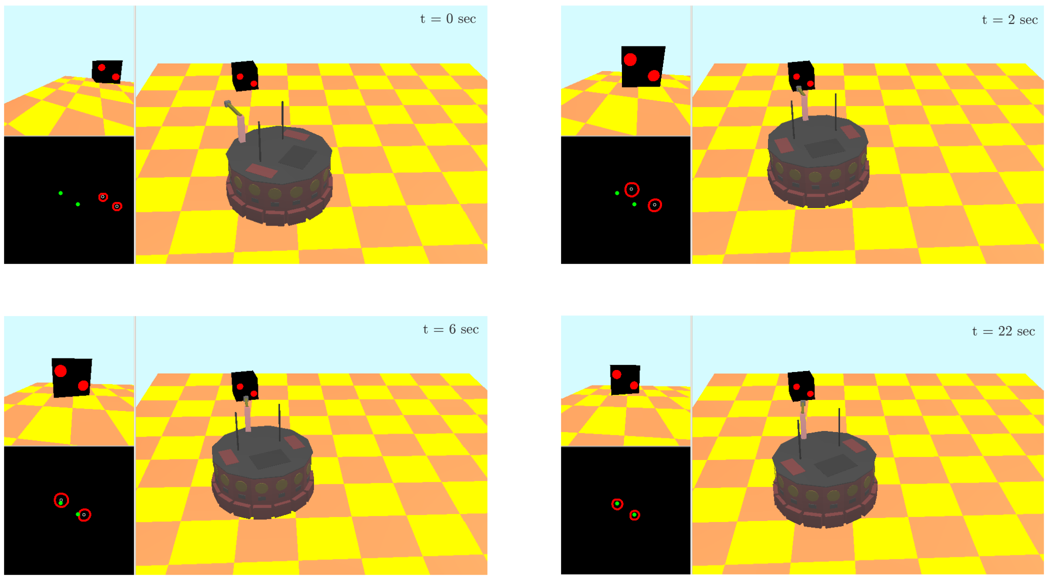


Figure 4.9: Snapshots of the visual servoing task with the TP method.

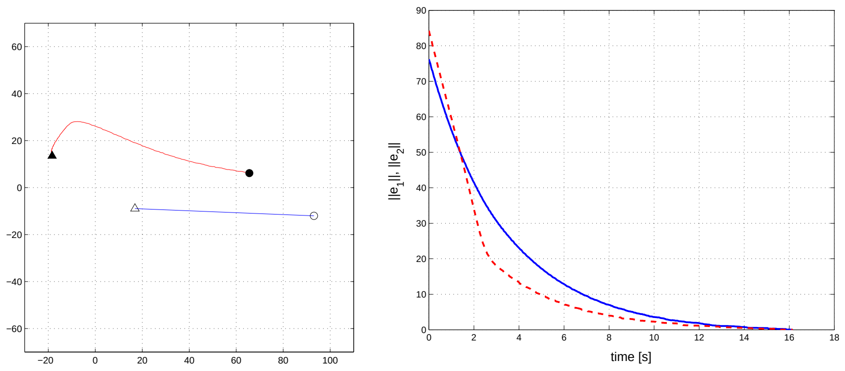


Figure 4.10: TP method. Left: motion of point features p_1 (solid blue line) and p_2 (dashed red line). Right: task errors norms $\|\mathbf{e}_1\|$ (solid blue line) and $\|\mathbf{e}_2\|$ (dashed red line) vs. time.

as they are mainly intended for cases when the (visual) task cannot be directly realized as a whole.

Simulation results

The TP method (2.12) has been applied with $\mathbf{K}_1 = 3\,\mathbf{I}_2$ and $\mathbf{K}_2 = 0.02\,\mathbf{I}_2$, and choosing $\mathbf{r}_1 = p_1$, i.e., the primary task is the regulation of the lower-right point feature, and $\mathbf{r}_2 = p_2$.

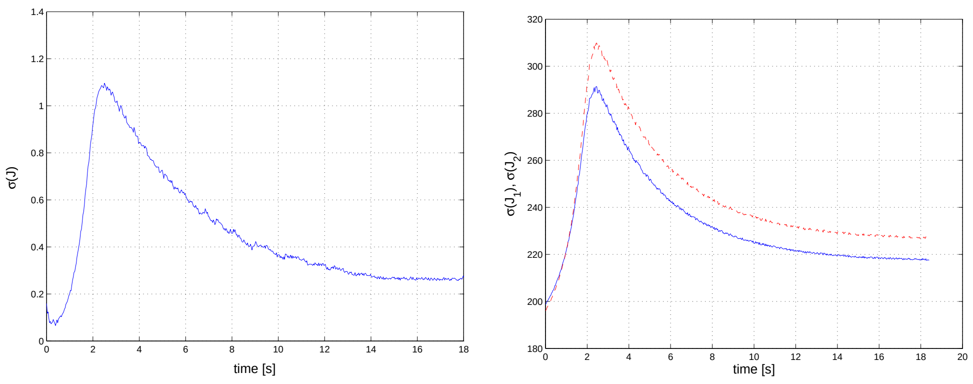


Figure 4.11: Singularity analysis during the visual task. Left: time evolution of the smallest singular value $\sigma(\mathbf{J}_{\text{img}})$. Right: smallest singular values $\sigma(\mathbf{J}_{\text{img}_1})$ (solid blue line) and $\sigma(\mathbf{J}_{\text{img}_2})$ (dashed red line).

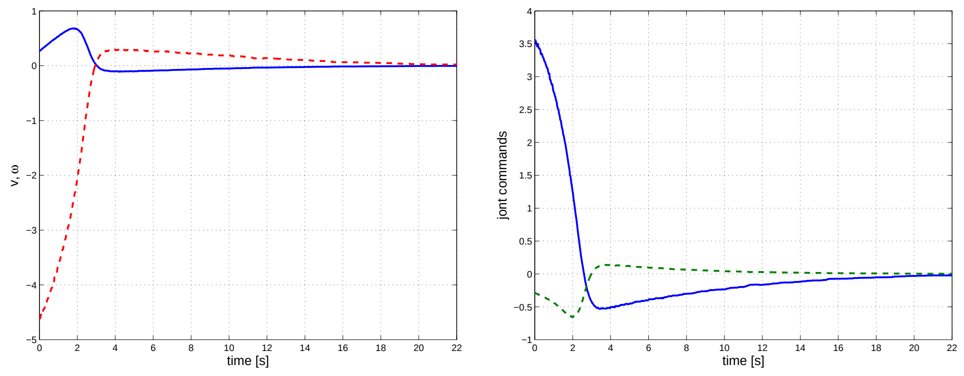


Figure 4.12: NMM control commands with the TP method. Left: linear velocity v (solid blue line) and angular velocity ω (dashed red line) of the platform. Right: manipulator joint velocities $\dot{q}_1$ (solid blue line) and $\dot{q}_2$ (dashed green line).

Figure 4.9 shows the motion of the NMM during visual servoing. In this case, the TP method is able to realize the task without encountering any singularity, but the motion of p_2 is no more on a straight line, as it can be seen from Fig. 4.10. This is a direct consequence of the fact that regulation of p_2 is addressed as a secondary task, and thus combined feature motions that would not be realizable are handled by relaxing the constraints on p_2. Nonetheless, the norm of the error on the second feature $\|\mathbf{e}_2(t)\| = \|p_2(t) - p_{2d}\|$ is still decreasing (right plot in Fig. 4.10), though decay is not exponential as with $\|\mathbf{e}_1(t)\|$.

In Fig. 4.11, we report the evolution of the smallest singular values $\sigma(\mathbf{J}_{\text{img}})$, $\sigma(\mathbf{J}_{\text{img}_1})$ and

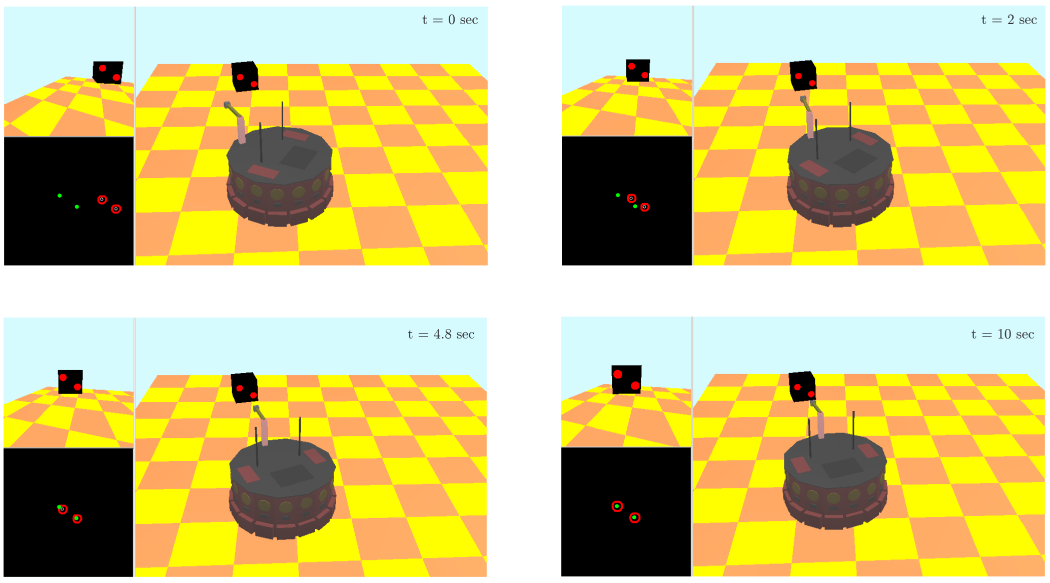


Figure 4.13: Snapshots of the visual servoing task with the TS method.

$\sigma(\mathbf{J}_{\mathrm{img}_2})$ during the servoing task. Note that matrix $\mathbf{J}_{\mathrm{img}}$ remains always close to singularity, while $\mathbf{J}_{\mathrm{img}_1}$ and $\mathbf{J}_{\mathrm{img}_2}$ are well conditioned throughout the motion. This confirms that a direct inversion of $\mathbf{J}_{\mathrm{img}}$ would not have provided satisfactory results. The velocity commands for the platform and the manipulator are shown in Fig. 4.12.

For comparison, we have applied also the TS method (2.17) and performed the visual task in two phases, achieving two degrees of redundancy during the first phase. In the reported simulation, we have chosen again $\mathbf{r}_1 = \mathbf{f}_1$ as the subtask to be realized during the first phase, and $\mathbf{r}_2 = \mathbf{f}_2$ as the completing subtask for the second phase. In order to keep the target as much as possible in front of the NMM, redundancy in the first phase has been used for minimizing the cost function

$$H_{TS}(\mathbf{q}) = \frac{1}{2}q_1^2.$$

Denoting by $\mathbf{u}_{H_{TS}}$ the expression (2.8) evaluated for $H = H_{TS}(\mathbf{q})$, the command sequence is implemented as follows:

I. $\mathbf{u}_I = \mathbf{J}^{\dagger}_{\mathrm{img}_1}\mathbf{K}_1\mathbf{e}_1 + (\mathbf{I}_4 - \mathbf{J}^{\dagger}_{\mathrm{img}_1}\mathbf{J}_{\mathrm{img}_1})\mathbf{u}_{H_{TS}}$, until $\|\mathbf{e}_1\| \leq \epsilon_1$ and $H_{TS}(\mathbf{q}) \leq \epsilon_2$, where $\epsilon_1 = 1$ and $\epsilon_2 = 0.001$;

II. $\mathbf{u}_{II} = (\mathbf{I}_4 - \mathbf{J}^{\dagger}_{\mathrm{img}_1}\mathbf{J}_{\mathrm{img}_1})\mathbf{J}^{T}_{\mathrm{img}_2}\mathbf{K}_2\mathbf{e}_2 + \mathbf{J}^{\dagger}_{\mathrm{img}_1}\mathbf{K}_1\mathbf{e}_1$, until the end of the servoing task.

In Fig. 4.13, four snapshots of the NMM motion are shown. In particular, the frame at $t = 4.8$ [s] in the sequence corresponds to the end of the first phase. The NMM is able to

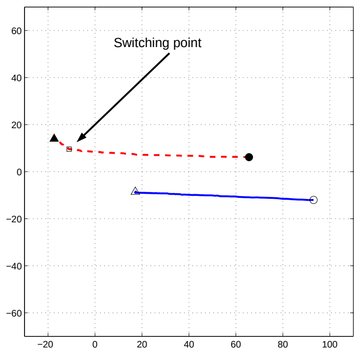


Figure 4.14: TS method. Motion of point features p_1 (solid blue line) and p_2 (dashed red line), with switching point of the latter.

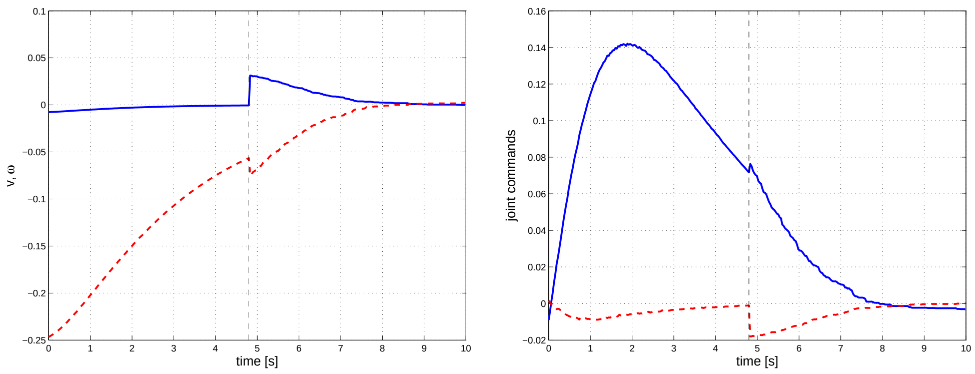


Figure 4.15: NMM control commands with the TS method. Left: linear velocity v (solid blue line) and angular velocity ω (dashed red line) of the platform. Right: manipulator joint velocities $\dot{q}_1$ (solid blue line) and $\dot{q}_2$ (dashed red line).

regulate the feature positions by keeping the target in front of the platform, thanks to the optimization of $H_{TS}(\mathbf{q})$ during the first phase. The motion of the features in the image plane is shown in Fig. 4.14, where the switching point between the two phases is indicated on the motion of p_2. Note that feature p_1 never switches, as its desired value is reached at the end of the first phase and then kept during the second phase. The velocity commands shown in Fig. 4.15 have a discontinuity at the switching time (vertical dashed line), but are considerably smaller than those produced by the TP method.

5
Estimation of Geometric and Camera Quantities

In the previous chapter, we discussed how Visual Servoing can provide a rich set of tools to control the pose of a robot through the elaboration of visual information. Among the different possibilities, IBVS is perhaps the best choice in terms of robustness w.r.t. calibration errors, control over image trajectories, and easiness of implementation, but at the expense of possible singularities/unstabilities that can be encountered in executing the visual task. As explained in Sect. 4.3.2 and Sect. 4.5, such stability issues are due to the local nature of most IBVS feedback laws, in particular because of the various uncertainties affecting the computation of the image Jacobian $\mathbf{J}_{\text{img}}$ in (4.20), such as

- poor knowledge of camera hand-eye calibration, i.e., the fixed pose of the camera w.r.t. its mounting point on the robot body. This uncertainty obviously only affects the geometric Jacobian $\mathbf{J}_M(\mathbf{q})$;

- poor knowledge of matrix $\mathbf{K}_C$, i.e., of the camera intrinsic parameters. This uncertainty only affects the interaction matrix $\mathbf{J}_r(\mathbf{m}(t), \boldsymbol{\chi}(t))$ in consequence of rough evaluation of the normalized image quantities $\mathbf{m}(t)$ (see Sect. 3.2);

- poor knowledge of the 3D information present in the vector of 3D quantities $\boldsymbol{\chi}(t)$.

Usually, an accurate estimation of the constant hand-eye and camera calibration parameters can be easily obtained by means of several off-line methods, so that the knowledge of these values does not represent a relevant issue in most applications. On the other hand, $\boldsymbol{\chi}(t)$ is made of unmeasurable time-varying quantities which need to be approximated if no external information about the 3D model of the target is present.

Estimating 3D quantities is part of the more general paradigm of motion and structure reconstruction, whose purpose is to design an identification scheme to estimate both the camera motion and the structure (i.e., the 3D geometry) of the scene. Several attempts have been made in the last years to solve separately these problems, i.e., by focusing more on the camera motion recovering or on the scene geometry reconstruction. Roughly speaking, most of the proposed solutions stem from two different points of view, namely the Computer Vision and Control Theory ones, so that different algorithms and methodologies have been often developed for the same final goals. Usually, Computer Vision methods work in a *discrete* setting, e.g., consider finite camera translations/rotations, and interpret the problem as a minimization task of a given linear/nonlinear cost function. The solution is typically found in an *off-line* fashion after a number of iterations, but then the obtained result is the 'best' one can achieve w.r.t. the boundary conditions of the specific case. On the other hand, Control Theory methods adopt an *online* or *incremental* approach, and are often designed in a continuous framework, i.e., by considering infinitesimal changes of the camera pose and of the scene. This is the case, for instance, of many solutions relying on (extended) Kalman filters [Gelb 1974; Maybeck 1979]. Therefore, as a downside, one only obtains convergence over time towards any solution that could have been found in one discrete step with the former approach, but, as a benefit, an estimate or approximation of the final value is continuously available online. Deciding whether one approach is superior to the other does not have a definitive answer, and a case-to-case analysis is typically required. However, in this Thesis only the Control Theory point of view is considered, since it suits better for implementing any reactive (and therefore online) visual control laws such as IBVS schemes.

As explained in Sect. 4.3.2, a common workaround for IBVS implementations is to perform the servoing by replacing $\boldsymbol{\chi}(t)$ with the constant value $\boldsymbol{\chi}_d$ relative to the final robot pose. Indeed, $\boldsymbol{\chi}_d$ can be provided during the learning stage when the desired image is stored, or estimated off-line by means of any (partial) pose estimation algorithm. However, other solutions involving online estimations of the unknown quantities are possible, such as the methods proper to the Control Theory class of 3D reconstruction algorithms. By pursuing such a methodology, we propose to replace $\boldsymbol{\chi}(t)$ with an estimate $\widehat{\boldsymbol{\chi}}(t)$ obtained as the output of a suitable *observer*, so that $\lim_{t\to\infty} \|\boldsymbol{\chi}(t) - \widehat{\boldsymbol{\chi}}(t)\| = 0$ for any (possibly wrong) initial guess $\widehat{\boldsymbol{\chi}}(t_0)$ [De Luca *et al.* 2007b; Robuffo Giordano *et al.* 2008]. The observation of $\boldsymbol{\chi}(t)$ is based on the idea that, since the motion of the feature on the image plane depends upon the current value of $\boldsymbol{\chi}(t)$, it is possible to estimate this value by comparing the measured motion with the one predicted by using the current estimate $\widehat{\boldsymbol{\chi}}(t)$, under the assumption of a perfect knowledge of the camera 3D motion and of its intrinsic parameters. Therefore, by interpreting $\boldsymbol{\chi}(t)$ as a continuous unknown state with known dynamics, we build an estimator which asymptotically recovers the actual value of $\boldsymbol{\chi}(t)$ associated to the selected set of features. The problem, however, cannot be attacked by means of classical observation schemes for linear systems [Luenberger 1971] because of the intrinsic nonlinear nature of the pin-hole perspective projection model (3.13). A solution must then be sought by borrowing techniques from the nonlinear observer theory which provides suitable tools to estimate unmeasurable time-varying states of known nonlinear dynamical systems.

To this end, Sect. 5.1 introduces the basic formulation of the *persistency of excitation* Lemma, a theoretical tool upon which all the subsequent observation schemes are designed. Next, Sect. 5.2 illustrates how the general formulation of the persistency of excitation Lemma can be exploited in order to estimate the value of $\boldsymbol{\chi}(t)$, in particular for the cases of point features and image moments. Furthermore, as a byproduct of this analysis, Sect. 5.3 shows that the hypothesis of a perfect calibrated camera can be partially relaxed. Indeed, the same structure of the observer can be extended to cover also the case of online reconstruction of the focal length which can be performed preliminary (and once for all), independently from $\boldsymbol{\chi}(t)$. Finally, simulation results validating the proposed methods are presented in Sect. 5.4. These numerical assessments are then resumed and further validated in Chapter 6 by means of experiments on real robots.

5.1 Persistency of excitation

As a preliminary step for the following developments, we recall the *persistency of excitation* Lemma whose proof can be found in [Marino & Tomei 1995].

Lemma 5.1. *Consider the linear time-varying system*

$$\begin{cases} \dot{\boldsymbol{\xi}} &= \mathbf{W}\boldsymbol{\xi} + \boldsymbol{\Omega}^T(t)\mathbf{z}, \qquad \boldsymbol{\xi} \in \mathbb{R}^n \\ \dot{\mathbf{z}} &= -\boldsymbol{\Lambda}\boldsymbol{\Omega}(t)\mathbf{S}\boldsymbol{\xi}, \qquad \mathbf{z} \in \mathbb{R}^l \end{cases} \tag{5.1}$$

where $\mathbf{W}$ *is an* $n \times n$ *Hurwitz matrix,* $\mathbf{S}$ *is an* $n \times n$ *symmetric positive definite matrix such that* $\mathbf{W}^T\mathbf{S} + \mathbf{S}\mathbf{W} = -\mathbf{Q}$*, with* $\mathbf{Q}$ *symmetric positive definite, and* $\boldsymbol{\Lambda}$ *is a* $l \times l$ *symmetric positive definite matrix. If* $\|\boldsymbol{\Omega}(t)\|$, $\|\dot{\boldsymbol{\Omega}}(t)\|$ *are uniformly bounded and the* ***persistency of excitation*** *condition is satisfied, i.e., there exist two positive real numbers* T *and* γ *such that*

$$\int_t^{t+T} \boldsymbol{\Omega}(\tau)\boldsymbol{\Omega}^T(\tau)\mathrm{d}\tau \geq \gamma\mathbf{I}_l > 0, \qquad \forall\, t \geq t_0, \tag{5.2}$$

then $(\boldsymbol{\xi}, \mathbf{z}) = \mathbf{0}$ *is a globally exponentially stable equilibrium point.* □

At first glance, this Lemma provides a stability condition for the sole class of linear dynamical systems in the form (5.1). Note that, however, despite the apparent linearity w.r.t. the state ($\boldsymbol{\xi}$, $\mathbf{z}$), formulation (5.1) can cover a wider range of systems thanks to the presence of the time-varying quantity $\boldsymbol{\Omega}(t)$ which can arbitrarily include additional (nonlinear) terms. As a matter of fact, this flexibility proves to be crucial when devising observation schemes for nonlinear systems, as in our case.

For our goals, the key idea in using this Lemma is the following: given a state vector $\mathbf{x} = [\mathbf{x}_m^T\ \mathbf{x}_u^T]^T \in \mathbb{R}^{n+l}$ where only the state subset $\mathbf{x}_m$ is directly measurable, design an update law for the estimated state $\widehat{\mathbf{x}} = [\widehat{\mathbf{x}}_m^T\ \widehat{\mathbf{x}}_u^T] \in \mathbb{R}^{n+l}$ such that, by letting $\boldsymbol{\xi} = \mathbf{x}_m - \widehat{\mathbf{x}}_m$ and $\mathbf{z} = \mathbf{x}_u - \widehat{\mathbf{x}}_u$ be the error sub-vectors, the associated error dynamics matches formulation (5.1). When this manipulation is possible, Lemma 5.1 guarantees exponential convergence of the error system, or, in other words, that values of the unmeasurable variables $\mathbf{x}_u$ can be inferred from knowledge of $\mathbf{x}_m$. In this context, condition (5.2) plays the role of an *observability* test, i.e., observation of $\mathbf{x}_u$ is possible iff there not exists a $\bar{t}$ such that $\forall t > \bar{t}$, $\|\boldsymbol{\Omega}(t)\| \equiv 0$. We wish to stress that convergence of the observation is not prevented if $\|\boldsymbol{\Omega}(t)\| \equiv 0$ for a *finite* period of time, since condition (5.2) is violated if and only if $\|\boldsymbol{\Omega}(t)\|$ definitely vanishes over time.

In the next section, we will show how Lemma 5.1 can be tailored to solve the problem of estimating the value of $\boldsymbol{\chi}(t)$ during the motion of the camera for the specific cases of point features and image moments. Note that, obtaining an explicit or 'physical' interpretation of (5.2) may be a nontrivial task in many practical situations. However, in the point feature case a direct interpretation is possible, thus providing a closed-form characterization of all camera motions that actually allow depth identification.

5.2 3D Observation

As stated at the beginning of the Chapter, the problem of estimating 3D quantities from camera measurements can be split in two conceptually different classes, namely estimation of the *camera motion* and estimation of the *scene structure*. The former problem is typically relevant when one needs to extract the so-called *egomotion* while moving through the environment. Among the many works on this topic, it is worth citing [Soatto 1994; Soatto *et al.* 1996; Ma *et al.* 2000] where the authors propose a set of recursive solutions endowed with the remarkable property of being independent of the structure of the target object, and therefore allowing direct estimation. For our needs, however, the latter problem of structure identification is more relevant, since, as stated in Sect. 3.1, the relative motion among the camera and the target is supposed known. Indeed, the target object is assumed fixed in the world, and the camera velocity twist can be known with high accuracy, being it mounted on the end-effector of a robot manipulator.

In the last years, several works have addressed the structure identification with known motion. Chaumette et al. [Chaumette *et al.* 1996] proposed a general methodology to recover the 3D information of several geometric primitives (points, lines, cylinders, spheres, etc.) by measuring the current values of the features, of the image motion (the feature time derivatives) and of the camera velocity twist. However, due to the presence of noise and discrete sampling, the extraction of the image motion is not trivial, and some constraints on the allowed camera motions must be considered. In [Matthies *et al.* 1989], two Kalman filter-based algorithms are derived and compared, the first estimating a continuous depth map of the scene, and the second extracting the depth of a discrete set of features. Both methods need the computation of the current image motion, and impose several constraints on the camera motion in order to simplify the problem. In particular, the second method assumes a camera which translates orthogonally to the optical axis (without rotations), so that the depth of the features is kept constant and the problem is considerably simplified. A similar approach is found in [Smith & Papanikolopoulos 1994], where, again, only lateral camera motions are allowed. Adaptive IBVS schemes are devised in [Conticelli *et al.* 1999] for a camera mounted on a nonholonomic mobile robot via online observation of a constant unknown parameter (the height of the object points). In fact, whenever the depth Z is kept constant during the camera motion [Matthies *et al.* 1989; Smith & Papanikolopoulos 1994], or the value of Z is related to any other fixed quantity [Conticelli *et al.* 1999], the problem of depth identification can be formulated in the adaptive control context, where several tools allow, under suitable hypothesis, to estimate an unknown constant parameter.

With respect to these works, we tackle the problem of structure identification without any preliminary constraint on the camera motion, and without the explicit need for image motion estimation; thus, the only information used is the current value of the features measured on the image plane and the current camera twist. The next sections will present the explicit derivation of suitable observer schemes for the particular cases of point features and image moments, respectively.

5.2.1 Observer design for point features

Consider a point feature $p = [p_u \; p_v]^T$ with associated depth Z. Both p and Z can be considered as time-varying states with dynamic equations given by the last row of (3.9), and (4.6), respectively. Note that, since only the first three columns of $\mathbf{J}_p$ in (4.6) are affected by the value of Z, a pure camera rotation does not bring any information useful for depth observation: a camera translation must be necessarily present. This intuitive consideration, already established (among the others) by [Inaba *et al.* 2000] in the context of the observability of dynamical systems with perspective outputs, will intrinsically affect all the proposed observation schemes in terms of satisfaction of the persistency of excitation condition. In addition, for the case of point features, it will explicitly characterize which camera motions are useless for the depth observation.

Let $\mathbf{x} = [p^T \; Z]^T \in \mathbb{R}^3$ be the complete state vector and $\mathbf{u} = [\mathbf{v}_C^T \; \boldsymbol{\omega}_C^T]^T \in \mathbb{R}^6$ be the input vector. Hence, the state dynamics are expressed by the driftless system

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} -\dfrac{1}{x_3} & 0 & \dfrac{x_1}{x_3} & x_1x_2 & -\left(1+x_1^2\right) & x_2 \\ 0 & -\dfrac{1}{x_3} & \dfrac{x_2}{x_3} & 1+x_2^2 & -x_1x_2 & -x_1 \\ 0 & 0 & -1 & -x_2x_3 & x_1x_3 & 0 \end{bmatrix} \mathbf{u} \\ \mathbf{y} &= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \end{aligned} \tag{5.3}$$

where the output vector $\mathbf{y} \in \mathbb{R}^2$ represents the measurable variables, i.e., the coordinates of the point p on the image plane. Consider the change of coordinates

$$\widetilde{\mathbf{x}} = \begin{bmatrix} x_1 \\ x_2 \\ \dfrac{1}{x_3} \end{bmatrix},$$

which is globally defined since $x_3(t) > 1$ (i.e., the point P is supposed to lie always in front of the image plane, otherwise the camera sensor, and hence the visual servoing, would fail)[1]. In the new coordinates, system (5.3) becomes

$$\begin{aligned} \dot{\widetilde{\mathbf{x}}} &= \begin{bmatrix} -\widetilde{x}_3 & 0 & \widetilde{x}_1\widetilde{x}_3 & \widetilde{x}_1\widetilde{x}_2 & -\left(1+\widetilde{x}_1^2\right) & \widetilde{x}_2 \\ 0 & -\widetilde{x}_3 & \widetilde{x}_2\widetilde{x}_3 & 1+\widetilde{x}_2^2 & -\widetilde{x}_1\widetilde{x}_2 & -\widetilde{x}_1 \\ 0 & 0 & \widetilde{x}_3^2 & \widetilde{x}_2\widetilde{x}_3 & -\widetilde{x}_1\widetilde{x}_3 & 0 \end{bmatrix} \mathbf{u} \\ \mathbf{y} &= \begin{bmatrix} \widetilde{x}_1 \\ \widetilde{x}_2 \end{bmatrix}. \end{aligned} \tag{5.4}$$

Since (5.4) is driftless and the output has dimension smaller than the state, its linear approximation at any point is unobservable. This is a consequence of the intrinsic nonlinear nature of (5.4) in the sense that any linear time-invariant approximation will lose the observability property. Hence, as stated before, a suitable nonlinear observer is then strictly needed in order to correctly address the problem.

According to the notation introduced in Sect. 5.1, let $\mathbf{x}_m = [\widetilde{x}_1 \; \widetilde{x}_2]^T \in \mathbb{R}^n$, $n = 2$, represent the measurable state components, and $\mathbf{x}_u = \widetilde{x}_3 \in \mathbb{R}^l$, $l = 1$, the unmeasurable one. System (5.4) takes the compact form

$$\begin{aligned} \dot{\mathbf{x}}_m &= \boldsymbol{\Gamma}_1(\mathbf{y}(t), \mathbf{u}(t))\mathbf{x}_u + \boldsymbol{\Pi}_1(\mathbf{y}(t), \mathbf{u}(t)) = \boldsymbol{\Gamma}_1(t)\mathbf{x}_u + \boldsymbol{\Pi}_1(t) \\ \dot{\mathbf{x}}_u &= u_3\mathbf{x}_u^2 + (y_2u_4 - y_1u_5)\mathbf{x}_u, \end{aligned}$$

[1]We recall that, in normalized coordinates, the image plane lies at distance $\lambda = 1$ from the camera optical center, see Sect. 3.2.

where

$$\mathbf{\Gamma}_1(t) = \begin{bmatrix} -u_1 + y_1 u_3 \\ -u_2 + y_2 u_3 \end{bmatrix}, \quad \mathbf{\Pi}_1(t) = \begin{bmatrix} y_1 y_2 u_4 - (1 + y_1^2) u_5 + y_2 u_6 \\ (1 + y_2^2) u_4 - y_1 y_2 u_5 - y_1 u_6 \end{bmatrix}$$

are functions of known quantities (system inputs and outputs).

Now let $\widehat{\mathbf{x}} = [\widehat{\mathbf{x}}_m^T \ \widehat{\mathbf{x}}_u^T]$ be the estimate of the (partially) unknown state $\widetilde{\mathbf{x}}$. We seek an update law which can yield an error dynamics close to formulation (5.1). To this end, the choice

$$\begin{array}{rcl} \dot{\widehat{\mathbf{x}}}_m & = & \mathbf{\Gamma}_1(t)\widehat{\mathbf{x}}_u + \mathbf{\Pi}_1(t) + \mathbf{K}_1(\mathbf{x}_m - \widehat{\mathbf{x}}_m), \quad \mathbf{K}_1 > 0 \\ \dot{\widehat{\mathbf{x}}}_u & = & u_3 \widehat{\mathbf{x}}_u^2 + (y_2 u_4 - y_1 u_5)\widehat{\mathbf{x}}_u + \mathbf{K}_2 \mathbf{\Gamma}_1^T(t)(\mathbf{x}_m - \widehat{\mathbf{x}}_m), \quad \mathbf{K}_2 > 0 \end{array} \tag{5.5}$$

yields the error dynamics

$$\begin{array}{rcl} \dot{\boldsymbol{\xi}} & = & -\mathbf{K}_1 \boldsymbol{\xi} + \mathbf{\Gamma}_1(t)\mathbf{z} \\ \dot{\mathbf{z}} & = & -\mathbf{K}_2 \mathbf{\Gamma}_1^T(t)\boldsymbol{\xi} + u_3(\mathbf{x}_u^2 - \widehat{\mathbf{x}}_u^2) + (y_2 u_4 - y_1 u_5)(\mathbf{x}_u - \widehat{\mathbf{x}}_u). \end{array} \tag{5.6}$$

If we set

$$\begin{array}{rcl} \mathbf{W} & = & -\mathbf{K}_1 \\ \mathbf{\Omega}(t) & = & \mathbf{\Gamma}_1^T(t) \\ \mathbf{\Lambda} & = & -\mathbf{K}_2 \\ \mathbf{S} & = & 1, \end{array}$$

system (5.6) is very close to the formulation in (5.1), the only difference being the last two terms in the $\mathbf{z}$ dynamics.

It is worth noting that, when

$$u_3(t) \equiv u_4(t) \equiv u_5(t) \equiv 0, \tag{5.7}$$

the two formulations match exactly and the global exponential stability of $(\boldsymbol{\xi}, \mathbf{z})$ is guaranteed, as long as the conditions of Lemma 5.1 are met. While we will thoroughly discuss such conditions in the forthcoming analysis, we would like to emphasize that (5.7) corresponds to a camera motion which keeps the depth Z constant. As explained before, in this case the problem is considerably simplified and can be attacked with various techniques. The purpose of our analysis is to show that (5.6) can converge also when (5.7) does not hold.

Proposition 5.1. *Using the observer (5.5), the origin of the error system (5.6) is exponentially stable if the conditions of Lemma 5.1 are verified.*

Proof. By letting $\mathbf{e} = [\boldsymbol{\xi}^T\ \mathbf{z}^T]^T \in \mathbb{R}^3$, it is useful to rewrite (5.6) as $\dot{\mathbf{e}} = \mathbf{A}(t)\mathbf{e} + \mathbf{g}(\mathbf{e},\, t)$ with

$$\begin{aligned}\mathbf{A}(t) &= \begin{bmatrix} -\mathbf{K}_1 & \boldsymbol{\Gamma}_1(t) \\ -\mathbf{K}_2\boldsymbol{\Gamma}_1^T(t) & 0 \end{bmatrix} \\ \mathbf{g}(\mathbf{e},\, t) &= \begin{bmatrix} 0 \\ 0 \\ (\mathbf{x}_u^2 - \widehat{\mathbf{x}}_u^2)u_3 + (y_2u_4 - y_1u_5)e_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2\mathbf{x}_u u_3 + (y_2u_4 - y_1u_5)e_3 - u_3e_3^2 \end{bmatrix}. \end{aligned} \tag{5.8}$$

We can consider $\mathbf{g}(\mathbf{e},\, t)$ as a perturbation term of the nominal system $\dot{\mathbf{e}} = \mathbf{A}(t)\mathbf{e}$ which, if (5.2) holds, is guaranteed by Lemma 5.1 to be globally exponentially stable. Note that $\mathbf{g}(\mathbf{e},\, t)$ is a vanishing perturbation, i.e., $\mathbf{g}(\mathbf{0},\, t) = \mathbf{0},\ \forall t$. Several tools are available for the stability analysis of globally exponentially stable systems with vanishing perturbations (see [Khalil 2002] for an overview). Generally, if $\|\mathbf{g}(\mathbf{e},\, t)\|$ is sufficiently small, the exponential stability is preserved, at least locally. Due to the boundedness of $\|\boldsymbol{\Gamma}_1(t)\|$ and $\|\dot{\boldsymbol{\Gamma}}_1(t)\|$, the system $\dot{\mathbf{e}} = \mathbf{A}(t)\mathbf{e}$ is an exponentially stable slowly varying linear system, and therefore there exists a suitable Lyapunov function $V(\mathbf{e},\, t)$ such that

$$\begin{gathered} c_1\mathbf{e}^T\mathbf{e} \le V \le c_2\mathbf{e}^T\mathbf{e} \\ \dot{V}(\mathbf{e},\, t) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial \mathbf{e}}\mathbf{A}(t)\mathbf{e} \le -c_3\|\mathbf{e}\|^2 \\ \left\|\frac{\partial V}{\partial \mathbf{e}}\right\| \le c_4\|\mathbf{e}\|, \end{gathered}$$

with $c_1 \dots c_4$ positive constants. To derive bounds on $\mathbf{g}(\mathbf{e},\, t)$ note that $0 < \mathbf{x}_u(t) < 1$, $|y_1(t)| \le M_1$ and $|y_2(t)| \le M_2$ where M_1, M_2 are the (finite) dimensions of the image plane[2], and, since $\|\boldsymbol{\Gamma}_1(t)\|$ is bounded, there exists a positive constant M_3 such that $|u_3(t)| \le M_3$, $|u_4(t)| \le M_3$ and $|u_5(t)| \le M_3$. Finally, if $|e_3| \le E$ we have $|e_3^2| \le E|e_3| \le E\|\mathbf{e}\|$. Hence,

$$\|\mathbf{g}(\mathbf{e},\, t)\| \le (2 + M_1 + M_2 + E)\, M_3\|\mathbf{e}\| = \gamma\|\mathbf{e}\|.$$

Using the Lyapunov candidate $V(\mathbf{e},\, t)$ for the perturbated system (5.8), we get

$$\dot{V}(\mathbf{e},\, t) \le -c_3\|\mathbf{e}\|^2 + \left\|\frac{\partial V}{\partial \mathbf{e}}\right\| \|\mathbf{g}(\mathbf{e},\, t)\| \le -c_3\|\mathbf{e}\|^2 + c_4\gamma\|\mathbf{e}\|^2.$$

If γ is small enough to satisfy the bound $\gamma < c_3/c_4$, $\dot{V}$ is negative definite and system (5.6) is exponentially stable. Note that, for given camera parameters (image plane size) and motion $(u_3,\, u_4,\, u_5)$ the constant γ only depends on E, i.e., the maximum value of $|e_3(t)|$. This can be made arbitrarily small by choosing the initial condition $e_3(t_0)$ inside a suitable level set

[2]We are implicitly assuming a camera motion such that the object of interest is always kept in the field of view.

$S_c = \{\mathbf{e} \in \mathbb{R}^3 | \, V(\mathbf{e}, t_0) \leq c\}$, since we have

$$E \leq \|\mathbf{e}(t)\| \leq \|\mathbf{e}(t_0)\| \leq \frac{V(\mathbf{e}, t_0)}{c_1} \leq \frac{c}{c_1}.$$

$\square$

Note that the above stability result is of a local nature, since convergence of the error to zero is only guaranteed in a suitable neighborhood of the origin. A less conservative estimate of this neighborhood may be obtained by considering that our observer will be obviously initialized with the measured values of the feature, so that $e_1(t_0) = e_2(t_0) = 0$. This implies that $|e_3(t)| \leq \|\mathbf{e}(t_0)\| = E$, so that

$$|e_3(t_0)| \leq \frac{c_3}{c_4 M_3} - (2 + M_1 + M_2)$$

guarantees exponential error convergence.

The conditions of Lemma 5.1 deserve some additional considerations. First of all the boundedness of $\|\mathbf{\Gamma}_1(t)\|$ and $\|\dot{\mathbf{\Gamma}}_1(t)\|$ requires that the input signal $\mathbf{u}(t)$ is bounded with bounded derivatives. As for condition (5.2), it has a deeper meaning: it essentially states that there must not exist a $\bar{t}$ such that $\forall t > \bar{t}$, $\|\mathbf{\Gamma}_1(t)\| \equiv 0$. By direct inspection of the expression of $\mathbf{\Gamma}_1(t)$, we can conclude that the persistency of excitation condition for depth observation fails if and only if

1. $\exists \bar{t} \,|\, \forall t > \bar{t}: \quad u_1(t) \equiv 0, \; u_2(t) \equiv 0, \; u_3(t) \equiv 0$, i.e., if no translations are involved in the camera motion;

2. $\exists \bar{t} \,|\, \forall t > \bar{t}: \quad \lambda u_1 = y_1 u_3, \; \lambda u_2 = y_2 u_3$, which is equivalent to

$$\frac{u_1}{u_3} = \frac{X}{Z}, \; \frac{u_2}{u_3} = \frac{Y}{Z}.$$

   This means that the camera is translating along the projection ray of the selected point p, and, thus, the projection of P on the image plane is kept constant during the motion.

It is interesting to note that such persistency of excitation condition, essential for the observation convergence, is basically due to the scale ambiguity present in every perspective system. Indeed, as discussed in Sect. 3.2, within a perspective system it is impossible to distinguish an object from the same object twice as big, twice as far and moving twice as fast. The condition of nonzero (and known) camera translational velocity introduces a scale information which is essential to disambiguate among all the equivalent states induced by the relation (3.14), and to successfully recover the actual feature depth. Note that, in this case, such property also implies the *necessity* of Lemma 5.1 requirements (which in general are only sufficient), i.e., depth can be recover *if and only if* (5.2) holds.

5.2.2 Observer design for image moments

In Sect. 4.3.1 we emphasized how image moments can provide a useful set of features since they encode in a compact way the fingerprint of a given shape, and their evaluation is typically free of the so-called correspondence problem. As with any other feature, however, the interaction matrix of a generic moment still depends of some 3D information. In particular, for a target object with planar limb surface,

$$\boldsymbol{\chi}(t) = \begin{bmatrix} A \\ B \\ C \end{bmatrix} = -\mathbf{n}/d,$$

where $\mathbf{n} = [n_x\ n_y\ n_z]^T \in \mathbb{S}^2$ is the plane unit normal and d the plane distance to the origin of $\mathcal{F}_C$ — see (4.9)–(4.11). Therefore, in the following we analyze the possibility of tailoring the formulation of Lemma 5.1 to obtain an online observation scheme for $[A\ B\ C]^T$. In particular, we discuss three possible solutions for such estimation depending on the initial assumptions made on the quantities directly measurable, and on target intrinsic geometry.

General case

Let $\mathbf{x}_m = [m_{i_1 j_1} \ldots m_{i_n j_n}]^T \in \mathbb{R}^n$ be a collection of n generic moments, and $\mathbf{x}_u = [A\ B\ C]^T \in \mathbb{R}^l$, $l = 3$. From (4.13), we can rewrite the $\mathbf{x}_m$ dynamics in the compact form

$$\begin{aligned} \dot{\mathbf{x}}_m &= \begin{bmatrix} \lambda_{A_1} & \lambda_{B_1} & \lambda_{C_1} \\ \vdots & \vdots & \vdots \\ \lambda_{A_n} & \lambda_{B_n} & \lambda_{C_n} \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} + \begin{bmatrix} \lambda_{D_1} \\ \vdots \\ \lambda_{D_n} \end{bmatrix} = \\ &= \boldsymbol{\Gamma}_2(t)\mathbf{x}_u + \boldsymbol{\Pi}_2(t). \end{aligned} \tag{5.9}$$

Hence, as done in the previous case, we define the update law for $\widehat{\mathbf{x}}_m$ as

$$\dot{\widehat{\mathbf{x}}}_m = \boldsymbol{\Gamma}_2(t)\widehat{\mathbf{x}}_u + \boldsymbol{\Pi}_2(t) + \mathbf{K}_1(\mathbf{x}_m - \widehat{\mathbf{x}}_m), \quad \mathbf{K}_1 > 0, \tag{5.10}$$

and get a $\dot{\boldsymbol{\xi}} = \dot{\mathbf{x}}_m - \dot{\widehat{\mathbf{x}}}_m$ error dynamics

$$\dot{\boldsymbol{\xi}} = -\mathbf{K}_1\boldsymbol{\xi} + \boldsymbol{\Gamma}_2\mathbf{z} \tag{5.11}$$

that matches exactly the first row of (5.1) with $\mathbf{z} = \mathbf{x}_u - \widehat{\mathbf{x}}_u$, $\mathbf{W} = -\mathbf{K}_1$ and $\boldsymbol{\Omega}^T(t) = \boldsymbol{\Gamma}_2(t)$. Note that Lemma 5.1 requires the boundedness of $\|\boldsymbol{\Gamma}_2(t)\|$ and $\|\dot{\boldsymbol{\Gamma}}_2(t)\|$. In our case, this is again guaranteed as long as the camera velocity $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ keeps bounded with bounded derivatives and a finite image plane size is assumed, so that the measured moments are bounded.

Now it remains to design an update law for $\widehat{\mathbf{x}}_u$ which can yield a $\dot{\mathbf{z}}$ dynamics as close as possible to the second row of (5.1). To this end, we first need an explicit expression of $\dot{\mathbf{x}}_u$.

From (4.11), it is

$$\dot{\mathbf{x}}_u = -\frac{\dot{\mathbf{n}}d - \mathbf{n}\dot{d}}{d^2} \tag{5.12}$$

and, since $\mathbf{n}$ is a free vector expressed in $\mathcal{F}_C$, from standard kinematics we have

$$\dot{\mathbf{n}} = -[\boldsymbol{\omega}_C]_\times \mathbf{n}. \tag{5.13}$$

Expression of $\dot{d}$ can be obtained as follows: among the points belonging to (4.9) consider point $P_n = -d\mathbf{n}$, i.e. the point on the plane which lies at a distance d along the direction of $\mathbf{n}$. From (4.9), it is $\dot{d} = -\dot{\mathbf{n}} \cdot P_n - \mathbf{n} \cdot \dot{P}_n$ and, since $\mathbf{n}$ and P_n are parallel, $\dot{\mathbf{n}} \cdot P_n = 0$. By exploiting the kinematics of P_n given by (3.9), we have

$$\mathbf{n} \cdot \dot{P}_n = \mathbf{n} \cdot (-\mathbf{v}_C - [\boldsymbol{\omega}_C]_\times P_n) = -\mathbf{n} \cdot \mathbf{v}_C,$$

where, again, the fact that $\mathbf{n}$ and P_n are parallel is used. In conclusion, we obtain

$$\dot{d} = \mathbf{n} \cdot \mathbf{v}_C. \tag{5.14}$$

By plugging (5.13) and (5.14) into (5.12), we get the searched relation

$$\dot{\mathbf{x}}_u = [\boldsymbol{\omega}_C]_\times \frac{\mathbf{n}}{d} + \left(\frac{\mathbf{n}}{d} \cdot \mathbf{v}_C\right) \frac{\mathbf{n}}{d}$$

which, using (4.11), can be explicitly rewritten in terms of (A, B, C) as

$$\begin{aligned} \dot{\mathbf{x}}_u &= \begin{bmatrix} A^2 & AB & AC \\ AB & B^2 & BC \\ AC & BC & C^2 \end{bmatrix} \mathbf{v}_C - [\boldsymbol{\omega}_C]_\times \mathbf{x}_u = \\ &= \boldsymbol{\Theta}(\mathbf{x}_u)\mathbf{v}_C - [\boldsymbol{\omega}_C]_\times \mathbf{x}_u. \end{aligned} \tag{5.15}$$

Hence, by choosing the update law

$$\dot{\widehat{\mathbf{x}}}_u = \boldsymbol{\Theta}(\widehat{\mathbf{x}}_u)\mathbf{v}_C - [\boldsymbol{\omega}_C]_\times \widehat{\mathbf{x}}_u + \mathbf{K}_2 \boldsymbol{\Gamma}_2^T \boldsymbol{\xi}, \quad \mathbf{K}_2 > 0, \tag{5.16}$$

we get a $\dot{\mathbf{z}}$ error dynamics

$$\dot{\mathbf{z}} = (\boldsymbol{\Theta}(\mathbf{x}_u) - \boldsymbol{\Theta}(\widehat{\mathbf{x}}_u))\mathbf{v}_C - [\boldsymbol{\omega}_C]_\times \mathbf{z} - \mathbf{K}_2 \boldsymbol{\Gamma}_2^T \boldsymbol{\xi} \tag{5.17}$$

which, by setting $\boldsymbol{\Lambda} = \mathbf{K}_2$ and $\mathbf{S} = \mathbf{I}_3$, results very close to the formulation in (5.1), the only differences being the first two terms in (5.17). The last step is to prove stability of the closed-loop error system (5.11)–(5.17) despite the presence of the unwanted terms in (5.17).

Proposition 5.2. *Using the observer (5.10)–(5.16), the origin of the error system (5.11)–(5.17) is exponentially stable as long as the conditions of Lemma 5.1 are verified, in particular condition (5.2).*

Proof. Let $\mathbf{e} = [\boldsymbol{\xi}^T \; \mathbf{z}^T]^T$ be the error vector and rewrite (5.11)–(5.17) as

$$\begin{aligned}\dot{\mathbf{e}} &= \begin{bmatrix} -\mathbf{K}_1 & \boldsymbol{\Gamma}_2 \\ -\mathbf{K}_2\boldsymbol{\Gamma}_2^T & \mathbf{0} \end{bmatrix} \mathbf{e} + \begin{bmatrix} \mathbf{0} \\ (\boldsymbol{\Theta}(\mathbf{x}_u) - \boldsymbol{\Theta}(\widehat{\mathbf{x}}_u))\mathbf{v}_C - [\boldsymbol{\omega}_C]_\times \mathbf{z} \end{bmatrix} = \\ &= \mathbf{A}(t)\mathbf{e} + \mathbf{g}(\mathbf{e},\, t)\end{aligned} \tag{5.18}$$

where we interpreted the term $\boldsymbol{\Theta}(\mathbf{x}_u) - \boldsymbol{\Theta}(\widehat{\mathbf{x}}_u)$ as a function of $\mathbf{e}$. Function $\mathbf{g}(\mathbf{e},\, t)$ can be seen as a perturbation term of the nominal system $\dot{\mathbf{e}} = \mathbf{A}(t)\mathbf{e}$ which is guaranteed to be globally exponentially stable by Lemma 5.1. Note that $\mathbf{g}(\mathbf{e},\, t)$ is a vanishing perturbation, i.e., $\mathbf{g}(\mathbf{0},\, t) = \mathbf{0},\ \forall t$. Therefore, if $\|\mathbf{g}(\mathbf{e},\, t)\|$ is sufficiently small, the exponential stability of (5.18) is (locally) preserved. Due to the boundedness of $\|\boldsymbol{\Gamma}_2(t)\|$ and $\|\dot{\boldsymbol{\Gamma}}_2(t)\|$, the nominal system is an exponentially stable slowly varying linear system, and therefore there exists a suitable Lyapunov function $V(\mathbf{e},\, t)$ such that

$$\begin{gathered} c_1 \mathbf{e}^T\mathbf{e} \le V \le c_2 \mathbf{e}^T\mathbf{e} \\ \dot{V}(\mathbf{e},\, t) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial \mathbf{e}}\mathbf{A}(t)\mathbf{e} \le -c_3\|\mathbf{e}\|^2 \\ \left\|\frac{\partial V}{\partial \mathbf{e}}\right\| \le c_4\|\mathbf{e}\|, \end{gathered}$$

with $c_1 \ldots c_4$ positive constants. Let $S_c = \{\mathbf{e} \,|\, V(\mathbf{e},\, t) \le c\}$ be a level set of function V. Since V is radially unbounded, S_c is a compact set. Due to the assumed boundedness of $(\mathbf{v}_C,\, \boldsymbol{\omega}_C)$, $\mathbf{g}(\mathbf{e},\, t)$ is (locally) Lipschitz and there exists a a positive constant M such that $\|\mathbf{g}(\mathbf{e},\, t)\| \le M\|\mathbf{e}\|$ in S_c. Using the Lyapunov candidate V for the perturbed system we get

$$\dot{V}(\mathbf{e},\, t) \le -c_3\|\mathbf{e}\|^2 + \left\|\frac{\partial V}{\partial \mathbf{e}}\right\| \mathbf{g}(\mathbf{e},\, t)\| \le -c_3\|\mathbf{e}\|^2 + c_4 M\|\mathbf{e}\|^2.$$

If M is small enough to satisfy the bound $M < c_3/c_4$, $\dot{V}$ is negative definite on S_c. Therefore, if the initial error $\mathbf{e}(t_0)$ is such that

$$\|\mathbf{e}(t_0)\|^2 \le \frac{V(\mathbf{e}(t_0),\, t_0)}{c_1} \le \frac{c}{c_1}, \tag{5.19}$$

system (5.11)–(5.17) converges exponentially to the origin. Note that a less conservative estimation on the initial error norm can be obtained by considering that observer (5.10)–(5.16) can be initialized with the measured states $\mathbf{x}_m$. In this case, (5.19) reduces to

$$\|\mathbf{e}(t_0)\|^2 = \|\mathbf{z}(t_0)\|^2 \le \frac{c}{c_1}.$$

□

This results demonstrates the possibility to use the measured moments and the known camera velocity to estimate vector $[A \;\; B \;\; C]^T$ without any special assumption on the shape of the object considered. Note, however, that the persistency of excitation condition (5.2) is supposed to hold. As explained before, this is equivalent to assume that $\|\mathbf{\Gamma}_2(t)\|$ does not definitely vanish over time. Structure of the $n \times 3$ matrix $\mathbf{\Gamma}_2(t)$ depends on the number and kind of moments considered for the estimation, as can be seen from (5.9). Therefore, choice of which moments to exploit for the estimation is of crucial importance in order to meet condition (5.2), and ongoing research is currently devoted to this topic. As a first evaluation, simulation results are presented in Sect. 5.4.1 in order to illustrate the observer behavior when area and barycenter are chosen as moments. This choice, for instance, can improve the overall convergence properties of the estimation w.r.t. the case of a target point feature, since the information relative to the area proves to be relevant to reduce the situations in which condition (5.2) is not met. In any case, since functions $\lambda_i(m_{kl}, \mathbf{v}_C)$ in (5.9) are such that $\lambda_i(m_{kl}, \mathbf{0}) \equiv 0$ [Chaumette 2004], the persistency of excitation requires, again, that the camera must necessarily move with a nonzero linear velocity $\mathbf{v}_C$ in order to have a converging estimation process.

Plane orientation n known

In some cases, it is possible to obtain plane orientation $\mathbf{n}$ by a direct evaluation. For instance, if the homography matrix $\mathbf{H}$ between the current and the desired views is available, $\mathbf{n}$ can be recovered by suitably decomposing $\mathbf{H}$. As explained in Sect. 3.3.2, computation of the homography typically requires the tracking and matching of several distinct points on the current/desired images, but there also exist techniques to obtain $\mathbf{H}$ from a dense unstructured object [Chesi *et al.* 2000; Benhimane & Malis 2004; Malis *et al.* 2004].

If $\mathbf{n}$ is known, estimation of $[A\ B\ C]^T$ is considerably simplified since the unmeasurable quantities only reduce to the plane distance d (see (4.11)). Indeed, in this case we can set $\mathbf{x}_m = [m_{i_1 j_1} \dots m_{i_n j_n}]^T \in \mathbb{R}^n$, as before, and $\mathbf{x}_u = 1/d \in \mathbb{R}^l$, $l = 1$. As a consequence, (4.13) can be rearranged as

$$\begin{aligned}\dot{m}_{ij} &= -\frac{n_x\lambda_A(m_{kl}, \mathbf{v}_C) + n_y\lambda_B(m_{kl}, \mathbf{v}_C) + n_z\lambda_C(m_{kl}, \mathbf{v}_C)}{d} + \lambda_D(m_{kl}, \boldsymbol{\omega}_C) = \\ &= \frac{\lambda(\mathbf{n}, m_{kl}, \mathbf{v}_C)}{d} + \lambda_D(m_{kl}, \boldsymbol{\omega}_C),\end{aligned} \tag{5.20}$$

and dynamics of $\mathbf{x}_m$ becomes

$$\begin{aligned}\dot{\mathbf{x}}_m &= \frac{1}{d}\begin{bmatrix}\lambda_1(\mathbf{n}, m_{kl}, \mathbf{v}_C) \\ \vdots \\ \lambda_n(\mathbf{n}, m_{kl}, \mathbf{v}_C)\end{bmatrix} + \begin{bmatrix}\lambda_{D_1} \\ \vdots \\ \lambda_{D_n}\end{bmatrix} = \\ &= \mathbf{\Gamma}_3\mathbf{x}_u + \mathbf{\Pi}_3.\end{aligned} \tag{5.21}$$

By choosing the update law

$$\dot{\hat{\mathbf{x}}}_m = \mathbf{\Gamma}_3\hat{\mathbf{x}}_u + \mathbf{\Pi}_3 + \mathbf{K}_1(\mathbf{x}_m - \hat{\mathbf{x}}_m), \quad \mathbf{K}_1 > 0, \tag{5.22}$$

we obtain the same $\dot{\boldsymbol{\xi}}$ error dynamics as in (5.11) with $\boldsymbol{\Omega}^T(t) = \boldsymbol{\Gamma}_3(t)$. Expression of $\dot{\mathbf{x}}_u$ can be derived from (5.14) as

$$\dot{\mathbf{x}}_u = -\frac{\mathbf{n} \cdot \mathbf{v}_C}{d^2} = -\mathbf{x}_u^2 \mathbf{n} \cdot \mathbf{v}_C,$$

from which, by designing the update law

$$\dot{\widehat{\mathbf{x}}}_u = -\widehat{\mathbf{x}}_u^2 \mathbf{n} \cdot \mathbf{v}_C + \mathbf{K}_2 \boldsymbol{\Gamma}_3^T \boldsymbol{\xi}, \quad \mathbf{K}_2 > 0, \tag{5.23}$$

we get the $\dot{\mathbf{z}}$ error dynamics

$$\dot{\mathbf{z}} = -(\mathbf{x}_u^2 - \widehat{\mathbf{x}}_u^2) \mathbf{n} \cdot \mathbf{v}_C - \mathbf{K}_2 \boldsymbol{\Gamma}_3^T \boldsymbol{\xi}. \tag{5.24}$$

The first term in (5.24) may be again considered as a vanishing perturbation term $\mathbf{g}(\mathbf{e}, t)$, so that exponential convergence of observer (5.22)–(5.23) can be proven by following the same arguments given for the general case (Sect. 5.2.2).

Concerning condition (5.2), the same former considerations about number and kind of moments to be included in $\mathbf{x}_m$ hold also in this case. There is, however, a slight difference which may be important in practical implementations: while $\boldsymbol{\Gamma}_2(t)$ in (5.9) is a $n \times 3$ matrix, $\boldsymbol{\Gamma}_3(t)$ is always a column vector of dimension n. Hence, in this case, it is sufficient that one component of $\boldsymbol{\Gamma}_3(t)$ does not vanish over time for the persistency of excitation condition to hold. In many practical situations, this can result in a milder constraint than requiring, as in the general case, full-rankness of matrix $\boldsymbol{\Gamma}_2(t)$ over time. Such difference is, of course, due to the assumed knowledge of $\mathbf{n}$ which reduces the number of unknown parameters to be estimated.

Case of a sphere

In the previous developments, we addressed the estimation of $\boldsymbol{\chi}(t)$ under the sole assumption that object $\mathcal{O}$ possesses a planar limb surface, but without posing other special requirements on its geometric structure. Of course, if some additional information about $\mathcal{O}$ is available, one can exploit this knowledge in order to obtain an improved estimation scheme tailored for the specific case. As an illustrative example, in this section we consider the design of the estimation algorithm when object $\mathcal{O}$ is a sphere. This case has also a practical relevance in the mobile robotics field when, e.g., robots are committed with visual tasks involving tracking/positiong w.r.t. a ball, and so on.

Consider a 3D sphere, with center $P_0 = [X_0 \ Y_0 \ Z_0]^T$ and radius R, represented by the equation

$$(X - X_0)^2 + (Y - Y_0)^2 + (Z - Z_0)^2 - R^2 = 0.$$

The sphere is an example of a 3D object with a planar limb surface, and, in this case, (4.10) becomes

$$\frac{1}{Z} = \frac{X_0}{K} p_u + \frac{Y_0}{K} p_v + \frac{Z_0}{K}, \tag{5.25}$$

where $K = X_0^2 + Y_0^2 + Z_0^2 - R^2$ [Espiau *et al.* 1992]. From (5.25) and (4.10), it follows

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \frac{1}{K} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} = -\frac{\mathbf{n}}{d}, \tag{5.26}$$

implying that $\mathbf{n}$ lies on the ray passing through the sphere center P_0. The projection of a sphere on the image plane is the ellipse

$$(X_0 p_u + Y_0 p_v + Z_0)^2 - K(p_u^2 + p_v^2 + 1) = 0, \tag{5.27}$$

with an equivalent expression in terms of image moments

$$\begin{aligned} &n_{02}p_u^2 + n_{20}p_v^2 - 2n_{11}p_u p_v + 2(n_{11}y_g - n_{02}x_g)p_u + 2(n_{11}x_g - n_{20}y_g)p_v + n_{02}x_g^2 + n_{20}y_g^2 \\ &-2n_{11}x_g y_g + 4n_{11}^2 - 4n_{20}n_{02} = 0, \end{aligned} \tag{5.28}$$

where $\bar{p}_g = [x_g \ \ y_g \ \ 1]^T$ is the ellipse barycenter in homogeneous coordinates, and n_{ij} are normalized centered moments of order $i + j$ [Chaumette 2004]. By equating (5.27) and (5.28), it follows

$$\begin{aligned} x_g &= \frac{X_0 Z_0}{Z_0^2 - R^2} \\ y_g &= \frac{Y_0 Z_0}{Z_0^2 - R^2}, \end{aligned}$$

which, plugged into (5.25), yields

$$Z_g = \frac{Z_0^2 - R^2}{Z_0} \tag{5.29}$$

as the barycenter depth, i.e., the depth of the point on the limb plane whose projection is $\bar{p}_g$. In order to evaluate $\mathbf{n}$ in terms of image quantities, one could hope that the 3D barycenter backprojection $P_g = Z_g \bar{p}_g$ and the sphere center P_0 were aligned. Indeed, in this case, it would be possible to obtain $\mathbf{n}$ as the direction of the measured barycenter $\bar{p}_g$. Unfortunately, while $x_g Z_g = X_0$ and $y_g Z_g = Y_0$, from (5.29) it is $Z_g \neq Z_0$, so that P_g and P_0 do not share the same 3D direction (actually, $Z_g < Z_0$, i.e., P_g is always in front of P_0). Note that, however, if $R \ll Z_0$, i.e., if the sphere radius is small compared to the distance of the sphere center from the camera, (5.29) can be approximated as $Z_g \simeq Z_0$, and (5.26) becomes

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \frac{1}{Z_g} \begin{bmatrix} \dfrac{x_g}{x_g^2 + y_g^2 + 1} \\ \dfrac{y_g}{x_g^2 + y_g^2 + 1} \\ \dfrac{1}{x_g^2 + y_g^2 + 1} \end{bmatrix} = \frac{\mathbf{n}_g}{Z_g}. \tag{5.30}$$

Therefore, under this approximation, the only unmeasurable quantity reduces to Z_g and it is possible to proceed similarly as in Sect. 5.2.2, by setting $\mathbf{x}_m = [m_{i_1 j_1} \dots m_{i_n j_n}]^T \in \mathbb{R}^n$ and

$\mathbf{x}_u = 1/Z_g \in \mathbb{R}^l$, $l = 1$. Dynamics of $\mathbf{x}_m$ and $\dot{\widehat{\mathbf{x}}}_m$ are given by (5.21) and (5.22), where d is replaced by Z_g, and $\mathbf{n}$ by $\mathbf{n}_g$. Furthermore, by using the last row of (3.9), we have

$$\begin{aligned} \dot{\mathbf{x}}_u &= -\frac{\dot{Z}_g}{Z_g^2} \simeq -\mathbf{x}_u^2 \dot{Z}_0 = \mathbf{x}_u^2(u_3 + Y_0 u_4 - X_0 u_5) = \\ &= \mathbf{x}_u^2 u_3 + \mathbf{x}_u(y_g u_4 - x_g u_5). \end{aligned} \tag{5.31}$$

The update law for $\widehat{\mathbf{x}}_u$ is then chosen as

$$\dot{\widehat{\mathbf{x}}}_u = \widehat{\mathbf{x}}_u^2 u_3 + \widehat{\mathbf{x}}_u(y_g u_4 - x_g u_5) + \mathbf{K}_2 \mathbf{\Gamma}_4^T \boldsymbol{\xi} \tag{5.32}$$

which yields the $\dot{z}$ error dynamics

$$\dot{\mathbf{z}} = (\mathbf{x}_u^2 - \widehat{\mathbf{x}}_u^2)u_3 + \mathbf{z}(y_g u_4 - x_g u_5) - \mathbf{K}_2 \mathbf{\Gamma}_4^T \boldsymbol{\xi}. \tag{5.33}$$

Since the first two perturbation terms in (5.33) are, again, vanishing for $\mathbf{z} = \mathbf{0}$, convergence of observer (5.22)–(5.32) can be proved an in the previous sections.

It is interesting to note that, for a sphere, the design of the observer structure is conceptually equivalent to the situation discussed in Sect. 5.2.2. Indeed, in both cases, the plane normal direction $\mathbf{n}$ is directly evaluated in terms of image data, and the only unknown quantity becomes the 'depth' of the target object. The only relevant difference is that the special geometric structure of the sphere allows a direct evaluation of $\mathbf{n}$, while in the previous (and more general) case a homography decomposition between current and desired view may be needed in order to obtain the same information.

5.3 Focal Length Estimation

As discussed in the previous section, a pure camera rotation is not useful for depth observation since in this case the motion imposed to the features does not depend on the associated 3D information $\boldsymbol{\chi}(t)$. Anyway, this motion does still depend on the camera intrinsic parameters $\mathbf{K}_C$, and in particular on the focal length λ (see (3.12)): therefore it is possible to exploit the same observer structure as before in order to estimate the constant value of λ without being affected by uncertainties on $\boldsymbol{\chi}(t)$. In principle, any image feature could be considered for such estimation, e.g., both point features and moments would be suitable. However, for the sake of illustration, in the following we focus only on the case of point features. Indeed, by considering point features, a 'physical' interpretation of the persistency of excitation condition (5.2) is possible, and, in addition, any equivalent formulation involving image moments, or other image quantities, can be tailored in straightforward way.

Since the goal of this section is the online observation of λ, the assumption of perfect knowledge of matrix $\mathbf{K}_C$ must be partially relaxed. In particular, we assume that only the values of $(u_0,\, v_0)$ are known, while considering λ as an unknown constant parameter. Therefore, by letting $p_u = \widetilde{p}_u - u_0$ and $p_v = \widetilde{p}_v - v_0$, i.e., by centering the pixel coordinates of $\widetilde{p}$ w.r.t. the camera principal point, the interaction matrix of p takes the form

$$\begin{bmatrix} \dot{p}_u \\ \dot{p}_v \end{bmatrix} = \begin{bmatrix} -\dfrac{\lambda}{Z} & 0 & \dfrac{p_u}{Z} & \dfrac{p_u p_v}{\lambda} & -\left(\lambda + \dfrac{p_u^2}{\lambda}\right) & p_v \\ 0 & -\dfrac{\lambda}{Z} & \dfrac{p_v}{Z} & \lambda + \dfrac{p_v^2}{\lambda} & -\dfrac{p_u p_v}{\lambda} & -p_u \end{bmatrix} \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \mathbf{J}_p(p,\, Z) \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix}. \tag{5.34}$$

Now, assume that a pure angular motion is imposed to the camera, and let $\mathbf{x}_m = p \in \mathbb{R}^n$, $n = 2$, $\mathbf{x}_u = [\lambda \;\, 1/\lambda]^T \in \mathbb{R}^l$, $l = 2$, be the measurable/unmeasurable state vectors, and $\mathbf{u} = \boldsymbol{\omega}_C \in \mathbb{R}^3$ be the input vector. Hence, in this case the state dynamics are expressed by the driftless system

$$\begin{aligned} \dot{\mathbf{x}}_m &= \boldsymbol{\Gamma}_5(t)\mathbf{x}_u + \boldsymbol{\Pi}_5(t) \\ \dot{\mathbf{x}}_u &= \mathbf{0} \\ \mathbf{y} &= \mathbf{x}_m, \end{aligned} \tag{5.35}$$

where

$$\boldsymbol{\Gamma}_5(t) = \begin{bmatrix} -u_2 & y_1 y_2 u_1 - y_1^2 u_2 \\ u_1 & y_2^2 u_1 - y_1 y_2 u_2 \end{bmatrix}, \quad \boldsymbol{\Pi}_5(t) = \begin{bmatrix} y_2 u_3 \\ -y_1 u_3 \end{bmatrix},$$

are functions of known quantities. Note that the introduction of the component $1/\lambda$ in $\mathbf{x}_u$, which at first glance may seem unnecessary, is aimed at obtaining a linear dependence on the unmeasurable states in (5.35).

Proceeding as in the previous section, we choose the update laws

$$\begin{aligned} \dot{\widehat{\mathbf{x}}}_m &= \boldsymbol{\Gamma}_5(t)\widehat{\mathbf{x}}_u + \boldsymbol{\Pi}_5(t) + \mathbf{K}_1(\mathbf{x}_m - \widehat{\mathbf{x}}_m), \quad \mathbf{K}_1 > 0, \\ \dot{\widehat{\mathbf{x}}}_u &= \mathbf{K}_2\boldsymbol{\Gamma}_5^T(t)(\mathbf{x}_m - \widehat{\mathbf{x}}_m), \quad \mathbf{K}_2 > 0, \end{aligned} \tag{5.36}$$

which, in this case, yield an error dynamics that matches exactly formulation (5.1) with $\mathbf{W} = -\mathbf{K}_1$, $\boldsymbol{\Omega}(t) = \boldsymbol{\Gamma}_5^T(t)$, $\boldsymbol{\Lambda} = \mathbf{K}_2$, and $\mathbf{S} = \mathbf{I}_2$. Hence, global convergence to the origin of $(\boldsymbol{\xi},\, \mathbf{z})$ is always guaranteed as long as conditions of Lemma 5.1 are met, in particular condition (5.2). Since $\boldsymbol{\Gamma}_5(t)$ is a 2×2 matrix, condition (5.2) implies that there must not exist a $\bar{t}$ such that, $\forall t > \bar{t}$ matrix $\boldsymbol{\Gamma}_5(t)\boldsymbol{\Gamma}_5^T(t)$ is not strictly positive definite. This can happen if and only if

$$\det \boldsymbol{\Gamma}_5(t) = y_1 y_2 u_2^2 - y_2^2 u_1 u_2 - y_1 y_2 u_1^2 + y_1^2 u_1 u_2 \equiv 0,\ \forall t > \bar{t}. \tag{5.37}$$

In order to fully understand the physical meaning of (5.37), it is useful to also look at its expression in terms of 3D coordinates of point P, i.e.,

$$\frac{\lambda^2}{P_z^2}(P_y u_4 - P_x u_5)(P_x u_4 + P_y u_5) = \frac{\lambda^2}{P_z^2}\,\eta_1\,\eta_2 \equiv 0,\ \forall t > \bar{t}. \tag{5.38}$$

Recalling that $P = [P_x\, P_y\, P_z]^T$ and $\boldsymbol{\omega}_C = [u_1\, u_2\, u_3]^T$, it is easy to see that η_1 is the third component of vector $[\boldsymbol{\omega}_C]_\times P$, while η_2 is the third component of vector $[\boldsymbol{\omega}_C]_\times P_{\pi/2}$, where

$P_{\pi/2} = [-P_y \, P_x \, P_z]^T$ is point P rotated $\pi/2$ radians about the camera $\mathbf{Z}_C$ axis. Therefore, from (5.37) and (5.38) we can state that the persistency of excitation condition for focal length observation is met if:

1. a rotation about the $\mathbf{X}_C$ or $\mathbf{Y}_C$ camera axes is present ($u_1 \neq 0$ or $u_2 \neq 0$). Note that a rotation about the optical axis $\mathbf{Z}_C$ is useless for the observation;

2. the feature point does not lie fixed at the center of the optical plane, i.e., $y_1 \neq 0$ or $y_2 \neq 0$;

3. the imposed rotation $\boldsymbol{\omega}_C$ induces on points P and $P_{\pi/2}$ a non zero linear velocity along the $\mathbf{Z}_C$ axis; i.e., these two points must either get closer to or further from the image plane as a consequence of the camera rotation.

5.4 Simulations

In the following, we present some simulation results meant to illustrate the properties of the observation techniques developed so far. The simulations are implemented in the MATLAB [Mathworks 2007] and Webots environments, under the assumption that the camera is carried by the end-effector of a robot system which can provide the chosen linear/angular motion. These numerical assessments will be further validated in Chapter 6 by means of experimental results obtained on real robots.

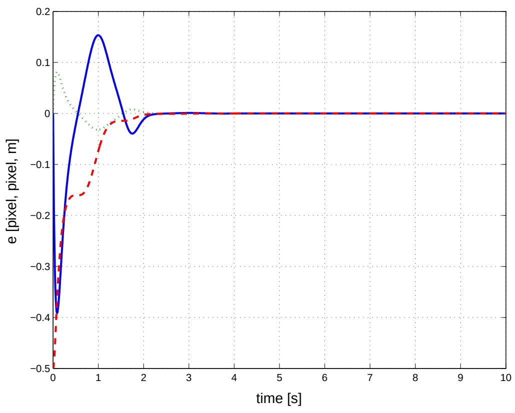
5.4.1 Observation of 3D quantities

Point features

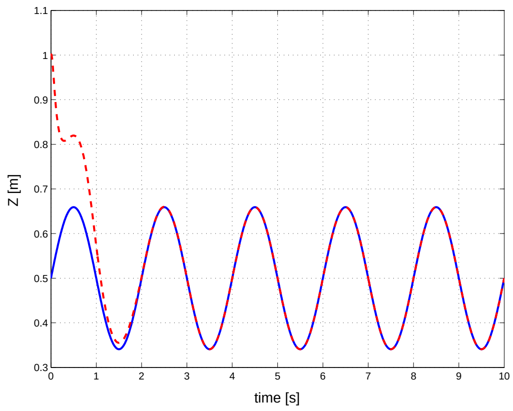
In order to show the performance of the depth observer (5.5) for point features, we begin with the simple case of a camera in sinusoidal motion along the $\mathbf{Z}_C$ axis, a case that, e.g., could not be addressed with the methods in [Matthies *et al.* 1989; Smith & Papanikolopoulos 1994] (see Sect. 5.2). The simulation data are:

$$\begin{array}{rcl} \left[\mathbf{x}_m^T(t_0)\;\mathbf{x}_u^T(t_0)\right]^T & = & [24 \quad -5 \quad 2]^T \\ \left[\widehat{\mathbf{x}}_m^T(t_0)\;\widehat{\mathbf{x}}_u^T(t_0)\right]^T & = & [24 \quad -5 \quad 1]^T \\ u_3(t) & = & 0.5\cos\pi t \\ \mathbf{K}_1 & = & 20\,\mathbf{I}_2 \\ \mathbf{K}_2 & = & 0.5 \end{array} .$$

Note that the first two components of the estimation error are initially zero because the feature position is measured and the observer is initialized accordingly. Figure 5.1 depicts the behavior of $\mathbf{e}(t)$ during the simulation and shows how the estimate of Z approaches the true value.

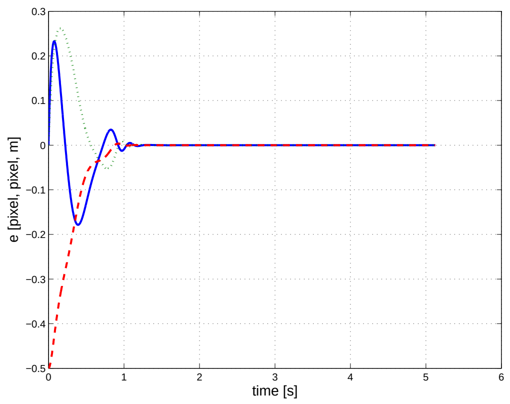


(a) Behavior of e_1 (solid blue line), e_2 (dotted green line) and e_3 (dashed red line) vs. time.

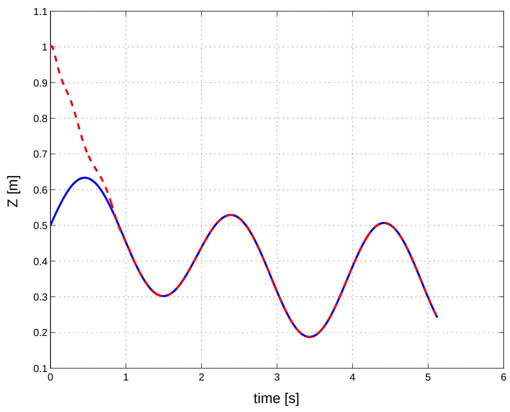


(b) True (solid blue line) and estimated (dashed red line) Z.

Figure 5.1: Observation with point features: **first** simulation.



(a) Behavior of e_1 (solid blue line), e_2 (dotted green line) and e_3 (dashed red line) vs. time.



(b) True (solid blue line) and estimated (dashed red line) Z.

Figure 5.2: Observation with point features: **second** simulation.

Convergence is reached after a transient of few seconds the motion along the camera optical axis.

Our second simulation involves a more complex camera motion consisting of a translation

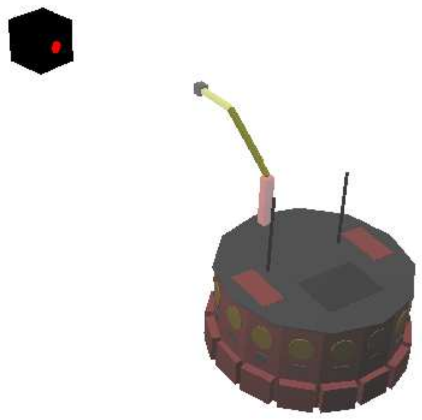


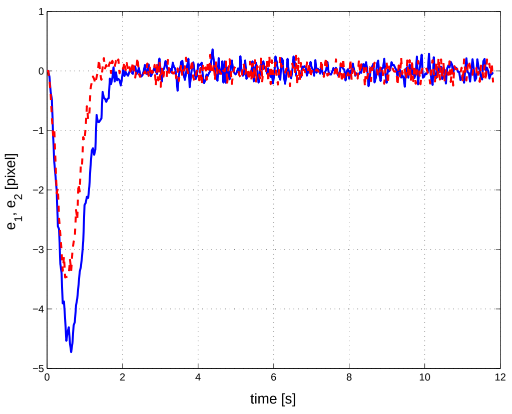
Figure 5.3: Observation with point features: **third** simulation. The camera is mounted on the end-effector of a NMM in a typical eye-in-hand configuration. The red dot on the target cube represents the point feature tracked during the simulation.

and a rotation about the $\mathbf{X}_C$ and $\mathbf{Z}_C$ axes. We set:

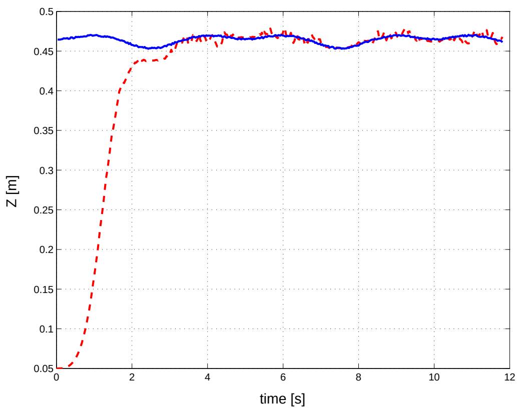
$$\begin{array}{rcl} \left[\mathbf{x}_m^T(t_0)\ \mathbf{x}_u^T(t_0)\right]^T & = & [10 \quad -10 \quad 2]^T \\ \left[\widehat{\mathbf{x}}_m^T(t_0)\ \widehat{\mathbf{x}}_u^T(t_0)\right]^T & = & [10 \quad -10 \quad 1]^T \\ u_1(t) & = & 0.1\cos 2\pi t \\ u_3(t) & = & 0.5\cos \pi t \\ u_4(t) & = & 0.6\cos \pi/2\, t \\ u_6(t) & = & 1 \\ \mathbf{K}_1 & = & 20\,\mathbf{I}_2 \\ \mathbf{K}_2 & = & 0.5 \end{array} .$$

Results of this simulation are shown in Fig. 5.2. Practically zero error is reached after 1 [s] of motion even if in this case the camera motion is quite complex. Note that, again, the first two components of the observation error are initially zero because the feature position is measured.

As an additional case study, we implemented the proposed algorithm in the Webots environment by considering a camera mounted on the end-effector of a mobile manipulator made of a unicycle-like platform carrying a 3R spatial manipulator (see Fig. 5.3). The idea was to test the performance of the proposed observer against the noise automatically introduced by the Webots engine. This noise is added on the image perceived by the camera and thus directly reflects on the feature extraction process. The objective is to estimate the depth of the target point (the red dot on the cube in Fig. 5.3), while the first and second link of the manipulator



(a) Behavior of e_1 (solid blue line) and e_2 (dashed red line) vs. time.



(b) True (solid blue line) and estimated (dashed red line) Z vs. time.

Figure 5.4: Observation with point features: **third** simulation.

move according to the velocity profiles:

$$\begin{aligned}\dot{q}_1 &= 0.2\sin 0.4\pi t\\ \dot{q}_2 &= 0.1\sin 0.8\pi t.\end{aligned}$$

The initial value of the estimated depth is set at $1/\widehat{x}_u(t_0) = 0.05$ and the gains were chosen as $\mathbf{K}_1 = 10\,\mathbf{I}_2$ and $\mathbf{K}_2 = 0.8$. Despite the noise, the observer is able to estimate accurately the actual value of the depth Z, as shown in Fig. 5.4. A video clip of this simulation can be found at `www.dis.uniroma1.it/~labrob/research/depth_est.html`.

Image moments

In this second set of simulations, we evaluate the performance of the observation schemes developed for image moments (Sect. 5.2.2). In particular, we focus on the case of a sphere (observer (5.22)–(5.32)). Indeed, as explained at the end of the previous section, this choice is equivalently representative, from the estimation point of view, of the scheme (5.22)–(5.23) for a known planar orientation, and ongoing research efforts are currently devoted to select a suitable set of moments for the general case of observer (5.10)–(5.16). The algorithms are implemented again in the Webots environment by considering a camera/NMM system as in Fig. 5.5.

In the first simulation, the moments used for the estimation are the area a and the barycenter $(x_g,\, y_g)$ measured from the projection of a sphere with radius $R = 0.07$ [m] and lying at a distance of about 0.4 [m] from the camera. Hence, in this case, it is $\mathbf{x}_m = [a\ x_g\ y_g]^T \in \mathbb{R}^3$, $\mathbf{x}_u = 1/Z_g$, and $\mathbf{\Gamma}_4(t) \in \mathbb{R}^3$. The robot is commanded with a periodic predefined motion

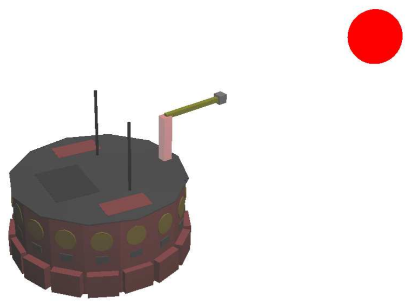
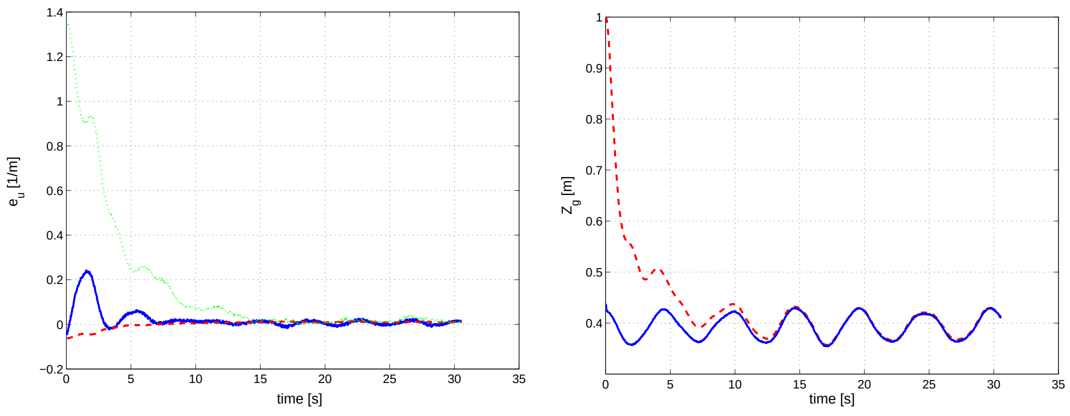


Figure 5.5: Observation with image moments. The camera is mounted on the end-effector of a NMM in a typical eye-in-hand configuration. The red sphere represents the target object tracked during the simulation.



(a) Behavior of $\mathbf{e}_u = \mathbf{x}_u - \widehat{\mathbf{x}}_u$ over time.

(b) True (solid blue line) and estimated (dashed red line) Z_g over time.

Figure 5.6: Observation with image moments: **first** simulation.

according to the velocity profiles $v(t) = 0.4 \sin 0.4\,\pi\, t$ and $\dot{q}_1(t) = 0.2 \sin 1.6\,\pi\, t$, where v is the platform linear velocity and $\dot{q}_1$ the first link velocity, and the observer is initialized with $\widehat{\mathbf{x}}_u(t_0) = 1/\widehat{Z}_g(t_0) = 1$ [m], and gains $\mathbf{K}_1 = 10\,\mathbf{I}_3$ and $\mathbf{K}_2 = 6000$. Results of the simulation are presented in Figs. (5.6–5.7). In particular, in Fig. 5.6(a) we report the behavior of the error vector $\mathbf{e}_u = \mathbf{x}_u - \widehat{\mathbf{x}}_u$, where the first term is evaluated according to (5.26), and the second is given by (5.30) with Z_g replaced by its estimate $\widehat{Z}_g$. Hence, one can check that, although neglecting the sphere radius R, observer (5.22)–(5.32) is able to recover the actual value of $[A\ B\ C]^T$ in an accurate way. As a comparison, Fig. 5.6(b) shows how the estimate $\widehat{Z}_g$ approaches the true value Z_g obtained from (5.29) after about 10 [s] of motion. Finally,

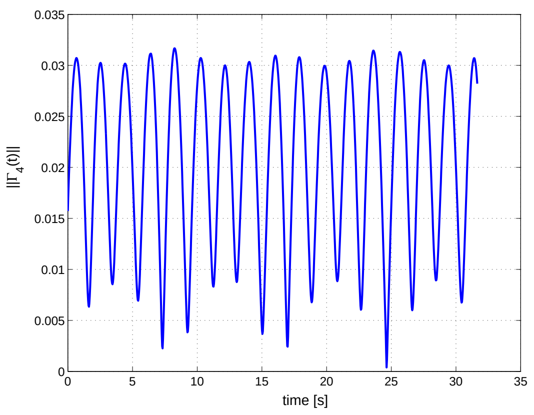
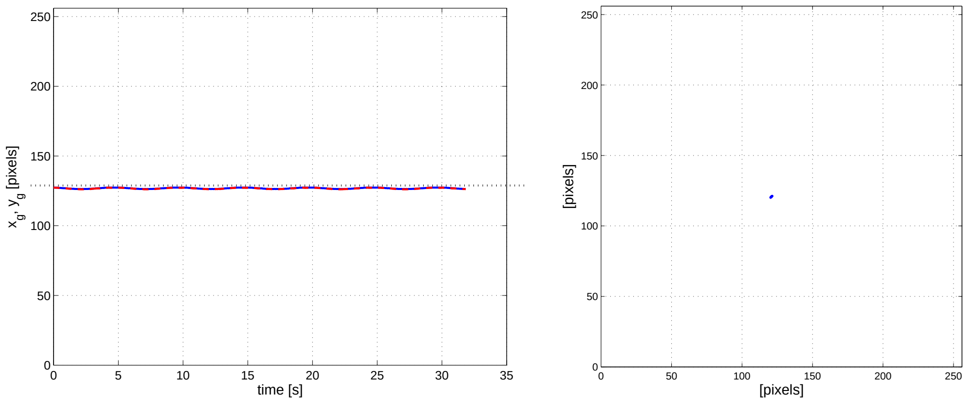


Figure 5.7: Observation with image moments: **first** simulation. Behavior of $\|\mathbf{\Gamma}_4(t)\|$ over time.



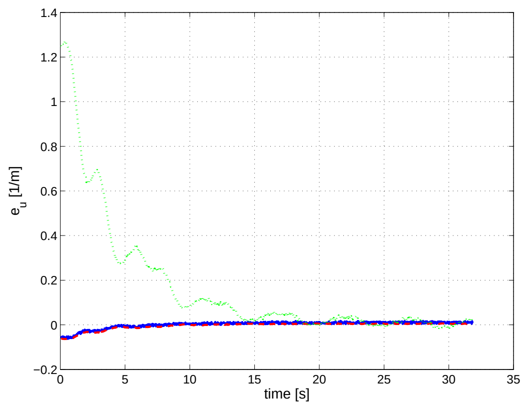
(a) Behavior of (x_g, y_g) over time. Note that the barycenter coordinates do not almost change during the motion.

(b) Image plane motion of (x_g, y_g). The sphere barycenter stays almost at the same spot.

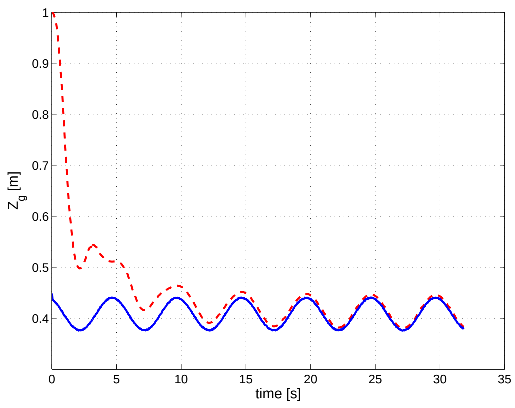
Figure 5.8: Observation with image moments: **second** simulation.

Fig. 5.7 depicts the behavior of $\|\mathbf{\Gamma}_4(t)\|$, showing that the choice of moments in $\mathbf{x}_m$ meets the persistency of excitation condition ($\|\mathbf{\Gamma}_4(t)\|$ does not definitely vanish over time).

In the second simulation, we considered the same situation as before but with a different motion imposed to the robot: we discarded the first link velocity command $\dot{q}_1$, while keeping the previous platform linear velocity command v. As a result, the camera moves is such a way that the sphere barycenter stays almost fixed at the center of the image, i.e., the center of the sphere lies on the camera optical axis during the backward/forward camera motion, see Figs. 5.8(a–b). This choice is meant to demonstrate the potential benefits of using moments for 3D structure estimation. Indeed, in this situation, an estimation scheme designed for a



(a) Behavior of $\mathbf{e}_u = \mathbf{x}_u - \widehat{\mathbf{x}}_u$ over time.



(b) True (solid blue line) and estimated (dashed red line) Z_g over time.

Figure 5.9: Observation with image moments: **second** simulation.

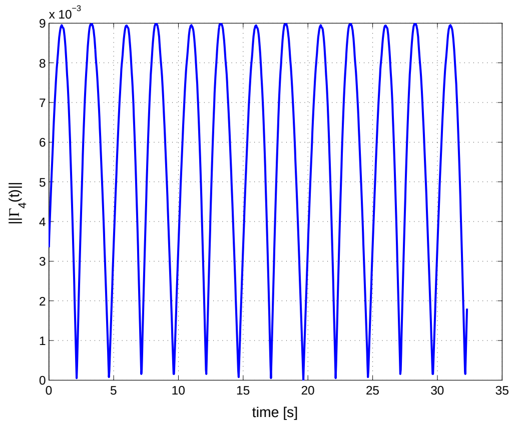


Figure 5.10: Observation with image moments: **second** simulation. Behavior of $\|\mathbf{\Gamma}_4(t)\|$ over time.

point feature (like the one proposed in [De Luca *et al.* 2007b]) could not correctly recover the feature depth since the camera linear velocity $\mathbf{v}_C$ and the point feature projection ray would be almost coincident, thus yielding an ill-conditioned problem (see Sect. 5.2.1). On the other hand, exploiting the area a besides barycenter (x_g, y_g), makes the estimation possible. This can be verified from Figs. (5.9–5.10) which show, as before, the good convergence properties of observer (5.22)–(5.32). In particular, despite the unfavorable arrangement of the sphere/camera relative motion, the persistency of excitation condition is still satisfied, as can be checked from Fig. 5.10.

5.4.2 Observation of focal length

By following the outline of the previous section, we consider two numerical case studies with the aim of assessing the focal length observer (5.36).

Data of the first MATLAB simulation are:

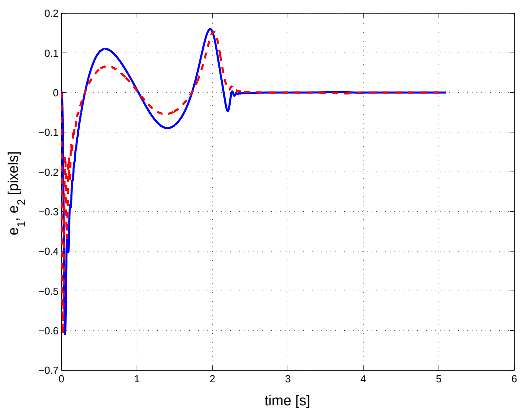
$$
\begin{array}{rcl}
\left[\mathbf{x}_m^T(t_0)\ \mathbf{x}_u^T(t_0)\right]^T & = & [10 \quad -10 \quad 128 \quad 1/128]^T \\
\left[\widehat{\mathbf{x}}_m^T(t_0)\ \widehat{\mathbf{x}}_u^T(t_0)\right]^T & = & [10 \quad -10 \quad 0 \quad 0]^T \\
u_1(t) & = & 0.5\cos\pi/2t \\
u_2(t) & = & 0.3\sin\pi t \\
\mathbf{K}_1 & = & 50\,\mathbf{I}_2 \\
\mathbf{K}_2 & = & \begin{bmatrix} 5000 & 0 \\ 0 & 1 \end{bmatrix}
\end{array} .
$$

In Fig. 5.11(a–c), the evolution of the error vector $\mathbf{e}(t)$ is reported, from which we can check that convergence is practically reached after 4 seconds of motion. It is worth noting that the behavior of $e_3(t)$ appears a bit erratic, alternating fast decreases (around $t = 2$ [s]), with nearly flat transients (at $t = 1$ [s] or $t = 3$ [s]). This is in close relationship with the persistency of excitation condition. Indeed, whenever matrix $\mathbf{\Gamma}_5(t)$ is far from singularity, the observation process can converge quickly, while when $\mathbf{\Gamma}_5(t)$ is close to ill-conditioning, convergence nearly stops. By defining $\sigma(\mathbf{A})$ as the smallest singular value of a matrix $\mathbf{A}$, we reported in Fig. 5.11(d) the evolution of $\eta_1(t)$, $\eta_2(t)$ and $20\sigma(\mathbf{\Gamma}_5(t))$ vs. time (the factor 20 is introduced to obtain a comparable scale between $\sigma(\mathbf{\Gamma}_5(t))$ and $\eta_1(t)$, $\eta_2(t)$). As expected, the slow convergence phases correspond to a small value of $\sigma(\mathbf{\Gamma}_5(t))$ which goes to zero when either $\eta_1(t) \to 0$ or $\eta_2(t) \to 0$.

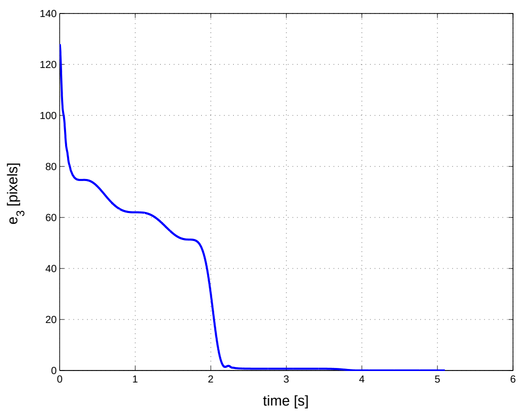
In the second simulation, run in Webots, we modeled the camera as a fully actuated free-flying object in order to easily impose a given angular motion, see Fig. 5.12 for a snapshot of the system. The simulation data are

$$
\begin{array}{rcl}
\left[\mathbf{x}_m^T(t_0)\ \mathbf{x}_u^T(t_0)\right]^T & = & [0.35 \quad -20.32 \quad 128 \quad 1/128]^T \\
\left[\widehat{\mathbf{x}}_m^T(t_0)\ \widehat{\mathbf{x}}_u^T(t_0)\right]^T & = & [0.35 \quad -20.32 \quad 0 \quad 0]^T \\
u_4(t) & = & 0.2\cos 0.5\pi t \\
u_5(t) & = & 0.2\sin 0.7\pi t \\
\mathbf{K}_1 & = & 20\,\mathbf{I}_2 \\
\mathbf{K}_2 & = & \begin{bmatrix} 600 & 0 \\ 0 & 1 \end{bmatrix}
\end{array} .
$$

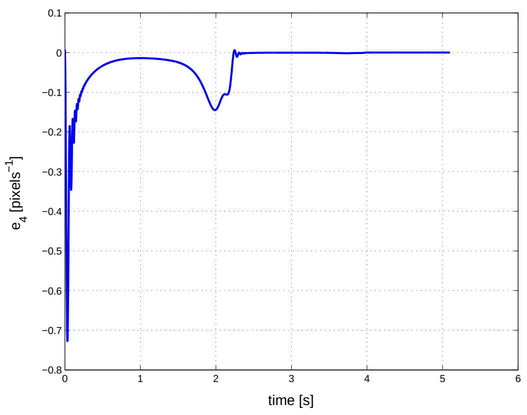
The behavior of $\mathbf{e}(t)$, depicted in Figs. 5.13(a–c), shows that convergence is reached after 5 [s]. of motion despite the presence of noise. Fig. 5.13 (d) allows again to evaluate the influence of $\sigma(\mathbf{\Gamma}_5(t))$ on the convergence rate of the algorithm.



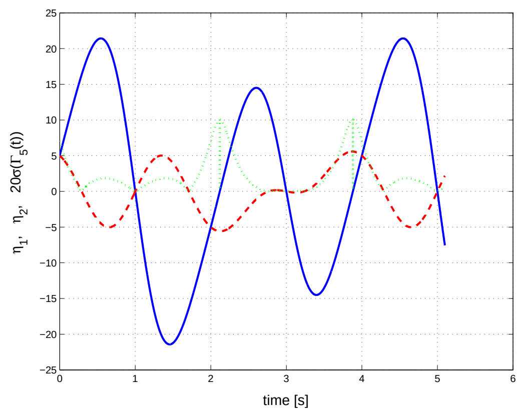
(a) Evolution of e_1 (solid blue line) and e_2 (dashed red line) vs. time.



(b) Evolution of $e_3 = \lambda - \widehat{\lambda}$ vs. time.



(c) Evolution of $e_4 = 1/\lambda - 1/\widehat{\lambda}$ vs. time.



(d) Evolution of η_1 (solid blue line), η_2 (dashed red line) and $20\sigma(\mathbf{\Gamma}_5(t))$ (dotted green line) vs. time.

Figure 5.11: Focal length observation: **first** simulation.

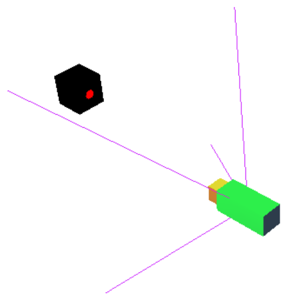
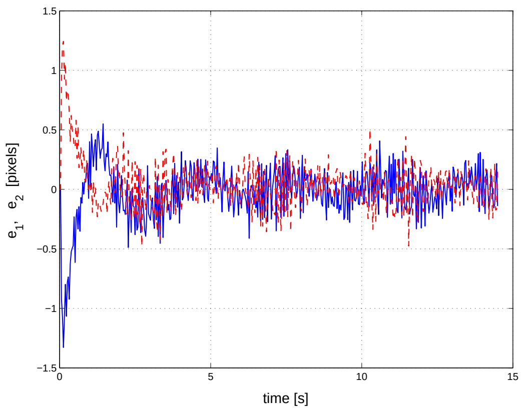
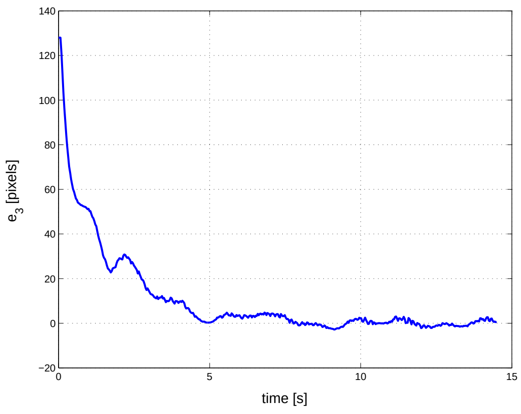


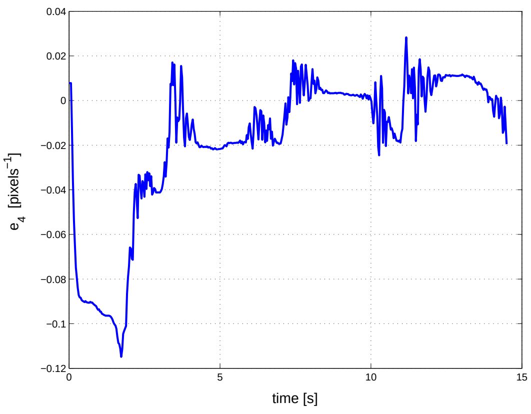
Figure 5.12: Focal length observation: **second** simulation. The camera is modeled as a fully actuated free-flying system. The red dot on the cube represents the feature point tracked during the motion.



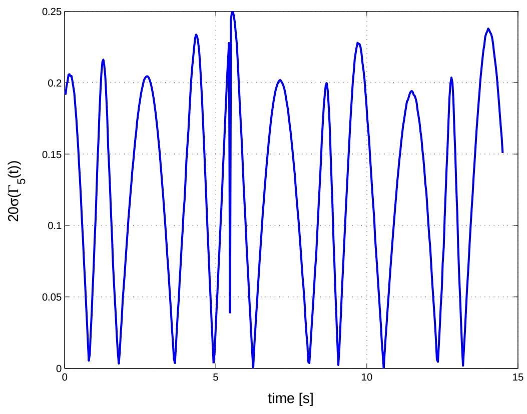
(a) Evolution of e_1 (solid blue line) and e_2 (dashed red line) vs. time.



(b) Evolution of e_3 vs time.



(c) Evolution of e_4 vs time.



(d) Evolution of $20\sigma(\mathbf{\Gamma}_5(t))$ vs time.

Figure 5.13: Focal length observation: **second** simulation.

Part III

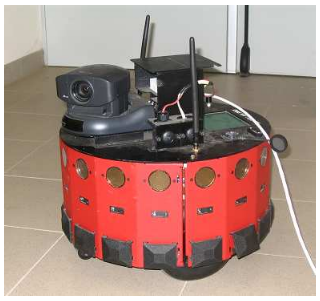
Visual Control: Experiments and Application

6
Experimental Validation

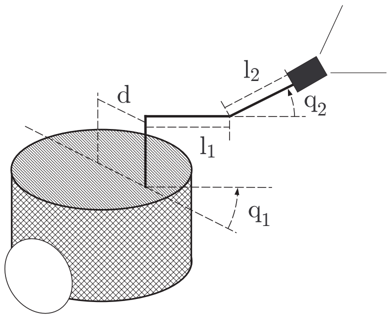
This Chapter is meant to provide an experimental validation of the theoretical tools developed in this Thesis. To this end, we propose a collection of results spanning the various topics introduced in the previous developments, namely

- kinematic modeling and control of fixed-base/mobile robot manipulators (Chapters 1–2);
- IBVS schemes based on point features and image moments (Chapter 4);
- on-line observation of 3D structure and camera intrinsic quantities (Chapter 5).

Backbone of this evaluation is the framework of Visual Servoing and, in particular, the subclass of IBVS methods. Indeed, as discussed in Chapter 4, IBVS implementations naturally embrace the aforementioned topics, in particular for what concerns redundancy resolution and structure identification. For instance, the numerical results given in Sect. 4.6 show that suitable exploitation of redundancy can considerably improve the overall visual task execution, by allowing realization of tasks that would be close to singularity if addressed altogether. This fact, and the other conclusions drawn in the previous assessments, are then resumed in the next sections, and further evaluated by means of experiments on real robots equipped with an onboard camera. In the same spirit, we propose an experimental validation of the observation



(a) MagellanPro with the onboard pan-tilt camera.



(b) Schematic view of the robot.

Figure 6.1: The robot used in our experiments.

schemes developed in Chapter 5, by focusing both on their convergence properties as stand-alone systems, and on their possible inclusion into the loop of IBVS feedbacks. The aim is twofold: to effectively show how 3D structure observation and IBVS can be integrated, and to point out the benefits of such integration in terms of stability and overall performance.

In the following, results involving use of redundancy and 3D estimation are presented in Sects. 6.1–6.2, while Sect. 6.3 reports an experimental validation of the focal length observation scheme developed in Sect. 5.3.

6.1 Experiments of redundancy exploitation

The experiments considered in this section have been conducted on the MagellanPro (Fig. 6.1(a)), a unicycle-like WMR equipped with a pan-tilt camera[1] [De Luca *et al.* 2008a]. By considering the pan-tilt unit equivalent to a (polar) 2R manipulator (Fig. 6.1(b)), this robotic system falls into the class on NMMs and is conceptually equivalent to the case considered in Sect. 4.6.2. Therefore, in the following we revisit the simulation results presented in Chapter 4 with the aim of reproducing similar boundary conditions on this experimental setup.

To this end, let the configuration vector $\mathbf{q}$ be partitioned as $\mathbf{q} = [\mathbf{q}_p \ \mathbf{q}_m]^T \in \mathbb{R}^n$, $n = 5$, where $\mathbf{q}_p = [x \ y \ \theta]^T \in \mathbb{R}^3$ represents the mobile platform configuration (position and orientation), and $\mathbf{q}_m = [q_1 \ q_2]^T \in \mathbb{R}^2$ the 'manipulator' joint variables (i.e., pan and tilt of the camera). Furthermore, let $\mathbf{u} = [\mathbf{u}_p \ \mathbf{u}_m]^T \in \mathbb{R}^p$, $p = 4$, be the induced partition of the robot velocity commands and assume $\dot{\mathbf{q}}_m = \mathbf{u}_m$, i.e., that any pan/tilt velocity can be arbitrarily specified. As for the mobile platform, it is $\dot{\mathbf{q}}_p = \mathbf{G}(\mathbf{q}_p)\mathbf{u}_p$, where the 3×2 matrix $\mathbf{G}(\mathbf{q}_p)$ is given by (2.19).

[1]Videos of these results are available at www.dis.uniroma1.it/~labrob/research/NMM_Visual_Servoing.html.

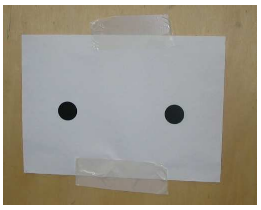


Figure 6.2: The target object used in our experiments.

With these settings, we obtain

$$\mathbf{J}_M(\mathbf{q}_m) = \begin{bmatrix} s_1 & -dc_1 - l_1 - l_2c_2 & -l_1 - l_2c_2 & 0 \\ c_1s_2 & ds_1s_2 & 0 & -l_2 \\ c_1c_2 & ds_1c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -c_2 & -c_2 & 0 \\ 0 & s_2 & s_2 & 0 \end{bmatrix}, \tag{6.1}$$

where c_i and s_i stand for $\cos q_i$ and $\sin q_i$, and the geometric quantities ($d = 0.135$ [m], $l_1 = 0.011$ [m], $l_2 = 0.0233$ [m]) are defined in Fig. 6.1(b).

As visual task, we consider again the regulation of two point features p_1, p_2 on the image plane, see Fig. 6.2 for a picture of the target object. Thus,

$$\mathbf{f} = [p_1^T \; p_2^T]^T \in \mathbb{R}^s, \quad s = 4, \tag{6.2}$$

and

$$\mathbf{J}_f(\mathbf{f}, \boldsymbol{\chi}) = \begin{bmatrix} \mathbf{J}_p(p_1, Z_1) \\ \mathbf{J}_p(p_2, Z_2) \end{bmatrix}. \tag{6.3}$$

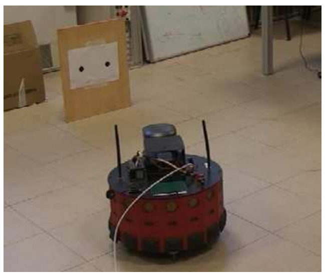
The image Jacobian $\mathbf{J}_{\text{img}}$ in (4.18) can then be evaluated from (6.1) and (6.3) where depth of the two feature points is estimated online via the observer (5.5). Implications of the coupling IBVS feedback/depth observation are extensively addressed in the next section. Here, we consider the estimated depth fully representative of the 'true' Z, and close the IBVS loop on this value.

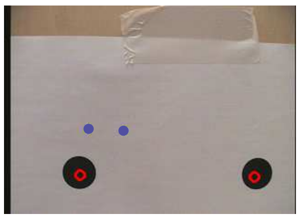
Note that, the dimension of the visual task defined in (6.2) matches the number of available robot commands ($s = p = 4$), so that the obtained image Jacobian $\mathbf{J}_{\text{img}}$ is a square 4×4 matrix.

6.1.1 Experiments with Task Priority

Although the task could be realized by direct inversion of $\mathbf{J}_{\text{img}}$ as in (2.1) with (2.3), the same issues reported in Sect. 4.6.2 affect this choice, i.e., lack of redundancy causes poor results when $\mathbf{J}_{\text{img}}$ is close to ill-conditioning. Consider, for example, the situation shown in Fig. 6.3(a–d)

(a) Initial external view.


(b) Final external view.


(c) Initial camera view.

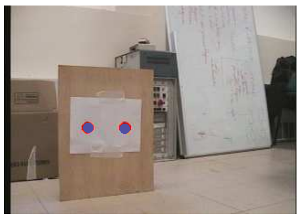
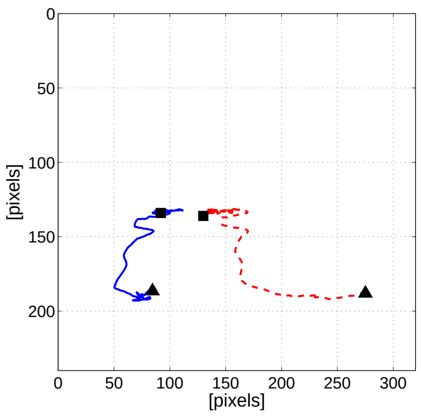
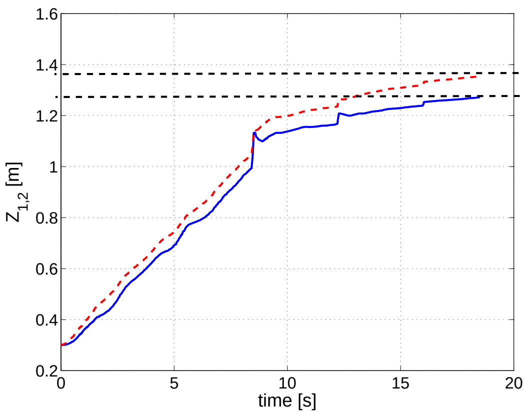

(d) Final desired camera view.

Figure 6.3: **TP experiment**. Initial and final robot views.



(a)



(b)

Figure 6.4: **TP experiment**. Left: motion, from ▲ to ■, of the two feature points on the image plane (p_1 is the blue solid line and p_2 the red dashed line). Right: Behavior of the estimated depths $\widehat{Z}_1(t)$ and $\widehat{Z}_2(t)$ over time. The dashed horizontal lines represent the final ground truth values of the depths.

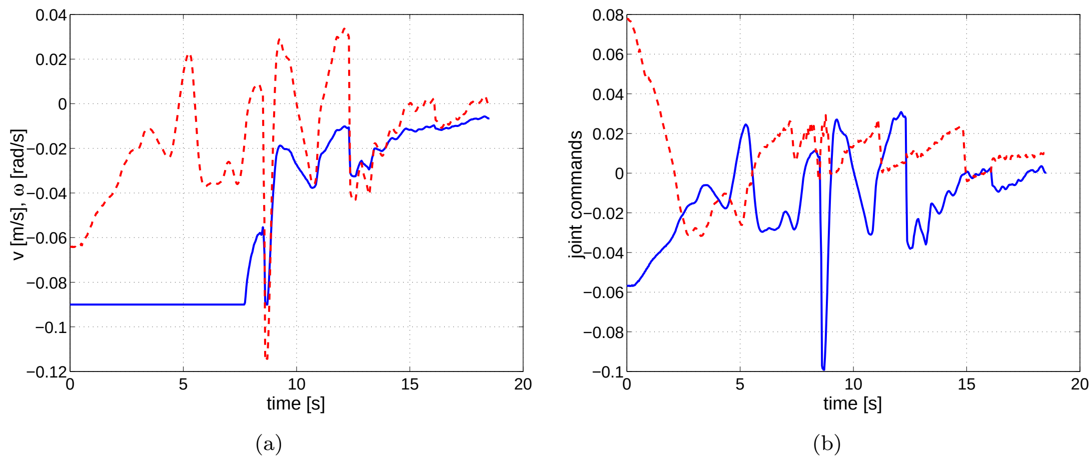


Figure 6.5: **TP experiment**. Left: platform linear velocity v (solid blue line) and angular velocity ω (dashed red line). Right: pan velocity $\dot{q}_1$ (solid blue line) and tilt velocity $\dot{q}_2$ (dashed red line).

corresponding to the following initial conditions:

$$\begin{cases} \mathbf{f}(t_0) & = [-0.4319\ 0.3803\ 0.6580\ 0.3893]^T \quad [\mathrm{m}] \\ Z_1(t_0) & = Z_2(t_0) = 0.21 \quad [\mathrm{m}] \\ q_1(t_0) & = 0.23 \quad [\mathrm{rad}] \\ q_2(t_0) & = -0.07 \quad [\mathrm{rad}]. \end{cases}$$

In this configuration, matrix $\mathbf{J}_{\mathrm{img}}$ is very close to singularity and its inversion would generate very large velocity commands for the robot. However, by splitting the whole task as in (2.12) with $\mathbf{r}_1 = p_1$ and $\mathbf{r}_2 = p_2$, the two 2×4 sub-Jacobians $\mathbf{J}_{\mathrm{img}_1}$ and $\mathbf{J}_{\mathrm{img}_2}$ relative to the individual feature points are well conditioned. Indeed, we have $\sigma(\mathbf{J}_{\mathrm{img}}) \sim 0.15$, $\sigma(\mathbf{J}_{\mathrm{img}_1}) \sim 227$ and $\sigma(\mathbf{J}_{\mathrm{img}_2}) \sim 246$. Therefore, we proceed as in Chapter 4, and adopt the TP method for the realization of task (6.2).

Figures 6.4(a–b), 6.5(a–b) and 6.6 show the results of the servoing realized with the TP control law (2.12) where we set $\mathbf{K}_1 = 2\,\mathbf{I}_2$, $\mathbf{K}_2 = 0.001\,\mathbf{I}_2$, $\mathbf{f}_d = [-0.3889\ \ 0.0801\ \ -0.1716\ \ 0.0915]^T$. We also used $v_{max} = 0.09$ [m/s], $\omega_{max} = \dot{q}_{1_{max}} = \dot{q}_{2_{max}} = 0.14$ [rad/s] as maximum allowed velocity commands. The task $\mathbf{f}$ is correctly executed (Fig. 6.4(a)), with simultaneous motion commanded to the platform (Fig. 6.5(a)) and the pan-tilt unit (Fig. 6.5(b)), despite the initial ill-conditioning of $\mathbf{J}_{\mathrm{img}}$. Figure 6.6 reports the time behavior of $\sigma(\mathbf{J}_{\mathrm{img}})$, $\sigma(\mathbf{J}_{\mathrm{img}_1})$ and $\sigma(\mathbf{J}_{\mathrm{img}_2})$: matrix $\mathbf{J}_{\mathrm{img}}$ remains always close to singularity, while $\mathbf{J}_{\mathrm{img}_1}$ and $\mathbf{J}_{\mathrm{img}_2}$ are well conditioned during the whole motion (note the different scales). This confirms the conclusions drawn in Sect. 4.6.2, i.e., that a direct inversion of $\mathbf{J}_{\mathrm{img}}$ would

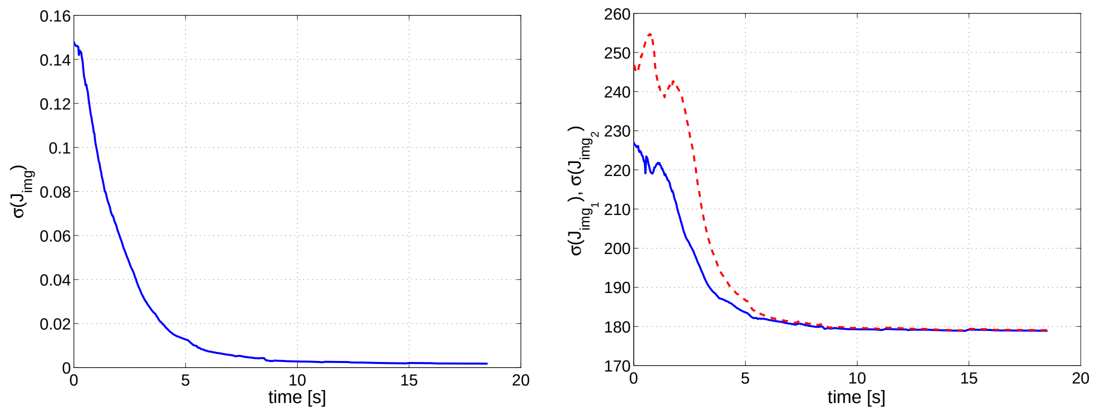


Figure 6.6: **TP experiment**. Singularity analysis during the visual task. Left: time evolution of $\sigma(\mathbf{J}_{\text{img}})$. Right: time evolution of $\sigma(\mathbf{J}_{\text{img}_1})$ (solid blue line) and $\sigma(\mathbf{J}_{\text{img}_2})$ (dashed red line).

not have provided good results in terms of execution performance, while the TP strategy is able to fulfill the given task.

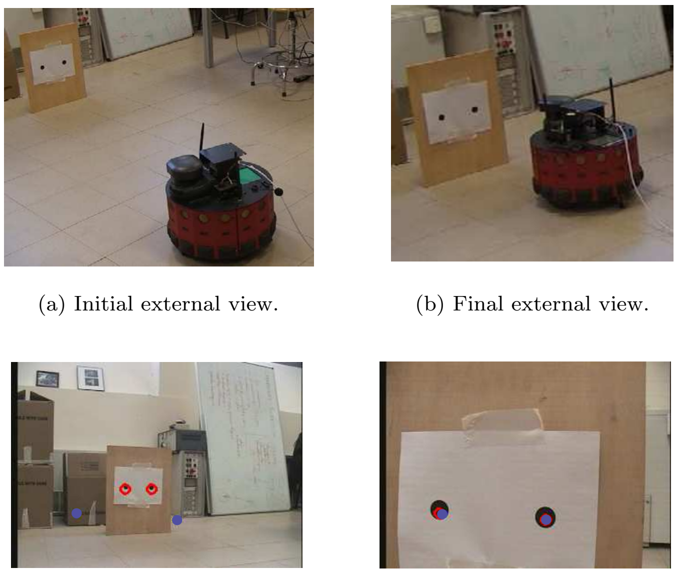
Furthermore, Fig. 6.4(b) shows the behavior of the two feature depths estimated via the observer (5.5), initialized with $\widehat{Z}_1(t_0) = \widehat{Z}_2(t_0) = 0.3$ [m]. There is a good convergence towards the final real depth values represented by two dashed horizontal lines. Finally note that the task is executed in about 18 [s] with a backward motion and a small clockwise rotation of the platform. The slightly erratic commands are a consequence of the slow sampling rate of the control architecture (30 [Hz]) and the presence of unmodeled disturbances due to the gaps between the floor tiles.

6.1.2 Experiments with Task Sequencing

Still in the spirit of reproducing, in an experimental setting, the numerical results of Sect. 4.6.2, we tested the TS method by decomposing the visual task (6.2) in two phases, being $\mathbf{f}_1 = p_1$ the variables regulated in the first phase and $\mathbf{f}_2 = p_2$ in the second phase — see (2.17). The two 'artificial' degrees of redundancy were exploited so as to keep the target as much as possible in front of the robot. This was again obtained by minimizing the cost function $H(\mathbf{q}) = \frac{1}{2}q_1^2$, i.e., the pan angle distance from the platform forward direction. The first phase switching time T_1 is set according to the value of q_1, so that the switch occurs when $|q_1(t)| < 0.1$ [rad][2].

Initial and final conditions of the robot are shown in Figs. 6.7(a–d). As can be seen from Fig. 6.7(a), the initial pan angle was intentionally set at about -90 [deg] (with the platform counter-rotated w.r.t. the target by a similar amount) in order to fully appreciate the null-space

[2]A termination condition on e_1 could also be added.



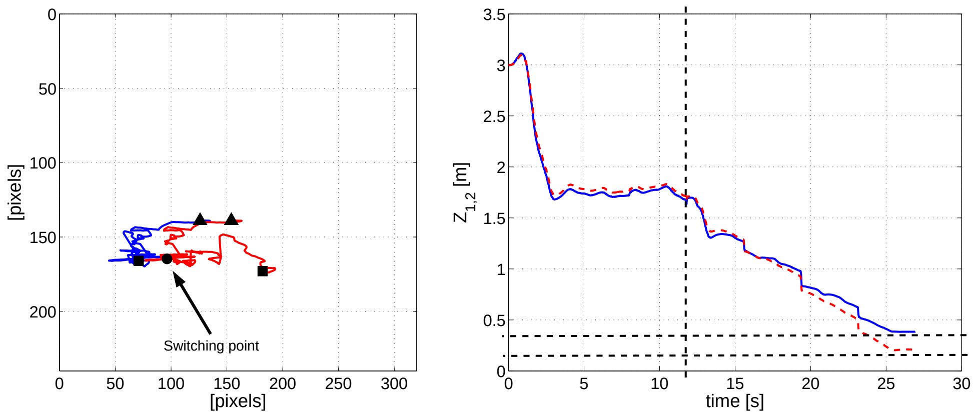
(a) Initial external view.

(b) Final external view.

(c) Initial camera view.

(d) Final desired camera view.

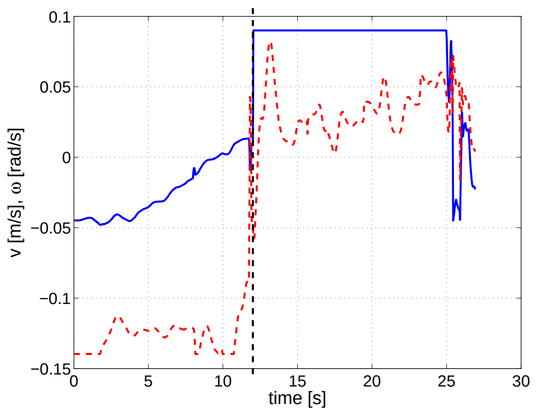
Figure 6.7: **TS experiment**. Initial and final robot views for the TS experiment.



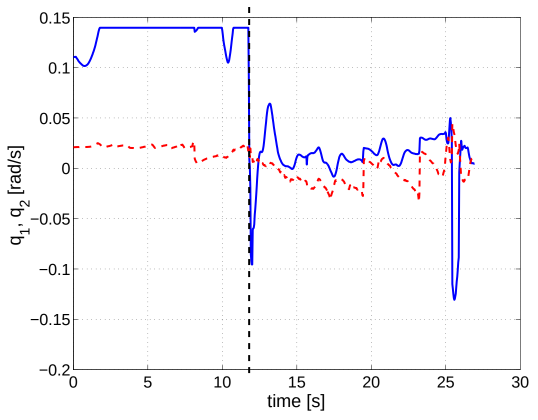
(a)

(b)

Figure 6.8: **TS experiment**. Left: motion, from ▲ to ■, of the two feature points on the image plane (p_1 is the blue solid line and p_2 the red dashed line). The black circle ● indicates the switching point between the two phases. Right: behavior of the estimated depths $\widehat{Z}_1(t)$ and $\widehat{Z}_2(t)$ over time. The dashed horizontal lines represent the final ground truth values of the depths.



(a)



(b)

Figure 6.9: **TS experiment**. Left: platform linear velocity v (solid blue line) and angular velocity ω (dashed red line). Right: pan velocity $\dot{q}_1$ (solid blue line) and tilt velocity $\dot{q}_2$ (dashed red line).

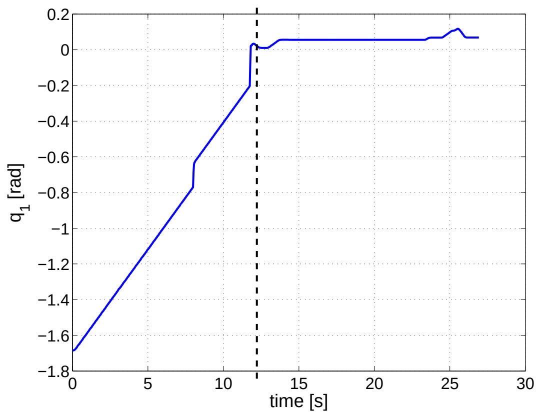


Figure 6.10: **TS experiment**. Behavior of $q_1(t)$ over time. The vertical dashed line indicates the switching point.

motion during the first phase. The initial conditions of the experiment are

$$\begin{cases} \mathbf{f}(t_0) & = [-0.1944\ 0.1087\ \ -0.0343\ 0.1087]^T \quad [\text{m}] \\ Z_1(t_0) & = Z_2(t_0) = 1.7 \quad [\text{m}] \\ q_1(t_0) & = -1.68 \quad [\text{rad}] \\ q_2(t_0) & = 0 \quad [\text{rad}], \end{cases}$$

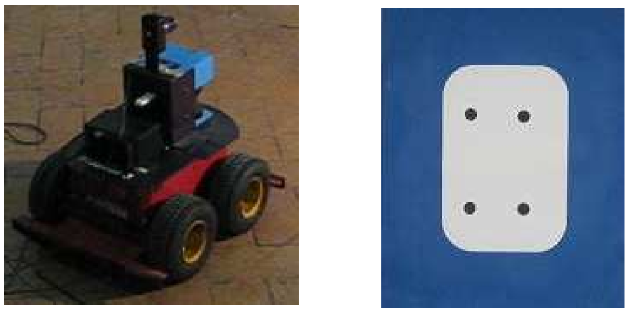
and we set in (2.17) $\mathbf{K}_1 = 1.5\,\mathbf{I}_2$, $\mathbf{K}_2 = 0.001\,\mathbf{I}_2$, $\alpha = 1.5$, and $\mathbf{f}_d = [-0.509\ 0.2631\ 0.1258\ 0.3031]^T$. The maximum command values were chosen as in the previous case. Results of the experiment are shown in Figs. 6.8(a–b), 6.9(a–b) and 6.10. During the first phase, angle q_1 is brought back to zero (Fig. 6.10) and feature point p_1 is kept close to its desired value, but no direct control is applied to the motion of p_2 (Fig. 6.8(a)). After the switching instant (represented in the plots by a dashed vertical line), the desired final position of feature point p_2 is recovered while keeping p_1 fixed to its reached desired location. Figures 6.9(a–b) show the velocity commands sent to the robot during the task execution. It is evident there the discontinuity due to the switch between the two phases. In particular, during the first phase the platform moves backwards and rotates clockwise in order to compensate for the counterclockwise pan motion; in the second phase, the platform mainly moves forward while slightly rotating to keep feature p_1 fixed on the image plane. Finally, in Fig. 6.8(b) the behavior of the estimated depths is reported. The observer was initialized with $\widehat{Z}_1(t_0) = \widehat{Z}_2(t_0) = 3$ [m] with an error of about 1.3 [m] from the actual initial depths (measured independently and not used in the experiment). Despite this rough approximation, the observer is able to recover the true values of $Z_{1,2}$ in about 3 [s] of motion (the fast initial transient on Fig. 6.8(b)), and yields, at the end of the task, depth estimates close to the real final depth values (the dashed horizontal lines).

6.2 Experiments of 3D structure observation

Having discussed the advantages of exploiting redundancy, in this section we focus on the benefits that arise from the coupling of IBVS feedback and 3D observation. Indeed, as discussed in Chapter 4, the local nature of most IBVS schemes can lead to instabilities that prevent task fulfillment. In this respect, our proposal is to replace $\boldsymbol{\chi}(t)$ with an estimate $\widehat{\boldsymbol{\chi}}(t)$ obtained from the output of observer (5.5) and its extensions. Note that, while in the linear domain the *separation principle* [Friedland 1986] would guarantee global stability of the coupling servoing/observer, when considering nonlinear systems this property is lost in general and convergence can be proved only locally, e.g., if the observer initial conditions are close enough to the true state values. Obtaining an analytical characterization of the actual stability region w.r.t. initial task/observer errors/states is a difficult problem due to the high nonlinearities present in the system dynamics and is currently object of ongoing research. Promisingly, the experiments reported hereafter show a good tolerance of the combined servoing/observer system w.r.t. observer initial errors, camera noise, and calibration uncertainties[3] [De Luca *et al.* 2008b]. Note that, choice $\boldsymbol{\chi}(t) = \widehat{\boldsymbol{\chi}}(t)$ instead of $\boldsymbol{\chi}(t) \equiv \boldsymbol{\chi}_d$ has at least two general advantages:

1. if $\boldsymbol{\chi}_d$ is available, IBVS stability domain can be enlarged by initializing the observer with $\widehat{\boldsymbol{\chi}}(t_0) = \boldsymbol{\chi}_d$, and by obtaining, in turn, a better approximation of $\widehat{\mathbf{J}}_{\text{img}}$ thanks to the observer convergence during the robot motion;

[3]Video clips of these experiments can be found at `www.dis.uniroma1.it/~labrob/research/depth_IBVS.html`.



(a) (b)

Figure 6.11: Robot and target object.

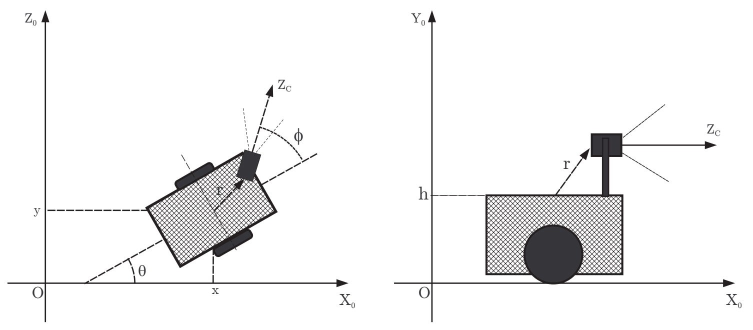
2. if $\boldsymbol{\chi}_d$ is not known, it is still possible to initialize the observer with a generic value, and then use $\widehat{\boldsymbol{\chi}}(t)$ to fulfil the task.

Last point can be particularly relevant in the case of navigation/exploration tasks for mobile robots equipped with cameras, whenever a set of locations is specified in terms of images acquired during the motion without the possibility to store at the same time the corresponding 3D information.

The following experiments have been realized on a unicycle-like robot equipped with a fixed camera mounted on its top (Fig. 6.11(a))[4]. This design can be seen as a particular NMM with $\mathbf{q}_m = \emptyset$, so that the modeling framework developed in Chapters 2–4 can be exploited effortlessly. As for the target to be tracked, we chose a vertical planar object with 4 black dots placed at the vertexes of a rectangular shape, see Fig. 6.11(b).

In order to obtain the analytic expression of matrix $\mathbf{J}_M$, let $\mathbf{q} = [x \ y \ \theta]^T \in \mathbb{R}^3$ be the platform configuration vector, $\mathbf{u} = [v \ \omega]^T \in \mathbb{R}^p$, $p = 2$, be the linear and angular platform velocities, vector $\mathbf{r} = [r_x \ r_y \ r_z]^T$ be the relative displacement between the unicycle reference point and O_C (the camera optical center), and ϕ be the angle between camera $\mathbf{Z}_C$ axis and platform main axis ($\mathbf{Z}_C$ lies on the horizontal plane, see Figs. 6.12(a–b)). With this notation,

[4]We wish to thank Prof. Prattichizzo and his team from the University of Siena for their kind support which made us possible to run these experiments.



(a) Top view. (b) Side view.

Figure 6.12: Definition of robot quantities.

we can obtain the following 6×2 matrix $\mathbf{J}_M$

$$\mathbf{J}_M = \begin{bmatrix} \sin\phi & -r_x\cos\phi - r_y\sin\phi \\ 0 & 0 \\ \cos\phi & r_x\sin\phi - r_y\cos\phi \\ 0 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix}$$

mapping the robot inputs $[v \;\; \omega]^T$ to the camera linear/angular velocity vector $[\mathbf{v}_C \;\; \boldsymbol{\omega}_C]^T$ expressed in the camera frame. The geometric data of the robot are:

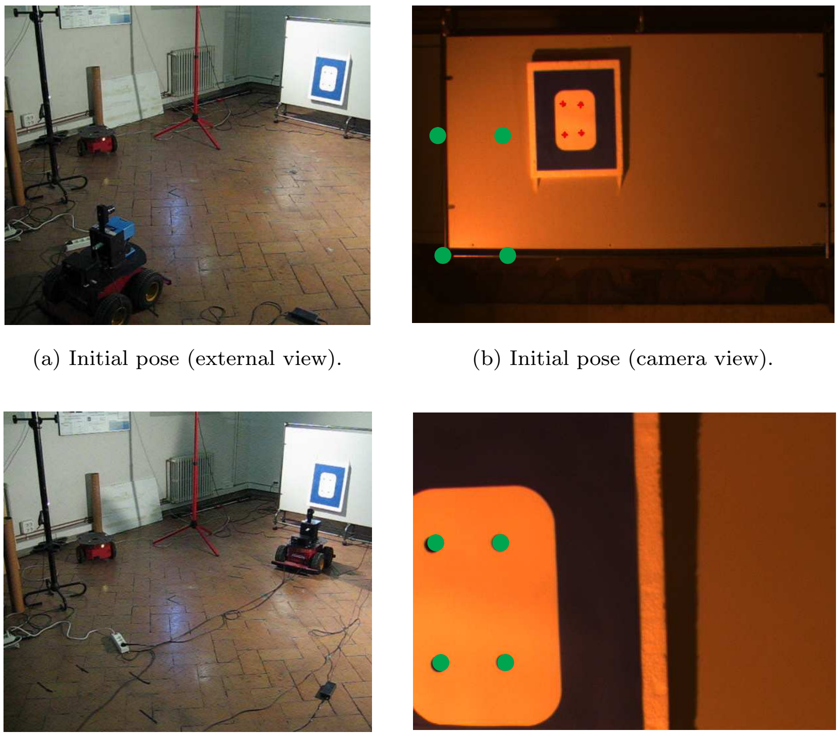
$$\begin{aligned} r &= [0.07\, 0.02\, 0.13]^T \text{ [m]} \\ \phi &= 0 \text{ [rad]}. \end{aligned}$$

6.2.1 Point features

In this first set of $g = 4$ experiments, we considered regulation of the $k = 4$ point features on the target object, see Fig. 6.13(b)–(d). The same servoing task (same initial and final robot poses) was run by considering different initial values for observer (5.5) with the aim of assessing the robustness and reliability of the integrated approach. In the following, we will use superscript index $j = 1 \dots g$ to denote the j-th experiment, and subscript index $i = 1 \dots k$ to denote the i-th point feature.

For this visual task we have $\mathbf{f} = [p_1^T \;\dots p_k^T]^T \in \mathbb{R}^s$, $s = 2k = 8$, matrix $\mathbf{J}_f(\mathbf{f},\, \boldsymbol{\chi}(t)) \in \mathbb{R}^{2k\times 6}$ is made of the stack of k point feature Jacobian $\mathbf{J}_{p_i}(p_i,\, Z_i(t))$ as in (4.7), and $\boldsymbol{\chi} = [Z_1 \dots Z_k]^T \in \mathbb{R}^k$. Since, in this case, no redundancy is present $(s > p)$, we use the simple feedback law

$$\mathbf{u} = \widehat{\mathbf{J}}^{\dagger}_{\text{img}}(\mathbf{f},\, \widehat{\boldsymbol{\chi}}(t))\mathbf{K}(\mathbf{f}_d - \mathbf{f}), \quad \mathbf{K} > 0, \tag{6.4}$$


(a) Initial pose (external view). (b) Initial pose (camera view).

(c) Final pose (external view). (d) Final pose (camera view).

Figure 6.13: **Point feature experiment**. External and camera views: the green dots represent the desired positions of the 4 points features.

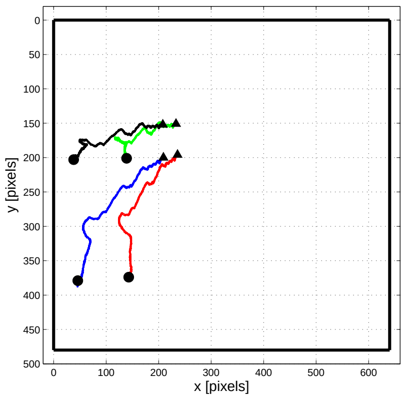
conceptually equivalent to (4.14) with inclusion of the robot kinematics $\mathbf{J}_M$. The initial pose of the robot is such that for each point feature i, $Z_i(t_0) \simeq 3.9$ [m], while at the desired pose $Z_{d_i} \simeq 0.98$ [m] and $\mathbf{f}_d = [-0.2509 \;\; 0.2123 \;\; -0.1594 \;\; 0.2075 \;\; -0.1632 \;\; 0.0443 \;\; -0.2585 \;\; 0.0462]^T$. Note that, in this case,

$$\widehat{\boldsymbol{\chi}}(t) = \left[\widehat{Z}_1 \dots \widehat{Z}_k\right]^T = \left[1/\widehat{\mathbf{x}}_{u_1} \dots 1/\widehat{\mathbf{x}}_{u_k}\right]^T,$$

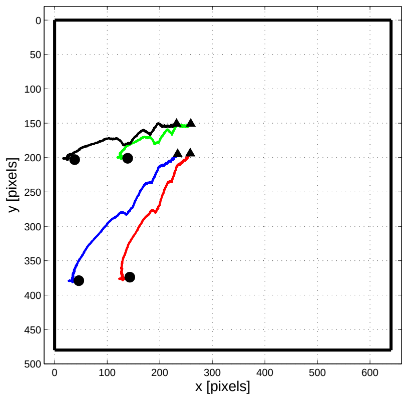
i.e., it can be directly computed as output of observer (5.5).

In each j-th experiment, the observer states are always initialized as $\widehat{\mathbf{x}}^j_{m_i}(t_0) = p^j_i(t_0)$, i.e., matching the measured initial feature positions, while, for the initial depth guesses, we considered 4 different values, one for experiment:

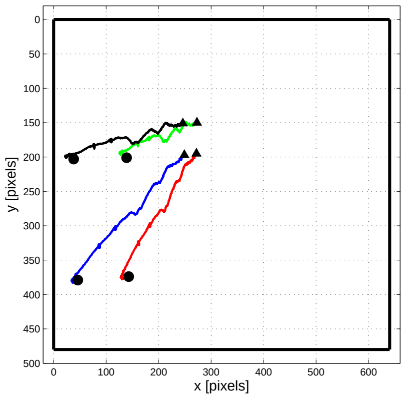
$$\begin{cases} \widehat{Z}^1_i(t_0) = 1/\widehat{\mathbf{x}}^1_u(t_0) & = & 1.6 \text{ [m]} \\ \widehat{Z}^2_i(t_0) = 1/\widehat{\mathbf{x}}^2_u(t_0) & = & 3 \text{ [m]} \\ \widehat{Z}^3_i(t_0) = 1/\widehat{\mathbf{x}}^3_u(t_0) & = & 5 \text{ [m]} \\ \widehat{Z}^4_i(t_0) = 1/\widehat{\mathbf{x}}^4_u(t_0) & = & 8 \text{ [m]}. \end{cases}$$



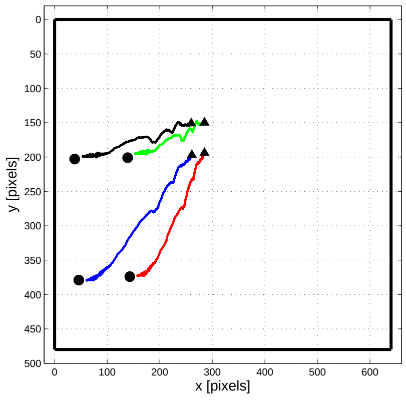
(a) $\widehat{Z}_i^1(t_0) = 1.6$ [m].



(b) $\widehat{Z}_i^2(t_0) = 3$ [m].



(c) $\widehat{Z}_i^3(t_0) = 5$ [m].



(d) $\widehat{Z}_i^4(t_0) = 8$ [m].

Figure 6.14: **Point feature experiment**. Feature trajectories during the four servoing experiments. The black thick box represents image plane boundaries.

Figures 6.14(a–d) show the feature motion on the image plane for each experiment where feedback/observer gains were chosen as $\mathbf{K} = 0.5\,\mathbf{I}_s$, $\mathbf{K}_1 = 48\,\mathbf{I}_2$ and $\mathbf{K}_2 = 44000$. The visual task is always completed in all the 4 cases, i.e., the feature points reach their desired positions without crossing the image plane boundaries (the black thick box in the plots). This is a direct consequence of the convergence properties of the observer which compensates for the initial wrong depth guesses and approximates the true $Z_i(t)$ values during the motion.

Convergence to the true depth values can also be checked on Fig. 6.15 which depicts the behavior of the estimated depths for each point feature i and each experiment j. Note that, despite the different initial values used in the experiments, at the end of the motion every $\widehat{Z}_i^j(t)$ approaches the final value $Z_{d_i} \simeq 0.96$ [m], i.e., the true depth at the desired pose (represented in the plot by a blue dashed horizontal line).

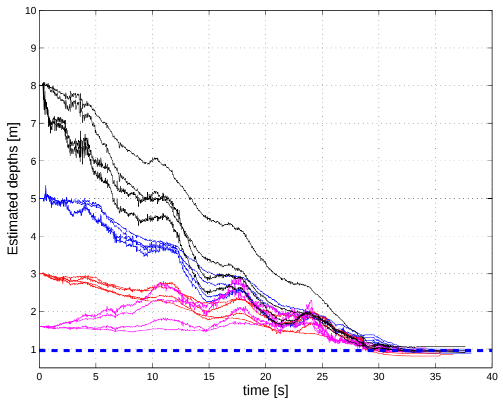


Figure 6.15: **Point feature experiment**. Evolution of the four estimated depths during the four servoing experiments. The blue horizontal line represents the (common) depth $Z_{d_i} = 0.96$ [m] at the desired pose.

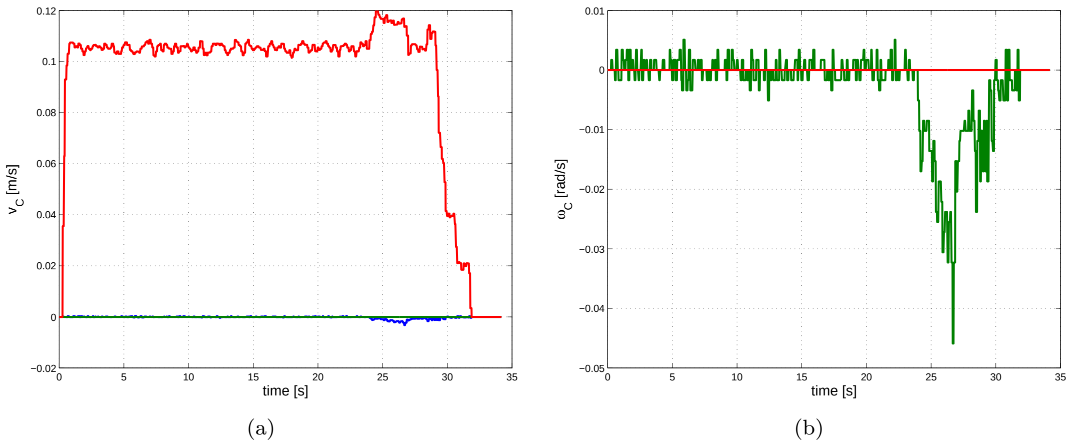
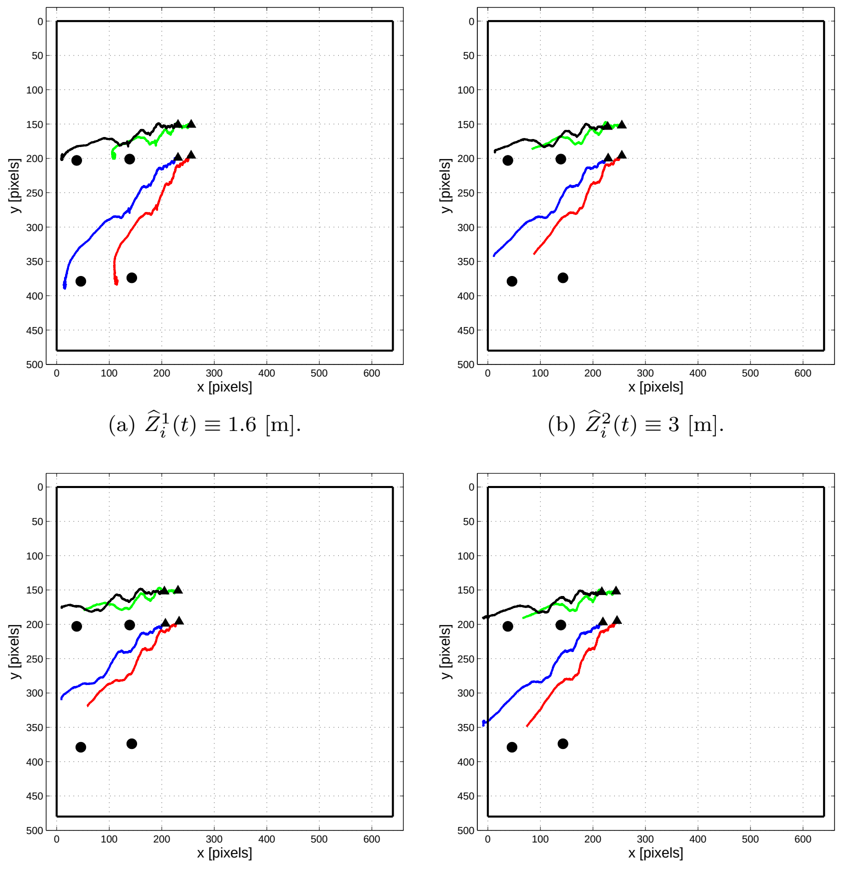


Figure 6.16: **Point feature experiment**. Camera linear velocity $\mathbf{v}_C$ (left) and camera angular velocity $\boldsymbol{\omega}_C$ (right) during the fourth experiment. The main components are v_{C_z} and ω_{C_y}, respectively.

It is also worth noting that the initial and final robot poses are such that, in each experiment, the camera linear motion is mainly along the $\mathbf{Z}_C$ axis, implying a continuously time-varying behavior for $Z_i(t)$. This can be verified in Fig. 6.16 where $\mathbf{v}_C$ and $\boldsymbol{\omega}_C$ relative to the fourth experiment are shown. Noisiness of $\mathbf{v}_C$ and $\boldsymbol{\omega}_C$ is a direct consequence of the numerical differentiation step needed to obtain the actual robot velocities from discrete sampling of wheel encoders. Hence, as claimed in Sect. 5.2.1 and numerically tested in Sect. 5.4.1, observer (5.5) confirms the ability to cope with a freely time-varying feature depth, and shows also good



(a) $\widehat{Z}_i^1(t) \equiv 1.6$ [m].

(b) $\widehat{Z}_i^2(t) \equiv 3$ [m].

(c) $\widehat{Z}_i^3(t) \equiv 5$ [m].

(d) $\widehat{Z}_i^4(t) \equiv 8$ [m].

Figure 6.17: **Point feature experiment**. Feature trajectories during the four servoing experiments without using the observer. The black thick box represents image plane boundaries.

robustness against noise.

Finally, in Figs. 6.17(a–d) we show the feature motion on the image plane when considering a constant depth during the servoing, i.e., without using the observer but just relying on prior knowledge of the scene. In order to make a comparison, we ran the same 4 experiments as before with $\widehat{Z}_i^j(t) \equiv \widehat{Z}_i^j(t_0)$, i.e., using the previous initial depth guesses as constant values during the motion. As a result, all the experiments failed in the sense that the servoing scheme was not able to fulfill the task while keeping the feature points inside the image plane boundaries. This, of course, was a consequence of the too rough approximation used in $\widehat{\mathbf{J}}_{\text{img}}$ due to wrong depth information.

6.2.2 Image moments

As an additional case study, we tested the integrated approach by considering a more sophisticated feature set, i.e., image moments instead of individual feature point coordinates — see Sect. 4.3.1. In particular, we relied upon discrete moments defined by the 4 points present in the target object used in the previous experiments (Fig. 6.11(b)). In order to control the robot motion, we chose a feature set made of $s = 3$ moments, namely

$$\mathbf{r} = \left[x_g \; y_g \; 1/\sqrt{a} \right]^T, \tag{6.5}$$

where $a = \mu_{20} + \mu_{02}$ is a measure of the area enclosed by the selected points [Tahri & Chaumette 2005]. As in the previous case, an approximation of $\boldsymbol{\chi}(t)$ is required for the actual computation of the moment interaction matrix, and a standard choice is $\boldsymbol{\chi}(t) \equiv \boldsymbol{\chi}_d$. However, one can again exploit observer (5.5) to get an indirect estimate $\widehat{\boldsymbol{\chi}}(t)$ during the servoing task. Indeed, rearranging (4.10) and considering the k selected points, we obtain the linear system

$$\begin{bmatrix} p_{u_1} & p_{v_1} & 1 \\ & \vdots & \\ p_{u_k} & p_{v_k} & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \dfrac{1}{Z_1} \\ \vdots \\ \dfrac{1}{Z_k} \end{bmatrix}$$

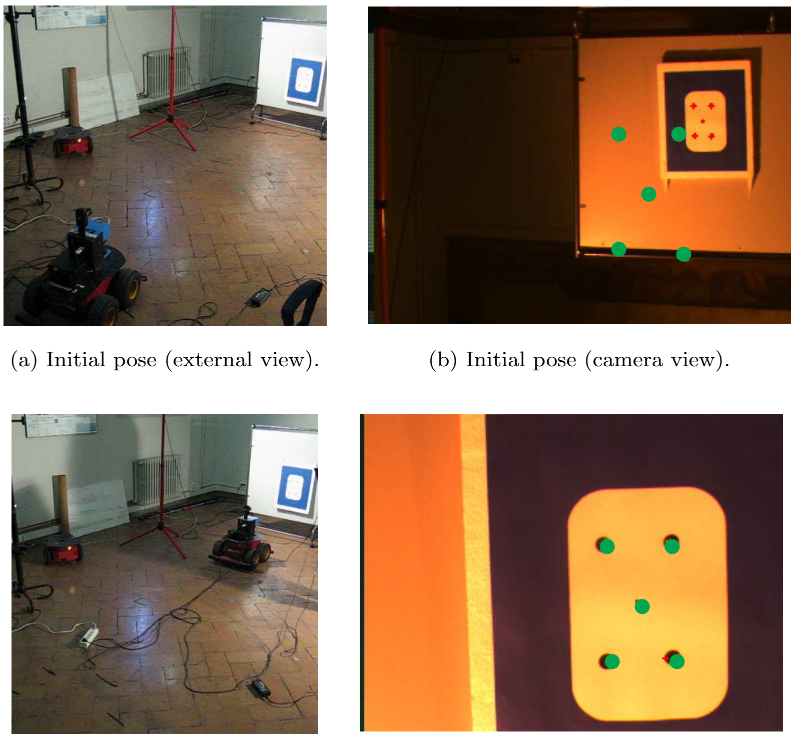
which can be easily solved as

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} p_{u_1} & p_{v_1} & 1 \\ & \vdots & \\ p_{u_k} & p_{v_k} & 1 \end{bmatrix}^{\dagger} \begin{bmatrix} \dfrac{1}{Z_1} \\ \vdots \\ \dfrac{1}{Z_k} \end{bmatrix} = \boldsymbol{\Phi}^{\dagger} \begin{bmatrix} \dfrac{1}{Z_1} \\ \vdots \\ \dfrac{1}{Z_k} \end{bmatrix}, \tag{6.6}$$

i.e., computing the 'best' plane passing through the given points. Since matrix $\boldsymbol{\Phi}$ is made of quantities directly measured on the image plane, an estimate of (A, B, C) is possible by replacing the true depths Z_i in (6.6) with the estimated $\widehat{Z}_i$ obtained from the observer. Therefore, in the following we will have

$$\widehat{\boldsymbol{\chi}}(t) = \begin{bmatrix} \widehat{A} \\ \widehat{B} \\ \widehat{C} \end{bmatrix} = \boldsymbol{\Phi}^{\dagger} \begin{bmatrix} \dfrac{1}{\widehat{Z}_1} \\ \vdots \\ \dfrac{1}{\widehat{Z}_k} \end{bmatrix}. \tag{6.7}$$

It is important to point out that the case of moments computed from a discrete set of coplanar points allows an easy observation of (A, B, C) also thanks to the special structure considered, i.e., the possibility to track and match the very same 3D points during the camera motion. Whenever such 'individual' tracking is not feasible or even possible, (6.7) cannot be directly used and the estimation techniques of Sect. 5.2.2 must be adopted to obtain $(\widehat{A}, \widehat{B}, \widehat{C})$.



(a) Initial pose (external view).

(b) Initial pose (camera view).

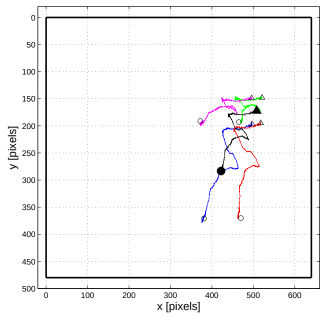
(c) Final pose (external view).

(d) Final pose (camera view).

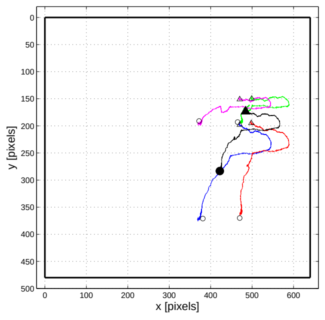
Figure 6.18: **Image moments experiment**. External and camera views. The green dots represent the desired positions of point features and relative barycenter (x_g, y_g).

Following the structure of the the previous section, we tested feedback (6.4) with $\boldsymbol{\chi}(t) = \widehat{\boldsymbol{\chi}}(t)$ in $g = 5$ different experiments, by keeping the same initial and final robot poses and varying the initial observer states. Figures 6.18(a–d) show the initial and final robot poses in terms of external and camera views. Note that in Fig. 6.18(b) and Fig. 6.18(d) the central dot is not a physical point, but it represents the computed barycenter of the real 4 point features. The initial pose of the robot is such that, for each point feature i, $Z_i(t_0) \simeq 4.1$ [m], while in the final pose we have $Z_{d_i} \simeq 0.9$ [m], $x_{g_d} = 0.1038$, $y_{g_d} = 0.1219$, and $a_d = 4.6217 \cdot 10^{-6}$.

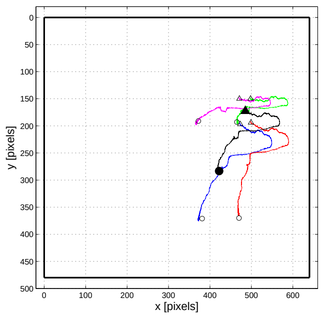
In each j-th experiment, the first two observer states were again initialized with the current



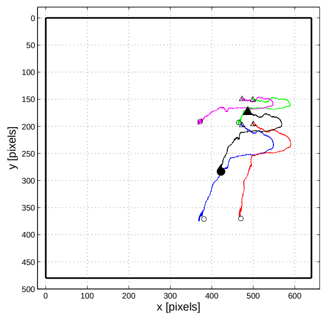
(a) $\widehat{Z}_i^1(t_0) = 0.9$ [m].



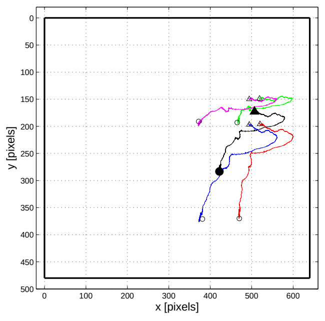
(b) $\widehat{Z}_i^2(t_0) = 3$ [m].



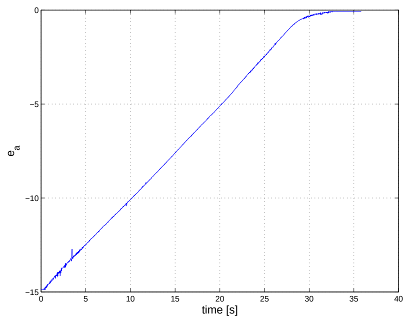
(c) $\widehat{Z}_i^3(t_0) = 5$ [m].



(d) $\widehat{Z}_i^4(t_0) = 8$ [m].



(e) $\widehat{Z}_i^5(t_0) = 16$ [m].



(f) $e_a(t)$

Figure 6.19: **Image moments experiment**. Feature trajectories during the 5 servoing experiments (Figs. (a–e)). The black thick box represents image plane boundaries. Figure (f) depicts the behavior of $e_a(t) = 1/\sqrt{a_d} - 1/\sqrt{a(t)}$ over time during the first experiment.

feature values measured on the image plane, while for the last state we chose:

$$\begin{cases} \widehat{Z}_i^1(t_0) = 1/\mathbf{x}_{u_i}^1(t_0) & = & 0.9 \text{ [m]} \\ \widehat{Z}_i^2(t_0) = 1/\mathbf{x}_{u_i}^2(t_0) & = & 3 \text{ [m]} \\ \widehat{Z}_i^3(t_0) = 1/\mathbf{x}_{u_i}^3(t_0) & = & 5 \text{ [m]} \\ \widehat{Z}_i^4(t_0) = 1/\mathbf{x}_{u_i}^4(t_0) & = & 8 \text{ [m]} \\ \widehat{Z}_i^5(t_0) = 1/\mathbf{x}_{u_i}^5(t_0) & = & 16 \text{ [m]} \end{cases} .$$

Figures 6.19(a–e) show the feature trajectories during the 5 experiments. White triangles/circles represent the initial/final positions of the k feature points, while a filled triangle/circle is used to denote the initial/final position of the barycenter (x_g, y_g), i.e., the feature over which we have ‘direct’ control (see (6.5)). In Fig. 6.19(f) we report, for experiment 1, the behavior of $e_a(t) = 1/\sqrt{a_d} - 1/\sqrt{a(t)}$, the task error relative to the third feature chosen for the servoing. The gains of feedback law and observer were set as in the previous section.

Again, we can verify that the integrated visual approach is able to fulfill the task despite the

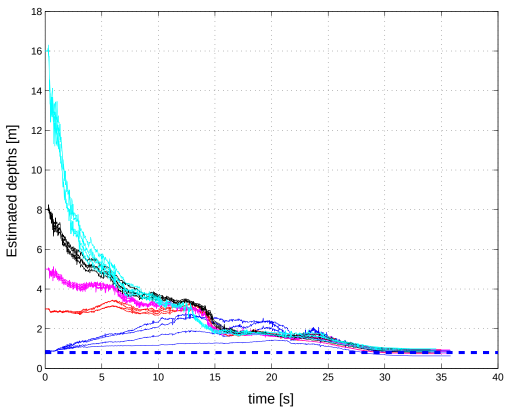
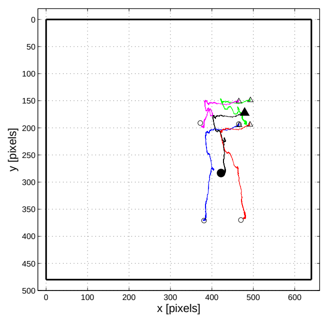


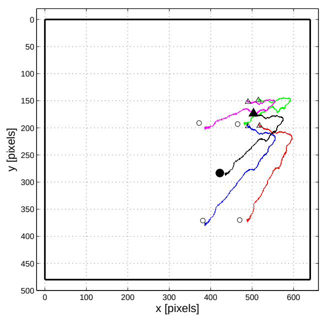
Figure 6.20: **Image moments experiment**. Evolution of the four estimated depths during the five servoing experiments. The blue horizontal line represents the common depth of the four feature points at the desired pose.

different initial depth guesses and the indirect observation of the plane parameters (A, B, C). Convergence to the true depth values can be checked in Fig. 6.20 where the behavior of the estimated depths for each point feature i and experiment j is shown. The blue dashed horizontal line in the plot represents the common depth value of the 4 feature points at the final robot pose.

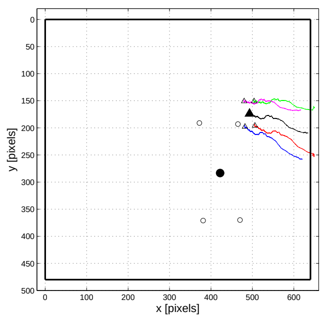
Finally, the same experiments were run by adopting a constant value for $\boldsymbol{\chi}(t)$, computed from (6.7) by setting $\widehat{Z}_i^j(t) \equiv \widehat{Z}_i^j(t_0)$. Results are shown in Figs. 6.21(a–e). It is interesting to note that in the first two cases (Figs. 6.21(a) and 6.21(b)) the servoing is still completed despite the approximation adopted in the feedback law, while in the other cases the features exit the image plane boundaries during the motion, causing failure of the task. Convergence of case (a), with constant depths $\widehat{Z}_i^1(t) \equiv Z_{d_i}$, i.e, set to the values relative to the final robot pose, is not surprising. Indeed, as discussed in the introduction, many previous work and experiments (e.g., [Chaumette 2004; Tahri & Chaumette 2005]) have shown that this setting allows the servoing fulfillment whenever the relative camera/target pose is inside the basin of stability of the servoing scheme (as it was in this case). On the other hand, convergence of case (b) shows a good tolerance of feedback (6.4) to depth/plane structure approximations. Indeed in this case $\widehat{Z}_i^2(t) \equiv 3$ [m], thus far from the true final depth values. As a comparison, in the previous section the same approximation led to the failure of the servoing task (Fig. 6.17(b)) and even a milder one yielded the same result (Fig. 6.17(a)). As for the remaining cases, Figs. 6.21(c–f) show that a rougher depth approximation prevents the convergence that was achieved with the integration of the depth observer (Figs. 6.19(c–f)).



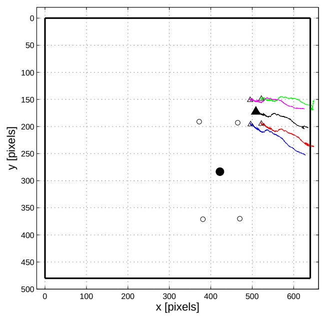
(a) $\widehat{Z}_i^1(t) \equiv 0.9$ [m].



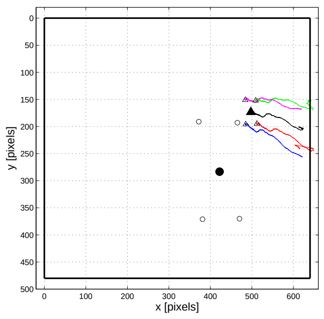
(b) $\widehat{Z}_i^2(t) \equiv 3$ [m].



(c) $\widehat{Z}_i^3(t) \equiv 5$ [m].



(d) $\widehat{Z}_i^4(t) \equiv 8$ [m].



(e) $\widehat{Z}_i^5(t) \equiv 16$ [m].

Figure 6.21: **Image moments experiment**. Feature trajectories during the 5 servoing experiments without using the observer. The black thick box represents image plane boundaries.

6.3 Experiments of focal length observation

In all the previous experiments, we have supposed full knowledge of the camera intrinsic parameters, i.e., of matrix $\mathbf{K}_C$ in (3.12), and in particular of the camera focal length λ. As discussed in Sect. 5.3, it is possible to design a suitable extension of the depth observer to preliminary estimate the value of λ independently from Z. Therefore, the purpose of this section is to provide an experimental validation to both the theoretical and simulation results presented in Sect. 5.3 and Sect 5.4.2.

With the robot chosen for our experiments, it is easy to check that a pure angular motion cannot be imposed to the camera. Indeed, for any platform angular velocity command ω, a resulting camera linear velocity will be always present due to the r_x and r_y components of the camera offset vector $\mathbf{r}$. However, the camera is not very far from the platform center, i.e., the values of r_x and r_y are small, and, as a consequence, the undesired camera linear velocity $\mathbf{v}_C$ can be considered negligible w.r.t. the imposed camera angular velocity $\boldsymbol{\omega}_C$. Therefore, we tested the algorithm by considering $\mathbf{v}_C$ as an additional external disturbance besides noise and

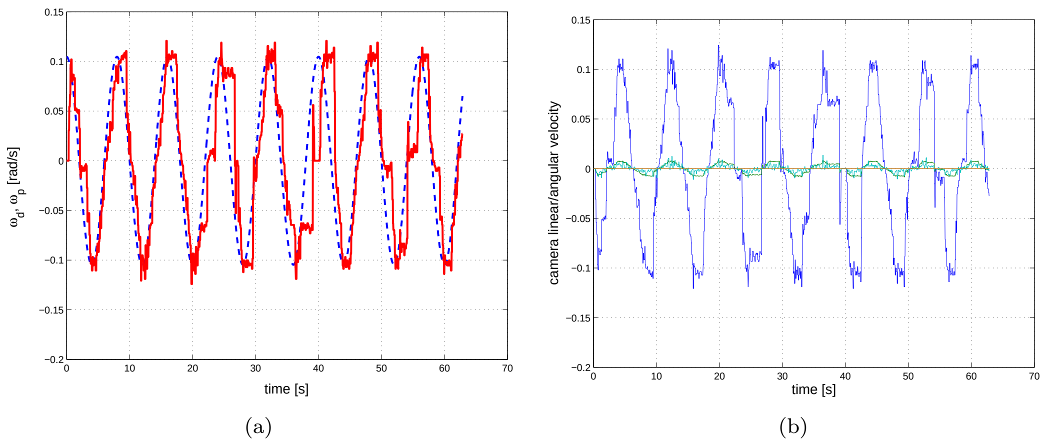


Figure 6.22: **Focal length observation experiment**. Left: ω_d (dashed blue line) and ω_p (solid red line) vs. time. Right: ω_{Cy} (solid blue line) and the other components of the camera linear/angular velocity vs. time.

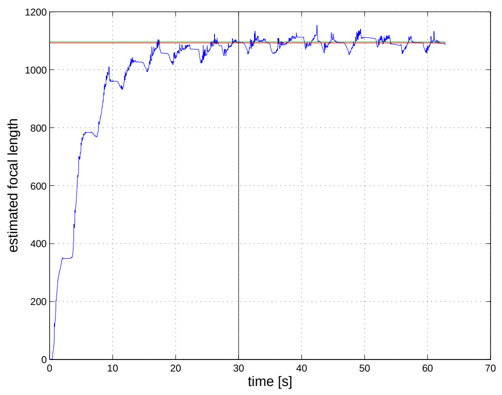


Figure 6.23: **Focal length observation experiment**. Evolution of $\widehat{x}_{u_1}(t) = \widehat{\lambda}(t)$ (solid blue line) vs. time. The estimate of λ reaches a steady state after about 30 sec of motion (vertical solid line).

modeling uncertainties. The feature point tracked during the experiment was the top-left black dot of the same planar target used in the previous experiments (see Fig. 6.11(b)).

The robot was commanded with the desired velocity profiles $v_d = 0$ [m/s], $\omega_d = 6\cos\pi/4t$ [deg/s]. Figure 6.22(a) shows the commanded ω_d and actual ω_p platform velocity during the experiment. Note that ω_p was obtained by numerical differentiation of the wheels encoder readings. In Fig. 6.22(b), the behavior of $\mathbf{v}_C$, $\boldsymbol{\omega}_C$ vs. time is reported. It is then possible to verify that the main camera angular motion (ω_{Cy} represented by the solid blue line) is the dominant component w.r.t. the undesired $\mathbf{v}_C$. The parameters of the focal length observation algorithm were set to:

$$\begin{array}{rcl}[\widehat{\mathbf{x}}_m^T(t_0)\;\widehat{\mathbf{x}}_u^T(t_0)]^T & = & [17.01\;32.99\;0\;0]^T \\ \mathbf{K}_1 & = & 30\,\mathbf{I}_2 \\ \mathbf{K}_2 & = & \left[\begin{array}{cc} 1500 & 0 \\ 0 & 0.01 \end{array}\right]\end{array}.$$

In order to have a reference value to compare our observation with, we also performed a standard off-line camera calibration by using the *MATLAB calibration toolbox* [Bouguet 2007]. The focal length value obtained at this preliminary stage was $\lambda_1 = 1096$ pixels.

Figure 6.23 shows the result of the observation process. The estimated focal length $\widehat{x}_{u_1}(t)$ (solid blue line) reaches a steady state after about $t = 30$ sec of motion (represented by a vertical line), despite of noise and other sources of disturbance. The average value of $\widehat{x}_{u_1}(t)$ after $t = 30$ [s] is $\lambda_2 = 1092.32$ pixels, thus very close to the value computed with the off-line technique. In Fig. 6.23, λ_1 and λ_2 are represented by the (almost coincident) horizontal green and red solid lines, respectively.

7 Application

In this Chapter, we present a potential industrial application which encompasses, among other aspects, the topics introduced in this Thesis, i.e., task-oriented kinematic modeling and VS feedback laws. In particular, we consider a combination of visual and force-torque feedback for automatic assembly of complex planar parts. A fixed-base robotic arm (the DLR light-weight robot) equipped with an onboard camera is committed with the task of looking for given parts on a table, picking them, and inserting them inside the corresponding holes on a target movable plate. In this context, we take advantage of VS techniques, and in particular of the HVS class, to achieve fine positioning of the robot manipulator during the approaching phase to the parts/holes.

Indeed, as seen in the previous Chapters, VS provides both robustness w.r.t. external disturbances and reactivity w.r.t. environmental changes, such as, for instance, the displacement of parts/plate during the task execution. Therefore, we chose to control the relative pose among robot end-effector and parts/holes by means of a suitable 6-dimensional task $\mathbf{r}$ made of visual and cartesian quantities, thus realizing a HVS scheme. The kinematic modeling and control framework for FBMs proposed in Chapters 1–2 is then exploited in conjunction with HVS formulation of Chapter 4 to design the needed feedback law. The resulting robot positioning accuracy, however, may not be high enough to accomplish tight assembly tasks as the ones

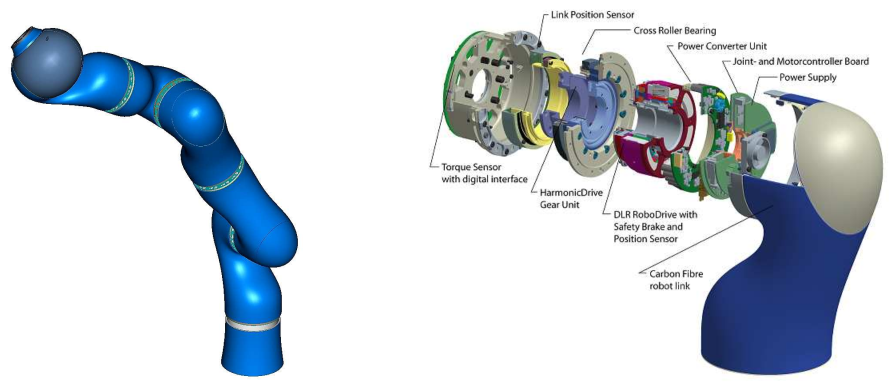


Figure 7.1: Left: CAD model of the 3-rd generation DLR light-weight robot. Right: detailed view of one joint. Courtesy of the German Aerospace Center (DLR).

considered for this application (clearance between parts and holes is less than 0.1 [mm]). Hence, force-torque sensors, which provide fast and high-resolution local information about the parts in contact, come into play during the final insertion phase, in a controlled compliance strategy able to successfully insert the parts despite possible positioning uncertainties of the visual loop.

The following sections are devoted to the relevant aspects of this application, namely, description of the experimental setup (robot, parts and plate), design of the HVS algorithm used for pose control, and relative experimental validation. Additional details on all the topics not covered here, such as description of the robust assembly strategy, can be found in [Robuffo Giordano *et al.* 2008] and references therein.

7.1 Experimental setup description

7.1.1 Robot manipulator

The FBM used for this application is the 3-rd generation DLR light-weight robot (LWR) — see Figs. 7.1(a–b). LWRs are kinematically redundant arms with seven dofs and a load to weight ratio of 1:1, and are designed for interaction with unstructured, everyday environments. Figure 7.2 shows a schematic view of the frames located at each joint according to the *Denavit-Hartenberg* (D-H) convention [Denavit & Hartenberg 1955]. Low weight and inherent joint compliance limit the interaction forces, even at high contact speed with the environment. Particularly relevant for our needs is the torque sensing integrated in each joint, allowing accurate, vibration free positioning and velocity control in the presence of elasticity and high performance impedance control during contact phases. The robot is able to switch within one

D-H Parameter LBR-KUKA
(Craig / Yoshikawa)

i	a_i[mm]	α_i [°]	d_i [mm]	θ_i [°]
0	0	0	0	0
1	0	90	310	0
2	0	-90	0	0
3	0	-90	400	0
4	0	90	0	0
5	0	90	390	0
6	0	-90	0	0
7	0	0	0	0

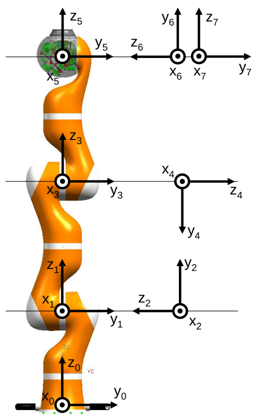


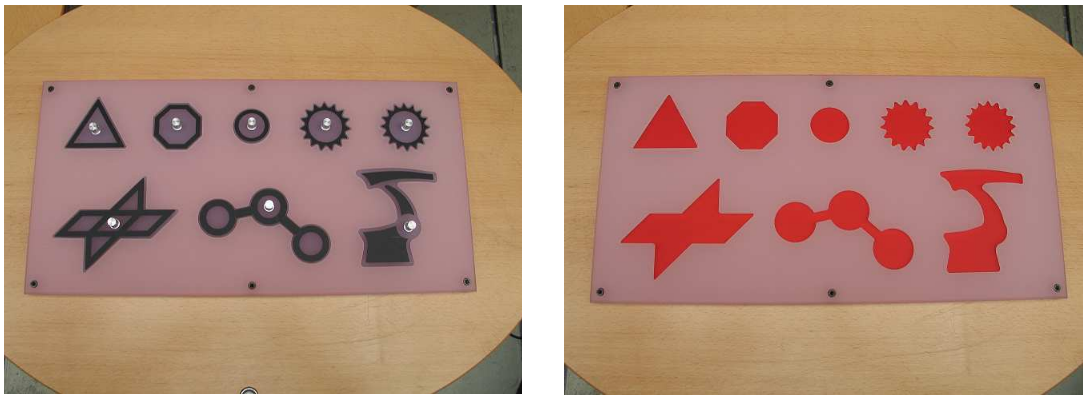
Figure 7.2: Schematic view of the seven joint frames chosen according to the *Denavit-Hartenberg* (D-H) convention. Courtesy of the German Aerospace Center (DLR).

control cycle (1 [ms]) between position/velocity control (required for high motion accuracy during the visual servoing phase) to impedance control (required for limiting the interaction forces and compensating the alignment errors during insertion).

7.1.2 Parts and plate

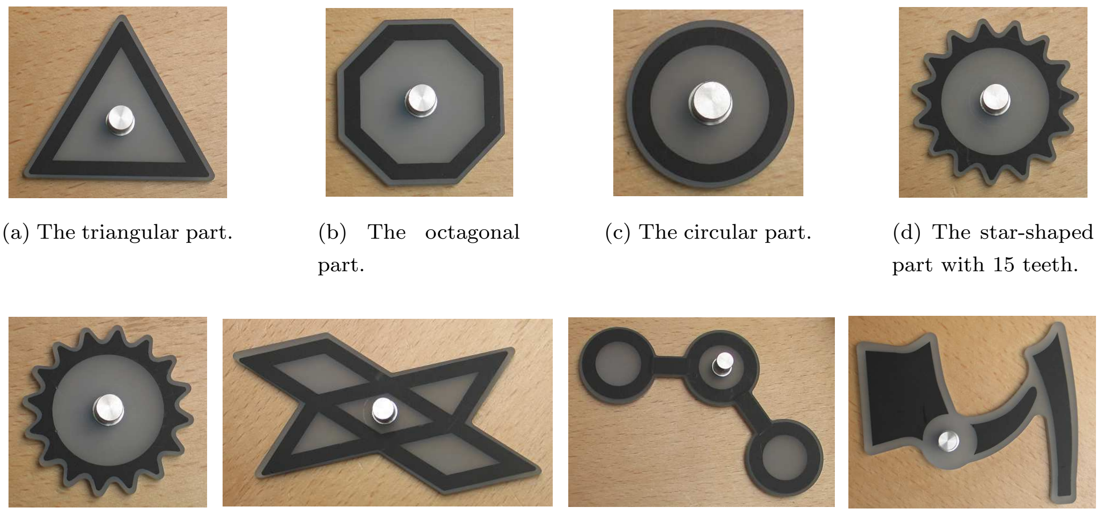
The eight parts and the plate used in our experiments are shown in Fig. 7.3(a–b) and Fig. 7.4(a–h). Each part has its shape marked with black tags useful for visual recognition, and a clearance of less than 0.1 [mm] w.r.t. the corresponding hole on the plate. Shapes of parts/holes range from simple geometrical primitives to more complex and nonconvex structures. Let $\mathcal{P} = \{p_1, \ldots, p_8\}$ and $\mathcal{H} = \{o_1, \ldots, o_8\}$ be the sets of parts and holes, respectively. With reference to Fig. 7.4, and starting from the upper left corner, we have the following shapes in order:

- o_1 (p_1): an equilateral triangle (Fig. 7.4(a));
- o_2 (p_2): a regular octagon (Fig. 7.4(b));
- o_3 (p_3): a circle (Fig. 7.4(c));
- o_4 (p_4): a star shape with 15 teeth (Fig. 7.4(d));
- o_5 (p_5): a star shape with 16 teeth (Fig. 7.4(e));
- o_6 (p_6): the DLR logo (Fig. 7.4(f));
- o_7 (p_7): the PAPAS logo (Fig. 7.4(g));
- o_8 (p_8): the KUKA logo (Fig. 7.4(h)).



(a) (b)

Figure 7.3: Left: plate filled with the eight planar parts used in our experiments. Right: empty plate with holes corresponding to the given parts.



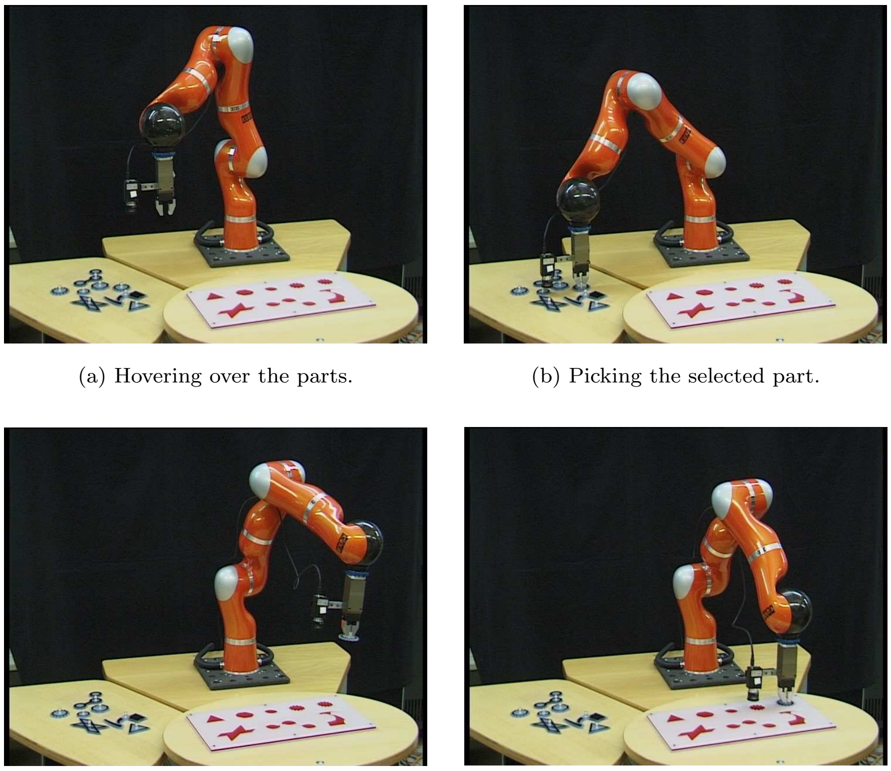
(a) The triangular part. (b) The octagonal part. (c) The circular part. (d) The star-shaped part with 15 teeth.

(e) The star-shaped part with 16 teeth. (f) The DLR part. (g) The PAPAS part. (h) The KUKA part.

Figure 7.4: Detailed pictures of the eight parts.

7.1.3 The overall task

As explained in the introduction, the global high-level task assigned to the manipulator is to locate, pick and insert each part into the corresponding hole on the plate (Figs. 7.5(a)–(d) illustrate a sample sequence). Obviously, such a complex task can hardly be tackled altogether



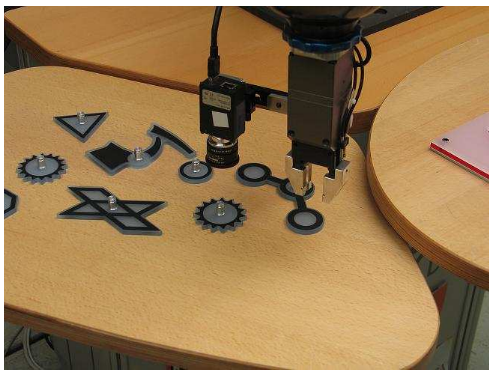
(a) Hovering over the parts. (b) Picking the selected part.

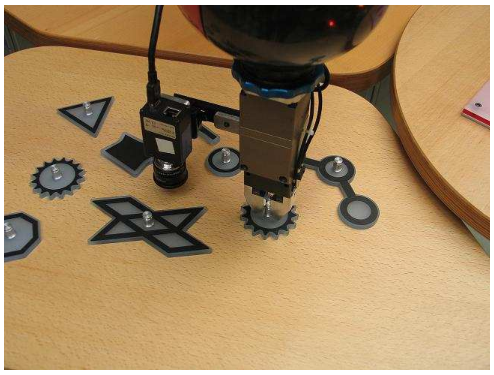
(c) Hovering over the plate. (d) Inserting the part.

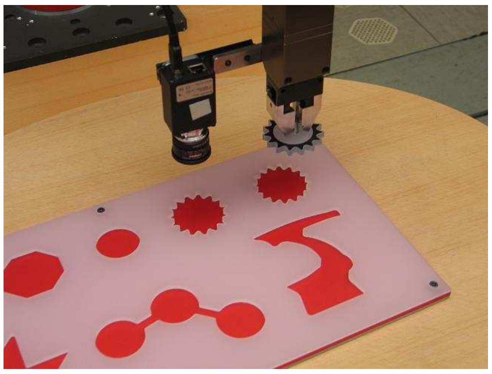
Figure 7.5: A typical sequence of operations. During the hovering motion (Figs. (a) and (c)) the visual system analyzes the scene and looks for a specific part/hole. As soon as the target shape is found, the HVS algorithm drives the robot towards the desired pose.

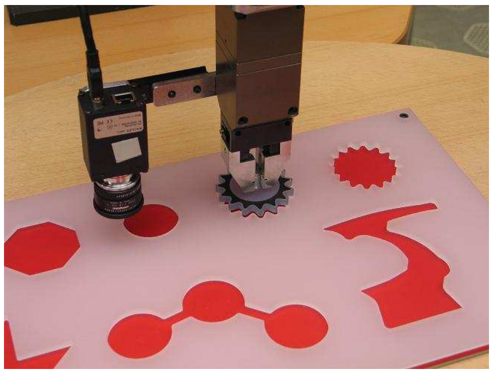
by a single control strategy, while a suitable subdivision can yield smaller and simpler subtasks to be fulfilled. A possible temporal decomposition is the following: for each part $p_i \in \mathcal{P}$ and corresponding hole $o_i \in \mathcal{H}$

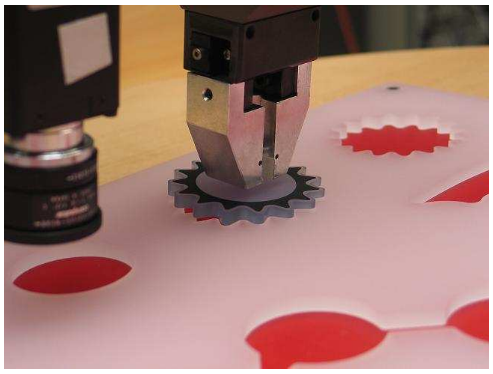
1. hover along a predefined trajectory until p_i is identified and located;
2. move to a suitable pose in order to pick p_i;
3. pick p_i;
4. hover over the plate along a predefined trajectory until o_i is identified and located;
5. move to a suitable pose in order to insert p_i;
6. insert p_i in o_i.


(a)


(b)


(c)


(d)


(e)

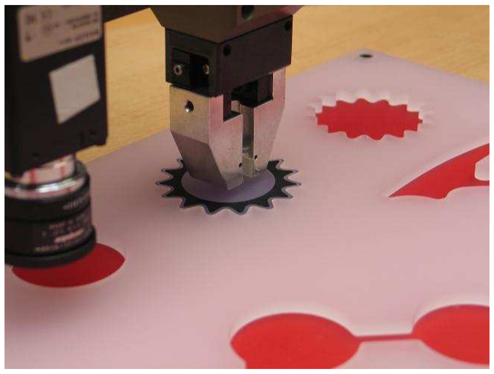

(f)

Figure 7.6: Detailed pictures of the picking/insertion phases. The HVS algorithm brings the camera over the selected part (step 2 — Fig. (a)), so that the robot can reliably pick it up (step 3 — Fig. (b)). During the insertion phase, the camera is positioned over the selected hole (step 5 — Fig. (c)) and then the assembly strategy fulfills the insertion (step 6 — Figs. (d–f)).

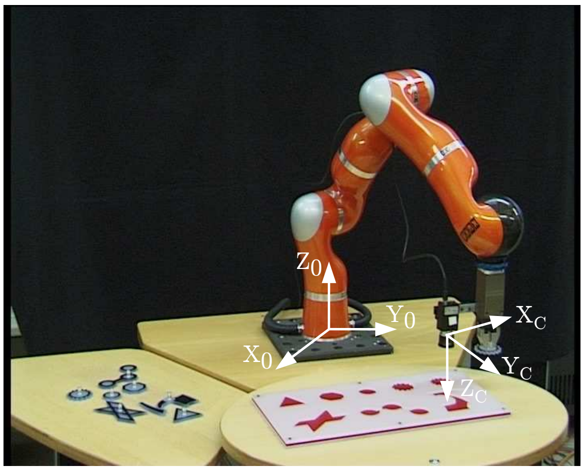


Figure 7.7: The world frame $\mathcal{F}_O : \{O;\, \mathbf{X}_O,\, \mathbf{Y}_O,\, \mathbf{Z}_O\}$ coincides with the manipulator base frame, and has axis $\mathbf{Z}_O$ pointing upwards. The moving frame $\mathcal{F}_C : \{O_C;\, \mathbf{X}_C,\, \mathbf{Y}_C,\, \mathbf{Z}_C\}$ is attached to the camera, with $\mathbf{Z}_C$ superimposed to the optical axis.

In particular, during steps 1 and 4, the vision system is committed with the on-line recognition of a target shape among the various parts/holes present in the images. The identification method adopted in our case relies on linear classification of *affine-invariant Fourier descriptors*, a set of parameters based on the *Fourier coefficients* extracted from the selected shapes. Details of this pattern classification methodology are reported in Appendix B.

Robot motion in steps 2 and 5 is governed by HVS techniques, while piece insertion in step 6 is realized through a robust force-torque controlled assembly strategy. Figures 7.6(a–f) show detailed pictures of the picking/insertion phases.

The next section illustrates the visual task definition and HVS feedback design used in the experiments.

7.2 Robot pose control

Goal of the pose control law is to position the manipulator close and precisely enough to the selected part/hole such that the part can be picked or the insertion strategy can be started. An example representative of both picking and insertion final poses is given in Fig. 7.7. The setup is arranged such that the (parallel) planes where parts and plate lie are fixed in $\mathcal{F}_O$ and perpendicular to $\mathbf{Z}_O$. Moreover, as shown in the figure, the final pose of the manipulator is always chosen such that $\mathbf{Z}_O = -\mathbf{Z}_C$, i.e., with the camera optical axis normal to the parts/plate plane and directed towards it.

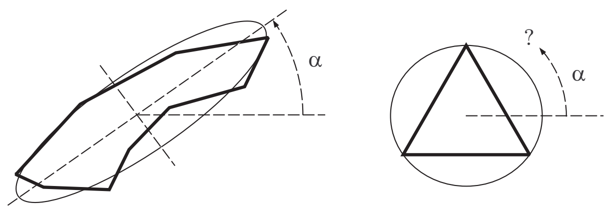


Figure 7.8: Main orientation α is well-defined for a generic shape (left), but is meaningless for a shape where minor and major axes are equal (right).

7.2.1 Visual task definition

Let $\mathbf{r} \in \mathbb{R}^s$, $s = 6$, be the visual task vector used for robot pose control. As discussed in Chapter 4, choice of which visual feature to include in $\mathbf{r}$ largely depends on the specific case one has to deal with, and can range from point coordinates, line parameters, ellipse centers and radii, etc. The complexity of the shapes considered in our experiments motivated us to adopt image moments as visual features for the servoing instead of individual points or other geometric primitives (see Sect. 4.3.1). In [Chaumette 2004], the use of moments for full pose control of a camera/robot is analyzed, showing that area a, barycenter (x_g, y_g) and main orientation α can be directly used to control four camera dofs, i.e., full translation and rotation about $\mathbf{Z}_C$. On the other hand, control of the remaining two dofs (the direction of $\mathbf{Z}_C$) is more involved, especially for symmetrical shapes, and the combined use of higher-order moments is typically required.

In our case, the circular symmetry of most shapes $(o_1/p_1 \ldots o_5/p_5)$ poses an additional difficulty to the control of the rotational dofs. Indeed, control of the direction of $\mathbf{Z}_C$ becomes even harder, and main orientation α loses its ability to discriminate among rotations about $\mathbf{Z}_C$. Consider Fig. 7.8: for a generic shape, α gives the major axis orientation of the ellipse which fits 'best' to the shape. Clearly, if a circular-like shape is considered, no major axis can be extracted and α becomes meaningless.

In order to overcome these issues, we chose to rely on moments only for what concerns the camera translational dofs (using area and barycenter), and to control the remaining three camera rotational dofs in cartesian space as explained in the following. Hence, $\mathbf{r}$ results in a mixture of 2D/3D quantities, so that the servoing scheme formally falls into the HVS class (Sect. 4.4).

Rotation about $\mathbf{Z}_C$ (one dof)

One interesting possibility is to replace main orientation α with an equivalent angular information obtained from Fourier coefficients. Indeed, Fourier coefficients are a valuable source of information since they encode in a condensed, discrete way, the complete shape of any closed curve $\mathcal{C}$. Therefore, apart from classification issues, they can also be exploited to obtain other shape characteristics.

Coming to our case, Fourier coefficients are used to estimate the similarity transformation parameters between two shapes, and in particular the relative rotation ρ. Assuming a camera optical axis almost perpendicular to the object plane, it is then possible to extract ρ up to a rotation ambiguity of $2\pi/\nu$ [rad], where ν is the degree of symmetry of the selected shape — see Appendix B. Angle ρ can then replace main orientation α for what concerns control of the rotation about $\mathbf{Z}_C$. When $\mathbf{Z}_C$ is almost perpendicular to the object plane, i.e., almost parallel to $\mathbf{Z}_O$, $\rho(t)$ behaves analogously to the camera roll angle[1] $\phi(\mathrm{t})$ in the sense that $\dot{\rho} = \nu\dot{\phi}$. Hence, by letting $\dot{\phi} = \mathbf{T}_\phi \boldsymbol{\omega}_C$, where 1×3 matrix $\mathbf{T}_\phi$ is the standard mapping from angular velocity $\boldsymbol{\omega}_C$ to Euler angle rate $\dot{\phi}$, we have

$$\dot{\rho} = \nu \mathbf{T}_\phi \boldsymbol{\omega}_C = \mathbf{J}_\rho \boldsymbol{\omega}_C, \tag{7.1}$$

where matrix $\mathbf{J}_\rho$ can be regarded as the Jacobian of ρ.

Direction of $\mathbf{Z}_C$ (two dofs)

In our setup, parts and plate plane orientations are fixed w.r.t. the manipulator base frame $\mathcal{F}_O$, hence it is possible to obtain the direction of $\mathbf{Z}_C$ w.r.t. $\mathcal{F}_O$, and thus w.r.t. the plane of parts and plate, directly through the robot forward kinematics and to control it in cartesian space. In order to avoid the usual singularity issues with Euler angles representations, we chose to control direction of $\mathbf{Z}_C$ directly on $SO(3)$. By letting R_{CO} be the rotation matrix of $\mathcal{F}_O$ w.r.t. $\mathcal{F}_C$, from (3.6) we have

$$\dot{\mathbf{R}}_{CO} = -[\boldsymbol{\omega}_C]_\times \mathbf{R}_{CO}. \tag{7.2}$$

Recalling that $\mathbf{Z}_C$ in $\mathcal{F}_O$ is the last row of $\mathbf{R}_{CO}$, by rearranging (7.2) we obtain

$$\dot{\mathbf{Z}}_C = \begin{bmatrix} \dot{r}_{31} \\ \dot{r}_{32} \\ \dot{r}_{33} \end{bmatrix} = \begin{bmatrix} -r_{21} & r_{11} & 0 \\ -r_{22} & r_{12} & 0 \\ -r_{23} & r_{13} & 0 \end{bmatrix} \boldsymbol{\omega}_C = \boldsymbol{\Omega} \boldsymbol{\omega}_C, \tag{7.3}$$

where r_{ij} stands for the (i,j)-th element of $\mathbf{R}_{CO}$. Note that matrix $\boldsymbol{\Omega}$ in (7.3) has always rank two reflecting the unit vector constraint of $\mathbf{Z}_C \in \mathbb{S}^2$. However, control over the direction of $\mathbf{Z}_C$

[1]Given a $\mathbf{ZYX}$ Euler angles representation of the orientation of $\mathcal{F}_C$ w.r.t. $\mathcal{F}_O$, roll angle ϕ is defined as the rotation about $\mathbf{Z}$ axis.

can be achieved by regulating only two of its components, yielding a well-posed problem. In our case, the robot final poses are designed with the camera optical axis perpendicular to the parts/plate plane, i.e., with $\mathbf{Z}_C = -\mathbf{Z}_O = [0\ 0\ -1]^T$ in $\mathcal{F}_O$ (see Fig. 3.1). Hence, a convenient choice is to regulate to zero the first two components $(r_{31},\ r_{32})$ of $\mathbf{Z}_C$ through the differential mapping

$$\begin{bmatrix} \dot{r}_{31} \\ \dot{r}_{32} \end{bmatrix} = \begin{bmatrix} -r_{21} & r_{11} & 0 \\ -r_{22} & r_{12} & 0 \end{bmatrix} \boldsymbol{\omega}_C = \mathbf{J}_{Z_C} \boldsymbol{\omega}_C, \tag{7.4}$$

where $\mathbf{J}_{Z_C}$ is the Jacobian of $(r_{31},\ r_{32})$. Note that regulation of $[r_{31}\ r_{32}]^T$ to $[0\ 0]^T$ through inversion of (7.4) admits the two stable equilibria $[0\ 0\ 1]^T$ and $[0\ 0\ -1]^T$ for $\mathbf{Z}_C$, depending on the initial vertical pointing direction of the optical axis. In our case, this ambiguity is avoided because the initial camera pose is always such that $\mathbf{Z}_C$ points downwards (the camera looks towards the parts/plate).

Having defined all the needed quantities, we let

$$\mathbf{r} = [x_g\ y_g\ a\ \rho\ r_{31}\ r_{32}]^T \in \mathbb{R}^6 \tag{7.5}$$

be the task vector used for robot pose control, and proceed to illustrate the control algorithm designed for regulation of $\mathbf{r}(t)$. Note that, as in any HVS scheme, $\mathbf{r}$ includes both image measurements $\mathbf{m}(t) = [a\ x_g\ y_g\ \rho]^T$ and 3D cartesian quantities $\boldsymbol{\chi}(t) = [r_{31}\ r_{32}]^T$. In our case, however, we do not obtain $\boldsymbol{\chi}(t)$ from a partial pose reconstruction step as supposed in the illustrative example of Sect. 4.4, but its value is computed as a function of the sole current manipulator joint state. As a result, while a parallel displacement of parts/plate can be fully recovered by using this visual task vector, an intentional tilting of plate would lead to failure of the positioning task because of the wrong value of $\boldsymbol{\chi}(t)$. Ongoing research efforts are currently devoted to obtain a task formulation fully dependent on visual quantities.

7.2.2 Control algorithm

Regulation of task (7.5) to a desired value $\mathbf{r}_d$ is addressed within the kinematic control framework, i.e., by inverting the velocity-level differential mapping $\dot{\mathbf{r}} = \mathbf{J}_r(\mathbf{m}(t),\ \boldsymbol{\chi}(t))\mathbf{J}_M(\mathbf{q})\mathbf{u}$ as in (4.20), with $\mathbf{q} \in \mathbb{R}^7$ being the manipulator joint configuration vector. Note that our robot is redundant w.r.t. task $\mathbf{r}$, with degree of redundancy one. Among the various techniques presented in Chapter 2, we chose the (PG) method (2.6), so that

$$\mathbf{u} = \mathbf{J}_r^{\dagger}\dot{\mathbf{r}} + \varsigma(\mathbf{I}_7 - \mathbf{J}_r^{\dagger}\mathbf{J}_r)\mathbf{u}_0, \quad \varsigma > 0. \tag{7.6}$$

Expression of the interaction matrix $\mathbf{J}_r$ can be obtained from

$$\dot{\mathbf{r}} = \begin{bmatrix} \multicolumn{2}{c}{\mathbf{J}_m} \\ 0 & \mathbf{J}_\rho \\ 0 & \mathbf{J}_{Z_C} \end{bmatrix} \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix} = \mathbf{J}_r \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix}. \tag{7.7}$$

Here, $\mathbf{J}_\rho$ and $\mathbf{J}_{Z_C}$ are defined in (7.1) and (7.4), and the 3×6 matrix $\mathbf{J}_m$ is the area a and barycenter $(x_g,\, y_g)$ interaction matrix

$$\begin{bmatrix} \dot{a} \\ \dot{x}_g \\ \dot{y}_g \end{bmatrix} = \mathbf{J}_m(m_{kl},\, A,\, B,\, C) \begin{bmatrix} \mathbf{v}_C \\ \boldsymbol{\omega}_C \end{bmatrix}$$

introduced in Sect. 4.3.1. Note that, in our case, $(A,\, B,\, C)$ can be directly evaluated. Indeed, as pointed out before, orientation and distance of parts/plate plane w.r.t. $\mathcal{F}_O$ is fixed and known, and the relative expression in $\mathcal{F}_C$ can be obtained via the manipulator forward kinematics.

In regulation tasks, one sets $\dot{\mathbf{r}} = \mathbf{K}(\mathbf{r}_d - \mathbf{r}) = \mathbf{K}\mathbf{e}$, $\mathbf{K} > 0$, as in (2.3), so that global exponential convergence of $\mathbf{e}(t)$ is guaranteed as long as the assumptions behind kinematic control hold — see end of Sect. 1.2. In our case, however, the specific constraints of the considered application required a fast overall execution time, which, in turn, implied fast transients for the robot motion. As a result, the aforementioned assumptions could not be met, and the neglected dynamics of the lower-level control loop, together with the limited actuator capabilities, modeling errors, and noise, proved to be a major limiting factor when imposing fast transients. In such cases, a large initial error coupled with a big $\mathbf{K}$ may lead to instability or, in the VS case, to the loss of visual features during the motion, causing failure of the task. Apart from reducing $\mathbf{K}$, such effects can be attenuated by avoiding large values for $\mathbf{e}(t)$, and in particular for $\mathbf{e}(t_0)$. To this end, we added a planning stage to the control law by defining an artificial signal $\mathbf{r}^*(t)$ which linearly interpolates the initial task value $\mathbf{r}(t_0)$ with the final desired value $\mathbf{r}_d$, thus obtaining $\mathbf{e}(t_0) = \mathbf{r}^*(t_0) - \mathbf{r}(t_0) = \mathbf{0}$. With these settings, vector $\dot{\mathbf{r}}$ becomes

$$\dot{\mathbf{r}} = \mathbf{K}(\mathbf{r}^*(t) - \mathbf{r}(t)), \tag{7.8}$$

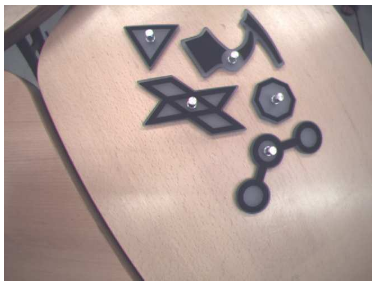
yielding a linear exponentially stable closed-loop error system driven by $\dot{\mathbf{r}}^*$

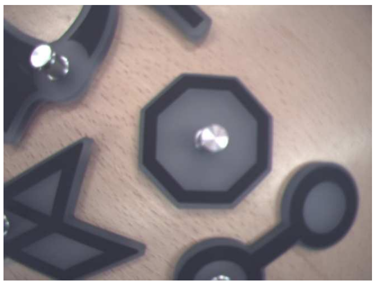
$$\dot{\mathbf{e}} = -\mathbf{K}\mathbf{e} + \dot{\mathbf{r}}^*.$$

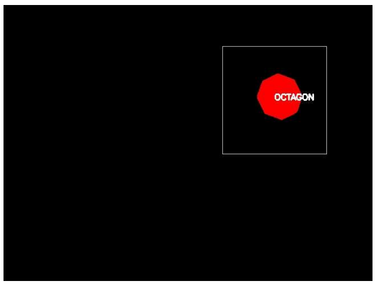
Finally, the optimization function $H(\mathbf{q})$, from which $\mathbf{u}_0 = \nabla_{\mathbf{q}} H(\mathbf{q})$ is derived and used in (7.6), is designed for joint limit avoidance as

$$H(\mathbf{q}) = \sum_{i=1}^{7} \left(\frac{q_i - \bar{q}_i}{q_{iM} - q_{im}} \right)^2,$$

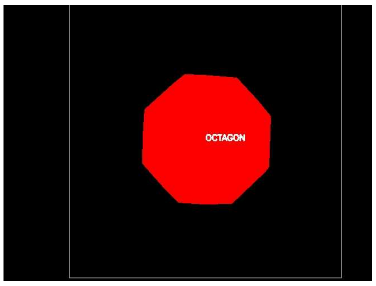
with $\bar{q}_i$, q_{iM} and q_{im} being joint i mean, max., and min. range value, respectively.


(a) Initial pose: raw camera image.

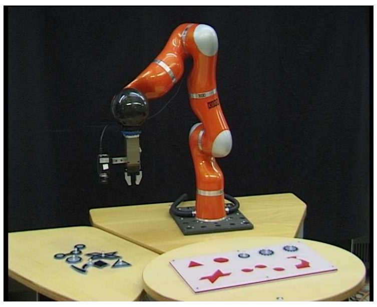

(b) Final pose: raw camera image.



(c) Initial pose: segmented camera image.



(d) Final pose: segmented camera image.


(e) Initial pose: robot configuration.

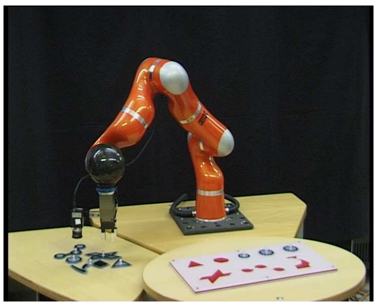

(f) Final pose: robot configuration.

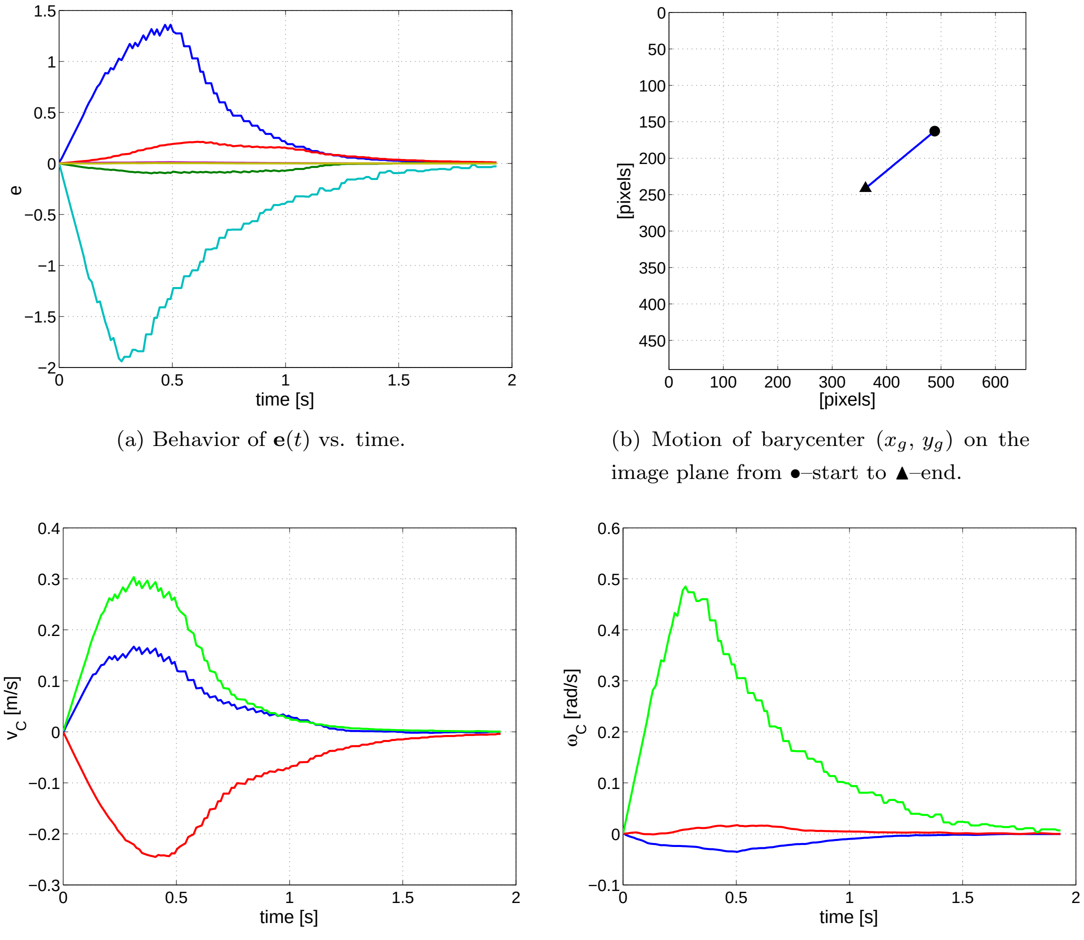
Figure 7.9: **Picking of octagon part**. Camera images and external views of the initial and final phases during the servoing. The overall motion lasts less than 2 [s].

7.3 Experimental results

In this section we present a selection of experimental data collected during the execution of a complete picking/insertion sequence[2].

Camera intrinsic (focal length, principal point, radial distortion) and extrinsic (hand-eye)

[2]Video clips of these experiments can be found at www.dlr.de/rm-neu/desktopdefault.aspx/tabid-3985//6199_read-8953.



(a) Behavior of $\mathbf{e}(t)$ vs. time.

(b) Motion of barycenter (x_g, y_g) on the image plane from ●–start to ▲–end.

(c) Camera linear velocity $\mathbf{v}_C$.

(d) Camera angular velocity $\boldsymbol{\omega}_C$.

Figure 7.10: **Picking of octagon part**. Experimental data. Thanks to the planning of $\mathbf{r}^*(t)$, the task error is initially $\mathbf{0}$ (Fig. (a)). As a consequence, no velocity jumps are commanded during the motion (Figs. (c–d)). The barycenter of the selected part moves along a straight line on the image plane (Fig. (b)), thanks to the decoupling properties of feedback (7.6)–(7.8).

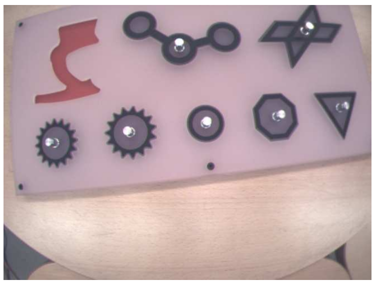
parameters were calibrated off-line using the CalDe and CalLab free softwares [Sepp *et al.* 2005; Strobl & Paredes 2005].

We consider first the approaching task to part p_2 (the octagon), see Figs. 7.9(a–f) for screenshots taken at the beginning and at the end of the HVS motion. As explained in Sect. 7.2.1, the circular symmetry of this part poses a challenge for visual control, since main orientation α cannot be used to control camera rotation about $\mathbf{Z}_C$. On the other hand, by using angle ρ from Fourier coefficients, we get $\rho(t_0) = 2.16$ [rad] and $\rho_d = 0.02$ [rad] as initial and final values (Figs. 7.9(c–d)). Recall that ρ measures shapes rotation up to their degree of symmetry ν, i.e., for an octagon with $\nu = 8$ a physical rotation of $2\pi/\nu$ [rad] results in a 2π [rad] rotation for ρ.

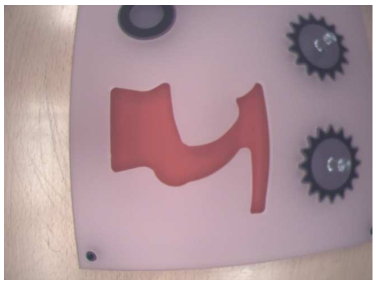
Figures 7.10(a–d) show some relevant quantities collected during the experiment. As can be seen from the plots, the overall motion is quite fast, lasting less than 2 [s] with a peak value of $\|\mathbf{v}_C\| \simeq 0.4$ [m/s] and $\|\boldsymbol{\omega}_C\| \simeq 0.5$ [rad/s]. In Fig. 7.10(a) the behavior of $\mathbf{e}(t) = \mathbf{r}^*(t) - \mathbf{r}(t)$ is reported. Note that, as expected, $\mathbf{e}(t_0) = \mathbf{0}$ because of the definition of reference signal $\mathbf{r}^*(t)$. As a consequence, no initial jump is present in the commanded camera velocity $(\mathbf{v}_C, \boldsymbol{\omega}_C)$ (Figs. 7.10(c), 7.10(d)). Finally, Fig. 7.10(b) shows the image plane motion of the octagon barycenter (x_g, y_g) during the servoing, which results in a straight line as a consequence of the decoupling properties of control (7.6)–(7.8).

Avoiding velocity jumps during fast transients, as in this experiment, has a major relevance for meeting the assumptions behind kinematic control. Indeed, a too high acceleration request, coupled with an high speed profile, could most likely violate such assumptions because of limited actuator capabilities, unmodeled dynamics of lower level feedbacks, and uncertainties in the robot dynamic model. As a result, close tracking of the 'ideal' velocity command $\mathbf{u}(t)$ would be prevented, potentially leading to failure of the visual task. It is, then, evident the benefit of having closed the loop on the artificial signal $\mathbf{r}^*(t)$ in terms of smoothness of the velocity commands.

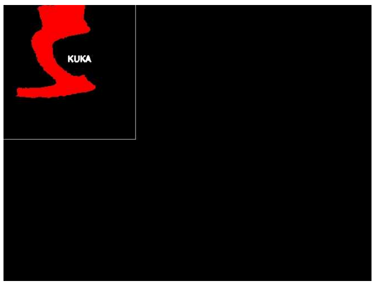
As an additional case, we present the approaching phase to hole o_8 (Figs. 7.11(a–f)). This shape is nonsymmetric ($\nu = 1$), and angle ρ matches the physical rotation about $\mathbf{Z}_C$. At the initial pose (Fig. 7.11(b)) $\rho(t_0) = 1.32$ [rad], and at the final pose (Fig. 7.11(e)) $\rho_d = 0.05$ [rad]. The overall motion lasts about 3 [s] with a peak value of $\|\mathbf{v}_C \simeq 0.45\|$ [m/s] and $\|\boldsymbol{\omega}_C\| \simeq 1.16$ [rad/s] (Figs. 7.12(c) and 7.12(d)).



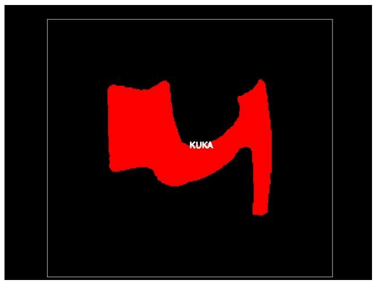
(a) Initial pose: raw camera view.



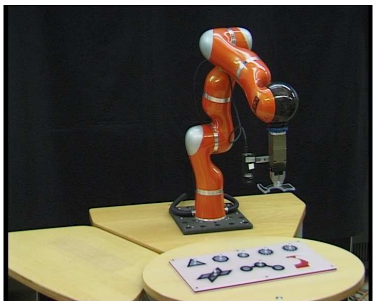
(b) Final pose: raw camera view.



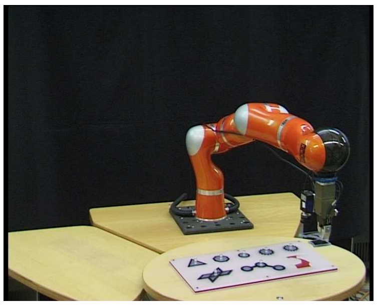
(c) Initial pose: segmented camera view.



(d) Final pose: segmented camera view.

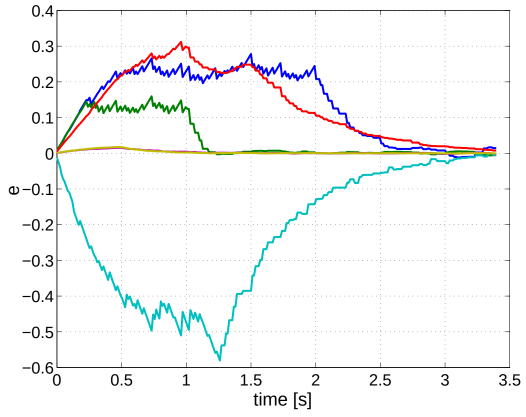


(e) Initial pose: robot configuration.

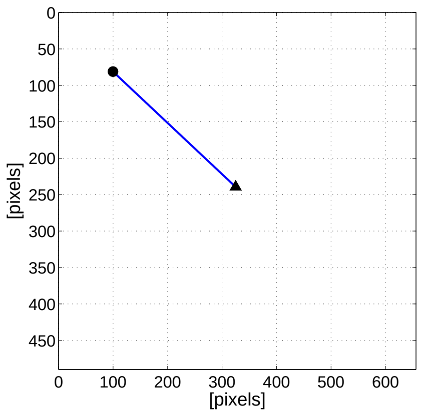


(f) Final pose: robot configuration.

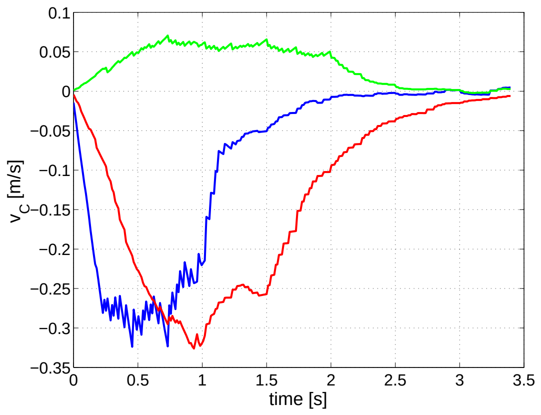
Figure 7.11: **Insertion of KUKA part**. Camera images and external views of the initial and final phases during the servoing. The overall motion lasts about 3 [s].



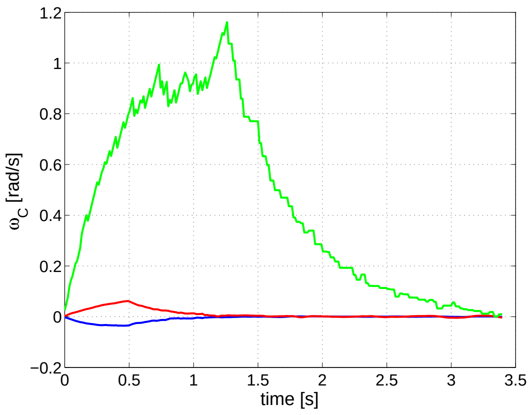
(a) Behavior of $\mathbf{e}(t)$ vs. time.



(b) Motion of barycenter (x_g, y_g) on the image plane from ●–start to ▲–end.



(c) Camera linear velocity $\mathbf{v}_C$.



(d) Camera angular velocity $\boldsymbol{\omega}_C$.

Figure 7.12: **Insertion of KUKA part**. Experimental data. Thanks to the planning of $\mathbf{r}^*(t)$, the task error is initially $\mathbf{0}$ (Fig. (a)). As a consequence, no velocity jumps are commanded during the motion (Figs. (c–d)). The barycenter of the selected part moves along a straight line on the image plane (Fig. (b)), thanks to the decoupling properties of feedback (7.6)–(7.8).

Figures 7.12(a–d) show again some relevant quantities collected during the experiment. In particular, Fig. 7.12(a) depicts the behavior of $\mathbf{e}(t)$ over time, from which we can check that $\mathbf{e}(t_0) = \mathbf{0}$ yielding, as in the previous case, a camera velocity command without jumps. Furthermore, in Fig. 7.12(b) image motion of the barycenter of hole o_8 is shown, resulting as before in a straight line.

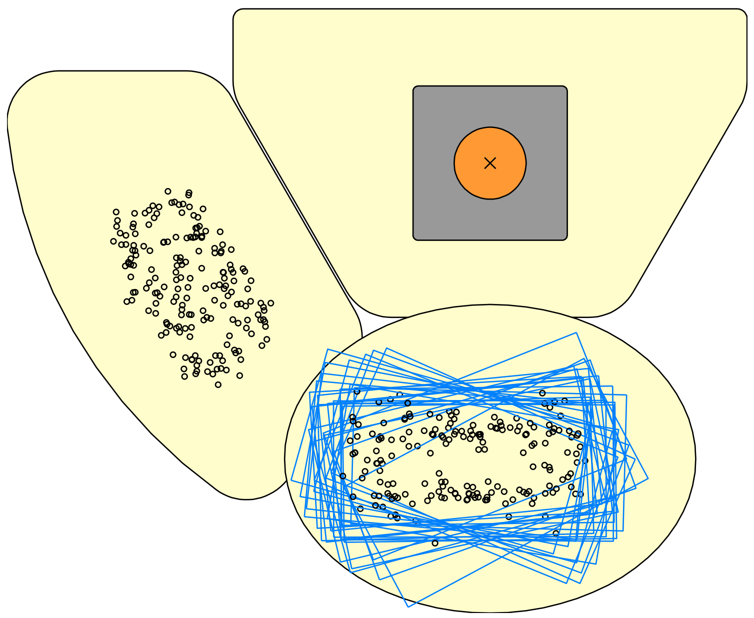


Figure 7.13: Placement of the parts and the plate on the table for the statistical evaluation of the automated assembly.

Statistics for the complete sequence of operations (picking and insertion of all parts) were collected over 20 cycles, i.e., for a total of 160 assemblies. Altogether, the robot was able to insert 154 parts successfully with a success rate of 96.25%. In four cases the assembly failed because the vision system was not able to detect the part or hole reliably (lost tracking during motion three times, once the part was not found at all). In the remaining two cases, the servoing was completed successfully, but the insertion failed because of the assembly strategy. No error recovery was implemented for these experiments: in case of errors (like lost tracking) the current part was dropped and the sequence continued with the next part. Figure 7.13 shows a collective picture that summarizes the distribution of parts and plate for the 20 sequences, while experimental results of the insertion strategy can be found in [Stemmer *et al.* 2007].

Finally, we discuss the results of an intentional displacement of the plate during the insertion of piece p_7. Figures 7.14(a–l) show a sequence of screenshots taken during this experiment. Thanks to the HVS feedback, the robot is able to reject external disturbances such as unexpected displacements of parts/plate, and to eventually fulfill the insertion as soon as the plate stops moving. Indeed, it is easy to prove that any constant 'visual' disturbance $\mathbf{d}$ is attenuated at steady state by a factor $\mathbf{K}^{-1}$. Hence, the value of $\mathbf{K}$ can be exploited to tune the amount of tracking error of the HVS algorithm, so that the chosen features are kept within the boundaries of the image plane. Several relevant quantities of this experiment are reported in Figs. 7.15(a–d).

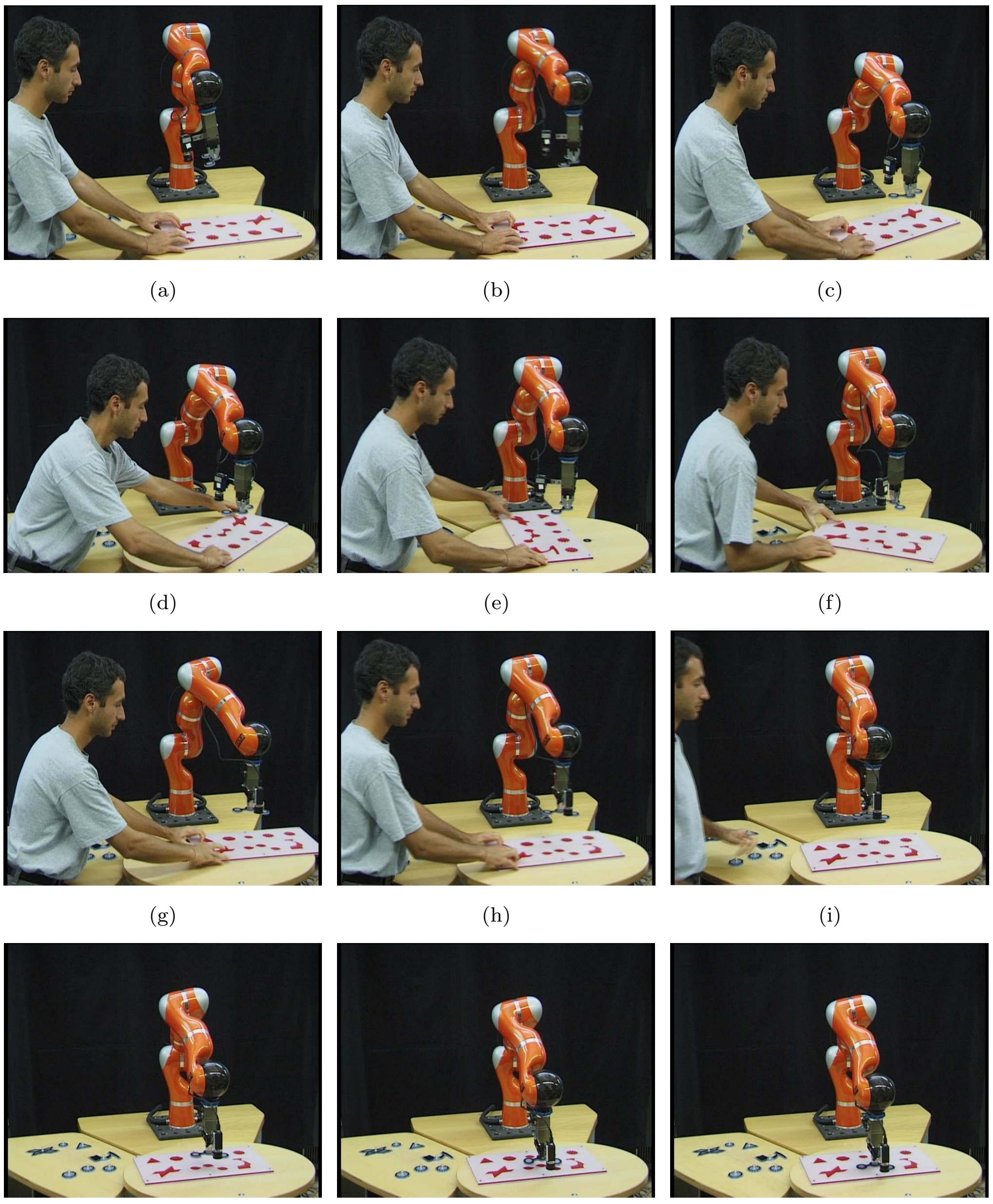
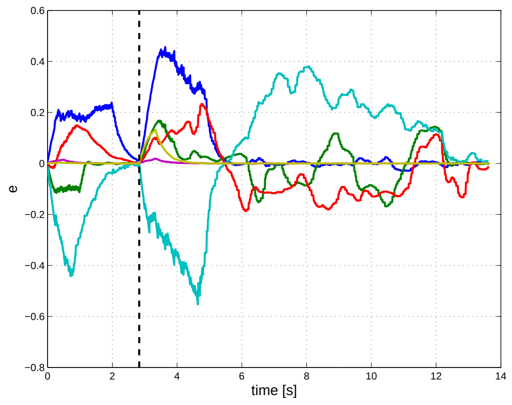
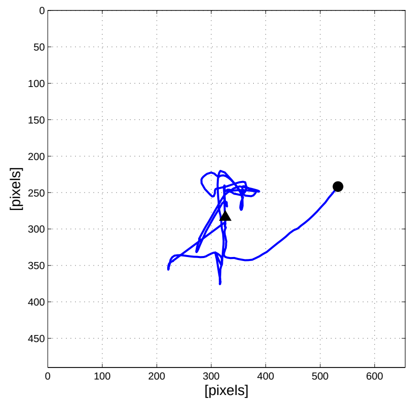

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

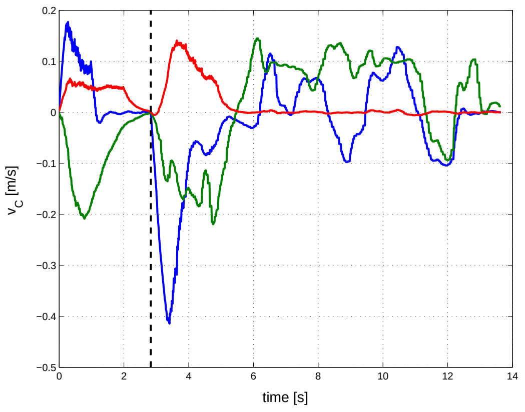
Figure 7.14: **Insertion of PAPAS part**. Despite the intentional displacement of the plate, seen as an external disturbance, the HVS algorithm does not lose track of the hole, and eventually reaches the correct final pose (Figs. (a–j)). The assembly strategy can then successfully fulfil the insertion of the part (Figs. (k–l)).



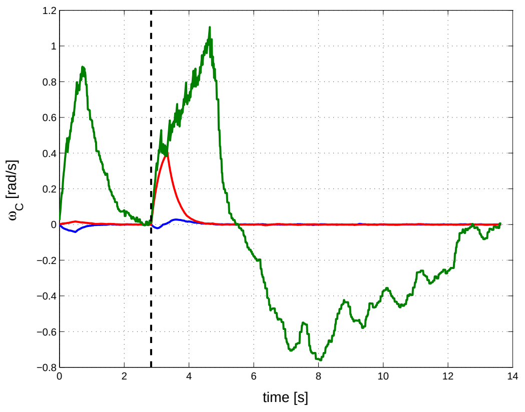
(a)



(b)



(c)



(d)

Figure 7.15: **Insertion of PAPAS part**. Experimental data. Note how the barycenter is kept inside the image plane (the boundaries of Fig. (b)) despite the external disturbance. A dashed vertical line represents the beginning of the displacement of the plate.

In particular, it is possible to check that the barycenter always lies inside the image plane boundaries (Fig. 7.15(b)). In the above plots, a dashed vertical line represents the beginning of the intentional displacement of the plate.

Conclusions

This final Chapter summarizes the main theoretical and applied achievements of the Thesis, and indicates a number of open points and possible future extensions of the work presented so far.

Summary

In this Thesis, we studied the problem of visual control for robot manipulators. The basic idea was to consider a camera as a sensor yielding a (nonlinear) output function of the scene, i.e., of 3D states subject to the Euclidean rigid body kinematics. Therefore, by interpreting image quantities as standard *task variables*, we addressed visual control from different points of view within the scope of Control Theory, namely:

(i) task-oriented kinematic modeling of robot manipulators, either with fixed or nonholonomic mobile base;

(ii) kinematic control of robot manipulators, with a special emphasis on redundancy exploitation;

(iii) merging of Visual Servoing schemes with the aforementioned task-oriented modeling;

(iv) suitable exploitation of the proposed kinematic control techniques aimed at improving the overall execution performance of a visual task;

(v) development of a nonlinear observation framework able to estimate online the unmeasurable 3D quantities needed by Visual Servoing implementations;

(vi) analysis of the benefits, in terms of improved stability and performance, arising from plugging the observation schemes into Visual Servoing loops.

All the theoretical claims were validated through extensive simulations and experiments on real robots equipped with cameras. In particular, the reported experiments showed a strong adherence with the corresponding simulation results, thus fully supporting the theory behind the proposed methodology. Such methodology was further exploited within a potential industrial application developed at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR). In this context, kinematic control and Visual Servoing algorithms were combined in order to regulate the pose of a manipulator committed with pick-and-assemble tasks of complex planar parts on a movable target plate. The results demonstrated the effectiveness of the approach in terms of robustness and reactivity w.r.t. external 'disturbances', such as unexpected displacements of parts/plate. Additionally, evaluation of the performance in continuous operation mode contributed to positively assess the overall reliability.

In conclusion, we believe that the proposed integration of kinematic modeling/control techniques, Visual Servoing schemes, and nonlinear estimation tools can substantially improve the behavior or robots guided by vision. The long-term objective of this research is to build artificial systems capable of autonomous interaction with the environment like the humans are. For instance, a robot should be able to reach a location, pick an item, or assemble two objects without the need of simplifying assumptions on the world, or any additional 'help' besides the information collected from its own sensors. Of course, a full exploitation of vision is a necessary (but not sufficient) condition in reaching such a goal. Indeed, besides vision, other important facets must be addressed: elaboration of additional cues, like tactile information, and their fusion with other sensory data is mandatory. Furthermore, the ability to infer high-level properties of the world from raw information, or, in other words, the amount of environmental awareness possessed by a robot, is even more crucial.

While these fundamental steps are yet to be fully achieved, our integrated approach may be seen as an improvement in the right direction within the scope of visual control. Indeed, thanks to the proposed estimation tools, one can conceive a servoing scheme totally independent from any prior 3D knowledge, like depth or additional structure relative to the target or to the robot final pose. This possibility would be particularly relevant, for instance, in the case of environmental navigation/exploration tasks for mobile manipulators. In this scenario, a robot could store several images of interesting locations while exploring the environment, regardless of the extraction of corresponding 3D information — a step possibly difficult or even impossible without special assumptions/devices. Hence, when asked to reach again the stored locations, the robot could recover online the missing information through the proposed observation tools, thus greatly enhancing its capability to fulfill the visual task in full autonomy. The generality of our methodology allows to devise similar strategies also when addressing other tasks besides navigation, such as grasping or manipulation. Suitable exploitation of these possibilities will be the leading theme of our future research.

Open points

Like any research activity, the work presented in this Thesis is not free of open points yet to be resolved. For instance, no formal proof is given for the closed-loop stability of the integration VS feedback/observation schemes. The experiments reported in Chapter 6 are quite promising in this direction, but a rigorous analysis is anyway required. Indeed, the nonlinearities of both VS/observers exclude a trivial satisfaction of the *separation principle* proper to linear systems. Therefore, we most likely expect the overall closed-loop stability to have a local nature, and we are currently seeking an analytic characterization of the corresponding stability domain.

Still about the observation framework, additional effort must be devoted to the use of image moments. In Sect. 5.2.2 we derived a formal solution for the general case, i.e., without any assumption on the geometry of the target object. However, choice of which moments one should exploit so as to meet the persistency of excitation condition does not yet have a definitive answer. Our feeling is that, in many practical situations, the amount of noise and uncertainties will play a central role in deciding what moments can provide the most significant information. Current research activities are then aimed at shedding light on this topic.

A last point that, in our opinion, still deserves some attention is the problem of full state regulation of NMMs. Within our approach, NMMs are controlled at the output (task) level by taking into account the nonholonomy inside the task Jacobian. While this control design proves to be a sound choice for task execution, it does not allow to specify an arbitrary *configuration* $\mathbf{q}_p$ to be reached by the mobile platform. The resulting pose will be just a consequence of the overall motion. Note that no time-invariant and continuous feedback can achieve full state stabilization of nonholonomic systems such as NMMs, because of the topological limitations of Brockett's theorem [Brockett 1983]. Therefore different solutions must be sought, such as time-varying or discontinuous laws[3]. Some possibilities have already been explored in the last years [Conticelli *et al.* 1999; Fang *et al.* 2005]. Hence, our intention is to take advantage of these ideas in order to generalize our approach, so as to cover the case of full-pose stabilization of NMMs as well.

Future directions

A major challenge posed to the forthcoming research consists in the unification of vision and force information to obtain an hybrid visual/force control. Such a possibility would represent a relevant step in mimicking the human's way of manipulating objects in the world, i.e., by relying on both visual and tactile senses at the same time. Some attempts have already been made in the last years, see [Morel *et al.* 1998; Prats *et al.* 2005; Lippiello *et al.* 2007] and references

[3] An overview of these topics, involving nonholonomic planning and control, can be found in [Laumond 1998].

therein. In our case, we envisage the enhancing of the application proposed in Chapter 7 along these directions. The exploitation of force and vision feedback during the insertion phase, for instance, could significantly increase the robustness and performance of the assembly task — the robot could actually 'see' and 'feel' parts and holes. In order to achieve such a result, a suitable extension of our methods to the torque-level control is obviously required. To this end, the robot dynamical model must be explicitly taken into account in the feedback design, thus going beyond the kinematic control approach here adopted. An oversampling of the visual data seems also necessary for a sound merging of vision and force sensors. Indeed, common cameras provide images with frame rates up to about 50 [Hz], while a force control loop usually runs at 1 [KHz] or more. The use of predictive systems, like (Extended) Kalman Filters, could solve this issue while simultaneously smoothing the effect of noise present in any image stream.

Another interesting research topic arises from the problem of planning suitable image trajectories for the features included in the visual task. This issue, already introduced in Sect. 4.3.2, has a major relevance when, e.g., relying on *many* (≥ 4) feature points for pose control. In general, consistency with the constraint of rigid camera motion is not met, and unrealizable image velocities are fed to the control law. Some authors have tackled this problem from different points of view, see [Basri *et al.* 1998; Mezouar & Chaumette 2002; Allotta & Fioravanti 2005; Chesi & Hung 2007]. In this context, our goal is to find a theoretical solution which can automatically guarantee consistency with the constraints of rigid motion and perspective mapping. The ideas developed by [Soatto *et al.* 1996; Ma *et al.* 2000], based on exploitation of the Essential space formulation (3.17), seem to perfectly fit in this picture. Of course, some work is needed to tailor such approach to the case of visual control, so as to include, for instance, visibility or mechanical constraints in the overall formulation.

Finally, we believe that the 3D observation framework proposed in the Thesis could answer to other needs besides the ones considered in the previous Chapters. For instance, the growing number of Visual SLAM solutions [Lemaire *et al.* 2007; Silveira *et al.* 2007; Davison *et al.* 2007] could greatly benefit of online tools for 3D estimation free from constraints on the particular scene, a problem that can be tackled by the moment-based observation techniques of Chapter 5. The same considerations also hold for what concerns visual-based navigation tasks, like the recent proposals in [Remazeilles *et al.* 2004; Remazeilles & Chaumette 2007] where navigation is achieved by relying on an internal database of images stored off-line. Again, the possibility to obtain a continuous and convergent estimate of relevant 3D data can contribute to improve both the performance of the visual feedback and the understanding of surrounding environment.

Appendixes

A
Nonholonomic Constraints

THIS APPENDIX PRESENTS, in a compact way, the basic differential geometry tools needed to deal with nonholonomic constraints. Many of the concept introduced hereafter are taken from [Murray *et al.* 1994; Isidori 1995].

A.1 Tools from differential geometry

A smooth *vector field* $\mathbf{f}$ on a smooth manifold M local diffeomorphic to $\mathbb{R}^n$ is defined as a smooth map $\mathbf{f}: M \to T_{\mathbf{q}}M$ that associates to each point $\mathbf{q} \in M$ a vector $\mathbf{f}(\mathbf{q}) \in T_{\mathbf{q}}M$. Vector fields can define the right hand side (rhs) of a differential equation

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}). \tag{A.1}$$

In this case, associated to the vector field, we define the *flow* $\phi_t^{\mathbf{f}}(\mathbf{q})$ of $\mathbf{f}(\mathbf{q})$ as the map which yields the solution of (A.1) at time t starting from $\mathbf{q}$ at time t_0. Formally, the flow is a function $\phi_t^{\mathbf{f}}(\mathbf{q}) : M \to M$ that satisfies

$$\frac{d}{dt}\phi_t^{\mathbf{f}}(\mathbf{q}) = \mathbf{f}(\phi_t^{\mathbf{f}}(\mathbf{q})),$$

and meets the group property

$$\phi_t^{\mathbf{f}} \circ \phi_s^{\mathbf{f}} = \phi_{t+s}^{\mathbf{f}}$$

for all t and s, where $\circ$ stands for the composition of the two flows, namely $\boldsymbol{\phi}_t^{\mathbf{f}}(\boldsymbol{\phi}_s^{\mathbf{f}}(\mathbf{q}))$. The time derivative of a smooth function $V : M \to \mathbb{R}$ along the flow of $\mathbf{f}$ is given by

$$\dot{V} = \frac{\partial V}{\partial \mathbf{q}}\mathbf{f}(\mathbf{q}) = \sum_{i=1}^{n} \frac{\partial V}{\partial q_i} f_i,$$

and is referred to as the *Lie derivative* of V along $\mathbf{f}$:

$$L_{\mathbf{f}} V = \frac{\partial V}{\partial \mathbf{q}}\mathbf{f}(\mathbf{q}).$$

Given two vector fields $\mathbf{g}_1$ and $\mathbf{g}_2$, the map $\boldsymbol{\phi}_t^{\mathbf{g}_1} \circ \boldsymbol{\phi}_s^{\mathbf{g}_2}$ represents the composition of the flow of $\mathbf{g}_2$ for s seconds with the flow of $\mathbf{g}_1$ for t seconds. In general, it is $\boldsymbol{\phi}_t^{\mathbf{g}_1} \circ \boldsymbol{\phi}_s^{\mathbf{g}_2} \neq \boldsymbol{\phi}_s^{\mathbf{g}_2} \circ \boldsymbol{\phi}_t^{\mathbf{g}_1}$, i.e., the operation of flow composition is not commutative. Indeed, starting at $\mathbf{q}_0$ and following $\mathbf{g}_1$ for ϵ seconds, $\mathbf{g}_2$ for ϵ seconds, $-\mathbf{g}_1$ for ϵ seconds and $-\mathbf{g}_2$ for ϵ seconds yields

$$\mathbf{q}(4\epsilon) = \mathbf{q}_0 + \epsilon^2\left(\frac{\partial \mathbf{g}_2}{\partial \mathbf{q}}\mathbf{g}_1(\mathbf{q}_0) - \frac{\partial \mathbf{g}_1}{\partial \mathbf{q}}\mathbf{g}_2(\mathbf{q}_0)\right) + O(\epsilon^3). \tag{A.2}$$

The ϵ^2 term in (A.2) is defined as the *Lie bracket* of the two vector fields $\mathbf{g}_1$ and $\mathbf{g}_2$

$$[\mathbf{g}_1, \mathbf{g}_2] = \frac{\partial \mathbf{g}_2}{\partial \mathbf{q}}\mathbf{g}_1(\mathbf{q}_0) - \frac{\partial \mathbf{g}_1}{\partial \mathbf{q}}\mathbf{g}_2(\mathbf{q}_0)$$

and represents the infinitesimal motion (of order ϵ^2) that results from flowing around a square defined by $\mathbf{g}_1$ and $\mathbf{g}_2$. If $[\mathbf{g}_1, \mathbf{g}_2] = 0$ it can be shown that (A.2) becomes $\mathbf{q}(4\epsilon) = \mathbf{q}_0$ and $\mathbf{g}_1$ and $\mathbf{g}_2$ are said to *commute*. Some properties of the Lie brackets are:

(i) skew-symmetry: $[\mathbf{g}_1, \mathbf{g}_2] = -[\mathbf{g}_2, \mathbf{g}_1]$,

(ii) Jacobi identity: $[\mathbf{g}_1, [\mathbf{g}_2, \mathbf{g}_3]] + [\mathbf{g}_3, [\mathbf{g}_1, \mathbf{g}_2]] + [\mathbf{g}_2, [\mathbf{g}_3, \mathbf{g}_1]] = 0$,

(iii) chain rule: $[\alpha\mathbf{g}_1, \beta\mathbf{g}_2] = \alpha\beta[\mathbf{g}_1, \mathbf{g}_2] + \alpha(L_{\mathbf{g}_1}\beta)\mathbf{g}_2 - \beta(L_{\mathbf{g}_2}\alpha)\mathbf{g}_1$.

A *distribution* Δ assigns to each $\mathbf{q}$ a subspace of the tangent space in $\mathbf{q}$ $T_{\mathbf{q}}M$. A distribution can be defined by a set of m smooth vector fields $\mathbf{g}_1 \ldots \mathbf{g}_m$ as

$$\Delta = \mathrm{Im}\{\mathbf{g}_1, \ldots, \mathbf{g}_m\}.$$

Evaluated at any point $\mathbf{q} \in M$, the distribution defines a linear subspace of the tangent space

$$\Delta_{\mathbf{q}} = \mathrm{Im}\{\mathbf{g}_1(\mathbf{q}), \ldots, \mathbf{g}_m(\mathbf{q})\} \subset T_{\mathbf{q}}M.$$

A distribution is said to be *regular* if the dimension of $\Delta_{\mathbf{q}}$ does not vary with $\mathbf{q}$, and *involutive* if it is closed under the Lie bracket, i.e.,

$$\Delta \text{ involutive} \quad \Leftrightarrow \quad \forall \mathbf{g}_i, \mathbf{g}_j \in \Delta,\ [\mathbf{g}_i, \mathbf{g}_j] \in \Delta.$$

The *involutive closure* of a distribution, denoted $\overline{\Delta}$, is the closure of Δ under the Lie bracket operation, i.e., $\overline{\Delta}$ is the smallest distribution containing Δ such that $\forall \mathbf{g}_i, \mathbf{g}_j \in \Delta, \quad [\mathbf{g}_i, \mathbf{g}_j] \in \Delta$.

A.2 Integrability of nonholonomic constraints

A distribution Δ of constant dimension k is said to be *integrable* if for every point $\mathbf{q} \in M$, there exists a set of $n-k$ smooth functions $h_i : M \to \mathbb{R}$ such that, $\forall \mathbf{g}_j \in \Delta$

$$L_{\mathbf{g}_j} h_i = 0, \quad i = 1, \dots, n-k. \tag{A.3}$$

The hypersurfaces defined by the level sets

$$\{\mathbf{q} : h_1(\mathbf{q}) = c_1, \dots, h_{n-k}(\mathbf{q}) = c_{n-k}\} \tag{A.4}$$

are called *integral manifolds* of Δ. Condition (A.3) can be interpreted as requiring the distribution to be equal to the tangent space of h_i at the point $\mathbf{q}$. The following theorem gives a mean to check whether a distribution is integrable.

Theorem A.1 (Frobenius)**.** *A regular distribution is integrable if and only if is involutive.*

Thus, if Δ is an k-dimensional involutive distribution, then locally there exist $n-k$ functions $h_i : M \to \mathbb{R}$ such that integral manifolds of Δ are given by the level surfaces of $h = (h_1, \dots, h_{n-k})$. These level surfaces form a *foliation* of M, and a single level surface is called a *leaf of the foliation*.

A smooth *one-form* is a mapping $\mathbf{a}^T : M \to T^*_{\mathbf{q}}M$, where $T^*_{\mathbf{q}}M$ is the dual space of linear forms on $T_{\mathbf{q}}M$. One-forms are regarded as *covectors* and in local coordinates are represented as row vectors. The differential of a smooth function h is an example of one-forms

$$dh = \left[\frac{\partial h}{\partial q_1} \cdots \frac{\partial h}{\partial q_n}\right].$$

Not all one-forms, however, are necessarily differentials of smooth functions, and when this is the case the one-form is said to be *exact*. A codistribution A^T assigns a subspace of $T^*_{\mathbf{q}}M$ smoothly to each $\mathbf{q} \in M$. A codistribution can be defined by a set of k one-forms

$$A^T = \mathrm{Im}\{\mathbf{a}_1^T \dots \mathbf{a}_k^T\}.$$

As before, the dimension of a codistribution is the dimension of $A^T_{\mathbf{q}}$, and a codistribution is said to be regular if its dimension is constant for every $\mathbf{q}$. Given a set of k smooth independent one-forms $\mathbf{a}_i^T(\mathbf{q})$ which defined a regular codistribution A^T, there exist $n-k$ smooth vector fields $\mathbf{g}_j(\mathbf{q})$ such that

$$\mathbf{a}_i^T(\mathbf{q})\mathbf{g}_j(\mathbf{q}) = 0 \quad \forall i, j.$$

These vectors define a distribution $\Delta = (A^T)^\perp$ called the annihilating distribution of codistribution A^T, i.e.,

$$A^T \Delta = 0.$$

The following proposition shows how to to check whether a codistribution is exact, i.e., is made of exact one-forms, in terms of its annihilating distribution.

Proposition A.1. *A codistribution A^T is exact if and only if its annihilating codistribution $(A^T)^\perp$ is involutive.*

This result can then be used to determine the nonholonomy of a given set of k Pfaffian constraints as in (1.3). Indeed, the Pfaffian constraints are holonomic, i.e., can be integrated to a set of k smooth functions $h_i(\mathbf{q})$, iff the vector fields defined by columns of $\mathbf{N}(\mathbf{q})$ in (1.6) are involutive. In this case, it is

$$\mathbf{a}_i^T(\mathbf{q})\dot{\mathbf{q}} = dh_i\,\dot{\mathbf{q}} = \dot{h}_i(\mathbf{q}) = 0,$$

implying that the system trajectories are confined to the submainfold of $\mathcal{Q}$

$$\{\mathbf{q}:\ h_1(\mathbf{q}) = c_1, \dots, h_k(\mathbf{q}) = c_k\}.$$

B
Pattern Recognition

Pattern recognition is a very classical problem in computer vision and many different solutions have been proposed in the literature — see Chapter 4 for some references. This Appendix details *one* pattern recognition algorithm without any pretension of providing, at the same time, an exhaustive coverage of this multifaceted topic. Indeed, no general methodology exists, and a case-to-case analysis is always needed when devising strategies for specific situations. Therefore, we only focus on the experimental application presented in Chapter 7, and illustrate the chosen approach.

Usually, given a reference shape to be found in the image, the problem of pattern recognition can be split into three conceptually distinct steps:

(i) a first preprocessing stage is applied to the raw camera image in order to extract the contours of several regions (blobs) which match the searched shape. This step typically involves image binarization and contour tracing, or edge detection and grouping. Use of color information may support this process substantially;

(ii) subsequently, specific 'features'[1] are computed from each segmented blobs with the aim

[1]Note that these features must not be confused with the ones used for VS purposes, even though they can coincide in specific cases.

of obtaining a discrete set of parameters sufficiently representative of every shape, and as much as possible invariant w.r.t. changes of the camera point of view;

(iii) finally, these features are fed to a classifier, i.e., an algorithm able to decide the class corresponding to an input feature vector, or, in other words, to discriminate whether the searched shape is present or not in the input set of candidate blobs.

According to this subdivision, our algorithm consists of a preliminary image preprocessing step based on standard color binarization where pixels are marked as belonging or not to a hole/part. Holes are segmented by looking for their red background color, while, for parts, the black outer tags glued on each of them are considered (see Fig. 7.3(a)–(d)). The subsequent feature extraction and shape classification phases are based on the *Affine-invariant Fourier descriptors* and *linear MSE classification* theories and are explained in the following sections.

B.1 Affine-invariant Fourier descriptors

Consider two closed curves in the image plane $\mathcal{C}^0$ and $\mathcal{C}$ representing the boundary of a planar object under two different camera views. Under affine projection[2], $\mathcal{C}^0$ and $\mathcal{C}$ would be equivalent with respect to the affine transformation group: $C^0 \bowtie C$. Affine-invariant Fourier descriptors (AIFD) [Arbter 1989, 1990] provide a solution to the problem of finding a discrete, minimal, and complete set of parameters able to univocally encode the common shape information shared by $\mathcal{C}$ and $\mathcal{C}^0$ despite the different camera points of view. Hence, AIFD yield a suitable choice for shape classification issues and, thanks to their ordered discrete nature, can be easily compared in contrast to the initial contour representation with its inherent correspondence problem. In the following, a brief overview of AIFD is given.

Let $x^0 = u^0 + jv^0 \in \mathbb{C}$ and $x = u + jv \in \mathbb{C}$ be two corresponding points on $\mathcal{C}^0$ and $\mathcal{C}$, respectively. The affine transformation among them may be written as

$$x = ax^0 + bx^{0*} + c, \qquad aa^* - bb^* \neq 0 \tag{B.1}$$

where a, b, $c \in \mathbb{C}$ and $(^*)$ stands for the complex conjugation operation. Let $x(t)$ be a parametric representation of $\mathcal{C}$ with a suitable parameter $t \in \mathbb{R}$. Since $\mathcal{C}$ is closed, the curve can be traced an indefinite number of times, i.e., there exists a $T \in \mathbb{R}$ such that $x(t) = x(t+T)$, $\forall t$. Hence, $x(t)$ may be considered to be a periodic function with period T and admits a Fourier series expansion $x(t) = \sum_{k=-\infty}^{+\infty} C_k e^{j2\pi kt/T}$, with

$$C_k = \frac{1}{T}\oint_{\mathcal{C}} x(t) e^{-j2\pi kt/T}$$

[2]Affine projection model, sometimes called *weak perspective* model, is the best linear approximation of the standard pin-hole camera projection model (see Chapter. 3). The approximation is very close if the mean distance along the optical axis between the camera and the object is large compared to the distance variation.

being the Fourier coefficients. Let $x^0(t^0)$ be a parametric representation of $\mathcal{C}^0$ (the reference curve). In general, parameters t and t^0 do not coincide and (B.1) may be formally written as

$$x(t(t^0)) = ax^0(t^0) + bx^{0*}(t^0) + c, \qquad aa^* - bb^* \neq 0, \tag{B.2}$$

where $t(t^0)$ represents a generic mapping among the two parameters. If the parameterizing function is chosen such that it is linearly transformed under affine transformations, then $t(t^0) = \mu(t^0 + \tau)$ where μ is a scale factor and shift τ is a consequence of the different starting point in the two parameterizations. While the 'natural parameter' arc length does not meet this condition, a valid parameter can be obtained by integrating over area differentials along the curve, as:

$$t = \frac{1}{2}\int_{\mathcal{C}} \det\left(u'(\xi)v_\xi(\xi) - v'(\xi)u_\xi(\xi)\right)\mathrm{d}\xi$$

where ξ is any initial parameter (e.g., arc length), $u_\xi = \mathrm{d}u/\mathrm{d}\xi$, $v_\xi = \mathrm{d}v/\mathrm{d}\xi$, $u' = u - u_C$, $v' = v - v_C$, and

$$u_C + jv_C = \frac{2}{3}\frac{\oint_{\mathcal{C}} x(\xi)(u(\xi)v_\xi(\xi) - v(\xi)u_\xi(\xi))\mathrm{d}\xi}{\oint_{\mathcal{C}} u(\xi)v_\xi(\xi) - v(\xi)u_\xi(\xi)\mathrm{d}\xi}$$

is the area center of gravity of the region enclosed by curve $\mathcal{C}$.

Choosing this parameterization, and owing to the linearity of (B.2), it is possible to show that the Fourier coefficients of $\mathcal{C}$ and $\mathcal{C}^0$ are related by

$$C_0 = aC_0^0 + bC_0^{0*} + c \tag{B.3}$$

$$C_k = z^k(aC_k^0 + bC_{-k}^{0*}), \quad k \neq 0, \tag{B.4}$$

where $z(\tau) = e^{-j2\pi\tau/T^0}$. Note that (B.4) is independent of the translation parameter c. In order to obtain a quantity completely free of all the affine parameters (a, b, c), i.e., an *affine-invariant Fourier descriptor*, we set

$$Q_k = \frac{C_kC_p^* - C_{-k}^*C_p}{C_pC_p^* - C_{-p}^*C_{-p}} = z^{k-p}Q_k^0 \tag{B.5}$$

with $p \neq 0$ fixed, $C_p \neq 0$, and $k = \pm1, \pm2, \ldots$. Descriptors Q_k, which can be computed directly from the parametric representation $x(t)$ of $\mathcal{C}$, are affine-invariant and complete. They do not lose any 'intrinsic' shape information w.r.t. the class represented by $\mathcal{C}^0$. Anyway, they depend on the starting point shift τ between parameterizations t and t^0. While there exist techniques to safely remove from (B.5) the dependence on z, a simpler choice is to just consider the magnitude of Q_k, i.e.,

$$|Q_k| = |Q_k^0|. \tag{B.6}$$

Descriptors $|Q_k|$ are affine and starting point shift invariant, but are not complete (phase information is lost), implying that boundaries belonging to different objects may share the same descriptors. However, in practice, their high selectivity guarantees a low ambiguity risk and high robustness against noise [Fenske 1993]. Therefore, also because of their easy implementation, we chose them as features for our shape classification problem.

Apart from pattern recognition issues, Fourier coefficients can also be used to estimate the similarity transformation parameters between two shapes, and in particular the relative rotation ρ. Assuming a camera optical axis almost perpendicular to the object plane, the affine mapping between $\mathcal{C}$ and $\mathcal{C}^0$ reduces to a similarity transformation

$$x(t) = |a|e^{j\rho}x^0(t^0) + c.$$

From (B.4), we have

$$C_k = z^k |a| e^{j\rho} C_k^0, \quad k = \pm 1, \pm 2, \ldots,$$

which, by letting ϕ_k denote the phase term of $C_k = |C_k|\phi_k$, implies

$$\phi_k = z^k e^{j\rho} \phi_k^0.$$

Rotation ρ can then be determined from two Fourier coefficients C_q and C_r as

$$e^{j(r-q)\rho} = \left(\frac{\phi_q}{\phi_q^0}\right)^r \left(\frac{\phi_r^0}{\phi_r}\right)^q$$

up to an ambiguity of $(r-q)$, i.e., a rotation ambiguity of $2\pi/(r-q)$ rad. It is worth noting that, if the selected shape has a degree of symmetry ν (with $\nu = 1$ for nonsymmetric cases), $r = q + \nu$ is a valid choice, and thus a 'de facto' unique solution for angle ρ can be found by using this method. As explained in Sect. 7.2.1, we use angle ρ in place of other rotation-related quantities, such as main orientation α, for what concerns rotation control about the camera optical axis $\mathbf{Z}_C$.

B.2 Shape classification

With the sole exception of part p_8 (KUKA logo), the tags on the parts have exactly the same shape of the corresponding holes (Figs. 7.4(a–h)). Therefore, we can collect all the shapes to be recognized in a set of 9 classes

$$\mathfrak{S} = \{tri,\, oct,\, cir,\, s15,\, s16,\, dlr,\, pap,\, kk1,\, kk2\}, \tag{B.7}$$

where the last two classes describe the distinct KUKA part/hole. The goal of the classifier algorithm is to decide for one of these classes given as input a feature vector $\boldsymbol{\psi}$ describing a candidate shape. We chose to consider $n_\psi = 33$ different descriptors $|Q_k|$ for the classification task, and to define vector $\boldsymbol{\psi}$ as

$$\boldsymbol{\psi} = [|Q_{-18}| \ldots |Q_{-2}|\, |Q_2| \ldots |Q_{17}|]^T \in \mathbb{R}^{n_\psi}.$$

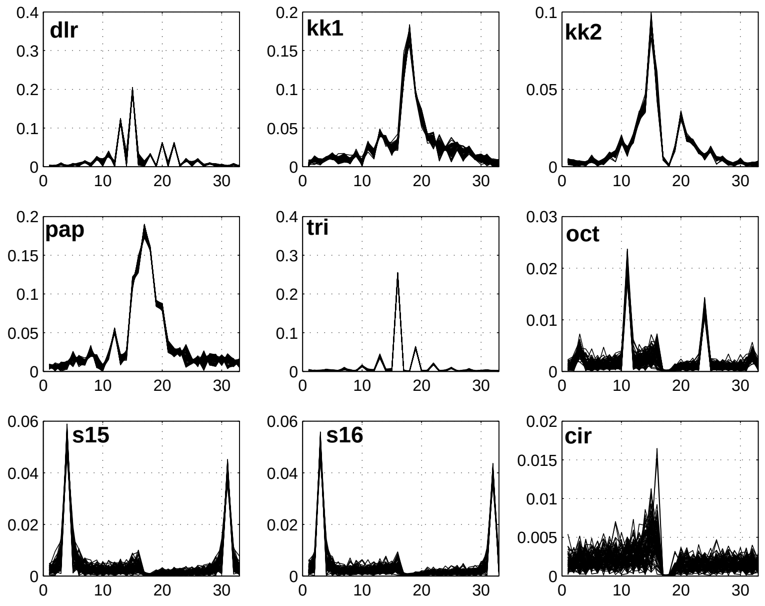


Figure B.1: Superimposed values of vector $\boldsymbol{\psi}$ for each shape and for each sample $i = 1 \dots N_i$. Despite the different camera points of view, low variation is experienced, and specific patterns arise from class to class. Therefore, the features included in $\boldsymbol{\psi}$ are suitable for being used in a classification step.

Now assume that, for a class $i \in \mathfrak{S}$, N_i distinct observations are taken from different camera views, resulting in N_i feature vectors $\boldsymbol{\psi}_i$. Let the $n_\psi \times N_i$ matrix $\boldsymbol{\psi}^{[N_i]} = [\boldsymbol{\psi}_1 \dots \boldsymbol{\psi}_{N_i}]$ be the collection of such feature vectors. Because of the invariant properties of Fourier descriptors $|Q_k|$, we expect the columns of $\boldsymbol{\psi}^{[N_i]}$ to match almost completely, with the only discrepancies due to noise or lack of validity of the affine projection model. Figure B.1 shows the superimposed values of the columns of $\boldsymbol{\psi}^{[N_i]}$, with $N_i = 80$, for all the classes in $\mathfrak{S}$. From these plots, we can verify the low variation of individual feature elements within each class despite noise and different viewing angles (robustness), and the presence of specific patterns from class to class (selectivity). These preliminary considerations motivated us to use a set of linear discriminant functions, obtained by the so-called MSE (minimum squared error) approach [Duda *et al.* 2000], in order to solve the classification problem. This method is optimal in a least-square sense and requires less computational effort than more sophisticated techniques.

The idea behind the MSE approach is the following: for each class $i \in \mathfrak{S}$, look for a weight vector $\boldsymbol{\theta}_i$ and a weight scalar β_i which are least-square solution of the linear system

$$\begin{cases} \boldsymbol{\theta}_i^T \boldsymbol{\psi}_i + \beta_i & = & 1 \\ \boldsymbol{\theta}_i^T \boldsymbol{\psi}_j + \beta_i & = & 0, \quad j \neq i, \end{cases} \tag{B.8}$$

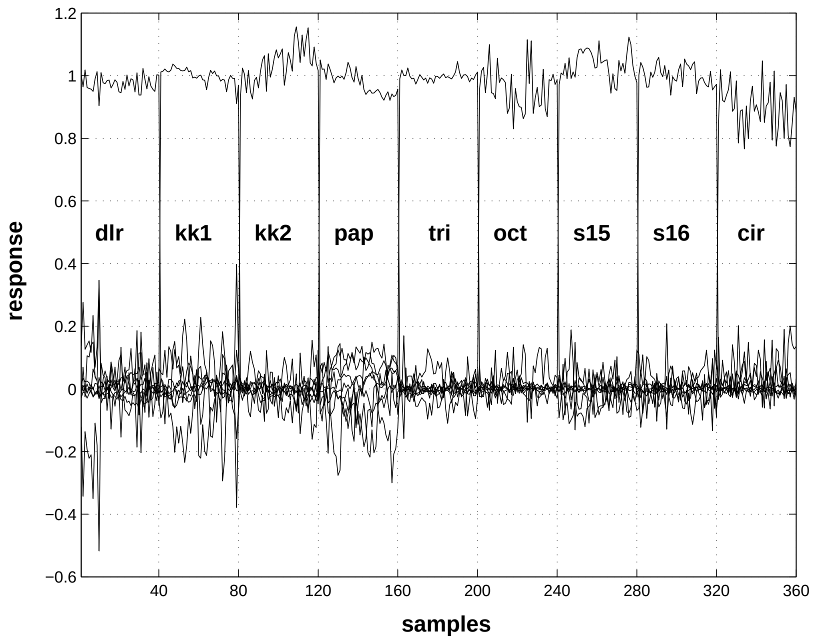


Figure B.2: MSE approach. The response to a set of 360 testing samples shows that the chosen classification algorithm is able to robustly discriminate among the shapes in $\mathfrak{S}$.

or, in other words, try to divide the set of all feature vectors $\boldsymbol{\psi}$ in two subsets: those belonging to class i, and those belonging to any other class. Assume, as before, that, for every class i, N_i observations are available for a total of $N = \sum_{i=1}^{9} N_i$ samples. By changing to homogeneous coordinates, we define the $(n_\psi + 1) \times N$ 'feature matrix'

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}^{[N_1]} & \cdots & \boldsymbol{\psi}^{[N_9]} \\ 1 & \cdots & 1 \end{bmatrix}$$

and the $9 \times N$ 'class matrix'

$$\boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{\gamma}^{[N_1]} & \cdots & \boldsymbol{\gamma}^{[N_9]} \end{bmatrix},$$

where $\boldsymbol{\gamma}^{[N_i]} = \underbrace{[\boldsymbol{\gamma}_i \cdots \boldsymbol{\gamma}_i]}_{N_i}$, and $\boldsymbol{\gamma}_i \in \mathbb{R}^9$ being a vector representing class i with a 1 in position i and zeros in all the other components. Hence, when considering all the N observation, system (B.8) can be rearranged in a compact form as

$$\boldsymbol{\Upsilon}\boldsymbol{\Psi} = \boldsymbol{\Xi} \tag{B.9}$$

where the $9 \times (n_\lambda + 1)$ matrix $\boldsymbol{\Upsilon}$, defined as

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{\theta}_1^T & \beta_1 \\ \vdots & \vdots \\ \boldsymbol{\theta}_9^T & \beta_9 \end{bmatrix},$$

is called *weight matrix*. Least square solution of (B.9) can be found as $\boldsymbol{\Upsilon} = \boldsymbol{\Xi}\boldsymbol{\Psi}^\dagger$ where the pseudoinverse of $\boldsymbol{\Psi}$ is used. After this step, done preliminarily on a set of suitable samples, any subsequent observation $\boldsymbol{\psi}$ can be directly tested through the weight matrix as $\boldsymbol{\Upsilon}\boldsymbol{\psi} = \widehat{\boldsymbol{\gamma}}$. If feature vector $\boldsymbol{\psi}$ belongs to class i, the output vector $\widehat{\boldsymbol{\gamma}}$ will (most likely) have its i-th component close to 1 and the others close to zero. Therefore, class i can be decided through inspection of $\widehat{\boldsymbol{\gamma}}$.

In order to assess the performance of the linear MSE approach, we tested it against $N_i = 40$ independent feature sets for every class i, for a total of 360 samples collected by varying continuously the camera pose.

By a suitable discriminant analysis, a subset of 10 significant features was identified among the components of vector $\boldsymbol{\psi}$. Figure B.2 shows the responses of the 9 linear discriminant functions (the components of $\widehat{\boldsymbol{\gamma}}$) to the set of $9 \times N_i$ test samples, each consisting of the selected 10 features. As expected, for each class there is only one component close to 1 with the others close to zero, so that class membership decision can be unambiguously taken.

References

Allotta, B. & Fioravanti, D. (2005). 3D motion planning for image-based visual servoing tasks. In: *Proc. 2005 IEEE Int. Conf. on Robotics and Automation*. 69, 160

Aloimonos, J. Y. (1990). Perspective approximations. *Image and Vision Computing* **8**(3), 179–192. 45

Arai, T. (1996). Robots with integrated locomotion and manipulation and their future. In: *Proc. 1996 IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*. 23

Arbter, K. (1989). Affine-invariant Fourier descriptors. In: *From Pixels to Feature* (Simon, J. C., ed.). Elsevier Science Publishers B.V. 168

Arbter, K. (1990). *Affininvariante Fourierdeskriptoren ebener Kurven*. Dissertation, Techn. Univ. Hamburg-Harburg. 168

Basri, R. (1996). Paraperspective ≡ affine. *Int. J. of Computer Vision* **19**(2), 169–179. 45

Basri, R., Rivlin, E. & Shimshoni, I. (1998). Visual homing: Surfing on the epipoles. In: *Proc. 1998 Int. Conf. on Computer Vision*. 160

Bayle, B., Fourquet, J.-Y. & Renaud, M. (2001). Génération des mouvements des manipulateurs mobiles: Etat de l'art et perspectives. *J. Européen des Systémes Automatisés* **35**(6), 809–845. 23, 29

Benhimane, S. & Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. In: *Proc. 2004 IEEE/RSJ Int. Conf. on Intelligent Robots Systems*. 56, 97

Birchfield, S. (2007). KLT: An implementation of the kanade-lucas-tomasi feature tracker. URL `http://www.ces.clemson.edu/~stb/klt`. 63

Born, M. & Wolf, E. (1999). *Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press. 45, 49

Bouguet, J.-Y. (2007). Camera calibration toolbox for matlab. URL http://www.vision.caltech.edu/bouguetj/calib_doc. 51, 136

Brockett, R. W. (1983). Asymptotic stability and feedback stabilization. In: *Differential Geometric Control Theory* (Brockett, R. W., Millman, R. S. & Sussmann, H. J., eds.). Birkhauser, pp. 181–191. 159

Campion, G., Bastin, G. & D'Andrea-Novel, B. (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Trans. on Robotics and Automation* **12**(1), 47–62. 23

Cervera, E., Pobil, A. D., Berry, F. & Martinet, P. (2003). Improved image-based visual servoing with three-dimensional features. *Int. J. of Robotics Research* **22**, 821–839. 69

Chaumette, F. (1990). *La relation vision-commande: Théorie et application à des tâches robotiques*. Ph.D. thesis, University of Rennes I. 64

Chaumette, F. (1998). Potential problems of stability and convergence in image-based and position-based visual servoing. In: *The Confluence of Vision and Control* (Kriegman, D., Hager, G. & Morse, A., eds.), vol. 237 of *LNCIS*. Springer Verlag. 63, 66

Chaumette, F. (2004). Image moments: A general and useful set of features for visual servoing. *IEEE Trans. on Robotics and Automation* **20**(4), 713–723. 62, 65, 97, 99, 133, 144

Chaumette, F., Boukir, S., Bouthemy, P. & Juvin, D. (1996). Structure from controlled motion. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **18**(5), 492–504. 89

Chaumette, F. & Hutchinson, S. (2006a). Visual servo control. I. Basic approaches. *IEEE Robotics & Automation Mag.* **13**(4), 82–90. 58

Chaumette, F. & Hutchinson, S. (2006b). Visual servo control. II. Advanced approaches. *IEEE Robotics & Automation Mag.* **14**(1), 109–118. 58

Chaumette, F. & Marchand, E. (2001). A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. *IEEE Trans. on Robotics and Automation* **17**(5), 719–730. 71

Chen, J., Dawson, D. M., Dixon, W. E. & Behal, A. (2005). Adaptive homography-based visual servo tracking for a fixed camera configuration with a camera-in-hand extension. *IEEE Trans. on Control Systems Technology* **13**(5), 814–825. 70

Chesi, G. & Hung, Y. S. (2007). Visual servoing: a global path-planning approach. In: *Proc. 2007 IEEE Int. Conf. on Robotics and Automation*. 69, 160

Chesi, G., Malis, E. & Cipolla, R. (2000). Automatic segmentation and matching of planar contours for visual servoing. In: *Proc. 2000 IEEE Int. Conf. on Robotics and Automation*. 56, 97

Chiacchio, P., Chiaverini, S., Sciavicco, L. & Siciliano, B. (1991). Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *Int. J. of Robotics Research* **10**, 410–425. 28, 29, 33, 34

Conticelli, F., Allotta, B. & Khosla, P. K. (1999). Image-based visual servoing of nonholonomic mobile robots. In: *Proc. of the 38th Conf. on Decision and Control*. 89, 159

Corke, P. I. (1994). Visual control of robot manipulators - a review. *Visual Servoing (K.Hashimoto, Ed.)* , 1–31. 58

Corke, P. I. (1997). *Visual Control of Robots: High-Performance Visual Servoing*. John Wiley & Sons. 58

Corke, P. I. & Hutchinson, S. A. (2001). A new partitioned approach to image-based visual servo control. *IEEE Trans. on Robotics and Automation* **17**(4), 507–515. 60, 71

Cyberbotics (2007). Webots. URL http://www.cyberbotics.com. 74

Davison, A. J., Reid, I. D., Molton, N. D. & Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **29**(6), 1052–1067. 160

De Luca, A., Ferri, M., Oriolo, G. & Robuffo Giordano, P. (2008a). Visual servoing with exploitation of redundancy: An experimental study. *Submitted to the 2008 IEEE Int. Conf. on Robotics and Automation* . 17, 116

De Luca, A., Lanari, L. & Oriolo, G. (1992). Control of redundant robots on cyclic trajectories. In: *Proc. 1992 IEEE Int. Conf. on Robotics and Automation*. 28

De Luca, A. & Oriolo, G. (1991). The reduced gradient method for solving redundancy in robot arms. *Robotersysteme* **7**(2), 117–122. 32, 33

De Luca, A., Oriolo, G. & Robuffo Giordano, P. (2006). Kinematic modeling and redundancy resolution for nonholonomic mobile robots. In: *Proc. 2006 IEEE Int. Conf. on Robotics and Automation*. 15, 29

De Luca, A., Oriolo, G. & Robuffo Giordano, P. (2007a). Image-based visual servoing schemes for nonholonomic mobile manipulators. *Robotica* **25**(2), 131–145. 15, 16, 29, 60

De Luca, A., Oriolo, G. & Robuffo Giordano, P. (2007b). On-line estimation of feature depth for image-based visual servoing schemes. In: *Proc. 2007 IEEE Int. Conf. on Robotics and Automation*. 16, 86, 108

De Luca, A., Oriolo, G. & Robuffo Giordano, P. (2008b). Feature depth observation for image-based visual servoing: Theory and experiments. *Submitted to the Int. J. of Robotics Research* . 17, 123

Dementhon, D. & Davis, L. (1995). Model-based object pose in 25 lines of code. *Int. J. of Computer Vision* **15**(1–2), 123–141. 60

Denavit, J. & Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *ASME J. of Applied Mechanisms* **22**(2), 215–221. 138

Duda, R. O., Hart, P. E. & Stork, D. G. (2000). *Pattern Classification*. Wiley-Interscience. 58, 171

Espiau, B. (1993). Effect of camera calibration errors on visual servoing in robotics. In: *Proc. 3rd Int. Symp. on Experimental Robotics*. 60

Espiau, B., Chaumette, F. & Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation* **8**(3), 313–326. 58, 62, 64, 65, 99

Fang, Y., Dixon, W. E., Dawson, D. M. & Chawda, P. (2005). Homography-based visual servo regulation of mobile robots. *IEEE Trans. on Systems, Man, and Cybernetics, Part B* **35**(5), 1041–1050. 159

Faugeras, O. (1993). *Three-Dimensional Computer Vision*. MIT Press. 11, 52

Faugeras, O. & Luong, Q.-T. (2001). *The Geometry of Multiple Images*. MIT Press. 52

Feddema, J. T. & Mitchell, O. R. (1989). Vision-guided servoing with feature-based trajectory generation. *IEEE Trans. on Robotics and Automation* **5**(6), 691–700. 60

Felleman, D. J. & Essen, D. C. V. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex* **1**(1), 1–47. 11

Fenske, A. (1993). *Affin-invariante Fourierdeskriptoren für Grauwertmuster*. Ph.D. thesis, Technische Universität Hamburg-Harburg. 170

Fourquet, J.-Y., Bayle, B. & Renaud, M. (2003). Manipulability of wheeled mobile manipulators: Application to motion generation. *Int. J. of Robotics Research* **22**(7-8), 565–581. 23

Friedland, B. (1986). *Control System Design: An Introduction to State-Space Methods*. McGraw-Hill. 123

Gardner, J. F. & Velinsky, S. A. (2000). Kinematics of mobile manipulators and implications for design. *J. of Robotic Systems* **17**(6), 309–320. 23

Gelb, A. (1974). *Applied Optimal Estimation*. MIT Press. 86

Geyer, C. & Daniilidis, K. (2001). Catadioptric projective geometry. *Int. J. of Computer Vision* **45**(3), 223–243. 45

Gonzalez, R. & Woods, R. (2002). *Digital Image Processing*. Prentice Hall. 58

Hill, J. & Park, T. (1979). Real time control of a robot with a mobile camera. In: *Proc. 1979 9th ISIR*. 58

Horn, B. (1986). *Robot Vision*. MIT Press. 45

Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IEEE Trans. on Information Theory* **8**(2), 179–187. 64

Huang, T. & Faugeras, O. (1989). Some properties of the E matrix in two-view motion estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **11**(12), 1310–1312. 54

Hutchinson, S., Hager, G. D. & Corke, P. I. (1996). A tutorial on visual servo control. *IEEE Trans. on Robotics and Automation* **12**(5), 651–670. 58

Inaba, H., Yoshida, A., Abdursul, R. & Ghosh, B. K. (2000). Observability of perspective dynamical systems. In: *Proc. of the 39th IEEE Conf. on Decision and Control*. 89

Intel (2007). Open source computer vision library (OpenCV). URL http://www.intel.com/technology/computing/opencv/index.htm. 63

Isidori, A. (1995). *Nonlinear Control Systems*. Springer-Verlag New York, Inc. 163

Khalil, H. K. (2002). *Nonlinear Systems*. Prentice-Hall, 3rd ed. 67, 92

Khatib, O. (1991). Cooperative manipulation in mobile robotic systems. In: *Proc. 1991 IEEE Int. Conf. on Robotics and Automation*. 23

Koenderink, J. J. & van Doorn, A. J. (1991). Affine structure from motion. *J. of Optical Society of America* **8**(2), 337–385. 45

Lamiraux, F., Bayle, B., Fourquet, J.-Y. & Renaud, M. (2002). Kinematic control of wheeled mobile manipulators. In: *Proc. 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 23, 29

Lapresté, J. T., Jurie, F., Dhome, M. & Chaumette, F. (2004). An efficient method to compute the inverse jacobian matrix in visual servoing. In: *Proc. 2004 IEEE Int. Conf. on Robotics and Automation*. 62

Laumond, J. P. (1998). *Robot Motion Planning and Control*. Springer-Verlag New York, Inc. 23, 159

Lavest, J., Rives, G. & Dhome, M. (1993). Three-dimensional reconstruction by zooming. *IEEE Trans. on Robotics and Automation* **9**(2), 196–207. 45

Lemaire, T., Berger, C., Jung, I.-K. & Lacroix, S. (2007). Vision-based slam: Stereo and monocular approaches. *Int. J. of Computer Vision* **74**(3), 343–364. 160

Li, Z. & Canny, J. F. (1993). *Nonholonomic Motion Planning*. Kluwer Academic. 23

Lippiello, V., Siciliano, B. & Villani, L. (2007). Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration. *IEEE Trans. on Robotics* **23**(1), 73–86. 159

Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature* **293**, 133–135. 54

Lowe, D. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence* **31**(3), 355–395. 60

Lucas, B. D. (1984). *Generalized Image Matching by the Method of Differences*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. 63

Lucas, B. D. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In: *Proc. 7th Int. Joint Conference on Artificial Intelligence*. 63

Luenberger, D. G. (1971). An introduction to observers. *IEEE Trans. on Automatic Control* **16**(6), 596–602. 87

Luenberger, D. G. (1984). *Linear and Nonlinear Programming*. Addison-Wesley. 31

Ma, Y., Kosecka, J. & Sastry, S. S. (2000). Linear differential algorithm for motion recovery: A geometric approach. *Int. J. of Computer Vision* **36**(1), 71–89. 88, 160

Ma, Y., Soatto, S., Košecká, J. & Sastry, S. S. (2004). *An Invitation to 3-D Vision*. Springer–Verlag New York. 46, 52

Maciejewski, A. A. & Klein, C. A. (1989). The singular value decomposition: Computation and applications to robotics. *Int. J. of Robotics Research* **8**(6), 63–79. 31

Malis, E. (2004). Improving vision-based control using efficient secondorder minimization techniques. In: *Proc. 2004 IEEE Int. Conf. on Robotics and Automation*. 67, 69

Malis, E., Benhimane, S., Mei, C., Silveira, G., Comport, A. & Vertut, B. (2004). ESM Visual Tracking. http://esm.gforge.inria.fr/ESM.html. URL `http://esm.gforge.inria.fr/ESM.html`. 56, 97

Malis, E. & Chaumette, F. (2000). 2-1/2-D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *Int. J. of Computer Vision* **37**(1), 79–97. 69

Malis, E. & Chaumette, F. (2002). Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods. *IEEE Trans. on Robotics and Automation* **18**(2), 176–186. 60, 69

Malis, E., Chaumette, F. & Boudet, S. (1999). 2-1/2-D visual servoing. *IEEE Trans. on Robotics and Automation* **15**(2), 238–250. 60, 61, 69

Malis, E., Chesi, G. & Cipolla, R. (2003). 2-1/2-D visual servoing with respect to planar contours having complex and unknown shapes. *Int. J. of Robotics Research* **22**(10–11), 841–854. 69

Malis, E. & Rives, P. (2003). Robustness of image-based visual servoing with respect to depth distribution errors. In: *Proc. 2003 IEEE Int. Conf. on Robotics and Automation*. 66

Mansard, N. & Chaumette, F. (2007). Task sequencing for high-level sensor-based control. *IEEE Trans. on Robotics* **23**(1), 60–72. 36

Marino, R. & Tomei, P. (1995). *Nonlinear Control Design: Geometric, Adaptive and Robust*. Prentice Hall, London. 87

Mariottini, G. & Prattichizzo, D. (2007). Image-based visual servoing with central catadioptric camera. *Int. J. of Robotics Research* In press. 45

Mariottini, G. L., Oriolo, G. & Prattichizzo, D. (2007). Epipole-based visual servoing for nonholonomic mobile robots. To appear in *IEEE Trans. on Robotics* . 71

Mathworks, T. (2007). Matlab. URL http://www.mathworks.com. 102

Matthies, L., Szelinski, R. & Kanade, T. (1989). Kalman filter-based algorithms for estimating depth from image sequences. *Int. J. of Computer Vision* **3**(3), 209–236. 89, 102

Maybank, S. (1992). *Theory of Reconstruction from Image Motion*. Springer-Verlag. 52

Maybeck, P. S. (1979). *Stochastic models, estimation, and control*. Academic press. 86

Mezouar, Y. & Chaumette, F. (2002). Path planning for robust image-based control. *IEEE Trans. on Robotics and Automation* **18**(4), 534–549. 69, 160

Michel, H. & Rives, P. (1993). Singularities in the determination of the situation of a robot effector from the perspective view of 3 points. Tech. Rep. RR-1850, INRIA. 68

Morel, G., Malis, E. & Boudet, S. (1998). Impedance based coimbination of visual and force control. In: *Proc. 1998 IEEE Int. Conf. on Robotics and Automation*. 159

Mukundan, R. & Ramakrishane, K. R. (1998). *Moment Functions in Image Analysis: Theory and Applications*. World Scientific. 64

Mundy, J. L. & Zisserman, A. (1992). *Geometric Invariance in Computer Vision*. MIT Press. 45

Murray, R. M., Li, Z. & Sastry, S. S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press. 22, 24, 26, 46, 163

Nakamura, Y. (1991). *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley. 23, 28, 29, 31, 34

Nenchev, D. N., Umetani, Y. & Yoshida, K. (1992). Analysis of a redundant free-flying spacecraft/manipulator system. *IEEE Trans. on Robotics and Automation* **8**(1), 1–6. 23

Oriolo, G. & Mongillo, C. (2005). Motion planning for mobile manipulators along given end-effector paths. In: *Proc. 2005 IEEE Int. Conf. on Robotics and Automation.* 30

Piepmeier, J. A., McMurray, G. V., & Lipkin, H. (2004). Uncalibrated dynamic visual servoing. *IEEE Trans. on Robotics and Automation* **20**(1), 143–147. 62

Pin, F. G., Morgansen, K. A., Tulloch, F. A., Hacker, C. J. & Gower, K. B. (1996). Motion planning for mobile manipulators with a non-holonomic constraint using the FSP method. *J. of Robotic Systems* **13**(11), 723–736. 23

Prats, M., Sanz, P. J. & del Pobil, A. R. (2005). Model-based tracking and hybrid force/vision control for the uji librarian robot. In: *Proc. 2005 IEEE/RSJ Int. Conf. on Robots and Intelligent Systems.* 159

Remazeilles, A. & Chaumette, F. (2007). Image-based robot navigation from an image memory. *Robotics and Autonomous Systems* **55**(4), 345–356. 160

Remazeilles, A., Chaumette, F. & Gros, P. (2004). Robot motion control from a visual memory. In: *Proc. 2004 IEEE Int. Conf. on Robotics and Automation.* 160

Robuffo Giordano, P., De Luca, A. & Oriolo, G. (2008). 3D structure identification from image moments. *Submitted to the 2008 IEEE Int. Conf. on Robotics and Automation* . 16, 86

Robuffo Giordano, P., Stemmer, A., Arbter, K. & Albu-Schäffer, A. (2008). Visual and force-torque controlled assembly of complex planar parts. *Submitted to the 2008 IEEE Int. Conf. on Robotics and Automation* . 17, 138

Russ, J. C. (1998). *The Image Processing Handbook.* CRC Press. 58

Sanderson, A. C. & Weiss, L. (1983). Adaptive visual servo control of robots. *Robot Vision* , 107–116. 58

Sanderson, A. C. & Weiss, L. E. (1980). Image based visual servo control using relational graph error signal. In: *Proc 1980 Int. Conf. on Cybernetics and Society.* 58

Schramm, F., Geffard, F., Morel, G. & Micaelli, A. (2007). Calibration free image point path planning simultaneously ensuring visibility and controlling camera paths. In: *Proc. 2007 IEEE Int. Conf. on Robotics and Automation.* 69

Sciavicco, L. & Siciliano, B. (2000). *Modelling and Control of Robot Manipulators.* Springer. 22, 26, 30, 71

Sepp, W., Fuchs, S. & Arbter, K. (2005). DLR Calibration Detection Toolbox (CalDe 0.99.0). URL http://www.dlr.de/rm/callab. 51, 149

Seraji, H. (1993a). Motion control of mobile manipulators. In: *Proc. 1993 IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*. 23

Seraji, H. (1993b). An on-line approach to coordinated mobility and manipulation. In: *Proc. 1993 IEEE Int. Conf. on Robotics and Automation*, vol. 1. 23

Seraji, H. (1998). A unified approach to motion control of mobile manipulators. *Int. J. of Robotics Research* **17**(2), 107–118. 23, 29, 31

Shamir, T. & Yomdin, Y. (1988). Repeatability of redundant manipulators: Mathematical solution of the problem. *IEEE Trans. on Automatic Control* **33**(11), 1004–1009. 28

Silveira, G., Malis, E. & Rives, P. (2007). An efficient direct method for improving visual slam. In: *Proc. 2007 IEEE Int. Conf. on Robotics and Automation*. 160

Smith, C. E. & Papanikolopoulos, N. P. (1994). Computation of shape through controlled active exploration. In: *Proc. 1994 IEEE Int. Conf. on Robotics and Automation*. 89, 102

Soatto, S. (1994). Observability/identifiability of rigid motion under perspective projection. In: *Proc. of the 33th IEEE Conf. on Decision and Control*. 88

Soatto, S., Frezza, R. & Perona, P. (1996). Motion estimation via dynamic vision. *IEEE Transaction on Automatic Control* **41**(3), 393–413. 88, 160

Stemmer, A., Albu-Schäffer, A. & Hirzinger, G. (2007). An analytical method for the planning of robust assembly tasks of complex shaped planar parts. *Proc. 2007 IEEE Int. Conf. of Robotics and Automation* , 317–323. 153

Strobl, K. H. & Hirzinger, G. (2006). Optimal Hand-Eye Calibration. In: *Proc. IEEE/RSJ Int. Conf. on Robots and Intelligent Systems*. Beijing, China. 72

Strobl, K. H. & Paredes, C. (2005). DLR Calibration Laboratory (CalLab 0.99.5). URL http://www.dlr.de/rm/callab. 51, 149

Stroebel, L. (1999). *View Camera Technique*. Focal Press. 45, 49

Sturm, P. F. & Maybank, S. J. (1999). On plane-based camera calibration: A general algorithm, singularities, applications. In: *Proc. 1999 IEEE Conf. on Computer Vision and Pattern Recognition*. 51

Tahri, O. & Chaumette, F. (2005). Point-based and region-based image moments for visual servoing of planar objects. *IEEE Trans. on Robotics* **21**(6), 1116–1127. 62, 65, 130, 133

Taylor, C., Ostrowski, J. & Jung, S. (2000). Robust vision-based pose control. In: *Proc. 2000 IEEE Int. Conf. on Robotics and Automation*. 59

Tomasi, C. & Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *Int. J. of Computer Vision* **9**(2), 137–154. 45

Tsai, R. Y. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE J. of Robotics and Automation* **3**(4), 323–344. 50, 51

Weiss, L., Sanderson, A. C. & Neuman, C. P. (1987). Dynamic sensor-based control of robots with visual feedback. *IEEE J. on Robotics and Automation* **3**(5), 404–417. 58, 60

Weng, J., Cohen, P. & Herniou, M. (1992). Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **14**(10), 965–980. 51

Wijesoma, S. W., Wolfe, D. F. H. & Richards, R. J. (1993). Eye-to-hand coordination for vision-guided robot control applications. *Int. J. of Robotics Research* **12**(1), 65–78. 58

Wilson, W. J., Hulls, C. C. W. & Bell, G. S. (1996). Relative end-effector control using cartesian position based visual servoing. *IEEE Trans. on Robotics and Automation* **12**(5), 684–696. 59

Yamamoto, Y. & Yun, X. (1999). Unified analysis on mobility and manipulability of mobile manipulators. In: *Proc. 1999 IEEE Int. Conf. on Robotics and Automation*. 23

Young, I. T., Gerbrands, J. J. & van Vliet, L. J. (1998). *Fundamentals of Image Processing*. Delft PH. 58

Zak, M. (1989). Terminal attractors in neural networks. *Neural Networks* **2**, 259–274. 35

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **22**(11). 50, 51