TÉCNICO LISBOA

Mobile Arm Visual Servoing for Object Manipulation

Soraia Mendes Ferreira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Doutor Pedro Daniel Dos Santos Miraldo
Prof. Doutor Rodrigo Martins de Matos Ventura
Mestre André Gonçalves Mateus

Examination Committee

Chairperson: Prof. Doutor João Fernando Cardoso Silva Sequeira
Supervisor: Doutor Pedro Daniel Dos Santos Miraldo
Member of the Committee: Prof. Doutor José António Da Cruz Pinto Gaspar

October 2017

Mobile Arm Visual Servoing for Object Manipulation

Soraia Mendes Ferreira

October 16, 2017

Acknowledgements

First of all, I would like to thank my supervisors, Doutor Pedro Miraldo and Professor Doutor Rodrigo Ventura, for all the guidance and knowledge transmitted, and also for the opportunity to achieve my goals throughout this journey.

A special thank to André Mateus that helped and contributed to the concretisation of this work.

I would like to thank all the members of SocRob@Home team, for giving me the opportunity to collaborate with them.

To all my colleagues from the 8th floor laboratory that somehow helped me during this journey. To all my friends who were always present, thank you for the encouragement and support.

To my parents, for always being there for me and making it possible for me to achieve my goals, for all the faith that they placed in me, and for all the support they gave me throughout this long and exhausting phase. To my little sister, who motivated me during this journey. To my grandparents José and Rosa, and my aunt Anita for all the affection and patience.

Last, but definitely not least, a special thanks to my boyfriend, Fábio, who supported me every single day along this journey. His constant encouragement, specially in the demotivating moments, gave me the strength to pursue and accomplish this work.

Abstract

Objects manipulation by a robotic arm inserted in mobile platforms, in unstructured environments, such as domestic environments, is a particularly challenging subject. Currently, the approach is to divide this question in two parts. The first one consists in implementing a plan trajectory to reach the object, avoiding any collision between the robot and other objects. The second one, is executing the defined trajectory in the environment. However, modification of the target position or obstacles may invalidate the planed trajectory. Thus, developing a closed loop control method, with constant feedback from the environment, to detect the targeted object, and potential obstacles, is essential. To this aim, the dynamic and non structured environment is perceived through vision sensing. We actuate in both joint space of the arm, and in the velocities of the mobile platform. We intend to extend the actual known real time navigation methods, applied to moving robots, to control the arm, attached to the mobile platform, to manipulate static objects, using vision.

This document presents the state-of-the-art of methods developed in this field and proposes a solution for the mobile manipulator control problem. The main goal is for the mobile manipulator to reach an adequate position that enable the manipulation of the targeted object in the world. To test and to analyse possible solutions, a simulator in Matlab is developed. Finally, the purpose of this work is to improve robots performance in manipulation during robotics competitions, like RoboCup, with the SocRob@Home team and ISR group.

Keywords: object manipulation, robotic arm, visual servoing, mobile manipulator.

Resumo

A manipulação de objetos usando um braço robótico inserido numa plataforma móvel, em ambientes não estruturados, tal como os ambientes domesticos, é um tema particularmente desafiante. Esta questão é atualmente abordada dividindo-se, numa primeira parte, no planeamento de uma trajetória que evite a colisão do braço com outros objetos e, numa segunda parte, na execução dessa trajetória. No entanto, a alteração da posição do objeto alvo, ou a alteração da posição de um obstáculo, invalida a trajetória planeada inicialmente. Pelo que, é necessário desenvolver um método de controlo em malha fechada, que receba feedback do ambiente em tempo real, para detetar o objeto alvo e os obstáculos. Neste trabalho, iremos usar a visão para receber a informação do ambiente dinâmicos e não estruturado, e assim atuar tanto no espaço de juntas do braço, como na movimentação da plataforma, onde este ultimo está inserirdo. Pretende-se estender os métodos de navegação em tempo real de robôs móveis, para o controlo do braço robótico, colocado na plataforma móvel, a fim de possibilitar a manipulação de objetos estáticos, em ambientes domésticos.

Este documento apresenta o estado da arte dos métodos desenvolvidos nesta área, e propõe uma abordagem ao problema de perceção e controlo do manipulador móvel. O principal objetivo é o manipulador móvel atingir uma posição adequada que lhe permita manipular o objeto alvo no mundo. Para testar e analisar possiveis soluções, foi desenvolvido um simulador em Matlab. Em particular, esta pesquisa é elaborada com vista a participar em competições de robôs domésticos como a RoboCup, através da equipa SocRob@Home do ISR (Instituto Superior de Robótica do IST).

Palavras-chave: manipulação de objetos, braço robótico, *visual servoing*,

manipulador móvel.

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

Contents

Chapter 1

Introduction

This document purposes a solution to objects manipulation by a mobile manipulator, tested in a simulator developed in MATLAB. In this chapter, we present motivations and a brief summary of the object manipulation history. To conclude this chapter, the goals of this work are stated.

1.1 Motivation

Initially, in object manipulation, the main goal of a robotic arm was to be used in industry, e.g. the automated robotic arms in assembly lines. However, recently, technological advances motivated by the man's desire to improve his daily life, had been increasing the need to adapt robotic arm object manipulation to domestic environment [9].

One of the main reason behind these needs (in domestic robotic field), is the growth of ageing population in industrialised countries, which implies an increasing need of medical and social services, as well as personal cares. Yet, those services are not always well accepted. Elderly people would prefer to live in a familiar environment, as long as possible. Furthermore, the dependency of someone else's cares also implies the decrease of self esteem and integrity. On the other side, the number of social workers is fewer than the elderly population, and the associated costs of a house care are especially high for most people [11]. In modern societies, active people spend less and less time doing household chores. Actually, during a study with 48 householders, domestic tasks most referred are time consuming

drudgery (vacuum, yard work, cocking, etc.), house-sitting, and personal attendance [12]. The study also highlights the lack of scientific understanding about the users needs and their desire to have robotic assistance at home.

Given those facts, assisting technology appears as an innovative solution. These recent technologies are defined as a set of tools capable of accomplishing tasks in domestic environments [11]. Many applications can be associated to it, going from treatment and rehabilitation of debilitated people, using long-term assistance devices, to companionship to alleviate feelings of loneliness [11]. In these cases, object manipulation is fundamental, and has to be well studied in order to be reliable. For elderly people, it has been established that the success of robotic devices depends on practical benefits. However, it depends even more on the cognitive and emotional components, that is, the way that people feels about robots [13].

On the other hand, these devices should comply with certain requirements and its functionalities must to be tested. In this perspective, several bench-marking competitions, like RockIn or RoboCup, have been created in order to assess specific features of the presented systems, and, in that way, to push technology further [14]. One of the biggest, in terms of participants and topics taken into account, is RoboCup@Home. The main goal is to keep up with domestic services progresses, in versatile robots, that can operate safely in all situations, found in a daily routine. Tests are performed in a dynamic way by evaluating simultaneous functionalities. Tab. 1.1 shows the scores of some fields analysed in RoboCup@Home [9].

Thought the table analysis, we noticed that the lowest scores are assigned to object recognition and object manipulation. Furthermore, in previous competitions the ISR robot could not grab the object. In fact (considering the current ISR robot object manipulation status), the object was recognised and the global trajectory was correctly planned, avoiding obstacles. Nevertheless, the last part of the process was not successfully executed. The arm was not correctly controlled or the vision of the object was not accurate. Hereupon, further study of the local displacement control, on the last phase of approximation, should be performed in order to improve current results.

Besides, this work can contribute to improve manipulators efficiency, if applied in modern industry. As a matter of fact, targeted objects needs to have a very

Table 1.1: Table of scores of Bench-marking adapted from [9].

Ability	2010
Navigation	33/20
Mapping	21/10
Person recognition	57/23
Person tracking	100/72
Object recognition	6/1
Object manipulation	29/8
Speech recognition	50/38
Gesture recognition	62/26
Cognition	17/3
Average	41.6/22.4

precise position to be correctly manipulated by robotic arms. This precision implies high costs related to the arms manufacture and maintenance. However, those costs can be reduced using Visual Servoing (VS) to localise the object position and move the arm according to visual feedback.

1.2 Brief history

This section presents an overview of robotics's history in the field of objects manipulation. In particular, in the field of mobile manipulators, domestic robots, and manipulation aids, mainly based on [15]. In a second part of this section, historic technical advances are exposed.

1.2.1 Robots

The object manipulation started with fixed robotic arms in industrial robots, in 1954. It was patented by George Devol in "Programmed article transfer". The first robot, Unimates (Fig. 1.1), was put into service in 1961, and begun to extract parts from die-casting machine, at General Motors plant.

However, the significant breakthrough was the 6 Degrees Of Freedom (DOF) Stanford Arm, designed as a research prototype in 1969 by Victor Scheinman. Later on, the IRB-6 combined all-electric-drives technology and a microcomputer

Figure 1.1: Robotic arm Unimates [1].

for programming and motion-control. Followed the SCARA, a four-axis low-cost manipulator with kinematic configuration allowing fast and compliant arm motions [1].

The introduction of the first microcomputer controlled robot allowed Unimation to create the PUMA (programmable universal machine for assembly), which came close to the dexterity of a human arm, when launched in 1979. However, manufactured industrial robots really took on its full meaning with the firsts assembly lines, in 1985. Then, appeared the KUKA light-weight robot (Fig. 1.2), which was the first seven axis arm design for human-robot cooperation [1]. Progresses, in this field, have been mostly guided by the needs of accomplishing tasks with very high precision, accuracy and speed. Manipulators must be strong enough do deal with very heavy objects and have to be synchronised with other arms movement, when working in assembly lines.

In terms of navigation, one of the first robots that integrated a robot arm in a mobile platform was the MORO (mobile robot, 1984, Fig. 1.3), developed at Frauenhofer IPA, Stuttgart. At the time, the platform had a wire-bound that allowed the robot to follow a wire buried in the floor. It was only in 1987, that a robot was able to combine navigation and objects manipulation. A robotic manipulator arm was mounted on a mobile platform, and the Handey robot became the first mobile manipulator. Indeed, the Handey robot was capable of recognising, choosing and grasping parts of unstructured pile of objects. Using motion planning, the grasped objects were placed on a commanded position. For that, it

Figure 1.2: KUKA light-weight robot [1]

required some inputs: it used a polyhedral world model including the objects to be manipulated and any other fixed objects in the environment [16]. Research and developments of mobile manipulators have been growing in recent years. This has enabled a broad use of automation and robotics, while simultaneously reducing costs. Nowadays, most of the industrial robots have integrated controllers, with PC-based processors, software for programming, communication, and simulation [1]. Moreover, in the past few years, vision systems for object identification, localisation, and quality control have been increasing. Mobile manipulators are capable of moving and interacting with the environment, and may be designed specifically to assist human beings in factories and, in particular, at home. The survey [17] provides a literature review of mobile manipulator research, with examples of experimental applications. Fig. 1.4 shows the time-line of mobile manipulator development [1].

Domestic robotics is a field of robotics that has been growing over the past decade. These robots are mainly used to achieve household tasks. The latest developments are mainly focused on domestic cleaning robotics, and a number of other smart applications, including robotic lawn mowing, ironing robots, and digital wardrobes [18]. Taking the example of cleaning robots, some of the technical challenges are: absolute positioning, area coverage in unknown, dynamic environments, sensor coverage for robust obstacle avoidance, error recovery, safety, and operation interface/human-robot interaction [18]. Sensor coverage is essential for safe, collision-free motion, and also for the observation of unknown parts of the

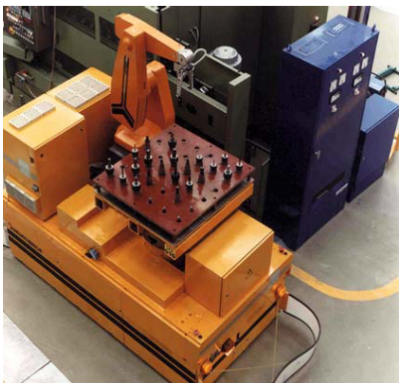


Figure 1.3: MORO robot [1]

environment. 3 Dimensional (3-D) sensor coverage is achieved by stereo-vision or by laser range finders and allows the robot to have a perception, which enables it to account for every known or unknown obstacle [18].

Nowadays, commercialised domestic robots includes, beyond cleaning robots, intelligent refrigerators, intelligent wardrobes, intelligent and connected devices for smarts home. However, these "little helpers" are coming slowly because of their expansive price and their limited capacities [18]. Furthermore, domestic robots still cannot manipulate objects in household tasks or, at least, they are not ready to be commercialised with that feature. Perhaps, the costs of such a robot is to high and its utilities are still limited.

On the other hand, manipulation aids, a category of assisting rehabilitation robotics, presents promising advances. Manipulation aids is divided in three sub-types: fixed, portable and mobile. Wheelchair manipulator arm systems are portable devices that helps users to manipulate objects, while navigating a home or a public place. One of those systems is a service manipulator (ARM, Exact Dynamics, Netherlands) that can be attached to the wheelchair's own joystick. It is particularly used by people who have muscular dystrophy, a high-level SCI, or cerebral palsy. Mobile autonomous systems with arms have a sensorimotor functionality, similar to that of a human being and serves people in performing menial physical tasks [19].

The quick overview of robots history (in particular the more recently domestic robots and manipulation aids) allows us to understand some characteristics needed

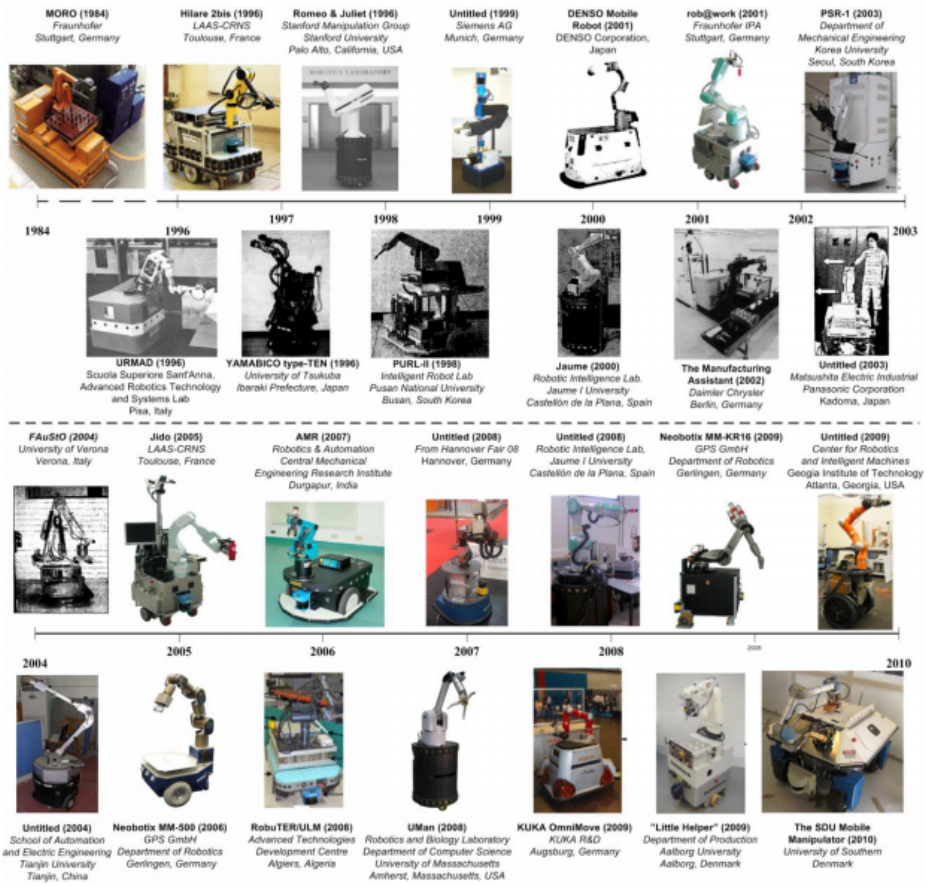


Figure 1.4: Time-line of mobile manipulator development [2]

to develop the techniques behind these achievements.

1.2.2 Techniques

In 1979, Hill and Parks denoted a technique that uses visual information in the feedback control (mainly applied to robotic manipulators) as VS [20].

In the field of image processing, one of the first applications (related to VS) was an automatic rocket tracking pan/tilt camera, i.e. a camera that was capable of remote directional and zoom control, able to keep the target in the centre of the image plane [21]. The target was first identified by a video-rate image processing hardware, which also updated the camera's orientation. The image resolution, took more and more importance as manufacturing task needed more precision.

Reducing the size of the correlation window, needed to track objects texture, was a purposed solution. A high-bandwidth planar controlled position was developed, for a micro-manipulator, by Weiss and Hollis [22], in 1989. For that purpose, they contributed to improve manipulators stability as the joint's vibrations are also reduced.

In the control field, in 1985, Weiss [23] discussed the idea of applying adaptive control to manipulators of 3-DOF. This technique established a relationship between robot pose and image features, in the Image-Based-Visual-Servoing (IBVS), when variations were not linear in time. However, problems of stability, accuracy, and tracking speed of manipulators delayed the progressions in the field. To solve these problems, Coulon and Nougaret [24] worked on a digital video processing system, using a TV camera (vidicon sensor), to find the location of the target in a processing window. This improvement also served to control the position of a XY mechanism, with a closed-loop controller.

Regarding objects manipulation, some significant progresses were also made. A 2-DOF manipulator was controlled using VS to complete the task of fruit picking [25]. In addition, ultrasonic sensors were used just before the robot reached the fruit. Thanks to the data provided by these sensors, visual information was extended, and distances during the final phase of the grasping were estimated [20]. For the task of part mating, a method of assembling of two or more component parts with mutually complementing shapes, a 2-DOF manipulator was also used. However, this time, a Jacobian approximation related the image-planed corrections to robot joint-spaced actions [26]. As the sample rate increased, the dynamic effects expanded when an eye-in-hand configuration was used. To tackle this problem, a closed loop control was used with an image-based control scheme, independent for each DOF [27].

In parallel, a real-time feature tracking and gaze controlled camera was experimented in a road vehicle guidance [28]. This achievement was followed by a great evolution in VS: catching flying objects [20]. The earlier tests consisted on catching a ball with an end-effector, assembled in a Puma 560 manipulator. For that purpose, a fixed-camera stereo-vision system was used [29]. Other achievements, like the catch of an object in space using a vision guided robot, emerged [30]. It was based on a proposed algorithm, that can be divided in two stages: 1) the

target was approached by estimating a coarse position; and 2) a more precisely tune was done, to match the robot acceleration and velocity with the targeted object [31]. Another achievement in VS, for catching moving objects, was the implementation of a tracking controller for a robot to pick items from a conveyor belt [32]. In order to do that, the camera was hand-helded and the interval update for visual information was fasten. To track an object moving at 250mm/s, the tracker speed was improved with the used of a 60 Hz fixed-camera stereo vision system. To improve this real-time tracking algorithm, a predictive filter was used in conjunction [33].

Finally, concerning IBVS, Feddema used an explicit feature-space trajectory generator and closed-loop joint control [34]. This approach was generalised, in 1991, and the trajectory became independent of target geometry [35]. 3-D tracking of static and moving objects with adaptive control was used to estimate the target distance [36]. During the same year, in a telerobotic environment (an environment where semi-autonomous robots were controlled from a distance) Humans could selected visual features as references for the task [37, 38].

More details about VS will be given in Sec. 2.1.

1.3 Problem Statement

In this section, we state the goals of this thesis. First, we define some goals to be reached in robotics competitions. Regarding object manipulation, the relevant tasks are all the tasks that involve contact between the object and the robotic hand (touching, holding, pushing, etc.). While navigating, the robot must avoid all objects coming on its path, unless the object is specified as the target. Relevant objects must be perceived and recognised by the robot, when they come in its field of view. It must be identified and localised in order to be reached [39]. These goals are evaluated through performance measurements using bench-marking methodology (based on repeatability, reproducibility, accuracy, and objectivity). The success and quality of task execution includes time of execution, quality of perception, quality of navigation and quality of manipulation [40].

This work will be focusing on the mobile manipulator control, just after recognising the object as a target and just before the arm grabs it. The main challenge

is to extend techniques used to manipulate objects, with fixed robotic arms, to an arm inserted on a moving platform, as it is the case for the Mbot robot, in the 8th floor laboratory, North tower-Campus Alameda. The Mbot robot came from the MOnarCH project (Multi-Robot Cognitive Systems Operating in Hospitals [41]), and was adapted. A Robai Cyton Gamma arm [42] was added to the Mbot body, to perform object manipulation. The path implementation and navigation avoiding obstacles, is already working on the robot. As well as the image processing needed to perceive the object, and to define it as a target. However, major improvements, in combining the movements of the body with the movements of the arm, are needed to perform object manipulation. Hereupon, we will be focusing on techniques to control the pose of the robot's end-effector, using visual information extracted from the image. The end-effector will have a camera to target the object. The goal is for the robot to reach the targeted object through the combined control between the moving platform velocities, and the arm joints velocities. To that aim, we intend to define a closed loop control feedback, using as input visual information provided by the camera (in real time). Solutions for the VS are tested in a simulator, developed in MATLAB. This simulator provides us with the robot trajectory, the position of the object represented by points in the world, and the projection of these points in the image. We can also analyse other relevant information such as velocities and features errors.

This document presents the theory behind the technique used, the research done to propose a suitable solution to our control problem, the implementation of the simulator with respective results and discussion.

1.4 Report's Outline

This report has five chapters. After this introduction (Chap. 1) in which we present the problem that we intend to solve, Chap. 2 explores the state-of-the-art in VS, namely: Image-Based, Pose-Based, and Hybrid techniques. The theoretical basis of image formation and control in robotics, in particular, the control of robot manipulators and vehicles are also explained. After what, Chap. 3 presents relevant information about Matlab, in particular about the toolbox used. The details, regarding our implementation, are divided in three important steps: the

control of the mobile platform, the control of the manipulator and, last but not least, the mobile arm control. Finally, Chap. 4 presents the simulations and the respective results are discussed. Chap. 5 concludes on this thesis and suggests some future improvements to this work.

CHAPTER 1. INTRODUCTION

Chapter 2

Theoretical fundamentals

The present chapter presents the state-of-art in terms of VS, applied to object manipulation using a robotic arm. The problem of positioning the arm is formulated as a VS problem. To do so, we first define theoretically VS. Then, we present the image formation problem, required to compute the VS. Last but not least, we explain the theory behind robotic manipulation and moving platforms.

2.1 Visual Servoing

VS relies on techniques from areas like image processing, computer vision, and control theory. Here, we present a definition of VS, Image-Based Visual Servoing (IBVS), Pose-Based Visual Servoing (PBVS), and Hybrid method. Different ways of classifying these techniques are possible. Several classifications are extensively discussed in [43]. Depending on the position of the camera (w.r.t. the robot), systems can be classified as:

- In eye-in-hand configuration: the camera is attached to the moving hand, observing the relative position of the target;
- In hand-to-eye configuration (stand-alone-camera): the camera is fixed in the world, observing the target and the motion of the hand.

In this section, we choose to classify the systems according to the space in which visual error is calculated. That is to say, if the error is calculated on the image space (2-D), we talk about IBVS. Otherwise, if it is calculated on the Cartesian

space (3-D), we talk about PBVS. When we use both methods, we talk about Hybrid method.

2.1.1 Definition

Chaumette defines VS as the control of a dynamic robotic system motion, using computer vision data for the input of the real-time closed-loop control schemes [44][10]. In other words, VS is the use of vision to extract features and to process images to control robots pose [20]. The introduction of visual feedback makes the system less sensitive to errors that could be generated by inadequate calibration, noise in the system, or changes in the environment. Indeed, contrarily to open-loop approaches, the use of visual information, acquired in real time, makes the control scheme more robust and adaptive. VS appears as an attractive technique for achieving manipulation tasks [45].

The main steps of the control scheme boil down to: 1) extract useful visual information to compute features; and 2) use them as inputs in the control loop (knowing that, the error between the current visual features and the desired value of those features is computed); and 3) the outputs (e.g. robot velocities), are calculated to reduce the error. Based on this explanation, and on [44] equations, a generalised mathematical definition of the error is given by:

$$e(t) = s(m(t), p) - d \tag{2.1}$$

In which s is a vector describing the set of visual features, depending on:

- $m(t)$, that is the computation from the image measurements, e.g. image coordinates of interest points, area, the centre of gravity, and other geometric characteristics of an object;
- p, that are other parameters like camera intrinsic parameters, the 3-D models of objects, or the pose between the camera and the environment [46].

Vector d is composed by the desired values of the features. Since we consider a static target, the desired features are constants. On the other hand, s only varies with the camera motion. The design of the vector s is discussed later in this chapter. To obtain the outputs, taking the example above, a velocity controller

can be design. For that, Chaumette, in [44], relates the time variation of s with the camera spacial velocity, v_c. So, the variation of s ($\dot{s}$) depends on the velocity of the camera frame v_c:

$$\dot{s} = L_x v_c, \tag{2.2}$$

where $v_c = (v_c, \omega_c)$ with v_c describing the instantaneous linear velocity and ω_c the instantaneous angular velocity, both relative to the origin camera frame. L_x is a k (number of visual features) by N (number of robot DOF) interaction matrix, also called features Jacobian in literature. This matrix is given in Sub. 2.1.2 and detailed in Appendix A. So, when the camera moves in space, the error changes and we can obtain the error variation by:

$$\dot{e}(t) = L_e v_c. \tag{2.3}$$

Considering L_e the interaction matrix for the error, the goal is to reduce the error. In [44], an exponential decay was introduced:

$$\dot{e}(t) = -\lambda e \tag{2.4}$$

From (2.3), the control law becomes:

$$v_c = -\lambda L_e^+ e \tag{2.5}$$

Where L_e^+ is Moore-Penrose pseudo-inverse matrix, that can be defined as:

$$L_e^+ = (L_e^T L_e)^{-1} L_e^T \tag{2.6}$$

when $L_e^T L_e$ is of full rank. Both L_e or L_e^+ are hard to be exactly extracted in real VS systems. Consequently, they have to be estimated as $\hat{L}_e$, in the control law. The pseudo-inverse of the interaction matrix can be estimated using different techniques. Two of the fundamentals approaches are IBVS and PBVS. What mainly differentiate the two approaches is the design of the visual features. In IBVS, $m(t)$ is a 2-D set of data immediately available in the image, while in PBVS $m(t)$ is a set of 3-D parameters estimated from the image measurements [44]. A third approach is the Hybrid approach, which is a combination of the two approaches.

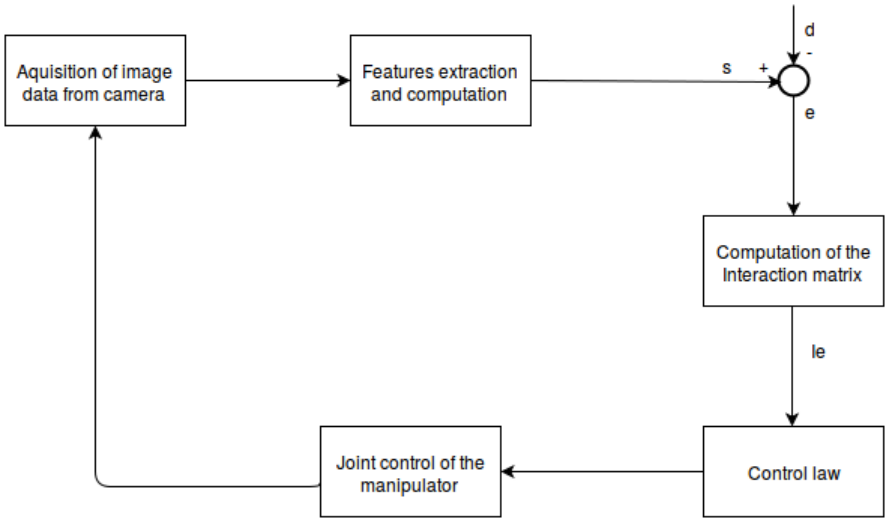


Figure 2.1: Simplified IBVS diagram

2.1.2 Image-Based-Visual-Servoing

Typically, IBVS use image measurements $m(t)$ (pixel coordinates, lines, moments of region, for example) converted to features using camera intrinsic parameters p. The error signal is measured directly from the image. The control law is based on the computed error between the set of current visual features and the desired value given by $\dot{s}$. The interaction matrix relates a 2-D point projected in the image from a 3-D point in the camera frame, by the depth of the point in the camera frame. In that way, no estimation of the targeted pose is needed [47] [10]. In IBVS systems, as it can be seen in Fig. 2.1, there is no direct control over the Cartesian velocities of the robot. Consequently, a map between image space velocities and velocities in the robot's workspace is needed. This mapping is done by the interaction matrix.

The authors of [44] construct the interaction matrix from a 3-D point coordinates (X, Y, Z). After projecting the 3-D points to the image space, imaging geometry is used to relate the velocity of the projected point, to the camera spacial velocity. Doing some mathematical manipulations (presented in appendix A), the interaction matrix L_x, related to an image point with coordinates (x, y), is obtained in (2.7), in which the depth Z appears in the coefficients of the interaction

matrix, related to the translational motions.

$$L_x = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{bmatrix} \tag{2.7}$$

This matrix has to be estimated when used in the control law, and the quantity of points chosen has to be enough to avoid singular configurations and local minima. That is to say, for a 6-DOF manipulator, there have to be at least 3 points so the interaction matrix has, at least, as many columns as the manipulator DOF. A study of several methods to approximate the iteration matrix, in IBVS, is presented in [10]. The classical method is to estimate the current depth Z of each point, at each iteration of the control scheme, using for example stereo vision. The interaction matrix is constructed as:

$$\hat{L}_e^+ = L_e^+ \tag{2.8}$$

However, another approach is to set only the desired depth Z* of each point for the desired pose, [10], choosing:

$$\hat{L}_e^+ = L_{e^*}^+ \tag{2.9}$$

where L_{e^*} is the value of L_e for the desired position $e = e^* = 0$ [44]. In [48], a vision-based robot control is formulated as a minimisation problem, leading to choose:

$$\hat{L}_e^+ = 1/2(L_e + L_{e^*})^+ \tag{2.10}$$

To see and compare the behaviour of the control system in practice, some tests are made using the three previous approaches, for the calculation of the interaction matrix in (2.8), (2.9) and (2.10). A rotational motion around the optical axis of the end-effector, with a camera located on it, is registered [44]. A good behaviour corresponds to a straight line trajectory in the image plan, without oscillations. Tab. 2.1 sums up the results obtained.

As it can be seen in Tab. 2.1, IBVS has some difficulties with motions when it comes to very large rotation. Other problems referred in literature, like problems of singularities in the Jacobian matrix, or the reach of local minima, can lead to problems in the control [49]. IBVS is also referred as a non-linear and highly coupled process [20]. A correct estimation of the interaction matrix in IBVS is important to provide stability to the control system, and avoid perturbations, when the robotic system reaches the desired pose.

Table 2.1: Performance tested for rotational motion around optical axis, for the 3 calculations of iteration matrix

	Positive results	Negative results
$\hat{L}_e^+ = L_e^+$	Satisfactory local behaviour when error is small	3-D behaviour not satisfactory, global behaviour unsatisfactory
$\hat{L}_e^+ = L_{e^*}^+$	Satisfactory local behaviour when error is small	Undesired translational motion
$\hat{L}_e^+ = 1/2(L_e + L_{e^*})^+$	Smooth trajectory, small oscillations (good performance)	No negative results registered

2.1.3 Pose-Based-Visual-Servoing

PBVS uses 3-D features instead of 2-D, as in IBVS, required to estimate the pose of the targeted object. Features are extracted from the image and used to determine the pose of the target with respect to the camera, in the Cartesian space [20]. The computation of the pose needs the intrinsic parameters of the camera and, at least, two views of the object, from different view points. That is to say, the 3-D parameters can be estimated from triangulation methods or epipolar geometry, as the current view and the image view are available [10]. Once the 3-D reconstruction of the object is established, the error is then computed in the task space and used in the control law (Fig. 2.2) [44]. The main advantage of this approach is the direct control of the camera trajectory, in Cartesian space [50].

Considering Eq. 2.1, s is now defined by the intrinsic parameters of the camera and the pose of the camera with respect to some reference coordinate frame [44].

In Fig. 2.3 and Fig. 2.4, three coordinate frames are considered: the current camera frame C, the world base frame W, and an object frame O, as reference frame of the object. A fourth coordinate system is considered in [44]: the desired camera frame, denoted as C^*. The orientation of C relative to $C*$ is given by the rotation matrix: ${}^{C^*}_{C}R$ [44]. The corresponding angle parametrisation Θ and the translation vector t, now defines s. The control law depends on how t is defined:

- If t is defined as a function of O (${}^{C}_{O}t$): $\Theta = 0$ for the desired s^* and the error

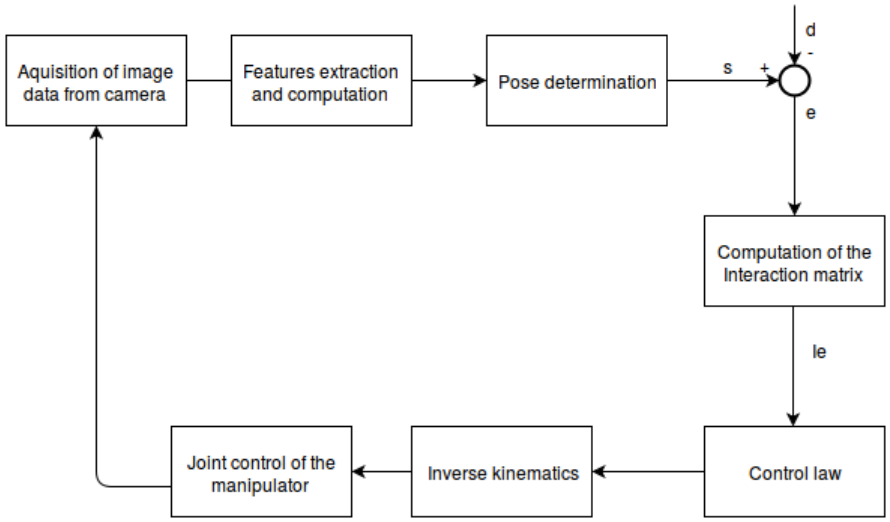


Figure 2.2: Simplified PBVS diagram

is computed between t and ${}^{C^*}_{O}t$. The interaction matrix is now:

$$L_e = \begin{bmatrix} -I_3 & [{}^{C}_{O}t]_x \\ 0 & L_\Theta \end{bmatrix} \tag{2.11}$$

in which I_3 is a 3x3 identity matrix and L_Θ is the matrix giving the information about the parametrisation of Θ, explained in [44]. Here, the control law depends on ${}^{C}_{O}t$, ${}^{C^*}_{O}t$ and Θ.

- If t is defined relative to C (${}^{C}_{C^*}t$): m^*=0 and $e = s$. The interaction matrix is now:

$$L_e = \begin{bmatrix} {}^{C^*}_{C}R & 0 \\ 0 & L_\Theta \end{bmatrix} \tag{2.12}$$

In that case, the control law depends on ${}^{C^*}_{C}R$ and ${}^{C}_{C^*}t$.

The same tests performed to evaluate IBVS are applied to PBVS. The results are shown in Tab. 2.2.

The trajectory is better followed in 3-D space than in image space due to the definition in Cartesian space. However, points can leave the image space due to the fact that the control is not directly done in the image. In PBVS, a correct estimation of the pose is crucial, because of the error regulation to 0. So, 3-D reconstruction depends directly on the calibration of the vision system and can

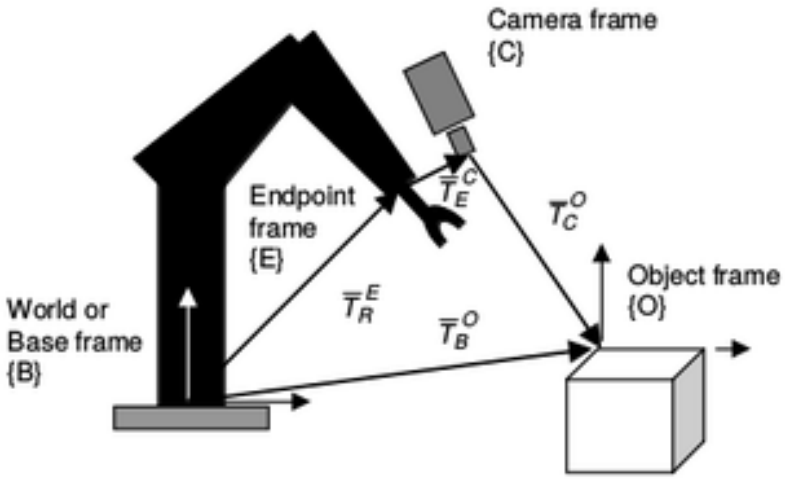


Figure 2.3: Relation between coordinates frames [3]

Table 2.2: Performance tested for rotational motion around optical axis, for the 2 definition of t

	Positive results	Negative results
${}^{C}_{O}t$	optimal object trajectory	camera trajectory not well described
${}^{C}_{C^*}t$	optimal camera trajectory	image trajectory less satisfactory, points leave the camera filed of view

lead to errors during the task execution [50]. Also, in [44], stability is analysed with the Lyapunov function, leading to the following observations:

- If the number of features chosen is equal to the number of camera degrees of freedom;
- If L_e and $\hat{L_e^+}$ have rank equal to the number of camera degree of freedom; and
- If approximation of $\hat{L_e^+}$ is not too coarse, then the global asymptotic stability of the system is ensured.

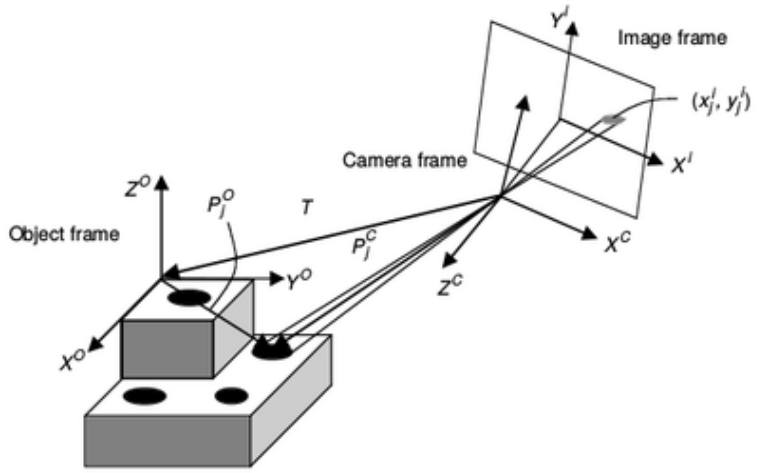


Figure 2.4: Projection of the object on the image [3]

As this approach has some major drawbacks, improvements are essential to overcome these problems, as for example, the points leaving the camera field of view. To that aim, we present another VS method in the next subsection.

2.1.4 Hybrid

A method that can deal with some performance problems, described above, is the hybrid scheme. This method consists in the combination of the 2-D (IBVS) and the 3-D (PBVS) servoing. The basic idea is to decouple the Z-axis motions (including both the translational and the rotational components) from the other DOF [49]. Then, separate controllers are implemented for these Z-axis motions. One method can control the rotational component, and the other control the translational component. There have been a few different approaches to Hybrid servoing, namely: 2-1/2-D servoing and partitioned VS [10].

The first method presented is 2-1/2-D servoing. This approach consists in a conjunction of PBVS and IBVS techniques, where IBVS is used to control the DOF related to the translational motion and PBVS provides the orientation control. In [10], authors constructs the following proposition: starting from the control law defined in (2.5), an available control for angular velocity ω_c, in PBVS method can

be defined as:

$$\omega_c = -\lambda\theta. \tag{2.13}$$

The goal is to control the several translational DOF by partitioning the interaction matrix into two matrices: L_v, matrix of linear velocities, and L_w, matrix of angular velocities, obtaining:

$$L_e = \begin{bmatrix} L_v & L_\omega \\ 0 & L_\Theta \end{bmatrix} \tag{2.14}$$

Denoting s as the feature vector, one can define its elements as:

$$s = \begin{bmatrix} L_v & L_w \end{bmatrix} \begin{bmatrix} v_c \\ w_c \end{bmatrix} \tag{2.15}$$

The desired translation control input is solved for an error with exponential decay:

$$\dot{e}_t = -\lambda e_t = \dot{s} = L_v v_c + L_\omega \omega_c \tag{2.16}$$

The following control law is obtained to express translational motion input, driving the error e_t to zero:

$$v_c = -{L_v}^{+}(\lambda e_t + L_\omega \omega_c) \tag{2.17}$$

This approach is analysed under the same conditions as for IBVS and PBVS. It shows good results in image space and shows similar behaviour to the one observed in the first PBVS approach (in Cartesian space [10]). In [3], the author resumes some of the drawbacks of this method, namely he highlights the importance of the homography matrix estimation, when using the 3-D model of the object. Some other conclusions have been made, based on the choice of s features, to keep the farther target points in the image field of view [10]. Also, the selection of redundant features serves to free the system of attractive local minima, using the 2-D homogeneous coordinates [10].

The second Hybrid method is called partitioned VS. In opposition to previous methods, that selected adequate visual features defined both in 2-D and in 3-D, this approach uses only features from the image. Here, the goal is to find a number of features corresponding to the number of DOF. Each feature corresponds to one of the DOF. In that way, the control problem can be reduce to "a pure, direct and simple linear control problem" [10]. In the tutorial [10], the motion related to the

optical axis is isolated through the partition of the interaction matrix, becoming:

$$\dot{s} = L_{xy} v_{xy} + L_z v_z, \tag{2.18}$$

Where, for the example of a 6-DOF manipulator, L_{xy} has the first, second, fourth, and fifth column of L_e, from (2.2), and L_v has the third and sixth. In that way, camera motion related to the optical axis and the camera motion related to the X & the Y axis are separated. Using the exponential decay for the error, as explained before, the control law obtained becomes:

$$v_{xy} = -{L_{xy}}^{+}(\lambda e(t) + L_z v_z). \tag{2.19}$$

The features chosen in s are the coordinates of image points, and the choice of v_z is discussed in [10]. The method is detailed and tested, in [49], exploring further positive and negative points.

A different combination of IBVS and PBVS is explored in [51], where PBVS is the core of the control system and IBVS method only adds information when required. In other words, IBVS is applied in the reduced area where a targeted point is considered in the limit of the image. After some simulations, the authors noticed that only a reduced number of features are necessary to keep the target into the image plane, simplifying the computation of the hybrid method.

After resuming the basic methods of VS, we present advantages and drawbacks of IBVS, PBVS and Hybrid methods in the Tab. 2.3. Analysing the information gathered in Tab. 2.3, we conclude that all techniques have some issues. However, in our solution proposal, we use IBVS. In fact, this method presents good stability for local behaviour. Moreover, the computation is less heavy than in other methods, as we only need to project the desired and current points onto the image plane, to compute the interaction matrix. The steps of image formation are detailed in the next section (Sec. 2.2).

2.2 Image Formation

In this section, we describe the image acquisition process, using perspective camera model. It is the most commonly used model in computer vision, since it models more accurately the behaviour of real cameras [5]. First, we present

Table 2.3: Comparisons between IBVS, PBVS and Hybrid techniques [3] [10]

	Advantages	Drawbacks
IBVS	Less computation needed. Good local behaviour	Presence of image singularities and camera-retreat problems.
PBVS	Better response to large translational and rotational camera motions. Free of the image singularities, local minima, and camera-retreat problem.	Undesired translational motion. Sensitive to camera and object model errors. Feedback and estimation more time consuming. No mechanism for regulating features in image space.
Hybrid	Better response for long-range three-dimensional motions. Cartesian camera motion and image plane trajectory can be controlled simultaneously. Good performance for large translational and rotational camera motions. Not require full pose estimation, geometric model of the object released	Performance depending on switching techniques. More computationally intensive

the model of a pinhole camera, that integrates the internal (or intrinsic) camera parameters, such as the focal length. Then, this camera model is extended to integrate external (or extrinsic) camera parameters, corresponding to the position and orientation of the camera w.r.t. the world coordinate system [52].

2.2.1 Perspective projection

Given a 3-D scene in the world, cameras project one (or more) 3-D point(s) onto the 2-D image plane. This section describes the process of image formation through a 3-D perspective projection. The main feature of the perspective projection is the conservation of straight lines, after the transformation. Image formation can be approximated with a simple pinhole camera model.

Fig. 2.5 shows the ideal representation of the pinhole camera model, considering a central projection of points in space, onto a image plane defined by $Z = f$, where

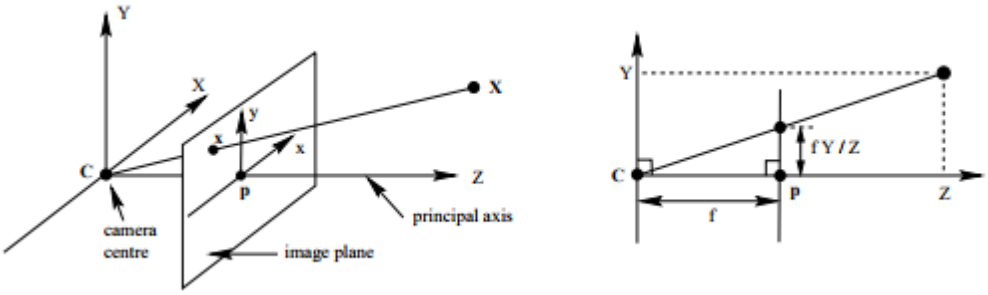


Figure 2.5: Ideal pinhole camera model [4]

f is the focal length. First of all, we have to consider the centre of projection as the origin of a Euclidean coordinate system. The optical centre C (or camera centre), located at the origin of this 3-D coordinate system, represents the point in which all the 3-D projection rays intersect. This principal axis is defined as the line perpendicular to the image plane, passing through the optical centre, originating from the camera centre. The principal axis intersects the image plane in the principal point p [4].

A point $\mathbf{X} = (X, Y, Z)^T$, defined in the coordinate system of the camera, is mapped on the image, dividing the coordinates X and Y by the point depth Z, and multiplying by the focal length f as described in (2.20) [4].

$$(X, Y, Z)^T \mapsto (f\frac{X}{Z}, f\frac{Y}{Z})^T \tag{2.20}$$

Using homogeneous coordinates, (2.20) can be written as:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.21}$$

(2.21) can be written as:

$$x = PX \tag{2.22}$$

where $\mathbf{X}$ is the world point, represented by the homogeneous 4-vector $(X, Y, Z, 1)^T$ and x is the image point represented by a homogeneous 3-vector. P is the 3×4 homogeneous matrix, given by (2.23), called camera projection matrix and defines the camera matrix parameters for the pinhole model of central projection [4].

These parameters need to be estimated by calibrating the camera with a series of measurements based on external 3-D points [5].

$$P = K[R|t] \tag{2.23}$$

The camera matrix P is composed the matrix K of intrinsic parameters and extrinsic parameters represented by the rotation matrix R and the translation matrix t [5].

2.2.2 Intrinsic parameters

The matrix K of intrinsic parameters, also called calibration matrix, is an upper triangular form matrix. We choose represent K as:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.24}$$

where f_x and f_y are independent focal length for the camera sensor x and y dimensions. If the sensor is not mounted perpendicular to the optical axis, the entry s encodes any possible bias between the sensor axes and the optical axes. (c_x, c_y) denotes the optical centre expressed in pixel coordinates [5]. Recalling (2.21) and (2.22), we can write:

$$x = K[I|0]X_{cam} \tag{2.25}$$

Where X_{cam} corresponds to the point $\mathbf{X} = (X, Y, Z, 1)^T$ in (2.21), expressed in the camera coordinate frame. As we have previously stated, we considered the camera centre in the origin of an Euclidean coordinate system. However, the points in 3-D space are expressed in the world coordinate frame. Consequently, we need to establish a relation between these two systems of coordinates.

2.2.3 Extrinsic parameters

Extrinsic parameters represent the external position and orientation of the camera in the 3-D world and relates the camera frame to the world frame. The point **0** in the world (Fig. 2.7) is represented by its coordinates as: $\mathbf{O} = [X\ Y\ Z]^T$.

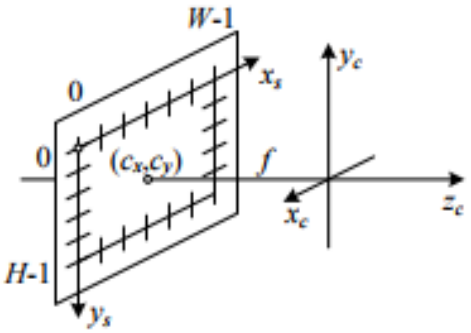


Figure 2.6: Simplified camera intrinsic parameters with focal length f, optical centre (c_x, c_y), image width W and image height H [5].

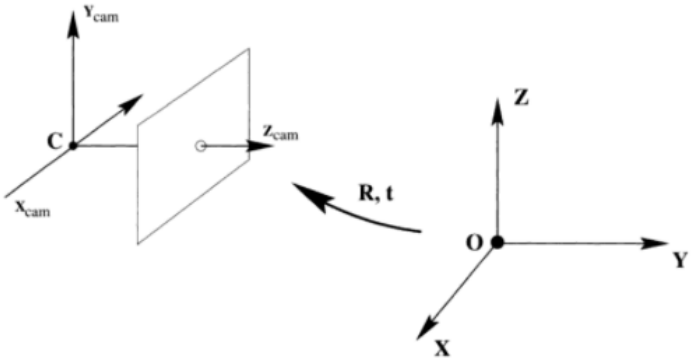


Figure 2.7: Transformation between the world coordinate frame and the camera coordinate frame [4].

We want to represent this same point in the camera frame. To do so, we apply the adequate homogeneous transformation described as follows:

$$X_{cam} = \begin{bmatrix} {}^{C}_{W}R & {}^{C}_{W}t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{O} \\ 1 \end{bmatrix} \tag{2.26}$$

where ${}^{C}_{W}R$ and ${}^{C}_{W}t$ are, respectively, the rotation and the translation parameter that define the transformation from the world frame to the camera frame [4].

Once we have defined the intrinsic and extrinsic parameters of the camera, we can compute the perspective projection, using the pinhole model described. With the points expressed in the image plane, we have the respective coordinates and depths needed to compute the interaction matrix (2.7). However, to apply IBVS to robots, we must defined the robots characteristics first.

2.3 Robotics

This section defines the mobile platform and the robotic arm manipulator, from a mathematical point of view. We will present equations that allow us to control the two systems and obtain the corresponding Jacobian matrices, to compute their input velocities.

2.3.1 Mobile platform

Mobile platforms are of two forms: omnidirectional or differential drive. The first one is defined as a holonomic robot as it can move and change its pose instantaneously, in all available directions. This kind of mobile platform has three controllable DOF: translation along X-axis, translation along Y-axis and rotation around Z-axis [15]. The following equations represents the omnidirectional vehicle kinematics:

$$X_k + 1 = X_k + \Delta X_k \tag{2.27}$$

$$Y_k + 1 = Y_k + \Delta Y_k \tag{2.28}$$

$$\theta_k + 1 = \theta_k + \Delta \theta_k. \tag{2.29}$$

The second kind of mobile platform is defined as non-holonomic. It cannot move freely in configuration space. The state in which the vehicle is, depends on the path followed to get there. A differential drive vehicle has two main wheels, each of which is attached to its own motor. A third passive wheel can also be placed in the rear, just to prevent the robot from falling over [15]. We define the position and orientation of the vehicle relative to the world frame as:

$$P = (x, y, \theta) \tag{2.30}$$

θ being the angle between the X axis of the fixed frame and the line of sigh of the vehicle (See Fig. 2.8). The differential drive vehicle can be simplified into a unicycle model. The kinematic model of the unicycle is define as:

$$\dot{x} = v cos(\theta) \tag{2.31}$$

$$\dot{y} = v sin(\theta) \tag{2.32}$$

$$\dot{\theta} = \omega \tag{2.33}$$

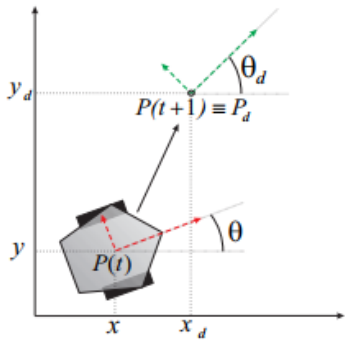


Figure 2.8: Unicycle 2-D representation [6]

where v and w are respectively the linear velocity and the angular velocity of the vehicle [6]. In matrix notation, we obtain:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{2.34}$$

Summing up:

$$\dot{P} = J_v V \tag{2.35}$$

This differential model, mapped in the vehicle Jacobian J_v, transforms velocities in the robot referential into velocities in the world referential. The robot has only 2-DOF, as suggested by the vehicle Jacobian dimension 3×2. In other words, for any control variable v and w in robot referential, the vehicle cannot move instantaneously in a direction perpendicular to the linear velocity vector. Eq. 2.36 proves the interdependence of each velocities, in the Cartesian space. Consequently, it exists a limited set of admissible velocities in each configuration [8] [53].

$$\frac{\dot{y}}{\dot{x}} = tan(\theta) \tag{2.36}$$

The control of a differential drive presents more restrictions and, therefore, it is more difficult to achieve than in an omnidirectional vehicle, which has 3-DOF.

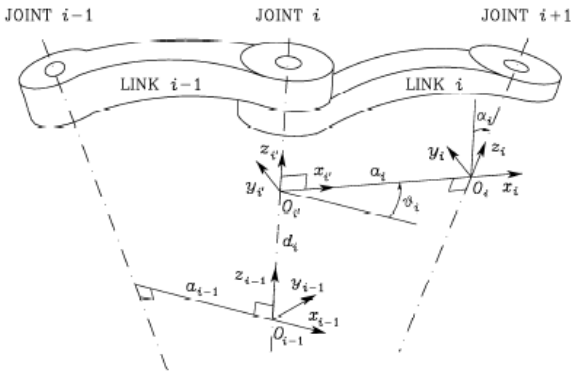


Figure 2.9: Model of serial manipulator links and joints [7]

2.3.2 Manipulator

A robot manipulator is defined as a set of rigid bodies linked to each others by joints. These joints allow relative movement between the rigid bodies and define the DOF of the robotic arm. The nature of the joint imposes a specific kind of movement of the associated links. Joints can be of revolution (defines simple rotations) or prismatic (defines simple translation over a fixed direction). Each joint links two rigid bodies to form a chain of links that defines the manipulator in series. The robotic arm has a base and an extremity. The base is defined as the link of the chain fixed in the environment, where it is inserted in. The other extremity of the cinematic chain, where the tool is inserted, is called end-effector [54] [53].

The position and orientation of the object to be grabbed by the end-effector, defines the overall movement of the manipulator. These movements can be described in joint space, given an angle value to each joint of the manipulator [53]. To this end, we previously have to define a model of the links and joints of the manipulator.

The DH-parameters are four parameters associated to the links of a kinematic chain, in space, illustrated in Fig. 2.9. Given the DH-parameters description, detailed in Tab. 2.4 and the set of frames that describes the link variables, we can apply consecutive transformations (rotations and translations) that relates all the links [7]. (2.37) is the product of the coordinate frame transform matrices, for each link. For a n-axis rigid-link manipulator, the forward kinematic solution gives

Table 2.4: Description of DH parameters

a_i	Distance from Z_i to Z_{i+1} along X_i
α_i	Angle between Z_i and Z_{i+1} around X_i
d_i	Distance from X_{i-1} to X_i along Z_i
θ_i	Angle between X_{i-1} and X_i around Z_i

the coordinate frame, or pose, of the last link.

$$ {}^{0}_{n}T = {}^{0}_{1}T \; {}^{1}_{2}T \; ... \; {}^{n-2}_{n-1}T \; {}^{n-1}_{n}T \tag{2.37} $$

The pose of the end-effector has 6-DOF in Cartesian space, 3 in translation and 3 in rotation, so robot manipulators commonly have 6 joints or DOF to allow arbitrary end-effector poses [54].

Instead of using direct kinematics, we can determine the directions of the end-effector movement as a function of joints velocities. The Forward instantaneous kinematics (or differential kinematics) are computed in the manipulator Jacobian matrix. Given a set of n functions f_i, the Jacobian is defined as the transformation of velocities in X ($X \subset \mathbb{R}^m$) into velocities in Y ($Y \subset \mathbb{R}^n$) through time derivative, expressed in matrix notation [53], as:

$$ \begin{bmatrix} \dot{y}_1 \\ \vdots \\ \dot{y}_n \end{bmatrix} = \begin{bmatrix} \frac{\delta f_1}{\delta x_1} & \cdots & \frac{\delta f_1}{\delta x_m} \\ \vdots & \ddots & \vdots \\ \frac{\delta f_n}{\delta x_1} & \cdots & \frac{\delta f_n}{\delta x_m} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_m \end{bmatrix} \tag{2.38} $$

More precisely, the matrix $\left[\frac{\delta f_i}{\delta x_j}\right]$, with $i = 1, \cdots, n$ and $j = 1, \cdots, m$ defines the Jacobian matrix. In the case of a serial manipulator with n joints, the Jacobian matrix maps the Cartesian velocities of the end-effector to the velocities in joints space [53], as follows:

$$ \left[\; \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z \; \right]' = J(\theta_1, \cdots, \theta_n) \left[\; \dot{\theta_1}, \cdots, \dot{\theta_n} \; \right]' \tag{2.39} $$

The Jacobian matrix can be separated into the Jacobian of linear velocities $J_p(\theta_1, \cdots, \theta_n)$ and the Jacobian of angular velocities, $J_o(\theta_1, \cdots, \theta_n)$ [53], as:

$$ J(\theta_1, \cdots, \theta_n) = \begin{bmatrix} J_p(\theta_1, \cdots, \theta_n) \\ J_o(\theta_1, \cdots, \theta_n) \end{bmatrix} \tag{2.40} $$

J_p can be calculated through the derivative of direct kinematics or by linear velocities propagation [53], as:

$$^{i+1}v_{i+1} = {}^{i+1}_{i}R \; {}^{i}v_i + {}^{i+1}_{i}R \; ({}^{i}\omega_i \times {}^{i} P_{i+1}) \tag{2.41}$$

J_o is obtained by the angular velocities propagation formula [53]:

$$^{i+1}\omega_{i+1} = {}^{i+1}_{i}R \; {}^{i}\omega_i + \dot{\theta}_{i+1} \; {}^{i+1}Z_{i+1}. \tag{2.42}$$

Having velocities calculated in one coordinate system, it is sometimes useful to express them in another one. For example, considering two referential F_A and F_B, the velocities in F_B can be expressed in F_A through the following equation [53], given by:

$$\begin{bmatrix} {}^{A}v \\ {}^{A}w \end{bmatrix} = \begin{bmatrix} {}^{A}_{B}R & 0 \\ 0 & {}^{A}_{B}R \end{bmatrix} \begin{bmatrix} {}^{B}v \\ {}^{B}w \end{bmatrix} = \begin{bmatrix} {}^{A}_{B}R & 0 \\ 0 & {}^{A}_{B}R \end{bmatrix} \; {}^{B}J\dot{\theta} = {}^{A}J\dot{\theta} \tag{2.43}$$

.

In practice, directions representing the robot motion are generated by the those differential models. That is why, the Jacobian analysis allows to infer about the movement limitations, when the robot is in a specific configuration. One of the reason to use the Jacobian is the exhaustive analysis of position models, when using direct kinematics. In fact, each point of the joint space has to be evaluated to find out the points of the work space (in world coordinate system) that can be reached. The Jacobian allows us to find the manipulator singularities, i.e. configurations in which the robot looses DOF in the movement [53]. A configuration at which the Jacobian is singular is referred to as a kinematic singularity. In such configuration, the robot cannot produce any movement in any direction (or set of directions) in work space. The movement of one joint, or the combination of many joints movements, do not produce any effect on the end-effector position or orientation [7]. Singularities are detected by calculating the Jacobian determinant, it is a squared matrix, or by checking the rank of a non quadratic Jacobian matrix. If $det(J) = 0$ or if the Jacobian is rank deficient, it indicates a loss of mobility of the manipulator end-effector. Indeed, end-effector velocities exist in this case which are unfeasible for any velocity commanded at the joints [15].

To sum up, the success of VS strongly depends on the accuracy of the computation of the interaction matrix. We have exposed the pros and cons of IBVS, PBVS, and Hybrid methods and summarised it in Tab. 2.3 (to better understand the differences between each method). Since the interaction matrix is directly related to the points coordinates in the image, it was necessary to detailed the image formation. The IBVS is to be applied to a robot, more precisely, to a mobile arm. Consequently, it was imperative to define the kinematics and Jacobian matrices of the mobile platform and the manipulator, in order to propose a suitable solution for the combined control.

CHAPTER 2. THEORETICAL FUNDAMENTALS

Chapter 3

Implementation

This chapter resumes the implementation process of IBVS applied to a mobile manipulator, proposed in this thesis. The code was developed using MATLAB, under the Peter Cork's Robotic toolbox. We divided this chapter in three parts: 1) we present some of the toolbox's functionality used to develop our code; 2) we detail how IBVS is applied to the manipulator and how IBVS is used to drive the vehicle; and 3) we explain how to join this two schemes to control the mobile manipulator.

3.1 Peter Cork Robotics Toolbox

Peter Cork Robotics Toolbox is a software package that uses MATLAB and provides many useful functionalities. It allows us to create and manipulate datatypes, such as homogeneous transformations, dynamics, and trajectories. The Toolbox is useful not only for simulation, but also to analyse results from experiments, with real robots [55]. It provides Jacobians, forward, and inverse kinematics functions, for arbitrary serial-link manipulators.

The manipulator used in our simulations, is the Puma 560 model (shown in Fig. 3.1). This robot was designed to have approximately the dimensions and reach of a human worker. It also has a spherical and wrist joints as humans have [8]. The model was created using workspace variables, which describe kinematic and dynamic characteristics of a Unimation Puma 560 manipulator, using the standard D-H convention. To visualise D-H parameters and other data (e.g. like

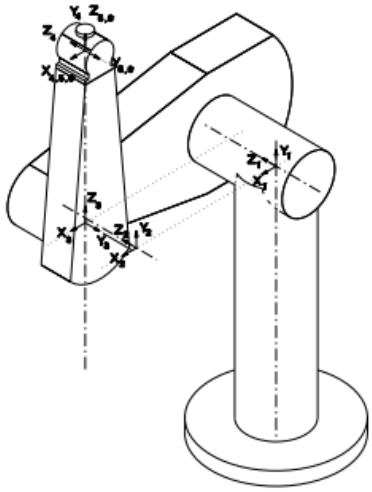


Figure 3.1: Puma 560 graphical model [8]

Joints	Limits in degrees
θ_1	[-160; 160]
θ_2	[-45; 225]
θ_3	[-225; 45]
θ_4	[-110; 170]
θ_5	[-100; 100]
θ_6	[-266; 266]

Table 3.1: Joints limits of Puma 560

the manipulator's base and tool poses Fig. 3.2), we run the `rtbdemo` from the toolbox (Fig. 3.3). It presents a graphical menu with demos that print tutorial text and MATLAB commands in the console window. In the menu, the Animation button stars by creating the Puma 560 model and defines a trajectory executed by the arm. This trajectory can be visualised graphically with the function `plot(q)`, where q is a 6×1 array, containing the joint's angles. Limits of arm joint's angles are defined in Tab. 3.1.

To understand better the manipulator's behaviour and its limitations, we tested some angles values to manually drive the robot around, using the function `p560.teach` (Fig. 3.4). Finally, to find the final joint angles of the robot, we use the function `p560.getpos()`.

```
>> p560

p560 =

Puma 560 (6 axis, RRRRRR, stdDH, fastRNE)
 viscous friction; params of 8/95;
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|     1.571|          0|
|  2|         q2|          0|     0.4318|          0|          0|
|  3|         q3|       0.15|     0.0203|    -1.571|          0|
|  4|         q4|     0.4318|          0|     1.571|          0|
|  5|         q5|          0|          0|    -1.571|          0|
|  6|         q6|          0|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+

grav =    0  base = 1  0  0  0   tool =  1  0  0  0
          0         0  1  0  0           0  1  0  0
       9.81         0  0  1  0           0  0  1  0
                    0  0  0  1           0  0  0  1
```

Figure 3.2: Puma 560 D-H parameters and characteristics

Using the function `fkine(q)`, the toolbox computes the forward kinematics of the robot. It returns the homogeneous transform between the base and the last link of the manipulator, and it will be used to obtain the end-effector pose in world frame.

Another useful function of the toolbox is the `manip([list of joints],q)`. It computes an homogeneous transform (4×4 matrix) that gives the transformation between the link frame of the first joint and the last joint of the list, as a function of the joint's variables of q. For example, if we wish to know the transformation from the base frame to the end-effector frame, we compute: `manip([6 5 4 3 2 1],q)`, where q is the vector of the 6 joint's angle values of the Puma 560.

The Jacobian function from the toolbox `jacob0(q)` relates differential joint coordinate motion to differential Cartesian motion. For an N-joint manipulator, the manipulator Jacobian is a $6 \times N$ matrix. After defining a particular joint angle configuration q for the robot, the function computes the Jacobian in the world coordinate frame. The Jacobian can also be computed in the end-effector frame, by using the function `jacobn(q)`. Our control scheme will require the inverse of the Jacobian, as we want to convert Cartesian velocities to joint angles velocities. At a manipulator singularity (or degeneracy) the Jacobian J becomes singular. So, the singularity of the Jacobian as to be checked. By computing the

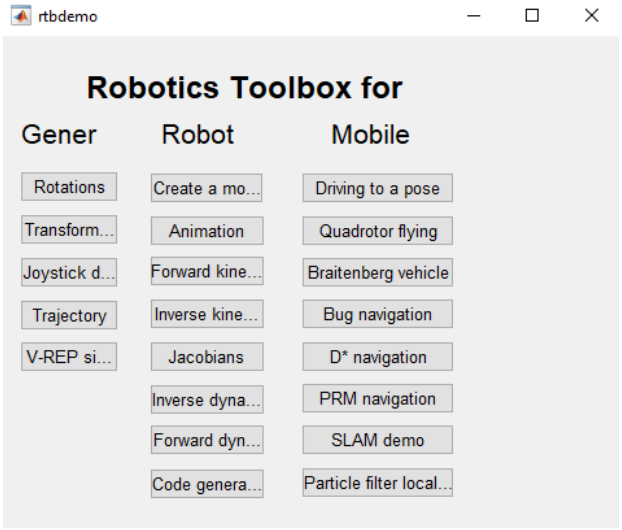


Figure 3.3: Toolbox Graphical Menu rtbdemo

determinant `det(J)`, if the result is null, the columns of the matrix are linearly dependent and, by consequence, the matrix is not full rank. We can also verify if the manipulator is in a position where it looses DOF, by computing the rank of the Jacobian. If it is inferior to the Jacobian columns, it exists dependencies between the columns, so DOF are lost. The singular value decomposition (`svd(J)`) gives the correspondent singular values of the Jacobian J, and is useful for a more detailed analysis of kinematics and dynamics.

Toolbox functions presented in this section are used to implement our simulator, in particular, the manipulator control part.

3.2 Mobile platform and manipulator control

In this section, the steps of the implementation process are detailed. First, we describe the vehicle control system for navigation and, then, the arm controller for object manipulation.

3.2.1 Vehicle IBVS

First of all, we define a simple differential drive model for the vehicle, using the function `DifferentialRobot.m`. This model computes the vehicle kinematics

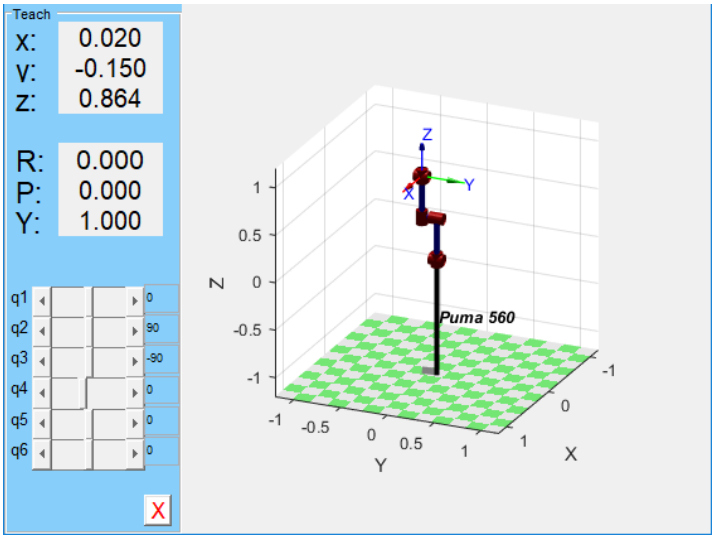


Figure 3.4: Puma 560 teach function

equations, described in (2.34), Sec. 2.3.1 and plot the vehicle pose in a 2-D plan. Then, we define the mobile platform initial position in the world, as $p = [x\ y\ \theta]'$. Similarly, we choose a final pose $R_x = [x_f\ y_f\ \theta_f]'$ to make sure that the vehicle can reach it. The vehicle pose, in the world frame, is given by the following homogeneous transform:

$$ {}^{W}_{V}T = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & x \\ sin(\theta) & cos(\theta) & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

Since the robot controller is based on image features, we have to simulate a camera in order to extract visual features, needed to compute the interaction matrix. We define four points in world frame, with coordinates $[X\ Y\ Z]$, in 4×4 homogeneous matrix from ${}^{W}P$. We consider for the camera pose, the same pose as the vehicle. In order to have the Z-axis of the camera pointing to the roof. Thus, the camera is expressed in world frame as in (3.1), i.e. ${}^{W}_{C}T = {}^{W}_{V}T$. Having the camera pose in world frame, we express points ${}^{W}P$, in the camera frame as in:

$$ {}^{C}P = {}^{C}_{W}T\ {}^{W}P. \tag{3.2} $$

Then, we project the points ${}^{C}P$ onto the image plan. Image coordinates (x, y) of

each point are stored in the feature's vector s (3.3).

$$s = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} \tag{3.3}$$

To compute the image pixel coordinates (u, v), we use the pinhole model described in Subsec. 2.2.1. Values of the focal lengths $f = (f_u, f_v)$, and of the image centre (c_x, c_y) are: $f = (671.13759, 680.77186)$ and $C = (2 \times 319.5, 2 \times 239.5)$. Pixels coordinates of the four points are stored in a 2×4 matrix uv:

$$uv = \begin{bmatrix} u_{x_1} & u_{x_2} & u_{x_3} & u_{x_4} \\ v_{x_1} & v_{x_2} & v_{x_3} & v_{x_4} \end{bmatrix} \tag{3.4}$$

The same process is executed for the vehicle final pose. Once we have expressed the image features and stored the current and desired features, corresponding to each image point coordinates, in the arrays s and s_x respectively. Then, we can compute the error between the features as:

$$e = s - s_x \tag{3.5}$$

This error will be the variable controlling the cycle.

The control cycle is initiated by an interaction matrix L_x (2.7). Then, the vehicle Jacobian J_v, relating the mobile platform Cartesian velocities and the mobile platform linear and angular velocities (v_x, w_z), is computed as follows:

$$J_v = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.6}$$

Similarly, the mobile platform control velocities are given by:

$$v_c = -\lambda L_e^+ e \tag{3.7}$$

In which

$$L_e^+ = (L_e^T \; L_e)^{-1} \; L_e^T \tag{3.8}$$

where $L_e = L_x \; J_v$. We compute the Moore-Penrose pseudo-inverse L_e^+, (3.8), a generalisation of the inverse matrix for non quadratic matrices, using the function `pinv()` from MATLAB library.

To drive the vehicle, we apply the calculated control, using the function `drive(obj,`v_c`,dt)` from the `DifferentialRobot.m` file. This function receives 3 parameters:

- obj: the robot object created by the *DifferentialRobot.m* file;
- v_c: the calculated velocity vector $v_c = [v_x \quad \omega_z]'$, where v_x is the linear velocity along the x axis and ω_z: the angular velocity in z axis; and
- dt is the time interval chosen.

The vehicle pose is updated at each iteration k, through:

$$\theta(k+1) = \theta(k) + \omega_z dt, \tag{3.9}$$

and:

$$\begin{bmatrix} x(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} x(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \end{bmatrix} v_c dt. \tag{3.10}$$

Once we have the vehicle new pose, we compute the new camera pose and we project the current points to the image plane. From here, we obtain the new features points and we compute the new error. Again, we visualised the vehicle trajectory in a 2-D plan, the current features points, in pixels, updated in the image, the error of each features, and the control velocities v_c at each iteration. Once we obtained conclusive results, i.e., we observed convergence of the current points and desired points in image, we analysed and found an adequate norm of the error to stop the cycle. We choose to end the VS control, when the norm of the error e was inferior or equal to 0.01.

3.2.2 Manipulator IBVS

The manipulator, called `manip` in the code, is defined using the Puma 560 model, presented in Sec. 3.1, with initial joints angles values, in the array $q =$

$[q_1\ q_2\ q_3\ q_4\ q_5\ q_6]$. Using the function `fkine(manip,q)`, we obtain the end-effector pose in world frame (for the arm configuration specified by the joints angles in q). Then, we placed a virtual camera rotated by 90^{o} in Y-axis, on top of the end-effector, in order to have the camera Z-axis oriented to the wall. The camera is expressed in the end-effector frame as:

$$ {}^{E}_{C}T = \begin{bmatrix} R_Y(\frac{\pi}{2}) & t(0,0,0.1) \\ 0_{1\times 3} & 1, \end{bmatrix} \tag{3.11} $$

We consider that the base frame is in the position $[0,0,0]$, defined in world frame. By consequence, we only need to compute the camera pose in base frame ${}^{B}_{C}T$ as:

$$ {}^{B}_{C}T = {}^{B}_{E}T\ {}^{E}_{C}T \tag{3.12} $$

From the initial camera pose, the initial features (coordinates of the four points previously defined in world frame) are computed in camera frame:

$$ {}^{C}P = {}^{C}_{B}T\ {}^{B}P \tag{3.13} $$

We project the points in ${}^{C}P$ onto the image plane and obtain the respective coordinates (x, y) of the projected points in the image. As before (see Subsec. 3.2.1), current features are stored in the vector s. We compute the image points coordinates in pixels in matrix uv to plot the image.

The next step is to define a final arm configuration, to make sure the goal can be reached, and to make sure that the final configuration is possible to reach. Having the final arm configuration defined, the same steps are executed to obtain the desired points coordinates in the image plane, stored in the vector s_x. The pixels coordinates of those desired points are stored in the matrix uv_x.

After calculating the error $e = s - s_x$, we initiate the cycle by computing the interaction matrix L_x. The interaction matrix is of size 8×6; the 8 lines that correspond to the 8 coordinates of the four points and the 6 columns corresponding to the 6-DOF of the manipulator. We obtain the manipulator Jacobian J_q in end-effector frame using the method `jacobn(q)` from the toolbox (Sec. 3.1).

Actually, we must express the manipulator Jacobian in the camera frame, since the control is done in the image plane, thought the interaction matrix L_x calculated in camera frame. To do so, we need the skew-symmetric matrix S and the

rotation matrix ${}^{C}_{E}R$, corresponding to the transformation from the end-effector to the camera frame. This transformation is fixed and defined in (3.11). From the transformation matrix ${}^{C}_{E}T$, we extract the end-effector coordinates in camera frame ${}^{C}_{E}X$, ${}^{C}_{E}Y$ and ${}^{C}_{E}Z$, and compute the skew-symmetric matrix as:

$$S = \begin{bmatrix} 0 & -{}^{C}_{E}Z & {}^{C}_{E}Y \\ {}^{C}_{E}Z & 0 & -{}^{C}_{E}X \\ -{}^{C}_{E}Y & {}^{C}_{E}X & 0 \end{bmatrix} \tag{3.14}$$

We also extract the rotation matrix ${}^{C}_{E}R$ from ${}^{C}_{E}T$, which allows use to construct the matrix:

$${}^{C}_{E}TS = \begin{bmatrix} {}^{C}_{E}R & S\,{}^{C}_{E}R \\ 0_{3\times 3} & {}^{C}_{E}R \end{bmatrix} \tag{3.15}$$

As discussed in Sec. 2.1.1, the control law is obtained as:

$$v_c = -\lambda L_e^{+} e, \tag{3.16}$$

in which

$$L_e^{+} = (L_e^T\ L_e)^{-1}\ L_e^T, \tag{3.17}$$

where $L_e = L_x\ {}^{C}_{E}TS\ J_q$.

Once we have calculated the velocities, we update the actual joints positions to the new ones. For that, we obtain the arm joints angles in the current configuration, using `getpos()` from the robotic's toolbox, and we add the new joints velocities. From the new end-effector pose in base frame (obtained by `fkine()` method), we project the points in the image plan, to update the current features, and to calculate the error between those new points and the desired ones. At each iteration, we visualise the manipulator movements in the world, the current points trajectory in the image plane, the error between the current and the desired features, the calculated linear velocities v_x, v_y and the angular velocity w_z of the end-effector, in Cartesian space.

Once we obtained satisfactory results of both manipulator and vehicle servoing, we joined the two control scheme to control the mobile manipulator.

3.3 Mobile manipulator control

In this section, we explain how we combine the two control schemes in order to have a unified control for the mobile manipulator. We start by defining an initial

position for the vehicle in the world $p = [x\ y\ \theta]'$. Then, we create the model of the manipulator (Puma 560) as described in Sec. 3.1. To attach the manipulator to the vehicle, we modify the pose of the manipulator base, giving it the same position and orientation as the vehicle, through the homogeneous transform:

$$ {}^{W}_{B}T = \begin{bmatrix} R_Z(\theta) & t(x,y,0) \\ 0_{1\times 3} & 1 \end{bmatrix}, \tag{3.18} $$

where $R_Z(\theta)$ is the rotation matrix of angle θ about Z-axis in world frame, and $t(x, y, 0)$ is the translation vector. This transformation is done by modifying directly the parameters in the manipulator model previously created.

Next, the initial joint's arm configuration is given by: $q = [q_1\ q_2\ q_3\ q_4\ q_5\ q_6]$. The same steps are executed to give a desired position to the vehicle, and the arm attached to it.

Once the initial and desired configuration are done, we place a virtual camera on top of the end-effector, facing the wall, i.e., we apply the homogeneous transformation:

$$ {}^{E}_{C}T = \begin{bmatrix} R_Y(\frac{\pi}{2}) & t(0,0,0.1) \\ 0_{1\times 3} & 1, \end{bmatrix} \tag{3.19} $$

corresponding to a rotation of 90^{o} about the Y-axis of the end-effector frame. To project the four points defined in 3-D world onto the image plan, we express those points in the camera frame, by applying the following transformations:

- First, we compute the end-effector pose in world coordinates ${}^{W}_{E}T$, by using the forward kinematics function `fkine(q)` already discussed in Sec. 3.1. This function gives the end-effector pose in homogeneous coordinates, taking into account all the joint's values from the base link to the last link of the arm;

- We express the camera pose in world coordinates by multiplying the homogeneous transforms of the camera in end-effector frame and the end-effector in world frame as:

$$ {}^{W}_{C}T = {}^{W}_{E}T\ {}^{E}_{C}T; \text{ and} \tag{3.20} $$

- We finally express points in camera frame as:

$$ {}^{C}P = {}^{C}_{W}T\ {}^{W}P. \tag{3.21} $$

Having the points expressed in camera frame, we can apply the perspective projection, to have the corresponding 2-D points coordinates in the image plane. As before, the 2-D coordinates of the current points are stored in the vector s and the desired ones are stored in the vector s_x.

Once again, the cycle is initialised with the interaction matrix L_x. Then, the function `jacobn(q)` computes manipulator Jacobian in the end-effector frame: J_q. In the previous section (Subsec. 3.2.1), we defined the vehicle Jacobian in (3.6). However, our goal is to combine the two previous Jacobians, in order to apply a unified IBVS system. To compute the combined Jacobian, most state-of-the-art methods took into account the dynamics of the mobile manipulator in their derivations (such as [56] [57]), which is not the case here, or involved task planning (e.g. [58]). Moreover, the calculations were not detailed enough for all the details to be understood and adapted to our case. In this thesis we follow the method proposed in [59], which we could adapt the combined Jacobian to our case.

First, as we have already stated, the manipulator base frame has the same orientation and position as the mobile platform. Considering the arm being at the centre of vehicle's gravity, we can establish that they have the same world coordinates, given by:

$$
\begin{aligned}
{}^{W}_{B}X &= {}^{W}_{P}X && (3.22)\\
{}^{W}_{B}Y &= {}^{W}_{P}Y && (3.23)\\
\theta_B &= \theta_P, && (3.24)
\end{aligned}
$$

where $[{}^{W}_{B}X \quad {}^{W}_{B}Y \quad \theta_B]'$ and $[{}^{W}_{P}X \quad {}^{W}_{P}Y \quad \theta_P]'$ represent respectively the base and the mobile platform frames positions and orientations in the world. By time differentiation, the relation between velocities of frame B and frame P, in the world is defined as:

$$
\begin{aligned}
\begin{bmatrix} {}^{W}_{B}\dot{X} \\ {}^{W}_{B}\dot{Y} \\ \dot{\theta}_B \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{W}_{P}\dot{X} \\ {}^{W}_{P}\dot{Y} \\ \dot{\theta}_P \end{bmatrix} && (3.25)\\
V_B &= J_{PB} \quad V_P. && (3.26)
\end{aligned}
$$

J_{PB}, from (3.26), is the Jacobian relating the arm base Cartesian velocities to the mobile platform Cartesian velocities. In our case, it is an identity matrix,

as we put the arm in the centre of gravity of the vehicle. By consequence, the general Jacobian that relates the arm based Cartesian velocities (which the mobile platform linear and angular velocities) is simply given by:

$$V_B = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ \omega_z \end{bmatrix}. \tag{3.27}$$

To include the six Cartesian velocities of the manipulator base frame B in the world, we have to extend the Jacobian as:

$$V_B = \begin{bmatrix} {}^W_B\dot{X} \\ {}^W_B\dot{Y} \\ {}^W_B\dot{Z} \\ {}^W_B\dot{\theta}_x \\ {}^W_B\dot{\theta}_y \\ {}^W_B\dot{\theta}_z \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ \omega_z \end{bmatrix}. \tag{3.28}$$

To combine the mobile platform Jacobian and the manipulator Jacobian, we have to include the effect that the mobile platform motion at frame B has on the end-effector at frame E. The method is divided in two steps: 1)we construct a matrix Jacobian ignoring the arm's joints motion, 2)we introduce the arm's joints velocities in the combined matrix.

This effect is determined by defining the velocity task vector of frame E, relative to the world frame W, as:

$$\begin{aligned}
{}^W_E\dot{X} &= {}^W_B\dot{X} - {}^B_E X sin(\theta)\dot{\theta} - {}^B_E Y cos(\theta)\dot{\theta} && (3.29)\\
{}^W_E\dot{Y} &= {}^W_B\dot{Y} + {}^B_E Y cos(\theta)\dot{\theta} - {}^B_E Y sin(\theta)\dot{\theta} && (3.30)\\
{}^W_E\dot{Z} &= {}^W_B\dot{Z} && (3.31)\\
{}^W_E\dot{\theta}_x &= {}^W_B\dot{\theta}_x && (3.32)\\
{}^W_E\dot{\theta}_y &= {}^W_B\dot{\theta}_y && (3.33)\\
{}^W_E\dot{\theta}_z &= {}^W_B\dot{\theta}_z, && (3.34)
\end{aligned}$$

where ${}^B_E X$ and ${}^B_E Y$ are the X and Y coordinates of the end-effector, relative to the arm base frame B. We obtain the homogeneous transformation, from the base link to the end-effector, by using the toolbox's function `manipulator.A([6 5 4 3 2`

1],q), from where we can extract the end-effector coordinates in base frame. ${}^{W}_{B}\dot{X}$, ${}^{W}_{B}\dot{Y}$, ${}^{W}_{B}\dot{Z}$ are the Cartesian velocities of the base B, relative to the world frame W, and the ${}^{W}_{E}\dot{X}$, ${}^{W}_{E}\dot{Y}$, ${}^{W}_{E}\dot{Z}$ are the Cartesian velocities of the end-effector E, relative to the world frame. The ${}^{W}_{E}\dot{\theta}_x$, ${}^{W}_{E}\dot{\theta}_y$, ${}^{W}_{E}\dot{\theta}_z$ are the rotation angles of the end-effector E relative to X, Y and Z axes of the world frame W. The mobile manipulator only moves in the horizontal plane, as we are not considering the movement of the arm's joints, the angular velocities ${}^{W}_{E}\dot{\theta}_x$ and ${}^{W}_{E}\dot{\theta}_y$ are null. The ${}^{W}_{E}\dot{\theta}_z$ is the same as the angular velocity of mobile platform $\dot{\theta}$. We can represent previous equations (Eq. 3.29 to Eq. 3.34) in matrix form:

$$V_e = \begin{bmatrix} {}^{W}_{E}\dot{X} \\ {}^{W}_{E}\dot{Y} \\ {}^{W}_{E}\dot{Z} \\ {}^{W}_{E}\dot{\theta}_x \\ {}^{W}_{E}\dot{\theta}_y \\ {}^{W}_{E}\dot{\theta}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -({}^{B}_{E}X sin(\theta) & + & {}^{B}_{E}Y cos(\theta)) \\ 0 & 1 & 0 & 0 & 0 & {}^{B}_{E}Y cos(\theta) & - & {}^{B}_{E}Y sin(\theta) \\ 0 & 0 & 1 & 0 & 0 & & 0 & \\ 0 & 0 & 0 & 1 & 0 & & 0 & \\ 0 & 0 & 0 & 0 & 1 & & 0 & \\ 0 & 0 & 0 & 0 & 0 & & 1 & \end{bmatrix} \begin{bmatrix} {}^{W}_{B}\dot{X} \\ {}^{W}_{B}\dot{Y} \\ {}^{W}_{B}\dot{Z} \\ {}^{W}_{B}\dot{\theta}_x \\ {}^{W}_{B}\dot{\theta}_y \\ {}^{W}_{B}\dot{\theta}_z \end{bmatrix} \tag{3.35}$$

$$V_e = J_{AE} V_B. \tag{3.36}$$

In (3.36), V_e is the vector of Cartesian velocities of the end-effector in the world. J_{AE} is the Jacobian that relates the Cartesian velocities of the end-effector frame, to the Cartesian velocities of the arm base frame, due to the mobile platform motion only. Now, if we substitute (3.28) in (3.35), we obtain:

$$V_e = \begin{bmatrix} cos(\theta) & -({}^{B}_{E}X sin(\theta) & + & {}^{B}_{E}Y cos(\theta)) \\ sin(\theta) & {}^{B}_{E}Y cos(\theta) & - & {}^{B}_{E}Y sin(\theta) \\ 0 & & 0 & \\ 0 & & 0 & \\ 0 & & 0 & \\ 0 & & 1 & \end{bmatrix} \begin{bmatrix} v_x \\ \omega_z \end{bmatrix} \tag{3.37}$$

$$V_e = J_{EP} V_p. \tag{3.38}$$

Eq. (3.38) is a simplification of Eq. (3.37), were J_{EP} is the Jacobian that relates the mobile platform Cartesian velocities to the Cartesian velocities of the end-effector in the world, without the motion of the arm. However, we want to express these velocities in the end-effector frame. For that, we transform the Jacobian calculated in world frame in order to express the Jacobian in end-effector frame.

To do so, we obtain the end-effector pose in world frame, using the `fkine(q)` function. From this matrix, we extract the end-effector coordinates ${}^{W}_{E}X$, ${}^{W}_{E}Y$ and ${}^{W}_{E}Z$ and compute the skew-symmetric matrix detailed in (3.39).

$$S = \begin{bmatrix} 0 & -{}^{W}_{E}Z & {}^{W}_{E}Y \\ {}^{W}_{E}Z & 0 & -{}^{W}_{E}X \\ -{}^{W}_{E}Y & {}^{W}_{E}X & 0 \end{bmatrix} \tag{3.39}$$

We also extract the rotation matrix R_e, which allows use to construct the matrix:

$$T = \begin{bmatrix} R_e & SR_e \\ 0_{3\times3} & R_e \end{bmatrix} \tag{3.40}$$

$$J_v = TJ_{EP}. \tag{3.41}$$

In (3.41), J_v relates the mobile platform Cartesian velocities to the Cartesian velocities of the end-effector, in the end-effector frame.

Finally, taking into account the arm joint's motion, the combined Jacobian J is given by:

$$J = [J_q \; J_v]. \tag{3.42}$$

This Jacobian relates the mobility of the mobile platform with the manipulation of the robotic arm, and relates the six Cartesian velocities of the end-effector to the joint angles of the arm and the mobile platform linear and angular velocities. As explained in Subsec 3.2.2, the combined Jacobian must be expressed in the camera frame. The necessary transformation ${}^{C}_{E}TS$ is given by (3.15) and applied in the control law:

$$v_c = -\lambda L_e^{+} e, \tag{3.43}$$

where $Le^{+} = (L_x \; {}^{C}_{E}TS \; J)^{+}$. The velocity vector is an array of 8×1, containing the 6 arm joints angles velocities $[\dot{q}_1 ... \dot{q}_6]'$ and the two linear and angular velocities of the mobile platform, respectively $[v_x \quad \omega_z]'$.

We drive the mobile platform and update the arm joint's angle, as explained in the previous Sec. 3.2. We project points onto the image (Sec. 3.2) to have the new features coordinates and update the error. The mobile manipulator motion is visualised in real-time (in 3-D world). The mobile platform trajectory is presented in 2-D plan and the current points are plotted in the image, at each iteration. The error features values, the end-effector velocities and the mobile

platform velocities are also represented in real time.

To summarise, in this chapter we detailed the implementation process, first by presenting the Peter Corke's robotic toolbox and then by exposing the three steps of the development: IBVS control of the mobile platform, of the manipulator and last, of the mobile manipulator. The next chapter presents simulations and an analyse of the corresponding results.

CHAPTER 3. IMPLEMENTATION

Chapter 4

Simulations

This chapter presents the results of experiments performed during the project, based on the theoretical fundamentals presented in Chap. 2. The IBVS control of the mobile platform, the manipulator, and the mobile manipulator are simulated in MATLAB, as presented in Chap. 3.

4.1 Mobile Robot

First, the performance of the mobile platform (using IBVS) is tested with the vehicle initial position in the origin of the world frame. By giving several different final positions to the vehicle (points in the image space), we make sure that the given final position is reachable. All tests are performed with the same four points ${}^{W}P$, defined in world frame. For simplicity, we consider that the camera's Z-axis is pointing to the roof, i.e. the camera has the same pose as the vehicle. By consequence, we do not have to consider different positions for the points in world frame, according to the orientation of the camera. The four points coordinates are given in matrix form:

$$ {}^{W}P = \begin{bmatrix} 5.25 & 5.25 & 5.25 & 5.25 \\ 1.75 & 1.25 & 1.25 & 1.75 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.1} $$

The parameters chosen for the tests are:

- A time interval $dt = 0.1$;

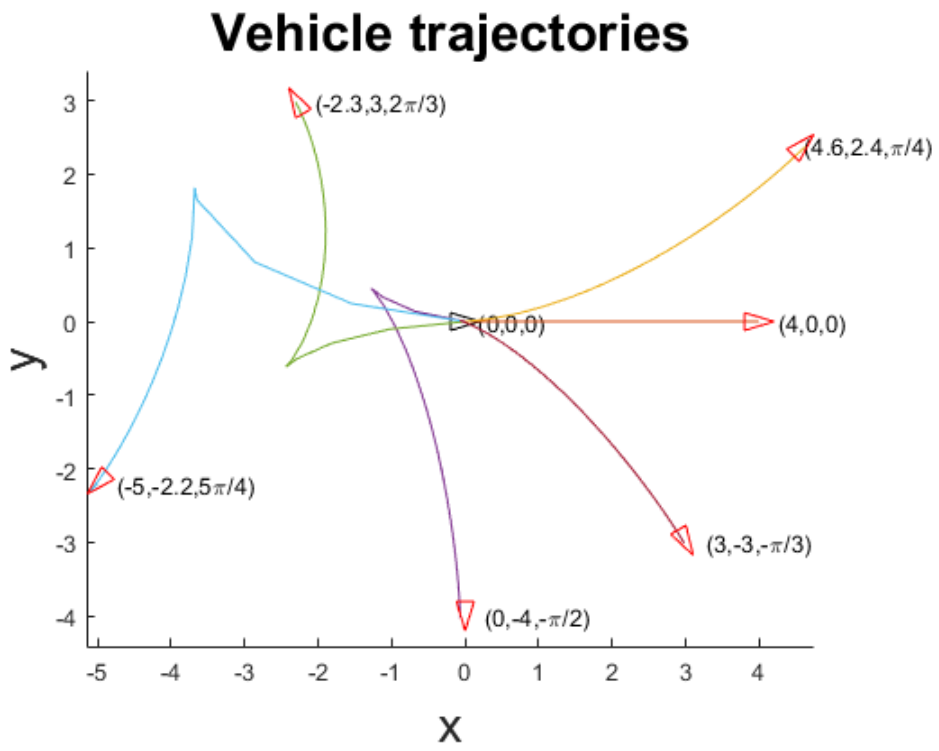


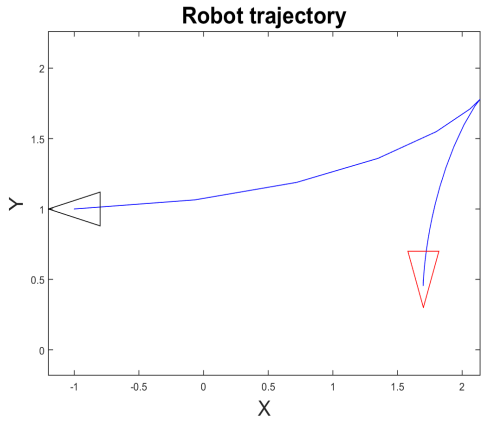
Figure 4.1: 2-D mobile platform trajectories with initial position $p = (0, 0, 0)$(black triangle) and several desired positions (red triangles).

- A control parameter $\lambda = 1$; and

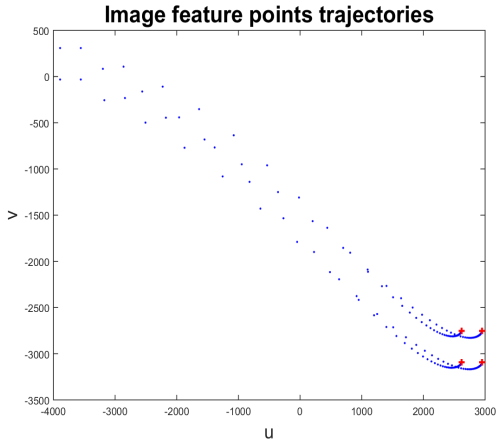
- A condition error $\|e\| >= 0.01$ to stop the control cycle.

The resulting mobile platform trajectories are shown in Fig. 4.1. As it can be seen by this figure, all the desired positions are reached.

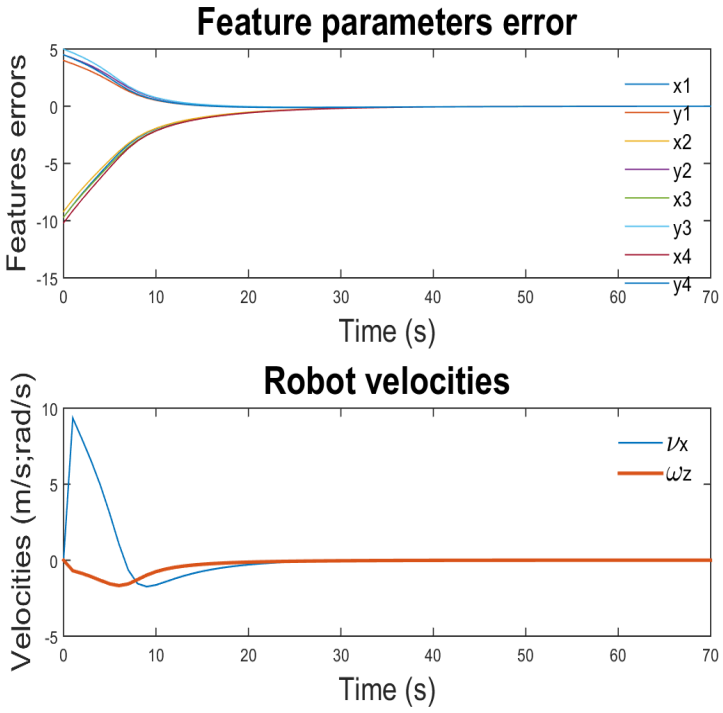
Given those successful results, we are interested in testing the performance of the vehicle control when the vehicle initial position and orientation are not null. For that purpose, we consider the same 4 points used in the previous simulations (4.1). Fig. 4.2 presents the results of the simulation run with mobile platform initial position $p = (-1, 1, \pi)$ and final position $pf = (1.7, 0.5, -\frac{\pi}{2})$. To confirm that results of Fig. 4.2 are not occasional, we show one more test with world points ${}^{W}P$



(a) 2-D mobile platform trajectory. The black and red triangles represent respectively the mobile platform initial and final positions.

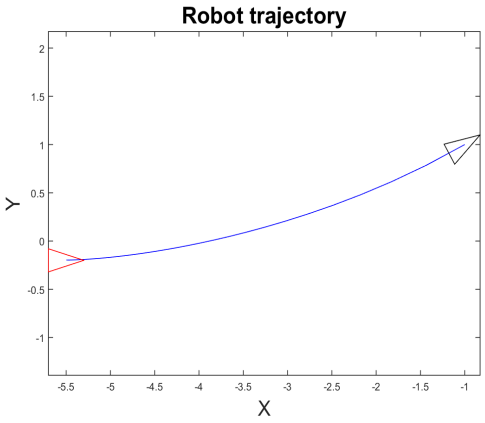


(b) Desired points (red crosses) and current points (blue points) trajectory in the image plane, in pixels.

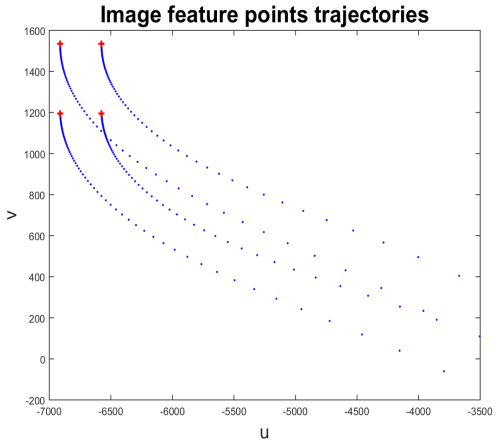


(c) Features errors over time, where $(x_1, y_1, ..., x_4, y_4)$ are the 4 image points coordinates. Mobile platform linear and angular velocities (v_x, ω_z) over time.

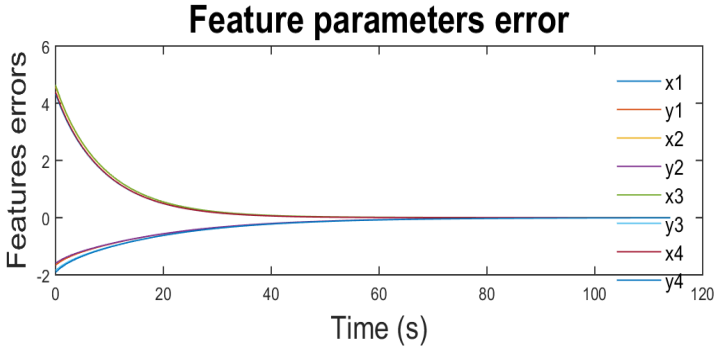
Figure 4.2: Results of mobile platform simulation with initial position $(-1, 1, \pi)$ and final position $(1.7, 0.5, -\frac{\pi}{2})$.

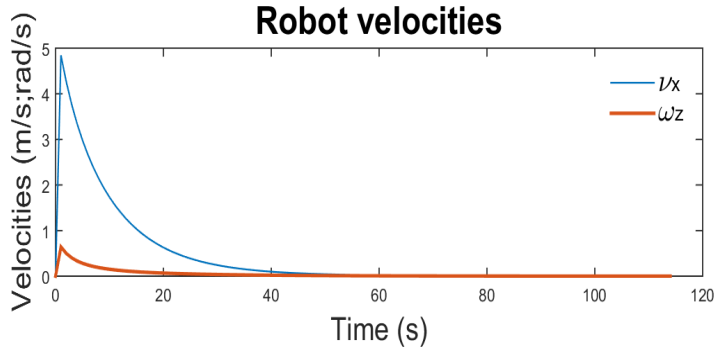


(a) 2D mobile platform trajectory. The black and red triangles represent respectively the mobile platform initial and final position.



(b) Desired points (red crosses) and current points (blue points) trajectory in the image plane, in pixels.





(c) Features errors over time, where $(x_1, y_1, ..., x_4, y_4)$ are the 4 image points coordinates. Mobile platform linear and angular velocities (v_x, ω_z) over time.

Figure 4.3: Results of mobile platform simulation with initial position $(-1, 1, \frac{\pi}{6})$ and final position $(-5.5, -0.2, 0)$.

with coordinates changed to:

$$^{W}P = \begin{bmatrix} -5.25 & -5.25 & -5.25 & -5.25 \\ 1.75 & 1.25 & 1.25 & 1.75 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.2}$$

The behaviour of the mobile platform is tested with an initial position of $p = (-1, 1, \frac{\pi}{6})$ and final position $pf = (-5.5, -0.2, 0)$. The results are presented in Fig. 4.3.

Both simulations results show the mobile platform 2D trajectory (Fig. 4.2a and Fig. 4.3a), the trajectory of the four points in the image (Fig. 4.2b and Fig. 4.3b), the error between the desired and the current features, and the robot velocities over the time (Fig. 4.2c and Fig. 4.3c). In both cases, current points converge to the desired ones (in the image plane) and the mobile platform reaches the final desired pose.

4.2 Manipulator

The performance of IBVS control on a robot manipulator (in this case a Puma 560) is tested for different arm joint angle values. For all the simulations, the parameters chosen are:

- A time interval $dt = 0.1$;
- A control parameter $\lambda = 1$; and
- A condition error $\|e\| >= 0.01$ to stop the control cycle.

First, we test the performance of the algorithm with four points, given in matrix form by:

$$^{W}P = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0.75 & 0.25 & 0.75 & 0.25 \\ 0.75 & 0.25 & 0.25 & 0.75 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.3}$$

The tests are initiated by choosing an initial arm joints configuration $q = [0\ 0\ 0\ 0\ 0\ 0]$, and several desired final configurations. Camera trajectories for those simulations are shown in Fig. 4.4. All those 6 simulations have successful

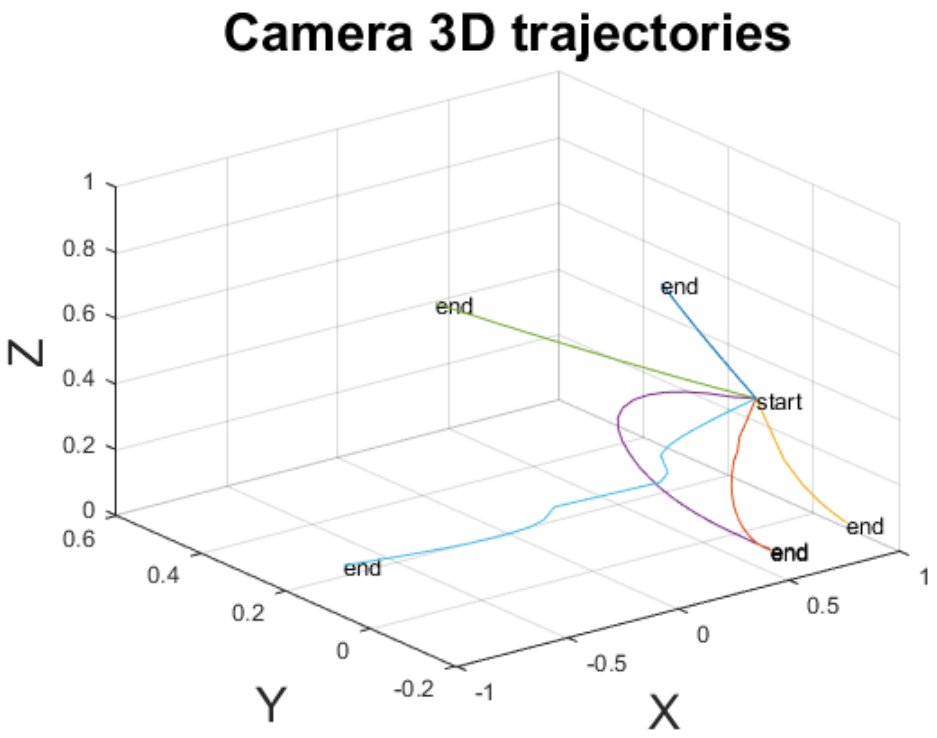


Figure 4.4: 3D camera trajectories of several desired final arm joints configurations, for the manipulator control simulations. The initial and final camera positions are represented respectively by the label *start* and *end*.

results, as the current points converged to the desired points, in the image plan, and the errors decay exponentially to zero with time, as explained in Sec. 2.1. The overall observed 3D trajectories were smooth, even with important movements of rotation, eventually needed at the beginning of the control.

Hereupon, we are interested in the observation of the manipulator control performance, for different initial and final configurations. Consequently, point coordinates in world frame are chosen by taking into account their final camera pose. Indeed, considering the camera rotation, with respect to the end-effector, we have to make sure that both initial and final configurations (chosen in the arm joints space) allow the camera to observe the points in those two poses. Fig. 4.5 and

Fig. 4.6, present the results of the simulation run with points described by:

$$ {}^{W}P = \begin{bmatrix} 0.75 & 0.25 & 0.75 & 0.25 \\ 1 & 1 & 1 & 1 \\ 0.75 & 0.25 & 0.25 & 0.75 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \tag{4.4} $$

and the configurations:

- $q = [\frac{\pi}{2}\ 0\ \pi\ 0\ \pi\ 0]$, for the initial joints angles values; and
- $qf = [\frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]$, for the final joints angles values.

As observed in Fig. 4.5c, the current features converge to the desired ones with smooth camera movements (Fig. 4.5d). The desired configuration is reached (Fig. 4.5b) by doing a translational movement in Y-axis, with a decreasing velocity (Fig. 4.6).

In addition, a different arm configuration is tested. Fig. 4.7 and Fig. 4.8 present the results of this simulation, run with the points described by:

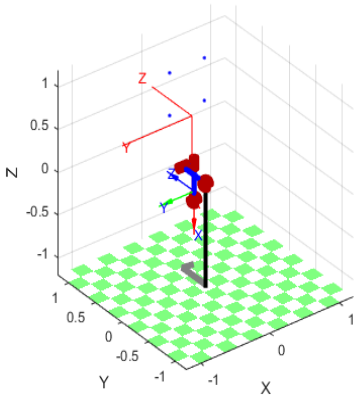
$$ {}^{W}P = \begin{bmatrix} 0.75 & 0.25 & 0.75 & 0.25 \\ -1 & -1 & -1 & -1 \\ 0.75 & 0.25 & 0.25 & 0.75 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \tag{4.5} $$

and configurations:

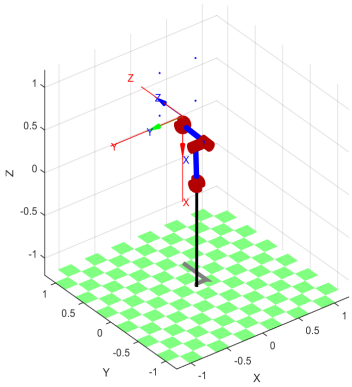
- $q = [-\frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]$, for the initial joints angles values
- $qf = [0\ -\frac{\pi}{2}\ -\pi\ -\frac{\pi}{6}\ -\frac{\pi}{6}\ -\frac{\pi}{3}]$, for the final joints angles values.

Initial velocities are higher than in the rest of the movement (as it can be observed in Fig. 4.8), which corresponds to the initial abrupt change of direction. The overall camera trajectory is smooth (Fig. 4.7d) and we observe that the system converge correctly in the image space, as we can see in Fig. 4.7c.

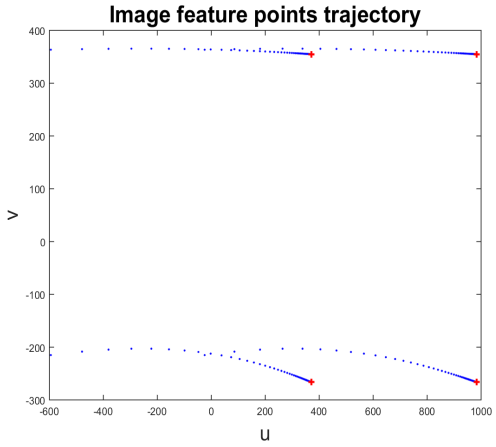
To sum up, we have to be careful when choosing the initial and final positions, for which the depth of the points can be correctly calculated in the interaction matrix.



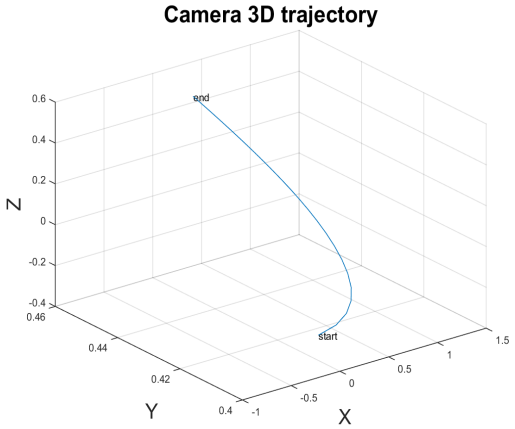
(a) Manipulator in initial configuration q. The actual camera frame (represented by the 3 arrows) does not match the red X-Y-Z axis representing the frame of the desired camera pose.



(b) Manipulator in final configuration qf. The actual camera frame (represented by the 3 arrows) matches the red X-Y-Z axis representing the frame of the desired camera pose.



(c) Desired points (red crosses) and current points (blue points) trajectory in the image plane, in pixels.



(d) 3D camera trajectory.

Figure 4.5: Results of Puma 560 simulations with initial configuration $q = [\frac{\pi}{2}\ 0\ \pi\ 0\ \pi\ 0]$ and final configuration $qf = [\frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]$.

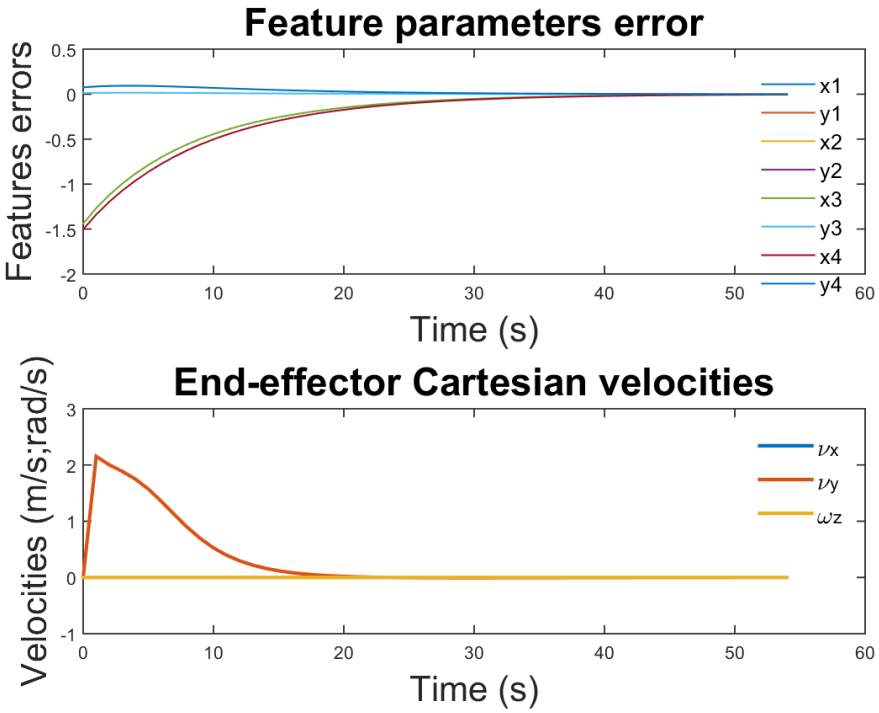


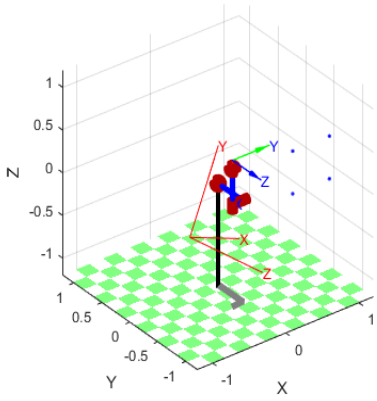
Figure 4.6: Features errors over time, where $(x_1, y_1, ..., x_4, y_4)$ are the 4 image points coordinates. End-effector linear and angular velocities (v_x, v_y, ω_z) over time.

4.3 Mobile manipulator

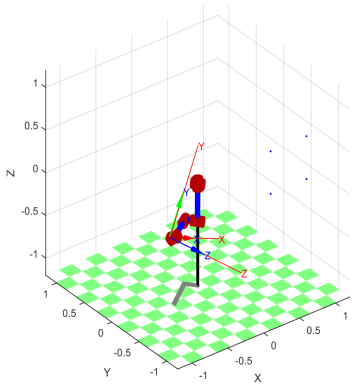
The performance of the IBVS control, developed for the mobile manipulator, is tested for different arm joint's angle values and mobile base positions. For all the simulations, the parameters chosen are:

- A time interval $dt = 0.1$;
- A control parameter $\lambda = 1$; and
- A condition error $\|e\| >= 0.01$ to stop the control cycle.

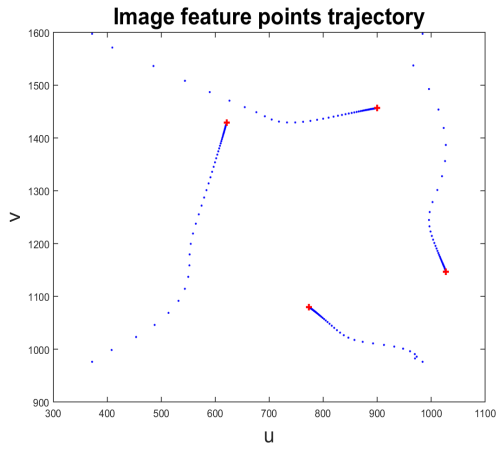
Tests are performed with four points. The position of the points has to be adapted according to the desired final position of the end-effector and camera (six distinct final position were considered). Simulations are initiated with the arm joints configuration $q = [0\ 0\ 0\ 0\ 0\ 0]$ and mobile platform position $p = [0\ 0\ 0]$. Camera



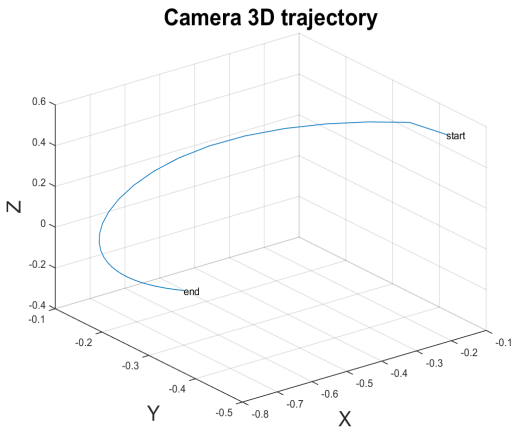
(a) Manipulator in initial configuration q. The actual camera frame (represented by the 3 arrows) does not match the red X-Y-Z axis representing the frame of the desired camera pose.



(b) Manipulator in final configuration qf, in a different view. The actual camera frame (represented by the 3 arrows) matches the red X-Y-Z axis representing the frame of the desired camera pose.



(c) Desired points (red crosses) and current points (blue points) trajectory in the image plane, in pixels.



(d) 3D camera trajectory.

Figure 4.7: Results of Puma 560 simulations with initial configuration $q = [-\frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]$ and final configuration $qf = [0\ -\frac{\pi}{2}\ -\pi\ -\frac{\pi}{6}\ -\frac{\pi}{6}\ -\frac{\pi}{3}]$.

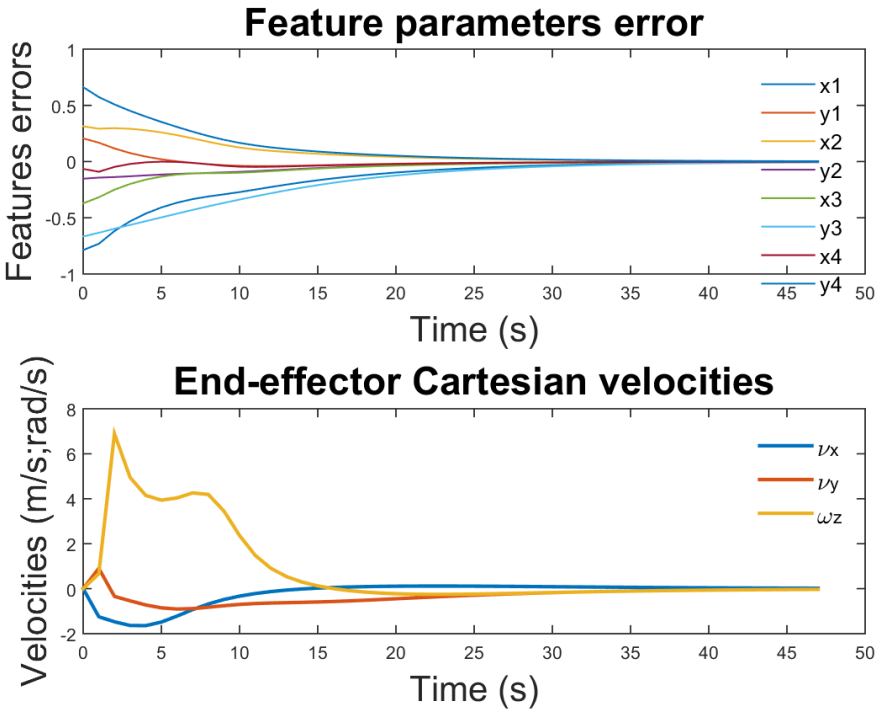


Figure 4.8: Features errors over time, where $(x_1, y_1, ..., x_4, y_4)$ are the 4 image points coordinates. End-effector linear and angular velocities (v_x, v_y, ω_z) over time.

trajectories for these simulations are represented in Fig. 4.9. All those 6 simulations have satisfactory results. We observed the convergence of the points in the image plan and errors decay exponentially to zero with time. Some observed 3-D trajectories are abrupt, with important movements of rotation, due to important changes of directions, in the beginning of the movement.

The mobile manipulator performance is also tested for different initial positions. One of these tests, which results are presented in Fig. 4.10 and Fig. 4.11, is performed with the points coordinates:

$$ {}^{W}P = \begin{bmatrix} 1.75 & 1.25 & 1.75 & 1.25 \\ 0 & 0 & 0 & 0 \\ 1.75 & 1.75 & 1.25 & 1.25 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.6} $$

The initial arm joints configuration q and mobile platform position p chosen for this test are:

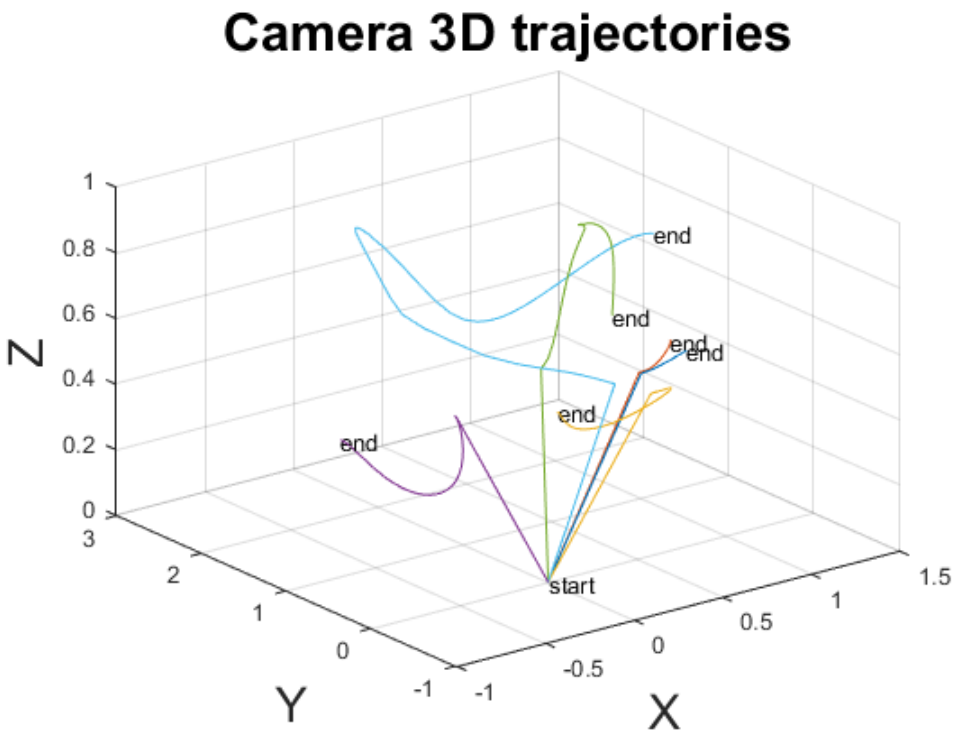


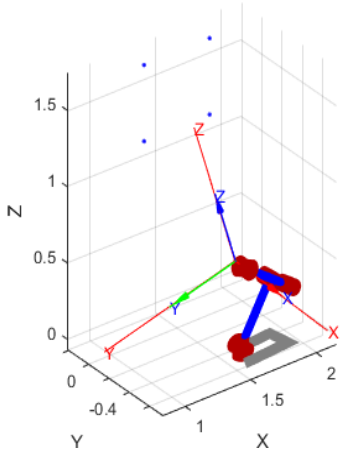
Figure 4.9: 3D camera trajectories for several desired final position of mobile manipulator. The initial and final camera positions are represented respectively by the label *start* and *end*.

- $q = [0\ 0\ \frac{\pi}{4}\ 0\ 0\ 0]$;
- $p = [0.5\ \ -1\ 0]$.

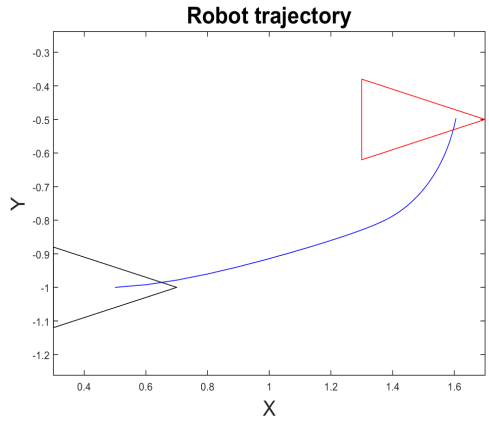
The final desired arm joints configuration qf and mobile platform position pf chosen are:

- $qf = [0\ \frac{\pi}{4}\ 0\ 0\ 0\ \frac{\pi}{3}]$;
- $pf = [1.5\ \ -0.5\ 0]$.

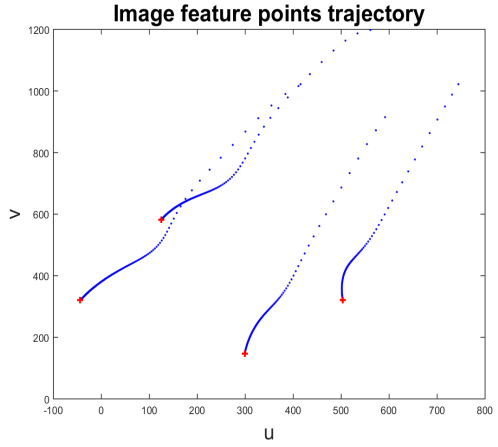
The features converge in the image plan (Fig. 4.10c) and the error between the current and desired features decay exponentially to zero (Fig. 4.11). The camera movements appear to be smooth during all the trajectory (Fig. 4.10d) and the mobile base get to the desired position, at the end of the control process (Fig. 4.10b).



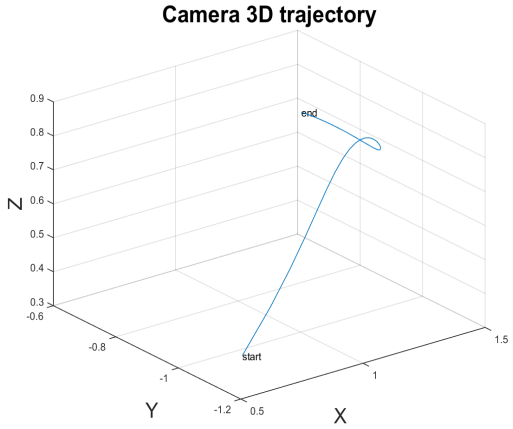
(a) Mobile manipulator in final configuration qf. The actual camera frame (represented by the 3 arrows) matches the red X-Y-Z axis representing the frame of the desired camera pose.



(b) Mobile manipulator base 2-D trajectory. The initial and final positions are represented respectively by the black and red triangle.



(c) Desired points (red crosses) and current points (blue points) trajectory in the image plane, in pixels.



(d) 3D camera trajectory.

Figure 4.10: Results of mobile manipulator simulations with initial configuration $q = [\frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]$ and position $p = (0, 0, 0)$. The final configuration $qf = [0\ \frac{pi}{4}\ 0\ 0\ 0\ 0]$ and position $pf = (0.5, 0.5, \frac{pi}{6})$.

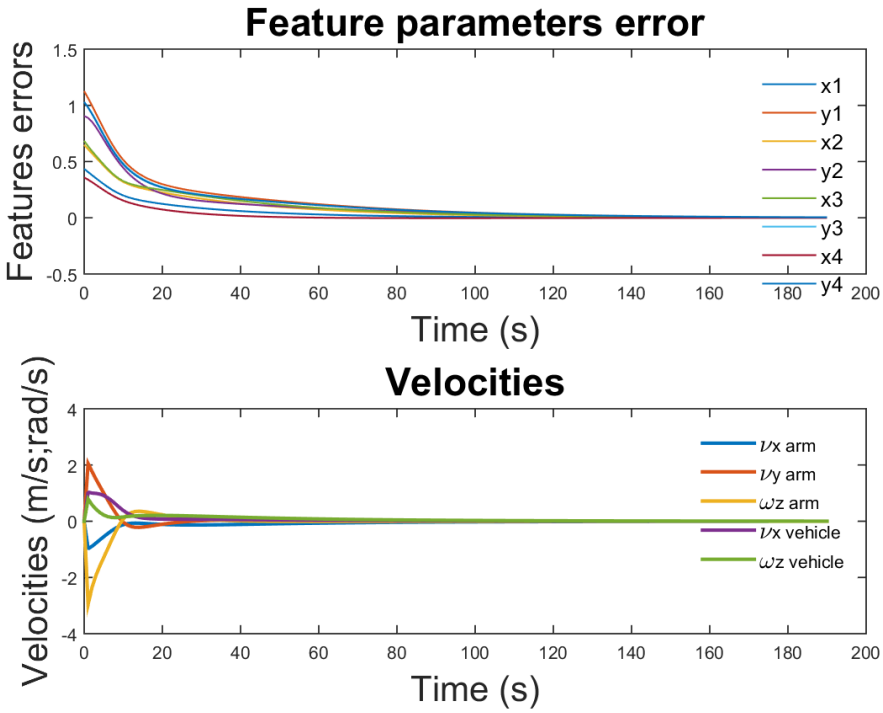


Figure 4.11: Features errors over time, where $(x_1, y_1, ..., x_4, y_4)$ are the 4 image points coordinates. Mobile platform and end-effector linear and angular velocities over time.

Given the success of the previous simulations, we want to test the control system with more than four points and a different depth. Point coordinates used are presented in matrix form in:

$$ {}^{W}P = \begin{bmatrix} 1.75 & 1.25 & 1.75 & 1.25 & 1 & 0.5 & 0.75 & 0.6 \\ 1.5 & 1 & 1.2 & 1 & 1.3 & 1.1 & 1.2 & 1.3 \\ 1.75 & 1.75 & 1.25 & 1.25 & 1 & 0.5 & 1.2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{4.7} $$

Initial arm joints configuration q and mobile platform position p chosen for this test are:

- $q = [\frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]$;
- $p = [0\ 0\ 0]$.

The final desired arm joints configuration qf and mobile platform position pf chosen are:

- $qf = [0\ \frac{\pi}{4}\ 0\ 0\ 0\ 0]$;
- $pf = [0.5\ 0.5\ \frac{\pi}{6}]$.

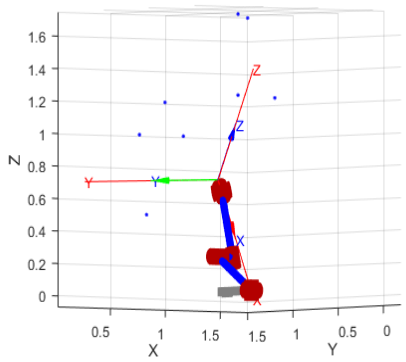
Results are shown in Fig. 4.12. The mobile manipulator is correctly controlled as we observe by the convergence of the points in the image plane (Fig. 4.12c), and the error is brought down to zero exponentially (Fig. 4.13). However, the mobile base is not in the final position at the end of the servoing. As the control is performed in the image, it exists some different positions for which the features converge in the image. Thus, results of the simulation appear to be satisfactory, even if initial high values of velocities are observed in Fig. 4.13, indicating abrupt changes of directions.

4.4 Discussion

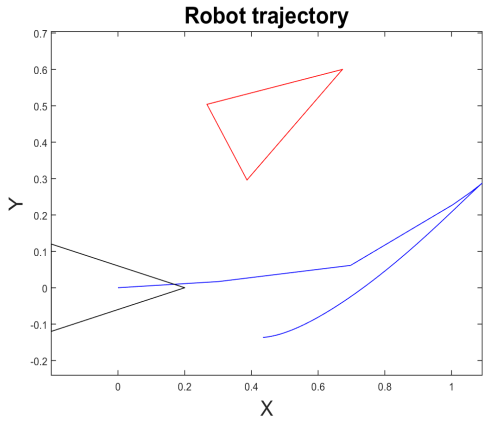
The results presented in previous sections (Sec. 4.1, Sec. 4.2, and Sec. 4.3) show an overall good performance of the IBVS applied to the mobile platform, to the manipulator and, finally, to the mobile arm. In the cases presented above, the convergence of the features in the image is always observed, with more or less smooth trajectories. Part of this success depends on the choice of the initial and final position, to project the points in the image. The Z-axis of the camera has to be oriented to the points.

In all the simulations, the convergence is observed after a reasonable amount of time, as the time interval chosen is of 0.1s. The longest time of convergence observed is 210s (Fig. 4.13), most probably due to the higher number of points in the image, which increases the time needed to compute the interaction matrix. Another factor could be the higher number of DOF on the mobile manipulator.

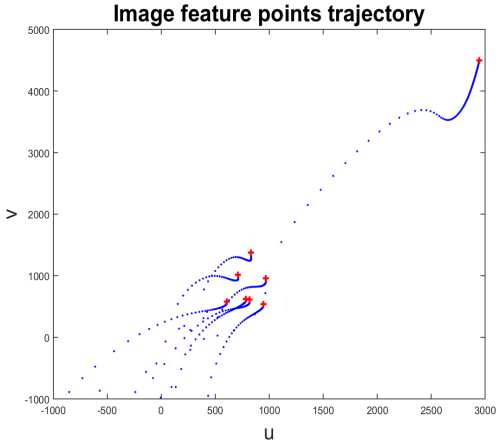
The overshoot in velocities curves are due to the important changes of robot direction and orientation, that occur at the beginning of the movement. In the mobile platform simulation, this phenomenon happens when the differences between the orientation of the desired final pose and the initial pose is greater than 90^{o} (Fig. 4.1). As the vehicle has only 2-DOF, it has to do an extra manoeuvre. This



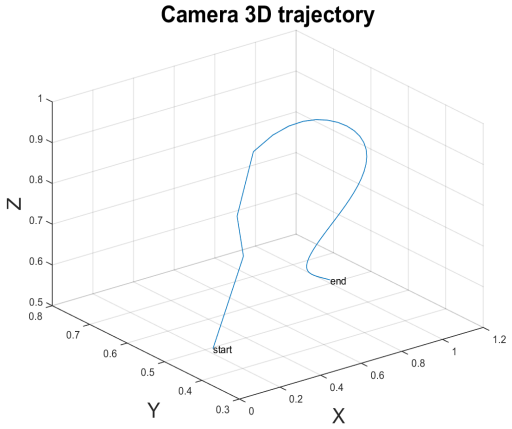
(a) Mobile manipulator in final configuration qf. The actual camera frame (represented by the 3 arrows) matches the red X-Y-Z axis representing the frame of the desired camera pose.



(b) Mobile manipulator base 2-D trajectory. The initial and final positions are represented respectively by the black and red triangle.



(c) Desired points (red crosses) and current points (blue points) trajectory in the image plane, in pixels.



(d) 3D camera trajectory.

Figure 4.12: Results of mobile manipulator simulations with initial configuration $q = [\frac{\pi}{2}\ 0\ 0\ 0\ 0\ 0]$ and position $p = (0, 0, 0)$. The final configuration $qf = [0\ \frac{\pi}{4}\ 0\ 0\ 0\ 0]$ and position $pf = (0.5, 0.5, \frac{\pi}{6})$.

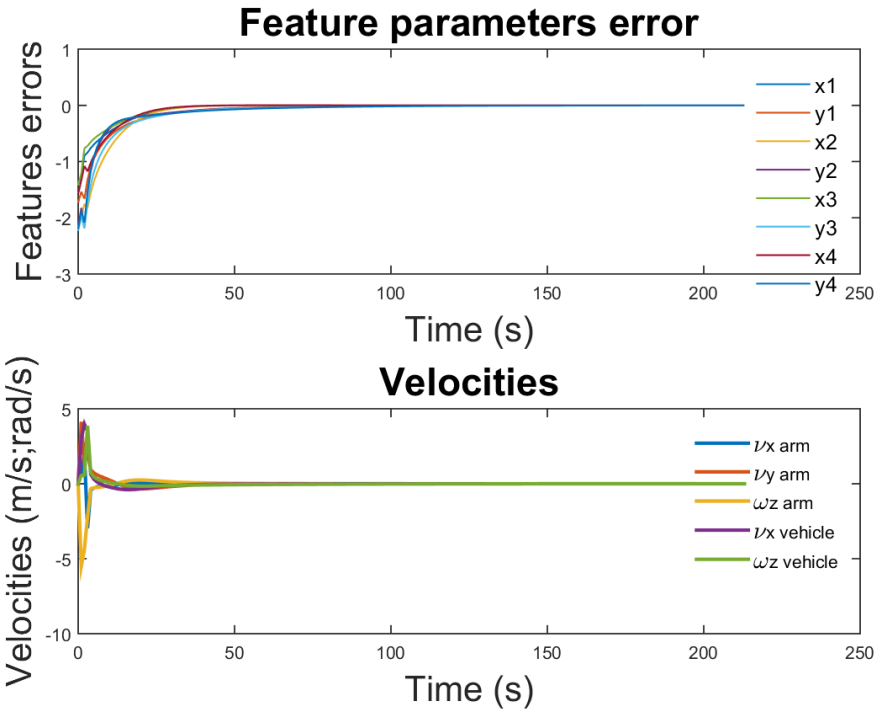


Figure 4.13: Features errors over time, where $(x_1, y_1, ..., x_4, y_4)$ are the image points coordinates. Mobile platform and end-effector linear and angular velocities over time.

could be avoided with a sliding movement of a 3-DOF omnidirectional platform. For the manipulator, the high velocity values can bring joints to the limit and cause damages to the manipulator.

When features converge in the image, for the mobile arm, it was observed that the mobile platform desired pose was not reached (Fig. 4.12b). As a matter of fact, the control is based on the error between the current and the desired features, in the image plane. Moreover, the mobile manipulator has redundant kinematics, since the camera has only 6-DOF (6 Cartesian velocities) to be controlled and the mobile arm has 8-DOF. Thus, the mobile arm base final pose, can be different from the desired one, as long as the norm of the error (between the current and desired image features) is less than 0,01. For each simulation, the rank of L_e^+ in Eq. (3.43), is verified at the end of the servoing, to check if the robot has lost any DOF, which has not been observed when the system converged. To sum up,

IBVS is successfully applied to the mobile manipulator, both when translational and rotational movements are needed.

During the implementation, the servoing was tested at each step, to let us identify cases where the system did not converge. On those cases, we verified that the chosen initial or desired pose were not adequate. In one of these pose (sometimes in both), points were not in the camera field of view and the depth of the points can not be estimated with precision.

In [44], only a local convergence is guaranteed. So, the system can be stuck in a local minimum, far away from the desired configuration. Thus, only local asymptotic stability can be obtained for IBVS. This phenomenon was observed several times for the vehicle servoing. Indeed, having only 2-DOF, it gets more complicated to ensure local asymptotic stability. Regarding the manipulator, convergence of the image features is almost always observed. As a matter of fact, the arm has 6-DOF which is theoretically sufficient to achieve any desired pose in a Cartesian taskspace, when using at least four points in the control scheme. However, global convergence can not be guaranteed either; the manipulator is not free from reaching joint limits or singularities and the arm can loose DOF in some configurations, meaning that the desired pose can not be achieved. Adding additional DOF, is one way to overcome this problem. [8].

Concerning the mobile manipulator, the redundancy is achieved by adding the 2-DOF of the mobile base. However, even that redundancy can not guarantee global convergence, as we observed the system reaching joints limits in some cases, or looses DOF in some configurations achieved during the servoing process.

Unfortunately, our system has some drawbacks regarding aspects that must be taking into account. The first important aspect that must be taken into account is the real camera field of view. In the present simulations, camera field of view was not limited. Only image dimensions limited the visualisation and projection of points. The second aspect that we have not treated is the robustness of the control system to random noise around the image points. Actually, when the camera is far from the desired points, the error between features is high and the noise is negligible, compared to the value of the error. The noise only influences the error when the camera is really close to the points. Last but not least, the third major aspect is the calculation of the points depth in the interaction matrix.

As a matter of fact, we considered the known depth of the points (given from the world coordinates matrix) at each iteration cycle, instead of estimating the depth. In a real environment, we have to estimate the depth of each point, which can be achieved by using stereo vision (a sequence of at least two images acquired by the robot movement). However, as we don't simulate noise around the points, the estimated depth, that would be given by the epipolar geometry, would be equal to the real depth of the points.

Chapter 5

Conclusion

This chapter concludes the work presented in this thesis and allows us to reflect about its development, considering its goals. To finish, we suggest some ideas for future work on this topic.

5.1 Conclusions

As stated on Chap. 1, the main goal of this thesis was to perform a control a mobile manipulator using visual information, acquired in real time, by a camera on top of the end-effector. The idea is to improve robots performance regarding objects manipulation. This progress can be useful as much as in industrial environments, as it is in domestic environments. In both, the robot is subjected to human interactions. Consequently, the servoing as to be as precise as possible and executed in real time, to avoiding possible obstacles.

After studying the VS state-of-the-art, IBVS appears as the adequate control scheme to compute errors directly from image features. Thus, this technique do not need neither a 3-D model of the scene, nor a perfect knowledge of the camera calibration matrix. Understanding the theoretical aspects of IBVS, image formation, and robotics (Chap. 2) was important to implement the simulator, to test the combined IBVS of a manipulator (Puma 560) and a differential drive vehicle (Chap. 3). We evaluated the performance of the servoing, at each step of the implementation. Results, in Chap. 4, were very satisfactory. The simulations also contributed to deepen of our knowledge about IBVS theory and robotics in general,

most specifically when applied to a mobile manipulator.

This work aims at contribute to the improvement of the mbot robot, present in the testbed in the North Tower 8th floor, in Alameda Campus, and used by the SocRob@Home team, in several competitions.

Unfortunately, we were not able to test the developed servoing for the mobile manipulator, in a real environment.

5.2 Future work

With the end of our study, we list a few points that (as a future work) could improve the results obtained in this thesis.

The IBVS control of the mobile manipulator could be improved by studying fine tuning parameters, to avoid possible local minima or singularities during the servoing, and to limit joint velocities preventing damages on the arm. Regarding the MATLAB simulator, in addition to the aspects mentioned in the discussion (Sec 4.4), we could implement the control scheme in Simulink, using blocks for each part of the servoing process. This improvement allied to a proper interface, can facilitate the use of the simulator; the servoing could be tested by choosing easily different inputs (number of points and coordinates of each one), changing the parameters variables (lower error for the tests, control parameter λ, time interval) or even choosing other robot model to be tested. For example, it would be interesting to create a model with a Cyton gamma arm, inserted in a pioneer robot, or even to use the mbot model, to approach the testing conditions of the real environment. More elaborated models, representing objects, instead of points could also be added.

Last, but not least, the ideal would be to include the IBVS directly on the mbot, to be used for objects manipulation, during competitions.

Bibliography

[1] J.Norberto Pires Martin Hägele, Klas Nilsson. *Field and Service Robotics*, volume 25, chapter Industrial Robotics, pages 964–969. Springer, 2014. vii, 4, 5, 6

[2] Simon Bøgh, Mads Hvilshøj, Morten Kristiansen, and Ole Madsen. Autonomous industrial mobile manipulation (aimm): from research to industry. In *42nd International Symposium on Robotics*, 2011. vii, 7

[3] F Janabi-Sharifi. Visual servoing: theory and applications. *Opto-Mechatronic Systems Handbook*, pages 15–1, 2002. vii, ix, 20, 21, 22, 24

[4] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. vii, 25, 27

[5] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer London, 2010. vii, 23, 26, 27

[6] *Control of unicycle type robots tracking, path following and point stabilization*. Citeseer, 2008. vii, 29

[7] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. Robotics: modelling, planning and control, ser. advanced textbooks in control and signal processing. *Springer*, 26:29, 2009. vii, 30, 32

[8] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011. vii, 29, 35, 36, 68

[9] Luca Iocchi, Javier Ruiz-del Solar, and Tijn van der Zant. Domestic service robots in the real world. *Journal of Intelligent & Robotic Systems*, 66(1):183–186, 2012. ix, 1, 2, 3

[10] François Chaumette and Seth Hutchinson. Visual servo control. ii. advanced approaches [tutorial]. *Robotics & Automation Magazine, IEEE*, 14(1):109–118, 2007. ix, 14, 16, 17, 18, 21, 22, 23, 24

[11] Sam Waller, Pat Langdon, and John Clarkson. Designing a more inclusive world. *Journal of Integrated Care*, 18(4):19–25, 2010. 1, 2

[12] Ja-Young Sung, Rebecca E Grinter, and Henrik I Christensen. Sketching the future: Assessing user needs for domestic robots. In *18th IEEE International Symposium on Robot and Human Interactive Communication (Toyama, JP*, pages 153–158, 2009. 2

[13] Amedeo Cesta, Gabriella Cortellessa, Vittoria Giuliani, Federico Pecora, Massimiliano Scopelliti, and Lorenza Tiberio. Psychological implications of domestic assistive technology for the elderly. *PsychNology Journal*, 5(3):229–252, 2007. 2

[14] Francesco Amigoni, Emanuele Bastianelli, Jakob Berghofer, Andrea Bonarini, Giulio Fontana, Nico Hochgeschwender, Luca Iocchi, Gerhard Kraetzschmar, Pedro Lima, Matteo Matteucci, et al. Competitions for benchmarking: Task and functionality scoring complete performance assessment. *Robotics & Automation Magazine, IEEE*, 22(3):53–61, 2015. 2

[15] Juanma De La Fuente, Julio Santiago, Antonio Román, Cristina Dumitrache, and Daniel Casasanto. *Handbook on Robotics*, volume 25. Springer, 2014. 3, 28, 32

[16] Tomas Lozano-Perez, Joseph L Jones, Emmanuel Mazer, Patrick A O'Donnell, Eric WL Grimson, Pierre Tournassoud, and Alain Lanusse. Handey: A robot system that recognizes, plans, and manipulates. *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, 4:843–849, 1987. 5

[17] Roger Bostelman, Tsai Hong, and Jeremy Marvel. Survey of research for performance measurement of mobile manipulators. *J. Natl. Inst. Stand. Technol.(2016, inpress)*, 2016. 5

[18] Kazuhiro Kosuge Erwin Prassler. *Field and Service Robotics*, volume 25, chapter Domestic Robotics, pages 1254–1256. Springer, 2014. 5, 6

[19] David J. Reinkensmeyer H.F. Machiel Van der Loos. *Field and Service Robotics*, volume 25, chapter Rehabilitation and Health Care Robotics, pages 1236–1238. Springer, 2014. 6

[20] Peter I Corke. Visual control of robot manipulators-a review. *Visual servoing*, 7:1–31, 1993. 7, 8, 14, 17, 18

[21] Alton L Gilbert, Michael K Giles, Gerald M Flachs, Robert B Rogers, and U Yee Hsun. A real-time video tracking system. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 47–56, 1980. 7

[22] TE Weber and RL Hollis. A vision based correlator to actively damp vibrations of a coarse-fine manipulator. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 818–825. IEEE, 1989. 8

[23] Lee Elliot Weiss, Arthur C Sanderson, and CP Neuman. Dynamic visual servo control of robots: an adaptive image-based approach. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pages 662–668. IEEE, 1985. 8

[24] PY Coulon and M Nougaret. Use of a tv camera system in closed-loop position control of mechanisms. In *Robot Vision*, pages 117–127. Springer, 1983. 8

[25] RC Harrell, DC Slaughter, and Phillip D Adsit. A fruit-tracking system for robotic harvesting. *Machine Vision and Applications*, 2(2):69–80, 1989. 8

[26] Rashpal S Ahluwali and Lynn M Fogwel. A modular approach to visual servoing. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 943–950. IEEE, 1986. 8

[27] Peter Corke and Richard Paul. Video-rate visual servoing for robots. In *Experimental Robotics I*, pages 429–451. Springer, 1990. 8

[28] Ernst Dieter Dickmanns and Volker Graefe. Applications of dynamic monocular machine vision. *Machine vision and Applications*, 1(4):241–261, 1988. 8

[29] R Bukowski, LS Haynes, Z Geng, N Coleman, A Santucci, K Lam, A Paz, R May, and M DeVito. Robot hand-eye coordination rapid prototyping environment. In *Proc. ISIR*, volume 16, 1991. 8

[30] G Skofteland and G Hirzinger. Computing position and orientation of a freeflying polyhedron from 3d. In *Proc. IEEE Conf. on Robotics and Automation*, 1991. 8

[31] Z Lin, V Zeman, and RV Patel. On-line robot trajectory planning for catching a moving object. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1726–1731. IEEE, 1989. 9

[32] DB Zhang, L Van Gool, and André Oosterlinck. Stochastic predictive control of robot tracking systems with dynamic visual feedback. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 610–615. IEEE, 1990. 9

[33] Peter K Allen, Billibon Yoshimi, and Aleksandar Timcenko. Real-time visual servoing. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 851–856. IEEE, 1991. 9

[34] John T Feddema and Owen R Mitchell. Vision-guided servoing with feature-based trajectory generation [for robots]. *Robotics and Automation, IEEE Transactions on*, 5(5):691–700, 1989. 9

[35] Won Jang and Zeungnam Bien. Feature-based visual servoing of an eye-in-hand robot with improved tracking performance. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2254–2260. IEEE, 1991. 9

[36] Nikolaos P Papanikolopoulos and Pradeep K Khosla. Shared and traded telerobotic visual control. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 878–885. IEEE, 1992. 9

[37] Frank Tendick, Jonathan Voichick, Gregory Tharp, and Lawrence Stark. A supervisory telerobotic control system using model-based vision feedback. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2280–2285. IEEE, 1991. 9

[38] Joseph S Yuan, Richard A MacDonald, and Felix HN Keung. Telerobotic tracker, July 17 1990. US Patent 4,942,538. 9

[39] Insitute for systems and robotics lisboa website, isrobonet, April 2016. Copyright 2015 David Afonso. 9

[40] Rockin website, April 2016. Copyright © 2013 RoCKIn project, contract no. FP7-ICT-601012. 9

[41] Joao Sequeira, Pedro Lima, Alessandro Saffiotti, Victor Gonzalez-Pacheco, and Miguel Angel Salichs. Monarch: Multi-robot cognitive systems operating in hospitals. In *ICRA 2013 workshop on many robot systems*, 2013. 10

[42] Cyton Gamma 1500 Arm Specifications, 2015. 10

[43] Maciej Staniak and Cezary Zieliński. Structures of visual servos. *Robotics and Autonomous Systems*, 58(8):940–954, 2010. 13

[44] François Chaumette and Seth Hutchinson. Visual servo control. i. basic approaches. *Robotics & Automation Magazine, IEEE*, 13(4):82–90, 2006. 14, 15, 16, 17, 18, 19, 20, 68

[45] Don Joven Agravante, François Chaumette, et al. Visual servoing for the reem humanoid robot's upper body. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5253–5258. IEEE, 2013. 14

[46] François Chaumette. Robot visual control. *Encyclopedia of Systems and Control*, pages 1188–1194, 2015. 14

[47] Francois Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The confluence of vision and control*, pages 66–78. Springer, 1998. 16

[48] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1843–1848. IEEE, 2004. 17

[49] Peter I Corke and Seth A Hutchinson. A new partitioned approach to image-based visual servo control. *Robotics and Automation, IEEE Transactions on*, 17(4):507–515, 2001. 17, 21, 23

[50] Ezio Malis. Survey of vision-based robot control. *ENSIETA European Naval Ship Design Short Course, Brest, France*, 2002. 18, 20

[51] Olivier Kermorgant and François Chaumette. Combining ibvs and pbvs to ensure the visibility constraint. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2849–2854. IEEE, 2011. 23

[52] Yannick Morvan. Multi-view video coding. 24

[53] Joao Sequeira. Introdução À robótica - caps 1 a 4 - monografia. Draft, Março 2011. 29, 30, 31, 32

[54] Peter I Corke. Toolbox, 2001. 30, 31

[55] P.I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, March 1996. 35

[56] Redwan Alqasemi and Rajiv Dubey. Combined mobility and manipulation control of a newly developed 9-dof wheelchair-mounted robotic arm system. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4524–4529. IEEE, 2007. 45

[57] Evangelos Papadopoulos and John Poulakakis. Planning and model-based control for mobile manipulators. In *Intelligent Robots and Systems,*

2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on, volume 3, pages 1810–1815. IEEE, 2000. 45

[58] Moslem Kazemi, Kamal Gupta, and Mehran Mehrandezh. Path planning for image-based control of wheeled mobile manipulators. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5306–5312. IEEE, 2012. 45

[59] Mustafa Mashali. *Kinematic Control of Redundant Mobile Manipulators*. University of South Florida, 2015. 45

Appendix A

Detailed interaction matrix calculation

Let $s = p = (x, y)$ be a point in the image plan, where $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$. By time differentiation, we obtain:

$$\begin{aligned} \dot{x} &= (\dot{X} - x\dot{Z})\frac{1}{Z} \\ \dot{y} &= (\dot{Y} - y\dot{Z})\frac{1}{Z} \end{aligned} \tag{A.1}$$

Relating the velocity of the 3-D point with the camera spatial velocity, we have:

$$\dot{X} = -v_c - \omega_c \times X \tag{A.2}$$

Becomes:

$$\begin{aligned} \dot{X} &= -v_x - \omega_y Z + \omega_z Y \\ \dot{Y} &= -v_y - \omega_z X + \omega_x Z \\ \dot{Z} &= -v_z - \omega_x Y + \omega_y X \end{aligned} \tag{A.3}$$

Relating A.1 and A.3, we obtain:

$$\begin{aligned} \dot{x} &= [-v_x - \omega_y Z + \omega_z Y - x(-v_z - w_x Y + w_y X)]\frac{1}{Z} \\ \dot{y} &= [-v_y - \omega_y X + \omega_x Z - y(-v_z - \omega_x Y + \omega_y X)]\frac{1}{Z} \end{aligned} \tag{A.4}$$

That becomes:

$$\begin{aligned} \dot{x} &= -\frac{v_x}{Z} - \frac{\omega_y Z + xX\omega_y}{Z} + \frac{\omega_z Y}{Z} + \frac{xv_z}{Z} + \frac{x\omega_x Y}{Z} \\ \dot{y} &= -\frac{v_y}{Z} - \frac{\omega_z X}{Z} + \frac{\omega_x Z + y\omega_x Y}{Z} + \frac{yv_z}{Z} - \frac{y\omega_y X}{Z} \end{aligned} \tag{A.5}$$

Recalling that $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$ and replacing on A.5, we obtain:

$$\begin{aligned} \dot{x} &= -\frac{v_x}{Z} - \omega_y(1+x^2) + \omega_z y + \frac{x v_x}{Z} + xy\omega_x \\ \dot{y} &= -\frac{v_y}{Z} - \omega_Z x + \omega_x(1+y^2) + \frac{y v_Z}{Z} - xy\omega_Y \end{aligned} \tag{A.6}$$

A.6 can be written as:

$$\dot{x} = L_x v_c \tag{A.7}$$

Finally, the iteraction matrix related to the point p is:

$$L_x = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & (1+y^2) & -xy & -x \end{bmatrix} \tag{A.8}$$