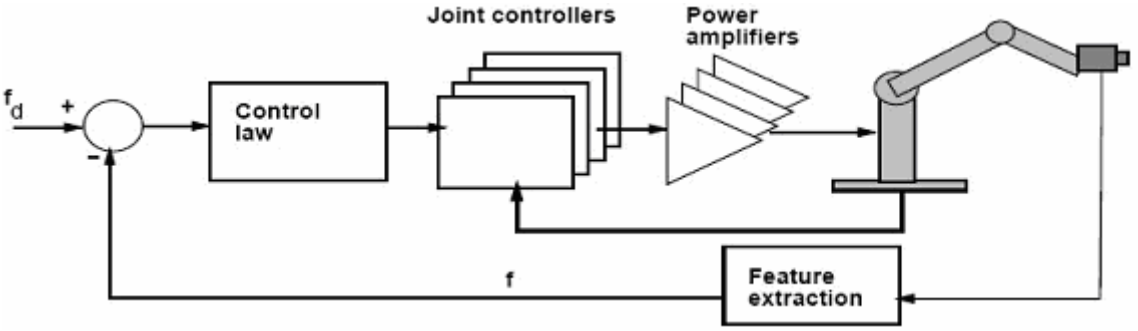
INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa



Evaluation of Visual Servoing Techniques for Robotic Applications

Ricardo Manuel Candeias da Silva Santos

Dissertation for obtaining the Degree of Master in
Electrical and Computer Engineering

Jury

President:	Doutor Francisco Miguel Prazeres da Silva Garcia
Supervisor:	Doutor Alexandre José Malheiro Bernardino
Assessor:	Doutor Carlos Jorge Ferreira Silvestre

September 2007

Evaluation of Visual Servoing Techniques for Robotic Applications

Ricardo Manuel Candeias da Silva Santos

Evaluation of Visual Servoing Techniques for Robotic Applications

Abstract

Control systems based on visual data input, termed visual servoing systems, are one of the most promising fields of research to develop a system as much autonomous as is possible.

In the last years, several methods have been proposed in this area, with their strengths and weaknesses. To clarify it, it is necessary to evaluate and compare them. In this thesis some visual servoing techniques are evaluated with the goal to elucidate in which situations each one of them is a good option to solve a specific task. The methods evaluated are: *position based visual servoing*, *image based visual servoing*, *2.5D visual servoing*, *decoupled visual servoing* and *shortest path visual servoing.*

Keywords

Visual servoing, robotic manipulator, computer vision, image noise, VISP.

Avaliação de Técnicas de Controlo Visual para Aplicações Robóticas

Resumo

O sistema de controlo baseado em imagem, denominado por controlo visual, é um dos mais promissores campos de investigação com vista a desenvolver um sistema o mais autónomo possível.

Nos últimos anos, vários métodos têm sido propostos nesta área, tendo estes os seus pontos fortes e fracos. Para melhor perceber como é o seu desempenho, é necessário avaliá-los e compara-los. Nesta tese, são avaliadas algumas técnicas de controlo visual com o objectivo de elucidar em que situações, cada uma delas é uma boa opção para resolver uma determinada tarefa. As técnicas avaliadas são: *controlo visual baseado em posição*, *controlo visual baseado em imagem*, *controlo visual 2.5D*, *controlo visual desacoplado* e *controlo visual segundo o caminho mais curto*.

Palavras-chave

Controlo visual, manipulador robótico, visão por computador, ruído de imagem, VISP

Preface

I would like to thank:

Kungliga Tekniska Högskolan of Stockholm (KTH) for all the conditions that the university gave me to do my project.

Danica Kragic and Ville Kyrki who helped me giving me all technical support needed.

Alexandre Bernardino for all the motivation and help that he gave me to do my project.

My parents and my sister for supporting me and encouraging me to take the experience of studying abroad.

Table of contents

Appendices

LIST OF FIGURES

List of Tables

List of Abbreviations

IBVS – Image based visual servoing;

PBVS – Position based visual servoing;

2.5D – 2.5D visual servoing;

CH – Corke & Hutchinson visual servoing;

SP – Shortest path visual servoing;

FPE – final pixels error;

NI – number of iterations;

MCD – maximum camera distance from the desired position;

MPE – maximum feature point excursion;

T – simulation time.

1 Introduction

In this section an introduction to the thesis is given. The field of research of the problem and the master thesis purpose are exposed. Some limitations of the present project and the outline of the thesis are also reported.

1.1 Background

Nowadays, technology is part of our lives. This has grown a lot in the last decades and it includes robotics. The systemization and automation of many processes have been possible only thanks to robots. In automation, one of the most important branches are manipulators, which can be built for the most diverse applications.

To recognize the environment in which a robot is in, it can use several kinds of sensors: sonars, lasers, contact sensors. These devices are normally built to get a very specific measure to solve a very specific task. A different kind of sensor that has been very developed in the last two decades is the video camera. The main reasons for that development in the last years are the reduction of the cameras cost and the increase of computers' processing speed, which now can run image processing algorithms in real time. The video camera has the advantage of allowing a bigger flexibility to the applicability of the data input and give the possibility to extract different kinds of information from the same sensor.

Visual data began to be used in robots control in the sixties, although at that time only in open-loop control, the processing restrictions did not allow much more. The close-loop control began to be viable only in the beginning of nineties. This kind of approach in robotics control is called *Visual Servoing*.

The main goal of the research in robots is to build a system as autonomous as possible. Even though visual data input is in its childhood, it already provides a certain level of independency to the robot. When could be possible to implement this approach with a higher accuracy could be viable, for example, putting robotic arms selecting the different kind of garbage for recycling, picking fruits, putting a vehicle of charge following a person or a car, creating a robotic arm able to change a damaged lamp on the road, without needs to hoist a man to do it and so on.

The flexibility, previously referred to classify the computer vision as data input, means that it could be easy to use the visual information to extract different kinds of information. However, it is equally advantageous, if the research done can be used easily, independently of the robot. In order to improve the *flexibility* and the *portability* of the research done in this area, in the last years a new software system has been developed, VISP. The VISP (VIsual Servoing Platform) has the goal of

integrating as much visual servoing tools as is possible and allowing a fast implementation of it in any specific robot with little work. This project was done using these C++ libraries, implementing some additional visual servoing techniques.

1.2 Description of the research area

In the last years, several new visual servoing techniques have been proposed trying to build control systems based on vision as good as possible. These methods are normally presented together with some experimentation to evaluate its performance. However, in some cases, those tests are not very exhaustive and they are done in comparison with a few other methods.

It is useful to have a global evaluation of the performance of several methods, under the same conditions. This kind of work was done by Gans, [2], however, more tests can be done, other features can be measured and other visual servoing methods can be evaluated.

1.3 Master Thesis Purpose

In this Master thesis, different kinds of visual servoing approaches are explored and their performances measured. The goal is to compare these methods and clarifying in which situations it is advantageous to implement each one of them. The first task of this thesis is to implement two visual servoing approaches on VISP, the *Corke & Hutchinson visual servoing*, [7], and the *shortest path visual servoing*, [5]. Then, the goal is to compare these, with other three methods already available on ViSP: *image based visual servoing*, [20], *position based visual servoing*, [20], and *2.5D visual servoing*, [1].

1.4 Limitations

With the goal to get results as close as possible to a real system, the tests were performed several times to try to remove outliers and get smooth results. A limitation found in simulations was the high computational complexity of the tests performed, which put some restrictions on their specifications. The tests were effectuated with a level of definition considered well enough for the parameters under evaluation. Another limitation was the high number of parameter to define, for example, the constant gains, the stop criterions, the camera parameters. The values that have showed a good average performance under all tests developed were chosen. However, if we would be interested in a small number of tests, for a specific task, these parameters could be chosen with a higher accuracy.

1.5 Outline

This thesis is organized as following. In Chapter 2 the generic tools of computer vision are exposed; it is explained the camera model used and how to estimate the camera displacement from two consecutive images. After that, Chapter 3 describes visual servoing and details the methods analyzed. It also exposes some recently developed work in this field, which could be improved using the knowledge provided by the studies undertaken in the current project. Subsequently, Chapter 4 includes the results obtained in this research. Finally, the conclusions and considerations regarding further research are presented in Chapters 5 and 6.

2 Computer Vision

For a robot operating in the field of industrial automation, whole environment that surrounds the robot is well-known. The challenge is developing an autonomous system that could act in one unknown environment, however, this is very far to be possible. To build a system like that, the first concern should be trying to get as much information as possible from the world. The sensor that best adapts to this requirement is vision. Vision acquisition also has the advantage of being able to be done in direct analogy with what is the main way that the humans extract information from the world.

Nowadays, the appearance of digital cameras allows an easy acquisition of images from the world and afterwards quickly links them to the robot's processor. After that, it is necessary to pre-process the image; this comprises typically noise-filtering, sharpening and restoration operations.

When the system obtains one image from the world, the 3D world is projected in a 2D dimension, which implies a consequent loss of information. The recourse to a second camera can minimize this loss of information, although that option brings other kind of problems.

When the image is acquired, it is necessary to identify the target. This could be hard whether there are many objects in the image or whether the background is textured. After localizing the target in the image, the object's features have to be extracted from it. These can be points, lines, circles, colors, grey levels, specific shapes. These features can be used, for example to localize the target in the world and perform pose estimation.

In this chapter, it is shown how to extract information from image data. First, we give one overview of the possible cameras system approaches that can be used; then the camera's model used in the work is explained. Finally, it is explained how to recover the camera displacement from two images using the *perspective projection* and the *homography matrix*.

2.1 Visual Information

The visual information can be obtained through monocular vision (one camera), stereo vision (two cameras) or a redundant system which uses several cameras, typically three or four, [31].

The first one, monocular vision, is the most popular because it is the cheapest and the most viable approach to run in real time. However, with only one camera, there is the problem that it is not possible measuring the depth. Stereo vision solves the former problem, but it introduces complexity in the algorithmic processing, which may not be tolerable in real time systems. Several cameras, are normally just used when a high accuracy is required and there is enough time for processing.

In this project only the monocular vision is approached. When only one camera is used, this can be done in two different ways: *Eye-to-hand Camera* **-** putting the camera looking to the end-effector and the target – and *Eye-in-hand Camera* – putting the camera in the last joint of the manipulator, close to its end-effector, looking to the target (see Figure 1).

Figure 1 – Computer Vision approaches using one camera: eye-in-hand (on left) and eye-to-hand (on right) configurations.

Eye-to-hand Camera

The eye-to-hand method was the first approach tried in visual servoing. This system requires an accuracy calibration because the control uses variables in the 3D Cartesian space; pose of the target in the relation to the camera and in relation to the end-effector. This computation increases the processing time which is undesired. Another problem in this method is that the robot cannot be collinear with the camera and the target, which would result in the object occlusion; in this sense, this adds one extra restriction in the robot displacement.

Eye-in-hand Camera

The eye-in-hand method is actually the most common approach in visual servoing because it gives more freedom to the robot than the other method. In this method it is necessary determine the transformation matrix between the end-effectors frame and the camera frame, although this is done only once, since both are strictly connected. This is the approach taken in this project.

2.2 Camera Modeling

The model of the camera used in this project is the pinhole camera model. This is one of the most commonly used model in computer vision due to its simplicity. This model is represented in Figure 2. The pinhole camera model assumes that the visual rays reflected from one object go through one hole – pinhole – and are projected in one planar surface – image plane.

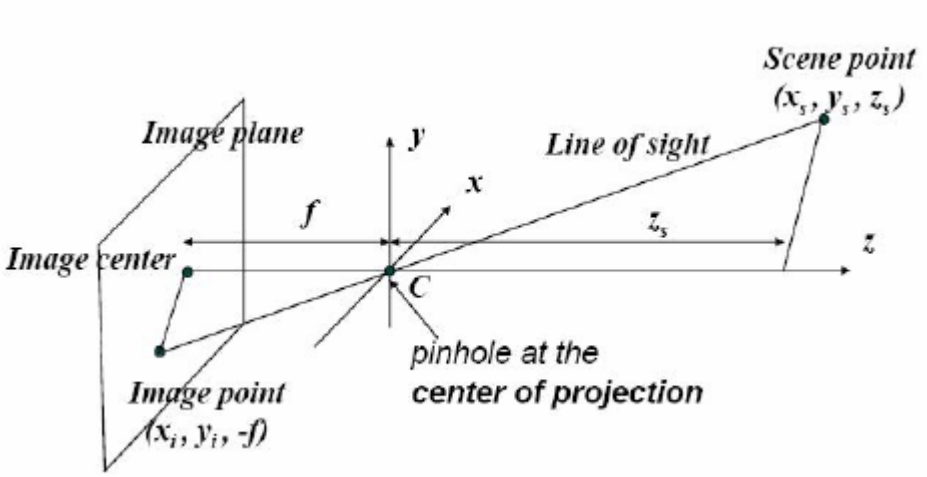


Figure 2 – Pinhole camera model.

In this model, the zero of z-axis lies in the center of projection and the image plane values have negative depth values. To avoid these negative values, this plane is displaced to the positive side of z-axis getting the model of Figure 3:

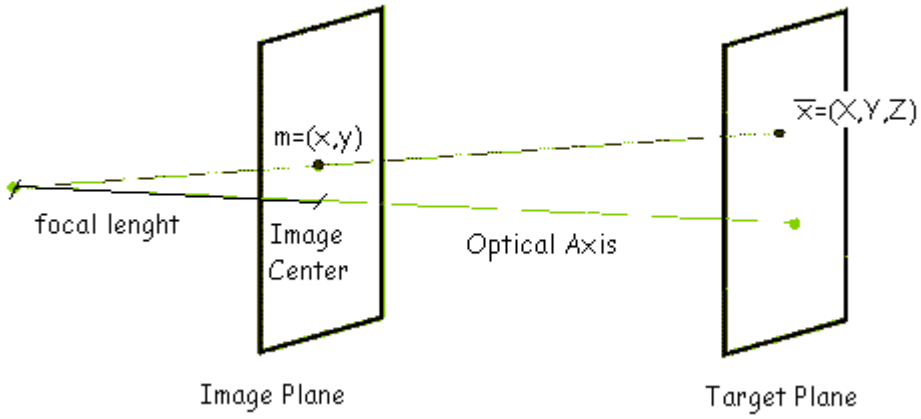


Figure 3 - Pinhole camera model with the image plane mirrored in the positive side of the optical axis.

In Figure 3 is possible to specify several definitions:

Focal length – distance between the image plane and the center of projection.

Optical axis – line that goes through the center of projection and is perpendicular to the image plane. This axis is also defined as collinear with the z-axis of the camera frame.

Image center – intersection between the optical axis and the image plane. In the literature it is also called *principal point*.

To define how is made the transposition from one point in the 3D space to the 2D image plane, it is used the perspective projection. The perspective projection has the advantage of could be specified as a linear relation, which makes the model easier to manipulate.

2.3 Perspective projection

The pinhole camera model associated to the perspective projection, [28], and [9], makes the transformation of points in Cartesian space to points in the image plane.

One point in the Cartesian space is represented in the world referential as ${}^0\bar{x} = (X, Y, Z)^T$. The metric coordinates of this point in the image plane are $\tilde{m} = (x, y, 1)^T$, which can be obtained by the relation

$$(x, y)^T = \left(\frac{f \cdot X}{Z}, \frac{f \cdot Y}{Z} \right)^T \quad \text{Equation 1}$$

and the image coordinates are $\tilde{p} = (u, v, 1)^T_{[pixels]}$

$$(u, v)^T = (x \cdot k_u, y \cdot k_v)^T \quad \text{Equation 2}$$

In which f is the focal length of the camera and k_u and k_v are the scaling factors along $\vec{x}$ and $\vec{y}$ axis. To facilitate the calculus, the focal length is defined equal to one, which does not interfere in the results. The relation of Equation 1 is non-linear and due to that, the homogeneous coordinates are used instead; ${}^0\bar{x} = (X, Y, Z, 1)^T$. The homogeneous coordinates have in the present case the advantage of allowing one linear approach which facilitates the data calculus.

The transformation of the point from the Cartesian space to the image plane, in homogeneous coordinates, is described by the following equations:

$$\lambda \tilde{m} = \left[{}^cR_0 \quad {}^ct_0 \right] {}^0\bar{x} \quad \text{Equation 3}$$

$$\tilde{p} = K \cdot \lambda \tilde{m} \quad \text{Equation 4}$$

In which λ is a gain factor, ${}^cR_0$ is the rotation matrix, [3x3], and ${}^ct_0$ is the displacement vector, [3x1], which represent the transformation from the world frame, (0), to the camera frame, (c).

K is the camera intrinsic parameters matrix and it makes the transformation between the normalized metric coordinates and the pixels coordinates. The matrix K is defined as:

$$K = \begin{bmatrix} f \cdot k_u & f \cdot k_\theta & u_0 \\ 0 & f \cdot k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$ Equation 5

In which k_θ is the distortion factor between both image directions. u_0 and v_0 are the coordinates of the principal point, the center of the image. As it can be seen in the definition of K, the value of the focal length can be incorporated in the image scaling factors. Then, it is possible to write the perspective projection matrix, P [pixels/meters].

$$\tilde{p} = K \cdot \begin{bmatrix} {}^{c}R_0 & {}^{c}t_0 \end{bmatrix} {}^{0}\bar{x}$$ Equation 6

$$P = K \cdot \begin{bmatrix} {}^{c}R_0 & {}^{c}t_0 \end{bmatrix}$$ Equation 7

2.4 Estimation of Homography Matrix

In the case of stereo vision, two cameras looking for the same target, the same point is projected in different places in both image planes. The homography matrix makes the transformation between the projections of one point in one image plane to another image plane. This transformation is defined by

$$\tilde{p}_2 = H_{12} \cdot \tilde{p}_1 <=> \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$ Equation 8

So, each point has the restrictions

$$x_2 = \frac{h_{11} \cdot x_1 + h_{12} \cdot y_1 + h_{13}}{h_{31} \cdot x_1 + h_{32} \cdot y_1 + h_{33}}$$ Equation 9

$$y_2 = \frac{h_{21} \cdot x_1 + h_{22} \cdot y_1 + h_{23}}{h_{31} \cdot x_1 + h_{32} \cdot y_1 + h_{33}}$$ Equation 10

which produce the relations

$$x_2 \left(h_{31} \cdot x_1 + h_{32} \cdot y_1 + h_{33} \right) = h_{11} \cdot x_1 + h_{12} \cdot y_1 + h_{13}$$ Equation 11

$$y_2 \left(h_{31} \cdot x_1 + h_{32} \cdot y_1 + h_{33} \right) = h_{21} \cdot x_1 + h_{22} \cdot y_1 + h_{23}$$ Equation 12

Then, it is possible to define for each point the relation

$$\begin{pmatrix} -h_{11}\cdot x_1 - h_{12}\cdot y_1 - h_{13} + h_{31}\cdot x_2\cdot x_1 + h_{32}\cdot x_2\cdot y_1 + h_{33}\cdot x_2 \\ -h_{21}\cdot x_1 - h_{22}\cdot y_1 - h_{23} + h_{31}\cdot y_2\cdot x_1 + h_{32}\cdot y_2\cdot y_1 + h_{33}\cdot y_2 \end{pmatrix} = 0 \qquad \text{Equation 13}$$

Imposing the restriction $h_{33} = 1$, it is possible to solve the Equation 8, if the relation of four points in both images is known, getting the following system of equations

$$\begin{bmatrix} x_1^1 & y_1^1 & 1 & 0 & 0 & 0 & -x_2^1\cdot x_1^1 & -x_2^1\cdot y_1^1 \\ 0 & 0 & 0 & x_1^1 & y_1^1 & 1 & y_2^1\cdot x_1^1 & -y_2^1\cdot y_1^1 \\ x_1^2 & y_1^2 & 1 & 0 & 0 & 0 & -x_2^2\cdot x_1^2 & -x_2^2\cdot y_1^2 \\ 0 & 0 & 0 & x_1^2 & y_1^2 & 1 & y_2^2\cdot x_1^2 & -y_2^2\cdot y_1^2 \\ x_1^3 & y_1^3 & 1 & 0 & 0 & 0 & -x_2^3\cdot x_1^3 & -x_2^3\cdot y_1^3 \\ 0 & 0 & 0 & x_1^3 & y_1^3 & 1 & y_2^3\cdot x_1^3 & -y_2^3\cdot y_1^3 \\ x_1^4 & y_1^4 & 1 & 0 & 0 & 0 & -x_2^4\cdot x_1^4 & -x_2^4\cdot y_1^4 \\ 0 & 0 & 0 & x_1^4 & y_1^4 & 1 & y_2^4\cdot x_1^4 & -y_2^4\cdot y_1^4 \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_2^1 \\ y_2^1 \\ x_2^2 \\ y_2^2 \\ x_2^3 \\ y_2^3 \\ x_2^4 \\ y_2^4 \end{bmatrix} \qquad \text{Equation 14}$$

Then, solving the former equation system, it is possible to get the homography matrix.

2.5 Estimation of Camera Displacement from the Homography Matrix

The homography matrix transforms a point from one image plane to another image plane. These images can be taken with two different cameras or, whether the target is static, with the same camera after a certain displacement. Assuming that the first camera is in the zero of the world, the two projection matrixes for the two cameras are:

$$P_1 = K_1 \cdot \begin{bmatrix} I_3 & 0_{3x1} \end{bmatrix} \qquad \text{Equation 15}$$

$$P_2 = K_2 \cdot \begin{bmatrix} {}^1R_2 & {}^1t_2 \end{bmatrix} \qquad \text{Equation 16}$$

To obtain the homography matrix from the previous projections, attending to Figure 4:

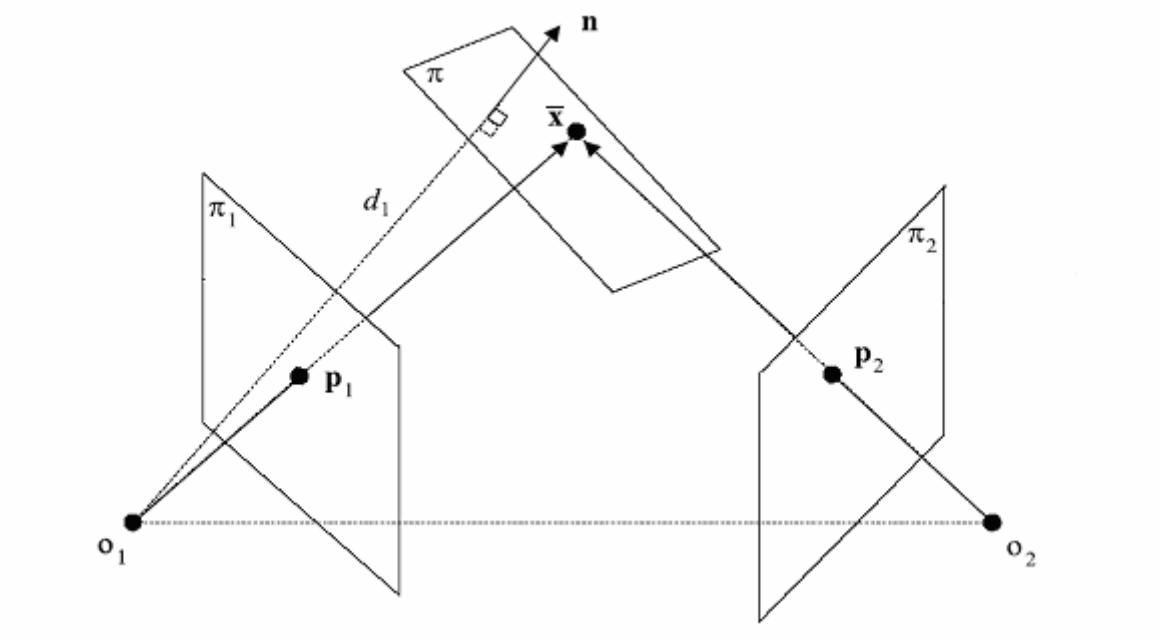


Figure 4 – Projection of one point in two image planes.

To simplify the task, the analysis is restricted to a planar object. The plane π is the object's plane which is not coplanar with any cameras' optical axis. This plane is defined by the normal *n* and the distance, *d*, from the plane until the optical center of camera.

Considering that $\bar{x}$ is one point in the object surface. That point is projected as $\tilde{p}_1$, $\tilde{p}_2$ in images' cameras planes. The transformation from the point $\tilde{p}_1$ to $\tilde{p}_2$ is made by homography matrix, H_{12}.

The point $\bar{x}$ verify projection equation of both cameras (Equation 6)

$$\lambda \tilde{p}_1 = K_1 \cdot \begin{bmatrix} I_3 & 0_{3x1} \end{bmatrix} \cdot \bar{x} \qquad \text{Equation 17}$$

and also the plane equation.

$$\begin{bmatrix} n^T & -d_1 \end{bmatrix} \cdot \bar{x} = 0 \qquad \text{Equation 18}$$

Where n^T is the normal vector of target's plane and d_1 is the distance from the first camera to the plane of the target (Figure 4).

Using matrix notation

$$\begin{bmatrix} \lambda \cdot K_1^{-1} \cdot \tilde{p}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} I_3 & 0_{3x1} \\ n^T & -d_1 \end{bmatrix} \cdot \bar{x} \qquad \text{Equation 19}$$

As $d_1 \neq 0$ and the plane π is not coplanar with the optical center of any of cameras, it is possible to isolate $\bar{x}$ in the Equation 19:

$$\bar{x} = \begin{bmatrix} I_3 & 0_{3x1} \\ \dfrac{n^T}{d_1} & -\dfrac{1}{d_1} \end{bmatrix} \cdot \begin{bmatrix} \lambda \cdot K_1^{-1} \cdot \tilde{p}_1 \\ 0 \end{bmatrix} = \lambda \begin{bmatrix} I_3 \\ \dfrac{n^T}{d_1} \end{bmatrix} \cdot K_1^{-1} \cdot \tilde{p}_1 \qquad \text{Equation 20}$$

Projecting this point in the second camera (Equations 6 and 16), the expression of the homography matrix is obtained:

$$\lambda \tilde{p}_2 = K_2 \begin{bmatrix} {}^1R_2 & {}^1t_2 \end{bmatrix} \cdot \begin{bmatrix} I_3 \\ \dfrac{n^T}{d_1} \end{bmatrix} \cdot K_1^{-1} \cdot \tilde{p}_1 = K_2 \cdot \left({}^1R_2 + {}^1t_2 \cdot \frac{n^T}{d_1} \right) \cdot K_1^{-1} \cdot \tilde{p}_1 \qquad \text{Equation 21}$$

$$\lambda \tilde{p}_2 = H_{12} \cdot \tilde{p}_1 \qquad \text{Equation 22}$$

$$H_{12} = K_2 \cdot \left({}^1R_2 + {}^1t_2 \cdot \frac{n^T}{d_1} \right) \cdot K_1^{-1} \qquad \text{Equation 23}$$

The camera displacement can be extracted from the homography matrix H (which it is represented in metric coordinates). To obtain H, it is used the relation:

$$H = K_2^{-1} \cdot H_{12} \cdot K_1 \qquad \text{Equation 24}$$

In Equation 23, it is shown that the homography matrix can be decoupled in translation and rotation displacement:

$$H = R + t \cdot \frac{n^T}{d_1} \qquad \text{Equation 25}$$

This mathematical operation expressed in Equation 25 is well known but not trivial and it is well explained in [10].

3 Visual Servoing

In this chapter, the theory about visual servoing used in this thesis is presented. Firstly is given an introduction and background about visual servoing. Then, the methods evaluated in the present work are defined. After that, some recent approaches developed to improve the efficiency of this kind of control system are discussed. At the end of this chapter some comparisons of some of the methods approached previously done are exposed.

3.1 Introduction

Visual servoing is the name given to the control techniques that use image data as input. It is a very large field of research and to build a system like that, it is necessary to join knowledge from many areas of research like robot modeling, control theory, computer vision and others, [20].

The main advantage of including vision in robotic systems is that it improves the flexibility and accuracy of these. In a visual servoing system, the robot should be able to extract from the image, all the environment's features which it needs to execute its task. There are two main approaches in visual servoing: image based visual servoing and position based visual servoing. Beyond these, there are also the hybrid systems which were developed to try to embody the best features of both approaches.

The two most common branches of application for visual servoing are: positioning of a robot in a desired position (typically with the goal of grasping some object) and following targets in movement, [32]. This thesis just considers the first one.

3.2 Background

The typical structure of a control system based in image as input is exposed in Figure 5

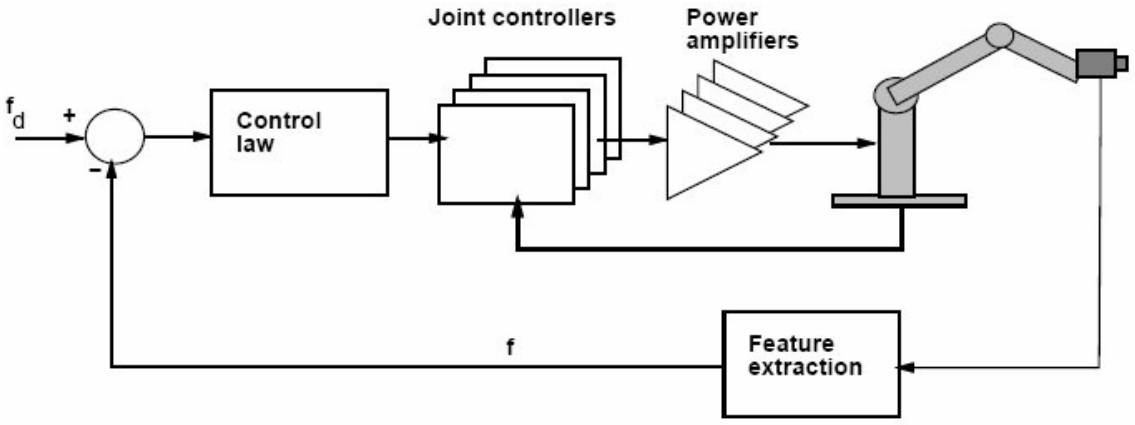


Figure 5 – Control scheme of a visual servoing system; image based visual servoing using the eye-in-hand approach.

The goal of a visual servoing system is to move a robot from one initial to one desired position, and keep it there using always a close-loop control scheme. In the beginning of the process, is provided to the system one image of the target with the camera in the desired position. Then, in the starting position, the camera obtains an image of the target. From that image, the necessary features, f, are extracted and compared with the desired ones extracted from the desired image, f_d; the error obtained from the difference between the desired and the current image features is used to compute the control law. The control signal is sent to the joints and a new image, with the camera in the new position, is taken. This is done until could be possible reducing to zero the error between the image features in the current and in the desired position.

Based on the image, it is possible to estimate parameters like the object's pose or the camera displacement and rotation. For that, the main challenge to solve is the image matching; how to know which point corresponds to which in two images. To solve the association problem, [9], many aspects need to be taken in account: whether the scene is textured, whether reflections or occlusion may occur, whether there are several objects in the scene, whether those are planar of curved. All these possibilities can make a big difference in the successes of the task, however, its study is out of the scope of this thesis. In the present work the target is assumed as perfectly identified.

The matching algorithms are basically divided in two approaches: correlation techniques (also called area based methods) and techniques based on object features.

The minimum sum of square differences method is used in the correlations methods, which tries to minimize the differences between the two images in a specific image window. It uses the following definition:

$$\min\left\{ SSD(u,v) = \sum_{n}\sum_{m}\left[I(u+m,v+n) - T(u,v)\right]^2 \right\} \qquad \text{Equation 26}$$

In which $I(u,v)$ is the current image, $T(u,v)$ the desired image and (m,n) a displacement along the (u,v) pixels directions.

In the case of techniques based on object features, those features are extracted from the images and then some similarity criterions are applied. The features based methods tend to work well when man-made objects, which have not a complex shape, are tracked. If it is desired to track a specific pattern, then the area methods have a better performance. The main problem of the area based method is that they only work well when the camera is almost perpendicular with the target plane, not performing well when the camera is too much inclined. Another problem of these area based methods is that their performance depends on the light conditions, which is a problem for environments that use natural light. If some occlusion in the scene can happen, the features methods are again more robust. In all cases, to get successful results it is necessary that both images are taken close. A big displacement between both images normally leads to loss of their correspondence. The area methods are not taken into account in this thesis and have been less used for generic applications.

For techniques based on object features, the first task to do, after an image is acquired, is to extract the information from it, *image processing*. That information can be points, lines, ellipses, differences in the texture, which basically means *edge detection*. This can be done using several methods; one possibility is to use the Laplacian operator, which detects changes in the grey levels of the image. This operator consists in convolving the image with the Laplacian mask matrix (Figure 6-a). An example of the application of this method can be seen in Figure 6.

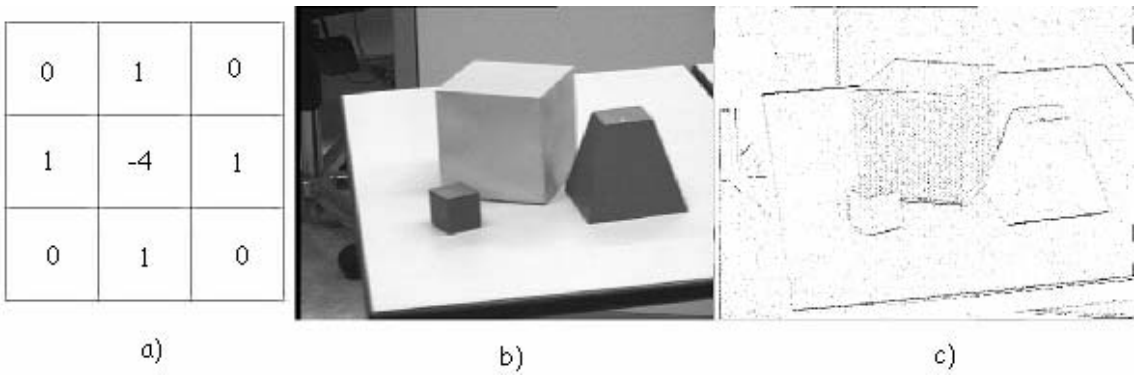


Figure 6 – Application example of the Laplacian operator to extract edges: a) Laplacian mask; b) Original image; c) Image after had been convolved with the Laplacian mask.

The main problem of this approach is that is very sensible to the noise. As can be seen in Figure 6-c, there are many isolated points (mainly in the biggest cube) which do not belong to any edge; beyond that if there is some occlusion in a part of the image, the edges do not appear connected. So it is necessary to define, which points correspond to real edges. For that, the starting point is an image

like in Figure 6-c, in which are defined the points which were identified as possible belonging to an edge. A way to search all the straight lines present in the image is finding all lines defined by every pair of points identified as a possibly belonging to an edge (Figure 6-c); it is then possible infer that the lines that appear represented more times correspond to the real edges. However, this algorithm it is not viable due to its high computational complexity. So instead of this, a better choice is using the Hough Transform which has the advantage that can be extended to detect curves. Due to the interest of the Hough Transform for visual servoing, it is described in the Appendix A.

It is possible to define several image feature parameters. An image feature parameter is a quantity value that can be extracted from each image. A good image feature is one that can be located unambiguously in different views of the scene. The choice of the image features (primitives) to use depends of the target and the image conditions. In [24] are given some hints to how to choose the most efficient features for a specific task. Only after had get those primitives, it is possible to start the task of interpretation of the surrounding robot's environment.

The extraction of features from the image can be done in a direct or indirect way. In the direct approach the variables to control lie in the 2D image plane coordinates, *image based visual servoing* (Figure 5). In the indirect approach it is necessary to extract the relevant features of the object from the image plane, and then, use a 3D model of the camera, and the target, to compute the position of the target and the camera in the world, *position based visual servoing*. In this second case, an extra block should be added in Figure 5 for *pose estimation* after the *feature extraction block* in the feedback loop.

In pose estimation, the position and orientation of the target in relation to the camera is computed. This is done from the object image and the *a-priory* knowledge it has of the object. There are several ways to build a model of the tracked target, which depend mainly of the object features. For objects with a well-defined and geometric shape the most common approach is use a wire-frame representation to define it; Figure 7-a ([31]), gives one example of this representation. An overview of modeling elementary polygons is given by Vincze in [25].

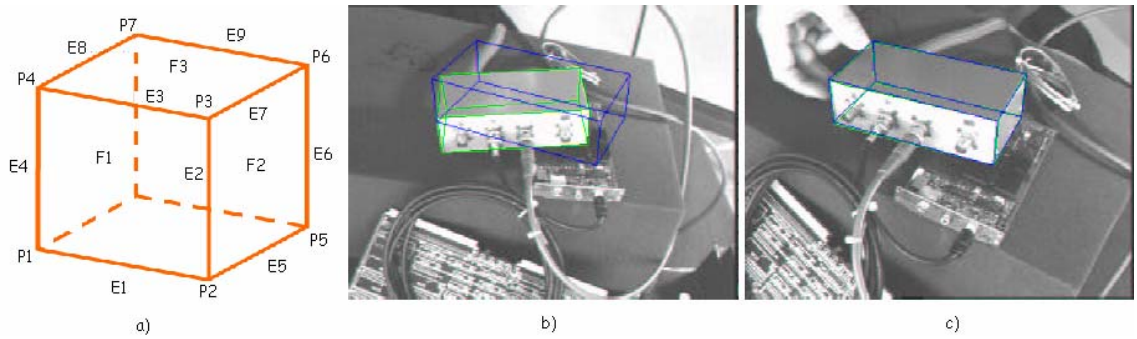


Figure 7 – Object model for position based visual servoing. a) Possible representation of the object tracked (P – points, E – edges, F - polygons); b) Projection of the object model in a not final position; c) Projection of the object model in the final position.

To estimate the position of the object in relation to the camera, the object model is projected in the image (only the visible edges), Figure 7-b (example extracted from [18]); then, some algorithms try to match the object edges obtained from the current image with the object model projection. Then, from the object model, it is possible to infer the spatial position of the object. Some of these matching algorithms are presented in [26] and [27]. In the desired, position the object edges and the object model projection should coincide, Figure 7-c. A survey of model-based approaches can be found in [15].

The goal of pose estimation is to have a fast, accurate and robust algorithm. However, normally it is not possible to get all these specifications at the same time. An analysis of the tradeoff between speed, accuracy and robustness in position based visual servoing methods was performed by Wilson in [29]. The pose estimation algorithms can be analytic, [22], and numerical, [23], and the last can be divided in iterative linear or non-linear.

The visual servoing control law structure is almost the same for all the methods, changing basically the features used for tracking and the quantity of object *a priori* knowledge used. The definitions presented in this chapter are generic for all the methods and their notation is subsequently presented.

The *feature vector*, s, is defined by the list of features, extracted directly or indirectly from the image, used in the control law. The *features error*, or *task vector*, e, is the difference between the current features vector and the desired features vector

$$e = s_{current} - s^{*}_{desired} \qquad \text{Equation 27}$$

For a static target, it is obtained the equality $\dot{e} = \dot{s}_{current}$.

To ensure the zeroing of the features errors in the task space, it is imposed a restriction in the control law defined by the following first order equation

$$\dot{e} = -\lambda \cdot e \qquad \text{Equation 28}$$

In visual servoing is necessary to make a transformation between the Cartesian space and the feature space. This is done by the *image Jacobian*, J, also called *interaction matrix,* L_s, which relates both dimensions

$$J_q = \frac{\partial e}{\partial q} \qquad \text{Equation 29}$$

The vector $q_{[1x6]}$ represents the position of the joints and ∂q their velocities.

Inverting the relation

$$\dot{e} = \dot{s} = J \cdot v = L_s \cdot v \qquad \text{Equation 30}$$

it is possible to obtain the control law

$$v = -\lambda \cdot J^{+} \cdot \left(s_{current} - s^{*}\right) \quad \text{Equation 31}$$

The difference between Equation 29 and Equation 30 is that the first is expressed in the joint space, $\left(q, J_q\right)$, and the second in the Cartesian space, $\left(v, J\right)$. The velocity, v, is a [1x6] vector, in which the first three components are associated with the translation, and the last three with the rotation, $v = \begin{bmatrix} v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{bmatrix}$. Typically J is a not square matrix, so in Equation 31, it is necessary to use the pseudo-inverse of J.

The velocity obtained in Equation 31 refers to the position of the camera, although normally, for example to pick up an object, it is desired control the position of the end-effector of the robotic arm instead. For that, it is necessary to change the reference point from the camera to the end-effector. This can be done with a transformation matrix, ${}^{e}V_c$, defined by

$${}^{e}V_c = \begin{bmatrix} {}^{e}R_c & \left[{}^{e}t_c\right]_x \cdot {}^{e}R_c \\ 0_3 & {}^{e}R_c \end{bmatrix} \quad \text{Equation 32}$$

Here, ${}^{e}R_c$ and ${}^{e}t_c$ are the rotation and translation between the camera frame and the end-effector frame, and $\left[{}^{e}t_c\right]_x$ is the anti-symmetric matrix associated to the vector ${}^{e}t_c$. The estimation of this matrix can be done offline.

To transform the Cartesian velocities, v, in joint velocities, q, it is necessary to use the robot Jacobian, J_R

$$v = J_R \cdot \dot{q} \quad \text{Equation 33}$$

In this sense, the image Jacobian of the end-effector, in the joint space, is given by

$$J_q = J \cdot {}^{e}V_c \cdot J_R \quad \text{Equation 34}$$

The control law given in the joint coordinates and in the end-effector frame is given by

$$\dot{q} = -\lambda \cdot J_q^{+} \cdot \left(s_{current} - s^{*}\right) \quad \text{Equation 35}$$

In visual servoing have been used a proportional controller, which it is enough to solve the task. The problem in the visual servoing are not related with the choice of the control (proportional, PD, PID) but with the other problems cited along this thesis.

3.3 Evaluated Methods

In this thesis were analyzed and evaluated five visual servoing methods. They are the *position based visual servoing* (PBVS), [20], the *image based visual servoing* (IBVS), [20], the *2.5D visual servoing* (2.5D), [1], the *Corke & Hutchinson visual servoing* (CH), [7], and the *Kyrki and Kragic Shortest Path* (SP), [5]. For an easier notation, in the rest of the thesis, the cited methods are called by their short-cuts in brackets.

3.3.1 Position Based Visual Servoing

In position based visual servoing, the features are extracted indirectly from the image; it is also called 3D visual servoing; the data is transposed from the 2D image plane to 3D Cartesian space. The image is used to localize the object in the world.

PBVS has some disadvantages: it is necessary to have a thorough knowledge about the target to build a specific model of it, which is only possible to apply to objects with a specific shape; this also reduces the applicability of the algorithm; other disadvantage is that, due to the control calculus are done in the Cartesian space, the system requires an accuracy calibration; another problem is that this computation, in the Cartesian space, also requires a high computational effort, which reduces the applicability of the algorithm in a real time system. Beyond this, there is not any control over the image plane, as could be seen in the end of current chapter, which means that the target might leave the camera's field of view. The stability analysis in this method is also hard to do.

On the other hand the main advantages of PBVS are the accuracy and robustness of its performance.

In PBVS, the feature vector is defined as

$$s = \begin{bmatrix} {}^{c}t_{c^*} \\ {}^{c}u_{c^*}\theta_{3x1} \end{bmatrix} \qquad \text{Equation 36}$$

${}^{c}t_{c^*}$ and ${}^{c}u_{c^*}\theta_{3x1}$ represent the translation and rotation from the initial to the desired position; in this last feature, ${}^{c}u_{c^*}$ are the rotation axis and θ_{3x1} are the rotation angles obtained from the rotation matrix. These features can be extracted from the transformation coordinates from the current camera's position to the desired, ${}^{c}T_{c^*}$. That is obtained from the composition of the transformation coordinates from the current position to the target with the desired position to it

$$ {}^{c}T_{c^*} = {}^{c}T_{o}\,{}^{o}T_{c^*} = \begin{bmatrix} {}^{c}R_{o} & {}^{c}t_{o} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{o}R_{c^*} & {}^{o}t_{c^*} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{c}R_{o}\,{}^{o}R_{c^*} & {}^{c}R_{o}\,{}^{o}t_{c^*} + {}^{c}t_{o} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{c}R_{c^*} & {}^{c}t_{c^*} \\ 0 & 1 \end{bmatrix} $$

Equation 37

In PBVS, the translation and rotation of the camera (end-efector) are computed separately and are given by

$$ {}^{c}\dot{t}_{c^*} = \begin{bmatrix} I_3 & 0_3 \end{bmatrix} \cdot v $$ Equation 38

$$ {}^{c}\dot{u}_{c^*}\theta_{3x1} = \begin{bmatrix} 0_3 & L_w \end{bmatrix} \cdot v $$ Equation 39

With

$$ L_w({}^{c}\dot{u}_{c^*}, \theta_{3x1}) = I_3 - \frac{\theta}{2} \cdot [u]_x + \left(1 - \frac{\sin c(\theta)}{\sin c^2\left(\frac{\theta}{2}\right)} \right) \cdot [u]_x^2 $$ Equation 40

$$ \sin c(\theta) = \begin{cases} 1 & \theta = 0 \\ \dfrac{\sin(\theta)}{\theta} & \theta \neq 0 \end{cases} $$

Then, the position based Jacobian can be defined as

$$ \dot{s} = \begin{bmatrix} {}^{c}\dot{t}_{c^*} \\ {}^{c}\dot{u}_{c^*}\theta_{3x1} \end{bmatrix} = \begin{bmatrix} I_3 & 0_3 \\ 0_3 & L_w \end{bmatrix} \cdot v = \begin{bmatrix} I_3 & 0_3 \\ 0_3 & I_3 \end{bmatrix} \cdot v = J_{3D} \cdot v $$ Equation 41

because $L_w^{-1}(u, \theta)$ can be approximate by the identity matrix, [10].

It is not strange that the image Jacobian is an identity matrix because both variables related by it, $(\dot{s}, v)$, are defined in the Cartesian space.

And then, the respective control law is

$$ v = -\lambda \cdot J_{3D}^{-1} \cdot \dot{s} $$ Equation 42

3.3.2 Image Based Visual Servoing

In image based robot control, the target features are extracted directly from the image, without any image interpretation. The features obtained from the object are primitives like ellipsis, lines, points or

moments. This control architecture does not need to make a 3D reconstruction of the target. This allows a reduction of the processing time of the algorithm and increases the sample rate which also reduces some computer vision problems. How to obtain the interaction model of IBVS is now presented.

Defining a generic point in the 3D space as:

$$\bar{x} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T \quad \text{Equation 43}$$

The velocity of that point in relation of a referential in movement with velocity v is

$$\dot{\bar{x}} = \begin{bmatrix} -I_3 & [\bar{x}]_x \end{bmatrix} \cdot v_{[1x6]} = \begin{bmatrix} \dot{X} & \dot{Y} & \dot{Z} \end{bmatrix}^T \quad \text{Equation 44}$$

In which $[\bar{x}]_x$ is the anti-symmetric matrix associated to the vector $\bar{x}$. A point in the image is defined in metric coordinates as

$$m = \begin{bmatrix} x & y \end{bmatrix}^T = \begin{bmatrix} \frac{X}{Z} & \frac{Y}{Z} \end{bmatrix}^T \quad \text{Equation 45}$$

Then, the derivation of this point in order to the time is

$$\dot{m} = \begin{bmatrix} \dot{x} & \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{\dot{X} \cdot Z - X \cdot \dot{Z}}{Z^2} \\ \frac{\dot{Y} \cdot Z - Y \cdot \dot{Z}}{Z^2} \end{bmatrix} = \frac{1}{Z} \cdot \begin{bmatrix} 1 & 0 & -\frac{X}{Z} \\ 0 & 1 & -\frac{Y}{Z} \end{bmatrix} \cdot \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \quad \text{Equation 46}$$

$$\dot{m} = -J_v(m) \cdot \dot{\bar{x}} \quad \text{Equation 47}$$

With

$$J_v(m) = \frac{1}{Z} \cdot \begin{bmatrix} -1 & 0 & \frac{X}{Z} \\ 0 & -1 & \frac{Y}{Z} \end{bmatrix} \quad \text{Equation 48}$$

Using the Equations 44 and 47, it is possible to obtain the relation

$$\dot{m} = -J_v(m) \cdot \begin{bmatrix} -I_3 & [\bar{x}]_x \end{bmatrix} \cdot v \quad \text{Equation 49}$$

$$\dot{m} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{X}{Z^2} & x \cdot y & -\left(1 + x^2\right) & y \\ 0 & -\frac{1}{Z} & \frac{Y}{Z^2} & 1 + y^2 & -x \cdot y & -x \end{bmatrix} \cdot v \quad \text{Equation 50}$$

In metric images coordinates:

$$\dot{m} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & x \cdot y & -\left(1+x^2\right) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -x \cdot y & -x \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$ Equation 51

$$\dot{m} = -J \cdot v = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{x}{Z} & -x \cdot y & \left(1+x^2\right) & -y \\ 0 & \frac{1}{Z} & -\frac{y}{Z} & -\left(1+y^2\right) & x \cdot y & x \end{bmatrix} \cdot \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$ Equation 52

Then, applying this approach to the IBVS, the feature vector, s, is defined as

$$s = \begin{bmatrix} x \\ y \end{bmatrix}$$ Equation 53

The variables x and y are the point's image plane coordinates and the error vector is

$$\dot{s} = \begin{bmatrix} x - x^* \\ y - y^* \end{bmatrix}$$ Equation 54

The relation between the features error and the velocity to send to the robot is

$$\dot{s} = J \cdot v$$ Equation 55

$$v = -\lambda \cdot J_{2D}^{+} \cdot \dot{s}$$ Equation 56

The Jacobian presented in the Equation 52 has the dimension of [2 x 6]. This is the Jacobian size to track one single point. For n points, the Jacobian dimension would be [n 2 x 6].

It is possible to compute the image Jacobian defined in Equation 52, in some different ways. The first possibility is to use at each iteration the real values of x, y and Z. The last parameter is hard to be measured only from the visual data information, although when any other information is provided, one way to solve this problem is to use a constant likelihood value, which normally performs correctly. Regardless of providing the most efficient computation of the Jacobian, using the real values at each iteration requires a great computation effort. Another way to compute the image Jacobian is to use a constant Jacobian; this can be computed with base on the information of the initial position which can cause large and unnecessary movements of the camera in its path; or with base on the desired

position, which sometime, can make the algorithm diverge; typically using the average value of both values $\left(\left(L_s^i + L_s^*\right)/2\right)$ results better than using only the information based in the initial or in the desired position. Another possibility is to use learning method to estimate the Jacobian (on-line or of-line), [16]. In this project the image Jacobian is always computed based on the real values of the image pixels and depth value, at each iteration.

To ensure the convergence of the control law, it is necessary that

$$J_{2D} \cdot J_{2D}{}^{+} > 0$$

This depends of the depth values. This means that for a Jacobian updated at each iteration, only a local convergence can be ensured. This local convergence is one of the problems of IBVS. The second main problem is the difficulty of this method in managing rotations around the optical axis. As can be seen in the experimental results of this thesis, each point in the image plane tries to make a straight line into the desired position. Because of that, in the case of rotations around the optical axis, the camera needs to retreat, and the correspondent displacement is as high as the rotation is close to 180º; for this value, the camera theoretically needs to retreat until the infinite – this problem in IBVS is known as Chaumette Conundrum. A global stability analysis of IBVS and also PBVS was investigated by Chaumette in [21].

To compute the control law for image based visual servoing, it is necessary to have at least three non-collinear points. However, doing it with a higher number of points increases its robustness.

In general, the accuracy in the image-based methods for static positioning is less sensitive to calibration than position based methods due to does not use the homography to compute the control law.

3.3.3 2 ½ D Visual Servoing

The 2.5D visual servoing method was presented by Malis [1] in 1999 and was developed with the goal of joining the best skills of the two main visual servoing approaches (IBVS and PBVS).

This hybrid method does not need any 3D model of the object and ensures the convergence of the control law in the whole task space. This method itself does not ensure that the target does not leave the camera's view field, which could happen mainly when the initial position of the camera is very far away from the target; however, this problem can be addressed including some additional techniques in the control law.

In the 2.5D, the task vector is defined as $e_{[6x1]} = \left(x, y, \log(\frac{Z}{Z^*}), {}^{c}u_{c*}\theta_{3x1} \right)$.

In which, x and y are the metric coordinates of a point, Z the current distance of the camera to that point and u_{3x1} and θ_{3x1} are the axis of the rotation and the angles that the camera needs to rotate.

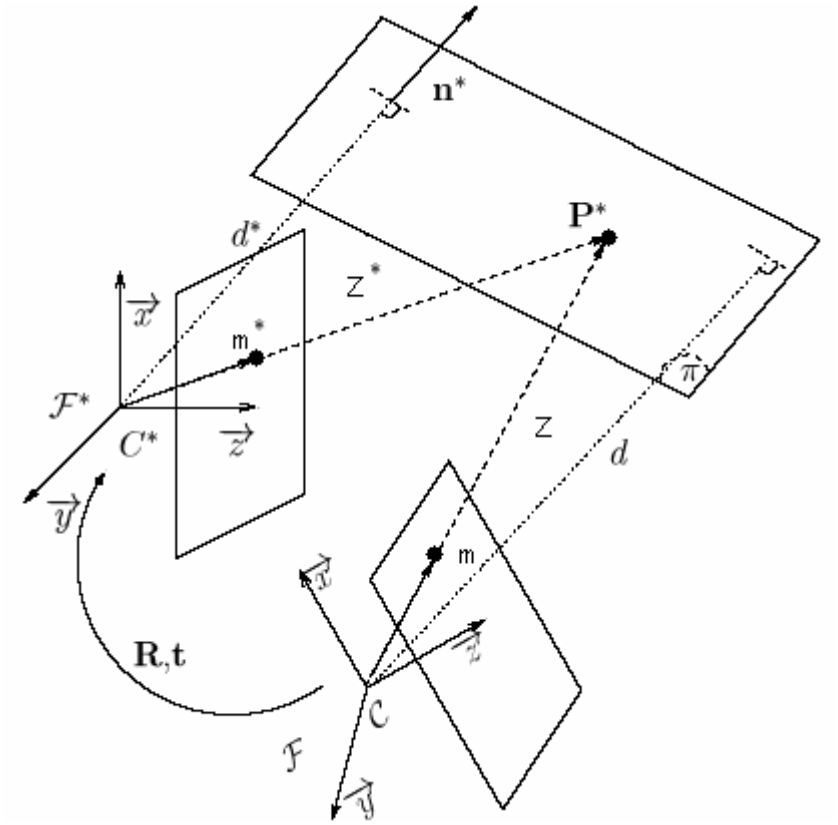


Figure 8 – Variables used in 2.5D visual servoing control law.

The velocity of one point, which lies on the target plane, π, relatively to one referential in movement, with velocity, v, is given by

$$\dot{\bar{x}} = \begin{bmatrix} -I_3 & [\bar{x}]_x \end{bmatrix} \cdot v \qquad \text{Equation 57}$$

In which $[\bar{x}]_x$ is the ant-symmetric matrix associated to the vector $\bar{x}$.

To control the orientation the ${}^{c}u_{c^*}\theta_{3x1}$ approach is used, which has the advantage of having no singularities in whole workspace. The time derivative of ${}^{c}u_{c^*}\theta_{3x1}$ (v) is:

$${}^{c}\dot{u}_{c^*}\theta_{3x1} = \begin{bmatrix} 0 & L_w \end{bmatrix} \cdot v \qquad \text{Equation 58}$$

In which L_w is defined as

$$L_w({}^c\dot{u}_{c^*}, \theta_{3x1}) = I_3 - \frac{\theta}{2} \cdot [u]_x + \left(1 - \frac{\sin c(\theta)}{\sin c^2\left(\frac{\theta}{2}\right)} \right) \cdot [u]_x^2 \qquad \text{Equation 59}$$

The extended image coordinates of a point in the target can be defined as

$$m_e = \begin{bmatrix} x & y & z \end{bmatrix}^T = \begin{bmatrix} \frac{X}{Z} & \frac{Y}{Z} & \ln(Z) \end{bmatrix}^T \qquad \text{Equation 60}$$

With the depth defined as $z = \log(Z)$. The time derivative of these coordinates is

$$\dot{m}_e = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T = \begin{bmatrix} \frac{\dot{X} \cdot Z - X \cdot \dot{Z}}{Z^2} \\ \frac{\dot{Y} \cdot Z - Y \cdot \dot{Z}}{Z^2} \\ \frac{\dot{Z}}{Z} \end{bmatrix} = \frac{1}{Z} \cdot \begin{bmatrix} 1 & 0 & -\frac{X}{Z} \\ 0 & 1 & -\frac{Y}{Z} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = -\frac{1}{d^*} \cdot L_v \cdot \dot{\bar{x}} \qquad \text{Equation 61}$$

With

$$L_v = \frac{1}{\rho} \cdot \begin{bmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{bmatrix} \qquad \text{Equation 62}$$

$$\rho = \frac{Z}{d^*} \qquad \text{Equation 63}$$

In which d^* represents the distance to the target in the desired position, see Figure 8.

Then, using the Equation 57, the time derivative of the coordinates is given by

$$\dot{m}_e = \frac{1}{d^*} \cdot L_v \cdot \begin{bmatrix} -I_3 & [x]_x \end{bmatrix} \cdot v \qquad \text{Equation 64}$$

$$\dot{m}_e = \begin{bmatrix} \frac{1}{d^*} L_v & L_{v,w} \end{bmatrix} = \begin{bmatrix} \frac{1}{d^*} \cdot \begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{pmatrix} & \begin{matrix} xy & -(1+y^2) & y \\ (1+y^2) & -xy & -x \\ -y & x & 0 \end{matrix} \end{bmatrix} \cdot v \qquad \text{Equation 65}$$

From Equations 58 and 65 it is possible to get the expression of the features error and the features error derivate.

$$e = \left[\left(m_e - m_e^* \right)^T \quad {}^c u_{c^*} \theta_{3x1} \right]^T$$ Equation 66

$$\dot{e}_{6x1} = J_{2.5D} \cdot v_{6x1}$$ Equation 67

With

$$J_{2,5D} = \begin{bmatrix} \frac{1}{d^*} \cdot L_v & L_{(v,w)} \\ 0 & L_w \end{bmatrix}$$ Equation 68

Then, the control law is:

$$v = -\lambda \cdot J_{2.5D}^{-1} \cdot e$$ Equation 69

$J_{2,5D}$ is a upper triangle matrix which its inverse is not hard to compute and $L_w{}^{-1}(u,\theta)$ can be approximate by the identity matrix, [10].

The control law is then defined as:

$$v = -\lambda \cdot \begin{bmatrix} \hat{d}^* \cdot L_v{}^{-1} & \hat{d}^* \cdot L_v{}^{-1} \cdot L_{(v,w)} \\ 0 & I_3 \end{bmatrix} \begin{bmatrix} m_e - m_e^* \\ {}^c u_{c^*} \theta_{3x1} \end{bmatrix}$$ Equation 70

3.3.4 Corke & Hutchinson Visual Servoing

The Corke and Hutchinson method, [7], was developed to address some of the IBVS problems, mainly the camera retreat problem, which occurs for pure rotations around the camera's optical axis. The idea of this image based partitioned method is to separate the control translation and rotation over the z-axis from the other degrees of freedom. The CH method has the drawback that its performance depends of the relative position of the features in the image.

The interaction model is defined, as in all the other methods, as

$$\dot{e} = J_{C\&H} \cdot v$$ Equation 71

In this case the error vector is defined as in IBVS

$$e = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} x - x^* \\ y - y^* \end{bmatrix}$$ Equation 72

Only one point is tracked, and (x , y) are their image plane coordinates.

In this method the translation, v_z , and the rotation, ω_z , velocities along/around z-axis are computed from the next features:

Theta Feature

To drive the camera inclination around the optical axis it is used the angle theta, θ. It is defined as the angle between the horizontal pixel axis and a straight line built from two points in the image, as is shown in Figure 9.

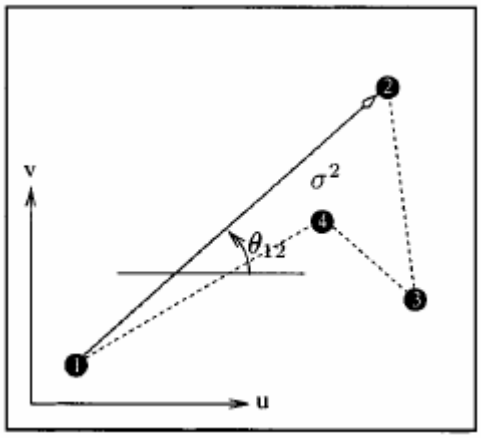


Figure 9 – Features used in the Corke and Hutchinson visual servoing control law.

This can be done with any pair of points, but the accuracy of the feature increases as far as the points lie on the image.

Sigma Feature

The control thought the z-axis is computed from the sigma-square feature, σ^2, which consists in the area of a polygon made from the image points; for that at least three not collinear points are needed. To use a feature with the unities of a length (in pixels) instead of the area, the feature used is the square root of the area of that polygon, $\sigma = \sqrt{area}$. If the camera is too far from the target (in relation of the desired position), the area, σ^2, of the polygon is smaller and the camera needs to move forward, if the area is too big, this needs to retreat.

The translational and rotational velocities along/around the optical axis are given by:

$$\begin{bmatrix} v_z \\ \omega_z \end{bmatrix} = \begin{matrix} \lambda_{Tz}\left(\sigma^* - \sigma\right) \\ \lambda_{\omega z}\left(\theta_{ij}^* - \theta_{ij}\right) \end{matrix} \qquad \text{Equation 73}$$

λ_{Tz} and $\lambda_{\omega z}$ are gain factors to adapt the speed of conversion along the associated directions. Then, it is possible to write the Equation 71 as

$$\dot{e} = -J_{xy} \cdot v_{xy} - J_z \cdot v_z$$ Equation 74

Where v_{xy} and v_z are the velocities associated to the respective indexes: $v_{xy} = \begin{bmatrix} v_x & v_y & \omega_x & \omega_y \end{bmatrix}$, $v_z = \begin{bmatrix} v_z & \omega_z \end{bmatrix}$, and J_{xy} and J_z are the colons of the image based visual servoing Jacobian associated with the respective indexes (Equation 52)

$$J_{xy} = \begin{bmatrix} -\frac{1}{Z} & 0 & x \cdot y & -\left(1+x^2\right) \\ 0 & -\frac{1}{Z} & 1+y^2 & -x \cdot y \end{bmatrix}$$ Equation 75

$$J_z = \begin{bmatrix} \frac{x}{Z} & y \\ \frac{y}{Z} & -x \end{bmatrix}$$ Equation 76

Then, the Equation 74 can be written in order of v_{xy}:

$$v_{xy} = -\left[J_{xy}^{+}\right]_{4x2} \left\{ \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \begin{bmatrix} \frac{x}{Z} & y \\ \frac{y}{Z} & -x \end{bmatrix} \cdot \begin{bmatrix} \dot{\sigma} \\ \dot{\theta} \end{bmatrix} \right\}$$ Equation 77

In the paper where this method was presented, [7], few tests were shown. I those tests, this technique performed perfectly for pure rotation around the optical axis. For generic motions, this method performed little better than IBVS, despite of the excursion of the features in the image plane are more complex than in IBVS. However these tests did not include high rotations around an axis coplanar with the target plane, which could make a big distortion in the image features. Those tests were also performed under perfect condition without a presence of any extern perturbation like noise or calibration errors.

3.3.5 Shortest Path Visual Servoing

The shortest path visual servoing was presented by Kyrki and Kragic, [5], with the goal of addressing two typical problems in the visual servoing: the possibility of the target leaving the camera's field of view and the risk of the joints working too close to their limits. This method was developed to the camera presents one straight line trajectory which goes from the initial to the desired position. This approach also solves the problem of the uncertain in the trajectory. The uncertainty of the trajectory occurs in several methods and in some cases can put their success in risk.

This method can be implemented as a 3D visual servoing method or a hybrid method. Here the mathematical description of the first is presented, which is implemented in this project. The difference between both approaches is just the way that the depth information is obtained. In the case of the hybrid approach, the depth estimation can be done as in the 2.5D visual servoing (Equation 63).

In the shortest path 3D, the position based control is used directly to control the translation of the camera. To control the rotation around x and y axis is used one virtual point which lies in the zero of the object frame; this point helps the control scheme to keep the target in the image. The rotation over the optical axis is controlled using the ${}^{c}u_{z\,c^*}\theta$, as in 2.5D visual servoing, which should be driven to zero.

The coordinates of the virtual point (center of the target plane) in the camera frame are $\left({}^{c}X_o, {}^{c}Y_o, {}^{c}Z_o\right)$ and (x, y) are the image coordinates of the same point.

The projection of the virtual point in the image plane is

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{{}^{c}Z_o} \cdot \begin{pmatrix} {}^{c}X_o \\ {}^{c}Y_o \end{pmatrix} \qquad \text{Equation 78}$$

And its velocity in the image screen is related with the camera velocity, v, by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = -\begin{pmatrix} -\frac{1}{{}^{c}Z_o} & 0 & \frac{x}{{}^{c}Z_o} & x \cdot y & -\left(1+x^2\right) & y \\ 0 & -\frac{1}{{}^{c}Z_o} & \frac{y}{{}^{c}Z_o} & \left(1+y^2\right) & -x \cdot y & -x \end{pmatrix} \cdot v \qquad \text{Equation 79}$$

The task vector is $e_{[6x1]} = \left({}^{c}X_{c^*}, {}^{c}Y_{c^*}, {}^{c}Z_{c^*}, x, y, {}^{c}u_{z\,c^*}\theta\right)^T$ and the transformation from the initial position to the desired can be obtain, as in PBVS, from the composition of the transformation coordinates from the current position to the target with the desired position to it

$${}^{c}T_{c^*} = {}^{c}T_o \, {}^{o}T_{c^*} = \begin{bmatrix} {}^{c}R_o & {}^{c}t_o \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{o}R_{c^*} & {}^{o}t_{c^*} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{c}R_{c^*} & {}^{c}t_{c^*} \\ 0 & 1 \end{bmatrix} \qquad \text{Equation 80}$$

The interaction model is given by

$$\dot{e}_{[6x1]} = \left({}^{c}\dot{X}_{c^*}, {}^{c}\dot{Y}_{c^*}, {}^{c}\dot{Z}_{c^*}, \dot{x}, \dot{y}, {}^{c}\dot{u}_{z\,c^*}\theta\right)^T = J_{SP} \cdot v \qquad \text{Equation 81}$$

The image Jacobian, J_{SP} is defined as

$$J = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -\frac{1}{Z} & 0 & \frac{x}{Z} & x \cdot y & -\left(1+x^2\right) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -x \cdot y & -x \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{Equation 82}$$

In which Z is the distance to the virtual point, ($^{c}Z_{o}$).

Expressing it in the Cartesian coordinates $\left(X,Y,Z\right)^{T} = \left(^{c}X_{o}, ^{c}Y_{o}, ^{c}Z_{o}\right)^{T}$

$$J = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -\frac{1}{Z} & 0 & \frac{X}{Z^2} & \frac{X \cdot Y}{Z^2} & -\left(1+\frac{X^2}{Z^2}\right) & \frac{Y}{Z} \\ 0 & -\frac{1}{Z} & \frac{Y}{Z^2} & 1+\frac{Y^2}{Z^2} & -\frac{X \cdot Y}{Z^2} & -\frac{X}{Z} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{Equation 83}$$

Then, the control law is given by

$$v = -\lambda \cdot J_{SP}^{-1} \cdot e \qquad \text{Equation 84}$$

Due to the simple mathematical definition that J_{SP} has, it is possible to define directly its inverse, J_{SP}^{-1}, which facilitate its implementation

$$J_{SP}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -\frac{X \cdot Y}{Z \cdot P} & \frac{X^2 \cdot Z^2}{Z \cdot P} & -\frac{Y}{P} & -\frac{X \cdot Y}{P} & \frac{X^2+Z^2}{P} & \frac{X}{Z} \\ -\frac{Y^2+Z^2}{Z \cdot P} & \frac{X \cdot Y}{Z \cdot P} & \frac{X}{P} & -\frac{Y^2+Z^2}{P} & \frac{X \cdot Y}{P} & \frac{Y}{Z} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{Equation 85}$$

With $P = \frac{X^2+Y^2+Z^2}{Z^2}$.

Attending in J_{SP} matrix, its determinant is $|Z| = \frac{X^2 + Y^2 + Z^2}{Z^2}$, which it is never zero, except when the camera is on the zero of the object frame, which it is not possible. This ensures global convergence of the method.

In the paper where this method was presented, [5], its performance is compared with two other shortest path methods: PBVS and Deguchi visual servoing, [6]. The simulations were done with six points on the target plane. Due to the goal of this thesis, only the results of the comparison between SP and the PBVS are shown.

For rotations around the optical axis, both methods had almost the same convergence time and the same feature excursion, i.e. maximum distance of a point from the center of the image. The same conclusions were gotten for a translation along the optical axis.

For a rotation around an axis perpendicular to the optical axis in the camera frame, both methods were a similar performance. The execution time was proportional to the rotation and in that case, the camera translation was almost zero.

For a rotation around an axis coplanar with the target plane (feature plane rotation), the SP had a little better performance than PBVS in execution time and in maximum features excursion.

For a generic motion the SP had also a better performance than PBVS in execution time and maximum features excursion; the difference is bigger with the increase of the initial angle of rotation; in these cases, in PBVS, the points often leave the camera's field of view.

All these evaluations were done for a perfect system, without a presence of any perturbation or an unknown parameter.

3.4 Related Work

The main goal of this thesis is to evaluate some visual servoing techniques in order to help improving the robustness of a control system based on image. Due to this, it is made one overview of some visual servoing approaches presented recently. These try to develop a more robust visual servoing system and could be helped to be developed and tested with the work developed in this thesis.

The image based visual servoing seems to be the most versatile method due to require less information to compute the control law. However, this also has some limitations and several partitioned or hybrid methods have appeared to try to solve them.

One way to build a more efficient image based system is to use the most efficient method in each situation. This means, to switch the control, depending on the task conditions. This requires having two control levels: a low level to compute the control law, where can be implemented several visual servoing methods in pipeline; a high level responsible to decide which of the methods should be used at each instant. In this case the global stability of the system is hard to ensure, however, under certain conditions, it is possible to prove its stability, [13], and [14]; for that, the first restriction is to ensure a soft change between both controllers.

This approach was tested by Gans, [12], with two controllers: one based on a homography method and other based on affine approximation.

In presence of noise the affine-approximation has a better performance than the controller based on the homography, although the first cannot be used for a motion over the x or y axis, only for displacements though the camera's optical axis.

In those tests, for the switching between the controllers, three rules were tested: the deterministic switch, in which, when the normal image plane vector, n, gets close enough to the vector $[0,0,1]^T$, the affine-approximation controller is used; the random switch, in which, at each iteration the controller to use is randomly chosen, having both equal probability to be elected; the biased random switch, in which, the probability of each controller be chosen depends on how close the vector n is from the configuration $[0,0,1]^T$. The simulations showed that the biased random switch approach performs better than the others.

In [36], it is implemented the same switching idea. In this case, it is addressed the possibility of the target leaving the camera's field of view. It also uses two controllers, chosen in function of where the points lie in the screen: whether all the points lie far enough from the boarder of the screen it is used the PBVS to compute the control; whether one of those points get out of a sub area in the center of the screen, the camera is steered by using some translation and rotation control law, which move the camera in order to push the points to inside that sub area. In the referred paper the stability, performance and the robustness of the proposed method are also analyzed.

The initial approach, in the beginning of the studies in visual servoing, was constraining all the degrees of freedom of the robot to the same task. However, in the beginning of the process, this is not necessary. The typical situation is that some features' errors arrive to the desired position (zero) before others, which implies some kind of waste of resources. Beyond that, the classical approaches choose a trajectory without knowing whether that is valid, this means, whether there are some kinds of restrictions on the path or, more recently, when this is done, the robot is not able to adapt to environment changes.

A more efficient way to use the system's resources is, in the beginning of the movement, when the robot is typically far away from the target, to use some degrees of freedom of the robot to perform secondary tasks. These called "secondary tasks" can be used mainly to improve the robustness of the control. These can include, for example, avoiding the joints limits, ensuring that the target remains in the camera's field of view or avoiding target occlusions. When the robot is close to the goal, then, it uses all the degrees of freedom of the robot to reach the desired situation. This approach is called *task sequencing*.

To use this approach it is necessary to ensure that all subtasks have different priorities and a secondary task does not disturb one which has a higher importance. To do it, it is used the *redundancy formalism* which is now explained, [3] and [37]. In the following deductions the Jacobians are always expressed in the joints space.

The main task e_1 and the secondary task e_2 are defined by the already known relation

$$\dot{e}_i = J_i \cdot \dot{q} \qquad \text{Equation 86}$$

and their image Jacobians are $J_i = \frac{\partial e_i}{\partial q}$. For $i = 1$, inverting the Equation 86 it is gotten

$$\dot{q} = J_1^+ \cdot \dot{e}_1 + P_1 \cdot z \qquad \text{Equation 87}$$

In which J_1^+ is the pseudo-inverse matrix of J_1 and P_1 the orthogonal projection operator onto the null space of J_1. From the Equation 87, it is possible to observe that any value of z does not disturb the task e_1. Then, it is possible defining a secondary task without disturbing the main one using the Equation 87 in Equation 86 and doing $i = 2$

$$\dot{e}_2 = J_2 \cdot \left(J_1^+ \cdot \dot{e}_1 + P_1 \cdot z\right) = J_2 \cdot J_1^+ \cdot \dot{e}_1 + J_2 \cdot P_1 \cdot z \qquad \text{Equation 88}$$

Solving the Equation 88 in order of z and introducing it in Equation 87, it is possible to obtain the control law for two orthogonal tasks with different priorities

$$\dot{q} = J_1^+ \cdot \dot{e}_1 + P_1 \cdot \left(J_2 \cdot P_1\right)^+ \cdot \left(\dot{e}_2 - J_2 \cdot J_1^+ \cdot \dot{e}_1\right) \qquad \text{Equation 89}$$

Such demonstration can be applied for several tasks:

$$\dot{q}_i = \dot{q}_{i-1} + \left(J_i \cdot P_{i-1}^A\right)^+ \cdot \left(\dot{e}_i - J_i \cdot \dot{q}_{i-1}\right) \qquad \text{Equation 90}$$

In which P_{i-1}^A is the projection onto the null-space of the augmented Jacobian: $P_0^A = I$, $P_i^A = P_{i-1}^A - \left(J_i \cdot P_{i-1}\right)^+ \cdot \left(J_i \cdot P_{i-1}\right)$. The $\dot{q}_i$ is the control value that realize the i higher priority tasks.

The task sequencing approach proposes a solution to increase the robustness of the system, dividing the global task into several subtasks. This is done adding and removing those subtasks from a stack, according to the conditions of the environment.

To build a system more robust to its environment it is possible to include restrictions on its path, adding more one subtask in the stack. This can be done using a cost function, V , to define where the robot is safety. This function can be defined in the joint space and is, for example, high for dangerous situations and low for the safety ones. For the control law, this task can be seen as an external force which pushes the system far from the dangerous configurations.

$$\dot{q} = -k \cdot g(q) = -k \cdot \nabla_g^T V \qquad \text{Equation 91}$$

In which $\nabla_g^T V$ is the gradient of the V .

Using predictive control it is possible to estimate when the system will be in danger with the actual stack of tasks, and change them to fix it. This prediction is done by the equation

$$\hat{q}(t+1) = q(t) + \Delta t \cdot \dot{q} \qquad \text{Equation 92}$$

The robot configuration in the present instant is $q(t)$, Δt is the time between two consecutive control input values and $\hat{q}(t+1)$ is the predicted robot configuration in the next iteration. If the cost function, V , gets above a certain threshold value it is time to change the task stack.

To decide which task should be removed from the stack of subtasks, two criterions can be used, see Figure 10.

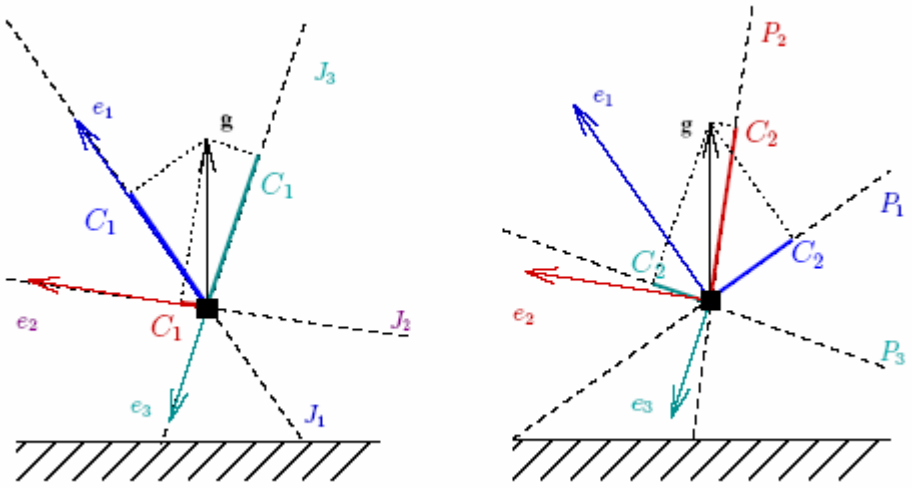


Figure 10 – Criterions to decide which task should be removed from the task stack, in the task sequencing approach: inner product between the gradient and each task (on the left) and projection of the gradient in the null space of each task (on the right).

The firth criterion is removing the subtask that is producing a velocity vector which is more in opposition with the gradient direction. This can be obtained by the inner product of the vectors

$$C_1 = -\frac{\left\langle J_i^+ e_i \middle| g \right\rangle}{\|e_i\|}$$ Equation 93

It is necessary divide by the norm of the task error to normalize the values, otherwise the task which is closest to its goal would be always removed. With this approach it is necessary to take care with the possibility of the denominator is zero.

The second criterion measure how independent is the gradient task from each one of the subtasks. This is done projecting the gradient onto the null space of each subtask

$$C = \|P_i g\|$$ Equation 94

$$P_i = I - J_i^+ \cdot J_i$$ Equation 95

In this case, the sign of the subtask control law is not taken into account.

After the danger is avoided, the predictive control is used again to know if the robot is far away enough from the singularity point.

This approach was successfully implemented and tested in a real robot by Mansard and is described in [3] and [4].

In that implementation, each time that it is introduced or removed one task, the task vector changes from the task vector e_A to e_B. As e and $\dot{q}$ are linearly linked by the image Jacobian, if there is a discontinuity in the e, this will conduce to an infinity acceleration in the joints. To prevent this, instead of using the traditional behavior equation $\dot{e} = -\lambda \cdot e$ is used

$$\dot{e} = -\lambda \cdot e + e^{-\mu t} \cdot \left(\dot{e}_A(t_0) + \lambda \dot{e}_B(t_0)\right)$$ Equation 96

In which t_0 is the instant of commutation in the control law, t is the instant after the last switch and the λ and μ are parameters which define the transition time between both tasks. Experimentally it was obtained that μ should be 10 times bigger than λ to a smooth switch.

Then, the total control law for a controller that realizes all tasks which are in the stack is

$$\dot{q} = \dot{q}_i + e^{-\mu t} \cdot \left(\dot{e}_A(t_0) + \lambda \cdot \dot{e}_B(t_0)\right) - k \cdot P_i^A \cdot q(g)$$ Equation 97

Theoretically the task sequencing, using the redundancy formalism can be done without that the secondary tasks affects the main task. However, for that, a full/coarse system calibration is required,

which can be not available. Beyond that, this normally needs one depth estimation, which it is not easy to obtain without *a priory* information. For these reasons a correct computation of the task Jacobian may be not available. In that case the secondary tasks are not perfectly projected onto the null space of the main task, which can imply that the secondary tasks could affect the main one. A possible way to address this loss of efficiency is estimating the Jacobian.

One of the first researches in learning Jacobians was done by Miller in [6]; he estimates a visual-motor model without needing *a priori* knowledge about the robot. This estimation was done off-line and had the drawback that requires training the system in the whole workspace.

Later, Hosoda, [16], developed a learning system using an on-line Jacobian learning. No *a priory* knowledge about the robot or the target is required, and the approach is independent of the number of cameras used. In that case, the Jacobian parameters were not estimated; it was just ensured that the initial image converges to the desired.

In the last years, the research in the learning methods has been increase even more. Some experiments demonstrate that the learning of the image Jacobian, J_q, is very hard to do, due to its intrinsic non-linearity, [30]. To facilitate the estimation of it, the task Jacobian, J_q, can be decomposed in

$$J_q = J \cdot J_R \qquad \text{Equation 98}$$

Where, as is already known, J is the image Jacobian in the Cartesian space, J_R is the robot Jacobian $\left(v = J_R \cdot \dot{q}\right)$ and J_q is the image Jacobian in the joint space. The task Jacobian in the Cartesian space, J, can be estimated using the homography matrix and the theory exposed in Chapter 2. To compute J_q it is necessary a full arm-eye calibration. In this sense, instead of learning the task Jacobian, J_q, is estimated the robot Jacobian, J_R, and then it is easy to compute the J from the homography matrix.

Mansard and Santos-Victor in [8], tested three methods to estimate the Jacobian: Broyden update, correlation and direct-inverse.

As is already well known, the Jacobian makes the following relation:

$$\Delta e = \hat{J}(t)\Delta x \qquad \text{Equation 99}$$

In which Δx represents a certain robot (camera) motion and Δe the correspondent features difference provoked by that motion.

Broyden update – the estimation of the Jacobian is given by

$$\hat{J}(t+1) = \hat{J}(t) + \alpha \frac{\left(\Delta e - \hat{J}(t)\Delta x\right)\Delta x^T}{\Delta x^T \cdot \Delta x} \quad \text{Equation 100}$$

α is a constant gain which regulates the update speed and $\Delta e - \hat{J}(t)\Delta x$ represents the estimation error. With this approach, it is necessary to take care with the possibility of instability when the motion is too small, which would provoke a zero in the denominator. A way to avoid it, it is for example, turn of the correction factor when the motion lies under a certain value.

Correlation – the least squares estimation is used to minimize the cost function

$$l = \sum_{i=0}^{t} \gamma^{i-t} \left(\Delta e - J\Delta x\right)^T \cdot \left(\Delta e - J\Delta x\right) \quad \text{Equation 101}$$

In Equation 101, t is the number of samples (iterations/movements). Its minimum value is obtained with

$$J = QR^{+} \quad \text{Equation 102}$$

In which Q and R are defined as

$$Q = \sum_{i=0}^{t} \lambda^{t-i} \Delta e_i^T \Delta x_i \quad \text{Equation 103}$$

$$R = \sum_{i=0}^{t} \lambda^{t-i} \Delta x_i^T \Delta x_i \quad \text{Equation 104}$$

The previous definitions of Q and R , are for an offline estimation when are known the values of all iterations. For an on-line estimation it is used

$$Q = \lambda Q + \Delta e_t^T \cdot \Delta x_t \quad \text{Equation 105}$$

$$R = \lambda R + \Delta x_t^T \cdot \Delta x_t \quad \text{Equation 106}$$

Direct-inverse – consist in the same technique as the method of *Correlation*, although in this case it is the inverse of the Jacobian, J^{+}, directly estimated, instead of the Jacobian. The cost function to minimize is

$$l = \sum_{i=0}^{t} \gamma^{i-t} \left(\Delta x - J^{+}\Delta e\right)^T \cdot \left(\Delta x - J^{+}\Delta e\right) \quad \text{Equation 107}$$

The main advantage of this last approach is that it is not necessary to invert the result obtained, reducing the inherent numerical errors and also the computation time.

In their experience all the methods presented were able to learn the Jacobians and to realize the sequencing tasks. The correlation method was the one which presented the best estimation.

The conclusion about the performance of these learning methods is that, when the system's parameters are well known, these methods are less efficient. However, for a coarse calibration the Jacobian learning methods are more efficient.

3.5 Previously Evaluations Done

The most global evaluation of visual servoing methods was done by Gans and presented in [2] and [17]. He evaluated the performance between IBVS, PBVS, 2,5D, CH and the Deguchi visual servoing. In his results only the first four methods are important in the context of the current project. He evaluated the influence of the image noise, and different depth estimation on the methods performance. Only results related with the noise are reported. Those are the ones which are in the scope of this thesis.

In his tests, IBVS had a problem with rotations around the optical axis (camera retreat) but generally it performed well in the presence of noise. All the other methods performed well in this test without any camera translation. In generic displacements, IBVS performed very well, even in the presence of noise and in the case of small motions.

IBVS performed badly in a rotation around an axis perpendicular with the optical axis in the camera frame. In this situation, the other methods generally had a good performance in their tasks. This test was not evaluated in the present thesis because it could not be tested in a real system. It is more like a theoretical experience because for little degrees of rotation, the target would leave the camera's field of view.

In those tests, 2.5D had presented a good performance but a higher loss of efficiency, in the presence of the noise, when compared with the IBVS and the CH methods.

CH proved that the camera retreat problem of IBVS can be solved. The CH performance had presented a little dependence with the increase of the noise. PBVS performance proved to be independent with the rotation or the translation values, however, it depends of the noise values.

4 Experimental Evaluation

In this chapter, the obtained experimental results are described. In the following subchapters, first we explain the methodology taken in the simulations. Then, the results are presented with the respective observations. This chapter finishes with the obtained conclusions.

4.1 Simulation Description

Before the results of the project developed are presented, this chapter describes the tests effectuated and the metrics used to evaluate the visual servoing techniques under analysis.

4.1.1 Simulation Methodology

To perform a trustful comparison between the visual servoing techniques, it is necessary to put them running in the same conditions. For that, it was necessary to adapt several parameters to put all the methods performing satisfactorily under the same test conditions; sometimes those parameters were hard to define due to the large possible combinations of them.

It was simulated a robotic arm with six degrees of freedom using the eye-in-hand approach. The desired camera position was set orthogonal with the target, being located one meter far from the target, oriented to it, and without any inclination. $P_* = [0, 0, 1, 0, 0, 0]$. The first three values of P_* correspond to the pose of the camera (x, y, z) and the last three to its orientation (θ_x, θ_y, θ_z).

To make the performance comparison between the visual servoing methods, it was planed to use an object consisting in four points lying in the target plane; this is the minimum number of points required to estimate the homography matrix and the camera displacement between two images. In order to get the clearest results, the best distribution for these points is in a square centered in the zero of the target frame. This occurs because this configuration allows the best way to see the evolution of the points in the image plane and also more easily avoid big condition numbers in the interaction matrix, which is desired.

However in SP method, the principal point is tracked, i.e. the point that lies in the center of the target plane. So, it is required that one of the points tracked lie in the center of the target frame. For that, it was thought to move the square of the point to one side of the target plane putting one of those points in (0, 0, 0) of the target frame; however, this would cause some problems in the measure of the features and a non-symmetric results for some camera motions. To address it, one fifth point located in the center of the target frame was added to the others four centered points.

The simulations measuring the influence of the image noise, η, in the performance of the several visual servoing techniques for different initial camera configurations. For that, it was added random noise, with variance ranging between 0 and 1, in the image plane coordinates of the points, (x, y). A noise of 1 pixel corresponds to one point to be laid in one pixel, when it should be laid in one of its neighbors. This noise can be due to light reflections, different kinds of illuminations in the target, bad quality of the camera and many other possible causes. The desired image, it was assumed as get it under perfect conditions and any noise was introduced on it.

In the simulations was assumed that the camera has unit focal length, as in the mathematical descriptions in the former chapters. The values of the scaling factors k_u and k_v were adapted in order to put the points adequately in the camera's field of view. The images were simulated with 256 by 256 pixels, which is a size that already allows a good definition for the common applications, without implying too much processing, as happens with bigger images.

In the simulations, the algorithm reaches the goal and is halted with success when the average pixels error is less than 1 pixel; this value corresponds to the maximum noise variance introduced in the system. The system is halted as not succeeded whether the camera had displaced more than 11 meters from the target along the z direction, or more than 2 along the x or y direction, whether had spent more than 200 iterations, or whether the average pixels error change less the 0.05 pixels after 5 consecutive iterations. To ensure that the camera does not collide with the target, it is also required that the camera does not get closer than 0,2 meters from it. To obtain smoother results, less influenced by outliers, each test was executed 50 times and it was made the average of its values. In the results presented, only the information considered useful are shown.

4.1.2 Tasks Tested

The tests performed are divided in two types: in the first tests, the performances of the algorithms are evaluated, when the initial camera position varies along only one dimension. After that, some generic motions are tested in which the initial cameral position vary along several degree of freedom. The tests effectuated were:

Rotation Around the Optical Axis

In the rotation around the optical axis test, the camera starts already in the desired position but with a variable initial inclination around the optical axis. The tests were performed for values of rotation between 5º and 355º.

Translation Along the Optical Axis

To measure the performance of the methods in a translation along the optical axis, it was simulated to set the camera in different initial positions along the optical axis. In this test the initial position varied between -0.5 m and + 0.5 m from the desired position (1 meters from the target, orthogonal with it, with alignment in the target direction and without any inclination).

Rotation Around an Axis Coplanar with the Target Plane

In the rotation around an axis coplanar with the target plane test, the initial camera position makes an acute angle with the target plane. In this test the camera starts its displacement in a position which lies in one circumference with the center in the zero of the target frame, and ray equal to 1 m (distance to the target in the desired position). This makes a big distortion in the image features and it is a hard test for the visual servoing techniques under evaluation. The tests were evaluated for initial angles between 0 and 70º. It was decided does not evaluate initial angles beyond 70º because, after that values, became unlikelihood could be possible to extract correctly features from a real image.

Translation Along an Axis Parallel to the Target Plane

In the translation along an axis parallel to the target plane test, it was evaluated the performance of the methods for an initial camera displacement parallel to the target plane. It was measured values of initial displacement only until 0.2 m because it was the maximum value that was possible to move the camera in the referred direction without making the points to leave the camera's field of view.

Generic Motions

For a final evaluation, it was performed tests in which is required that the camera moves along several dimension on the same time (translation and rotation). These were the most challenging tests made and its conclusion should be seen with more attention than the previous ones. In the generic motions tested the camera started its movement in the following positions:

Test 1: P_0 = [0.2, 0.2, 1.5, 0, 0, 0];

Test 2: P_0 = [0.2, 0, 1.5, 0, 0, 180];

Test 3: P_0 = [0, 0, 1.5, 30, 0, 90];

Test 4: P_0 = [0, 0, 1.5, 60, 0, 90].

4.1.3 Metrics Evaluated

Here are presented the metrics used to evaluate the performance of the visual servoing methods studied. Were used the following metrics: final pixels error (FPE), number of iterations (NI), maximum camera distance from the desired position (MCD), maximum feature point excursion (MPE) and time simulation (T).

Final Pixels Error

To measure the final pixels error, the pixel error of each point, is summed and this value divided by the number of points. It is the most important feature, and it is which shows whether the method had success in reach its goal. Any value under 1 pixel is equally considered successful and does not matter if those values are more or less close to 0.

Number of Iterations

This feature evaluates the number of iterations that the algorithm needed until be halted, as have had success or not.

Maximum Camera Distance from the Desired Position

To measure the maximum distance of the camera to the desired position, it is measured that distance at each iteration of the algorithm and reported the highest value obtained.

Maximum Feature Point Excursion

To measure this feature, it is computed the distance of each point tracked to the center of the image plane, at each iteration of the algorithm; the highest value measured is reported. The value of this feature in the desired position, with the points centered in the image, is 56 pixels.

Simulation Time

For each test, it was measured the execution time of each method. This was done in a different way than the other features. The execution time of a very fast algorithm is hard to compute with accuracy. Due to that, to measure it, each algorithm was executed 1000 times and the final result obtained divided by the same number.

4.2 Results

In this chapter the simulation results performed in the present project are exposed.

The results of the tests, in which the initial camera position vary along only one dimension are organized is following explained: first the evolution of the points in the image plane are presented; then, it is shown 3D graphics where are presented the performances of the features under analysis in function of the image noise and the different initial positioning; at the end are shown some tables with concrete values of the features evaluated to an easier interpretation of the results.

In the tests in which it is required that the cameral moves along several dimensions it is shown only the evolution of the points in the image screen, and some tables with the results for specific situations. Due to the large number of variables varying at the same time in these tests, it wouldn't be possible represent it in 3D graphics.

After each simulation chapter it is presented a briefing *Discussion*, in which are summarized the main conclusions of each test. At the end of this chapter, it is made a synthesis of the results and one global overview of all main features observed.

4.2.1 Rotation Around the Optical Axis

Figures 11 and 12 present the trajectories of the points in the image plane for rotations around the camera optical axis.

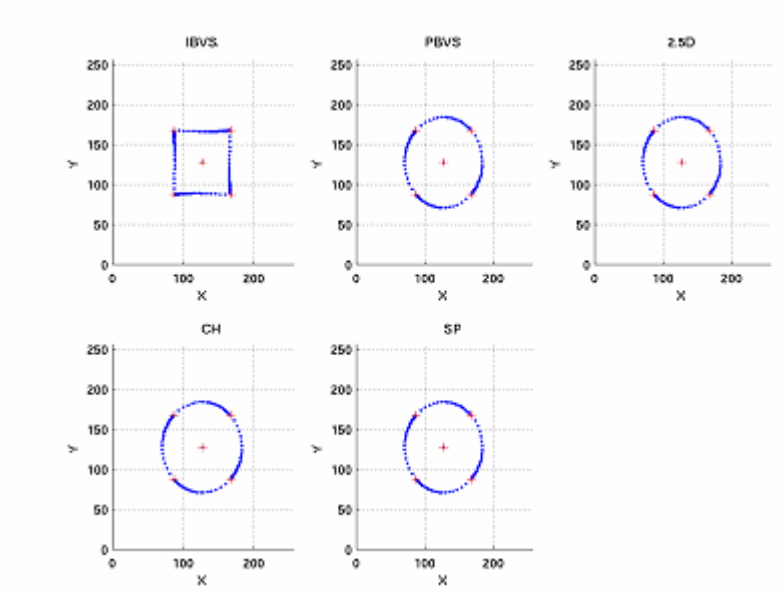


Figure 11 – Trajectory of the points in the image plane, for a rotation of 90º around the optical axis. The camera starts in the position P_0=[0,0,1,0,0,90] and there is not image noise.

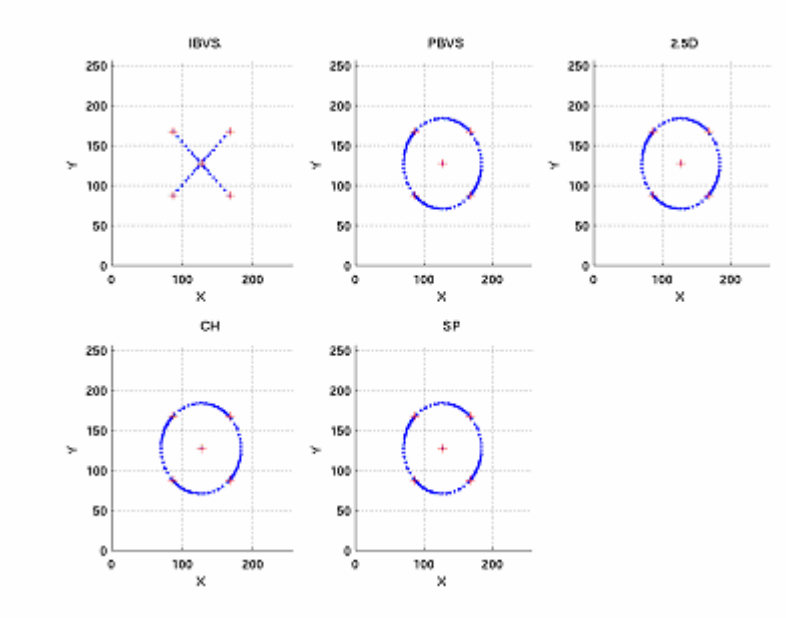


Figure 12 – Trajectory of the points in the image plane, for a rotation of 180º around the optical axis. The camera starts in the position P_0=[0,0,1,0,0,180]; there is not image noise.

4.2.1.1 Final Pixels Error

Figure 13 shows evolution of the final pixels error in function of the initial camera inclination around the optical axis. The figure also presents the performance of the methods with the increase of the image noise.

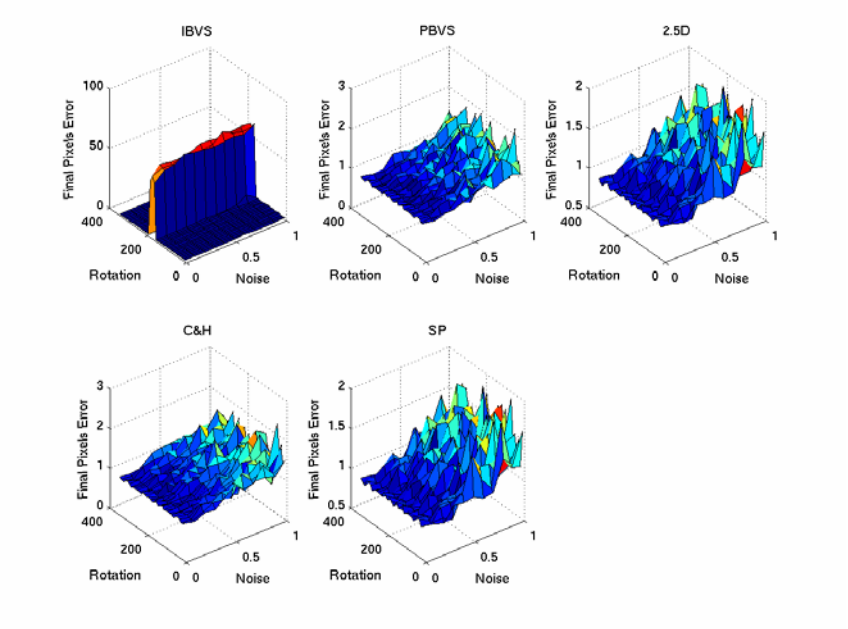


Figure 13 – Rotation around the optical axis; final pixels error in function of the camera rotation and image noise.

Here the phenomenon of camera retreat in IBVS can be perfectly seen. In the IBVS, the system can always converge until initial inclinations of around ±160º. After that value the final pixels error increases abruptly, and the algorithm is halted due to the excess of the camera retreat.

In all of the remaining techniques, the behaviors are very similar each other. They have some dependence with the increase of the noise, which makes the final pixel error to rise until 2 pixels. IBVS also shows some dependence with the increase of the image noise, although that is not visible in the presented scale.

Beyond IBVS, all the systems have an independent behavior with the level of the initial camera inclination.

4.2.1.2 Number of Iterations

Figure 14 presents the evolution of the number of iterations until the algorithms to be halted.

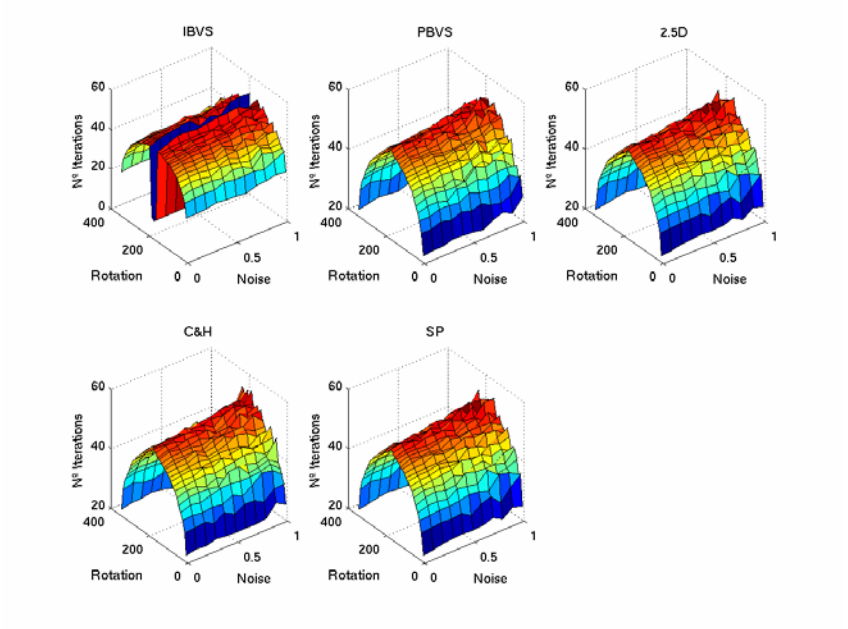


Figure 14 – Rotation around the optical axis; number of iterations in function of the camera rotation and image noise.

These graphics show that, in all methods, there is a clear dependence of the number of iterations with the level of initial camera inclination. In IBVS the number of iterations drops abruptly for levels of rotation close to 180º (above ±160º). This is also due to the phenomenon of camera retreat; for these values of rotation the camera retreats, crossing the maximum distance from the target allowed in the stop criterions, and the algorithm is quickly halted.

All systems have a slight dependence of the number of interactions with the increase of the image noise. In all methods, the number of interactions grows more or less from 50 iterations to around 55.

4.2.1.3 Maximum Camera Distance from the Desired Position

In Figure 15 can be seen the maximum camera distant from the desired position in the Cartesian space, during the execution of the visual servoing tasks.

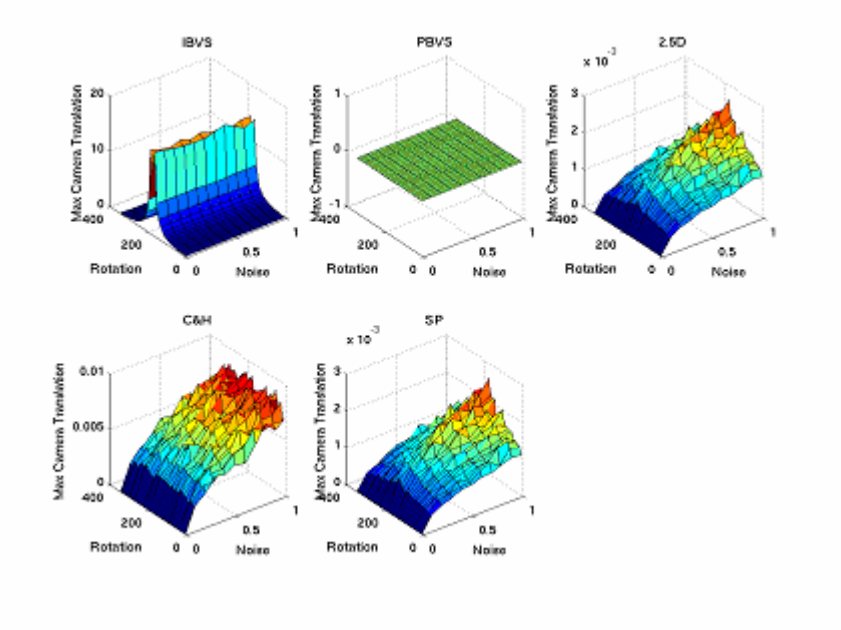


Figure 15 – Rotation around the optical axis; maximum camera distance from the desired position in function of the camera rotation and image noise.

The 2.5D, CH and SP have a little dependence of the parameter in analyze with the increase of noise, although looking to the scale this is insignificant (almost zero). For PBVS this value is always exactly zero. In the graphic of IBVS is possible to see how much the camera retreat for each level of initial camera inclination.

In 2.5D, CH and SP methods it is not shown any dependence of the values of the camera displacement with the levels of the initial inclination. This occurs because it is tracked a point which lies in the center of the image. Some displacement perpendicular to the optical axis appeared in tests made, when it was tracked a point which did not lie in the zero of the image plane.

4.2.1.4 Maximum Feature Point Excursion

Figure 16 exposes the values of the maximum feature point excursion with variation of the initial camera inclination and the image noise.

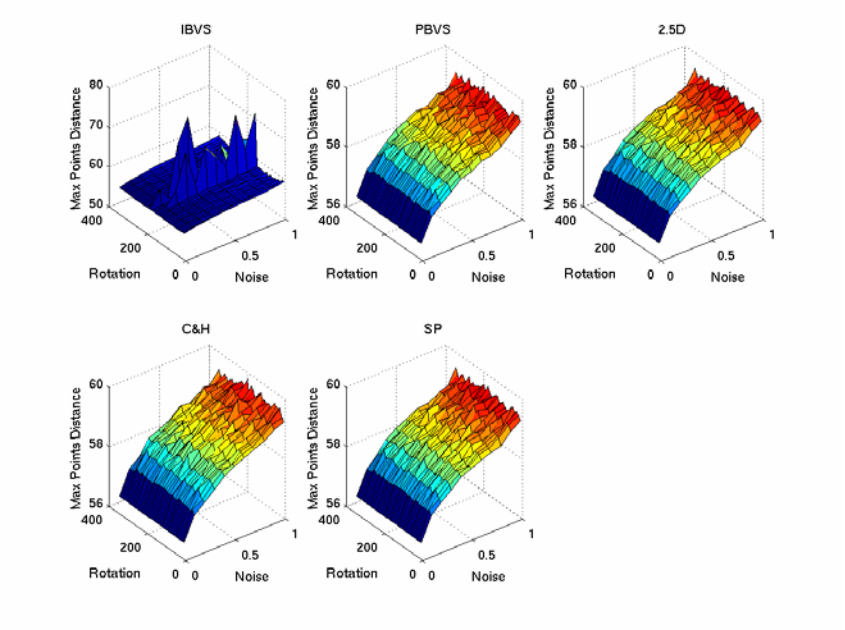


Figure 16 – Rotation around the optical axis; maximum feature point excursion in function of the camera rotation and image noise.

Once again, as the point tracked in the 2.5D, CH and SP methods lies on the zero of the target frame, the maximum feature point excursion of these methods is independent of the initial camera inclination. The same happens with PBVS, although this result would be the same wherever the points tracked lie.

The maximum feature point excursion rises a little in all the methods with the increase of the noise, from 56 pixels to around 59. IBVS has some exceptions close to levels of rotation around $\pm 180º$, which once again occurs due to the camera retreat phenomenon.

4.2.1.5 Discussion

In this test was evaluated the performance of each one of the visual servoing techniques in a pure rotation of the camera around its optical axis. It was possible to observe the problems of IBVS for this kind of test. In IBVS the system tries that, the points tracked move in the screen along one straight line, from the initial position to the desired one, see Figure 11. For that, the camera needs to retreat. In Figure 12 is shown the evolution of the points in the screen, for a rotation of 180º. In this case the camera theoretically needs to retreat until infinite, becoming the system unstable. For the other four

methods the motion in the Cartesian space is almost null and for PBVS it is exactly zero; all these performed satisfactorily in this test.

In the case of the maximum feature point excursion, 2.5D, CH and SP have not presented any dependency with the level of the rotation. This happens because it was tracked a point which lies in the center of the image. PBVS presented to have the same behavior but in this case, the results are independent of where the points tracked lie.

The number of iterations was almost the same for all methods. It had an important dependence with the initial camera inclination, and just a little with the increase of the noise. This occurs because it was chosen a too high value of the gain step, λ. The value of this parameter was not reduced in the simulations because it would make other problems in other tests, like for example, increasing the number of iteration needed or making the algorithms being easily stopped in local minimums. As said before, it was chosen a value for λ which could perform satisfactorily in all the tests executed.

Tables 1, and 2 also present a little dependence of the final error with the increases of the initial camera inclination in the presence of noise, 0.5 pixels, (with exception of IBVS). This behavior is almost the same in all methods, although Table 3 shows that in this specific test, the CH method could research a better final result than the other methods, handling better the presence of the image noise.

From Tables 1, 2, and 3, it is also possible to conclude that, with exception of the case in which IBVS does not converge, the SP method was always the fastest in execution time and after it the CH method.

P_0 = [0, 0, 1, 0, 0, 30], η = 0.5

Method	FPE [pixels]	NI	MCD [m]	MPE [pixels]	T [ms]
IBVS	0.99	34.6	0.040	57.77	17
PBVS	0.88	35.3	0	58.34	16
2.5D	0.98	34.7	0.001	58.32	17
C.&H.	0.98	34.4	0.005	58.323	14
S.P.	0.98	34.7	0.001	58.32	13

Table 1 – Rotation around the optical axis; performance of the methods with the camera starting in the position P_0=[0, 0, 1, 0, 0, 30] and image noise equal to 0.5.

P_0 = [0, 0, 1, 0, 0, 90], η = 0.5

Method	FPE [pixel]	NI	MCD [m]	MPE [pixel]	T [ms]
IBVS	0.97	43.5	0.466	57.49	19
PBVS	1.26	43.6	0	58.54	18
2.5D	1.35	43	0.001	58.60	18
C.&H.	1.32	43.9	0.005	58.72	15
S.P.	1.35	43	0.001	58.60	13

Table 2 – Rotation around the optical axis; performance of the methods with the camera starting in the position P_0=[0, 0, 1, 0, 0, 90] and image noise equal to 0.5.

P_0 = [0, 0, 1, 0, 0, 180], η = 0.5

Method	FPE [pixel]	NI	MCD [m]	MPE [pixel]	T [ms]
IBVS	56.50	9.0	10.954	57.83	13
PBVS	1.26	50.1	0	58.51	18
2.5D	1.31	49.9	0.002	58.63	20
C.&H.	1.04	51.1	0.006	58.70	15
S.P.	1.31	49.9	0.002	58.63	14

Table 3 – Rotation around the optical axis; performance of the methods with the camera starting in the position P_0=[0, 0, 1, 0, 0, 180] and image noise equal to 0.5.

4.2.2 Translation Along the Optical Axis

Figures 17 and 18 present the trajectories of the points in the image plane for translations along the camera's optical axis.

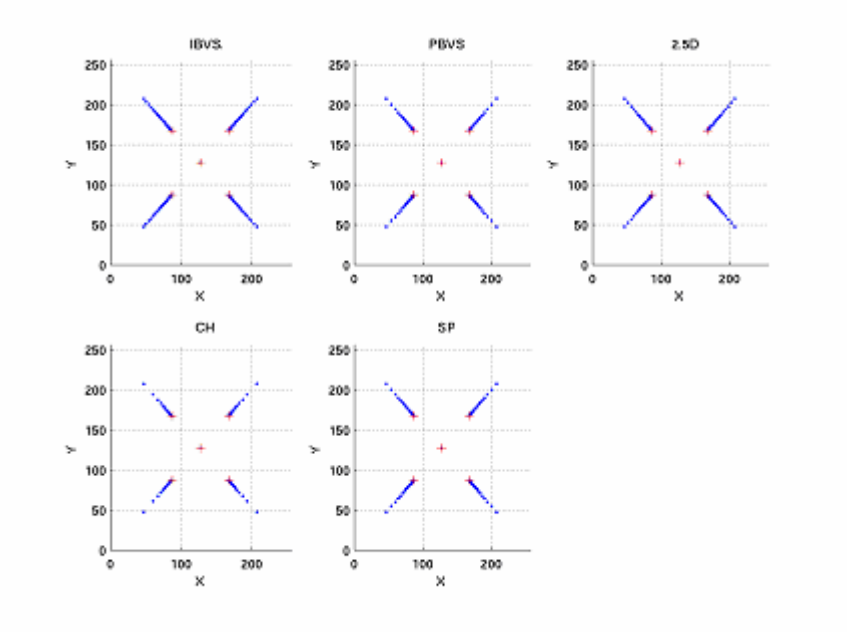


Figure 17 – Trajectory of the points in the image plane for a translation along the optical axis. The camera starts in the position P_0=[0,0,0.5,0,0,0] and there is not image noise.

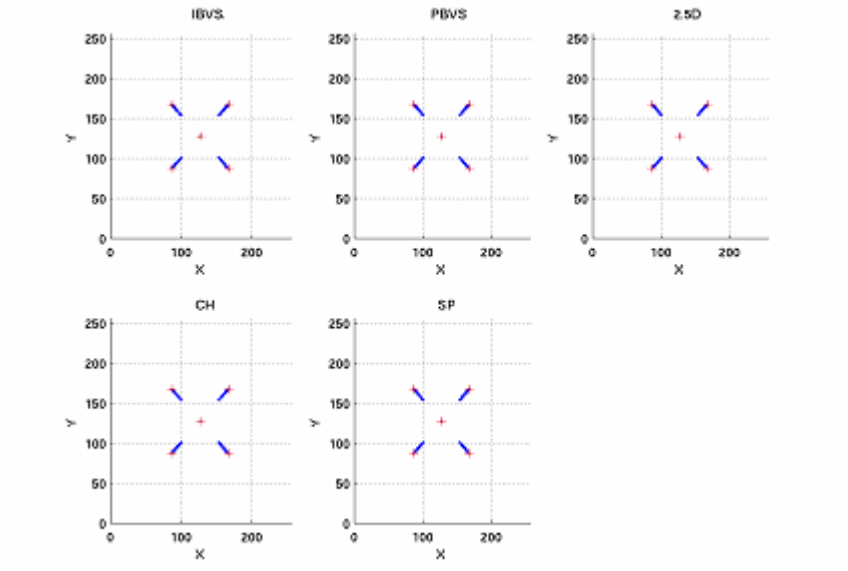


Figure 18 – Trajectory of the points in the image plane for a translation along the optical axis. The camera starts in the position P_0=[0,0,1.5,0,0,0]; there is not image noise.

4.2.2.1 Final Pixels Error

Figure 19 shows the values of the final pixels error for different initial camera positioning along the optical axis and values of noise added.

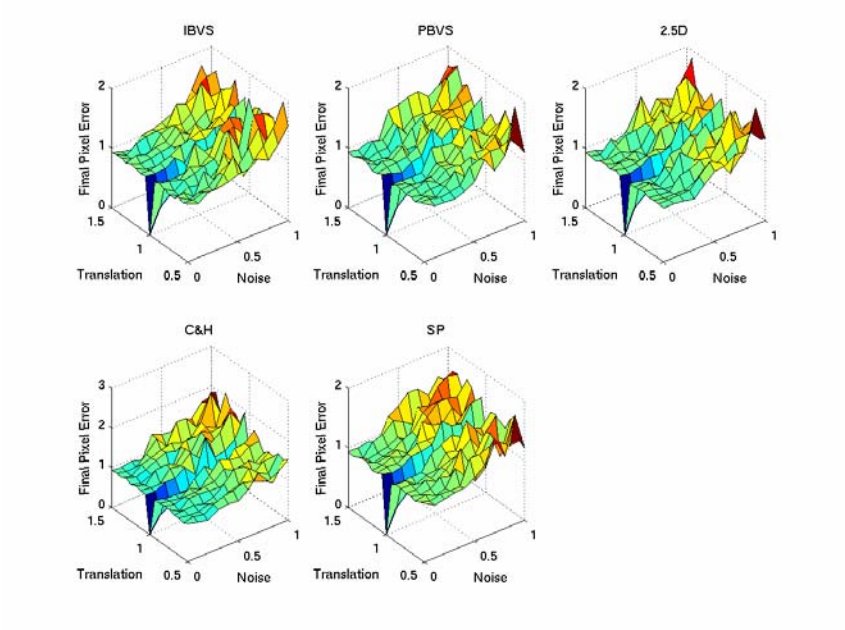


Figure 19 – Translation along the optical axis; final pixels error in function of the camera translation and image noise.

All methods had a similar performance. IBVS, PBVS, 2,5D and SP have a symmetric graphic though the translation axis. With low levels of noise, all these systems can reach the desired situation (final error below 1). With the increase of noise, these values increase until 2, and they are little higher for the highest displacements (± 0.5 m).

CH has a similar performance, although its graphic is not symmetric along the translation axis. The final error values are higher when, in the presence of the noise, the camera starts the movement farther away from the target; when the camera starts its displacement at 0.5 meters from the target, the final error, with the maximum values of the noise, is around 1.5 pixels, performing in this case, even better than the other four methods; when the camera starts its displacement at 1.5 meters from the target, the final error, with the maximum noise, is around 2 pixels.

4.2.2.2 Number of iterations

The number of iterations in function of the initial camera position along the optical axis and the noise are shown in Figure 20.

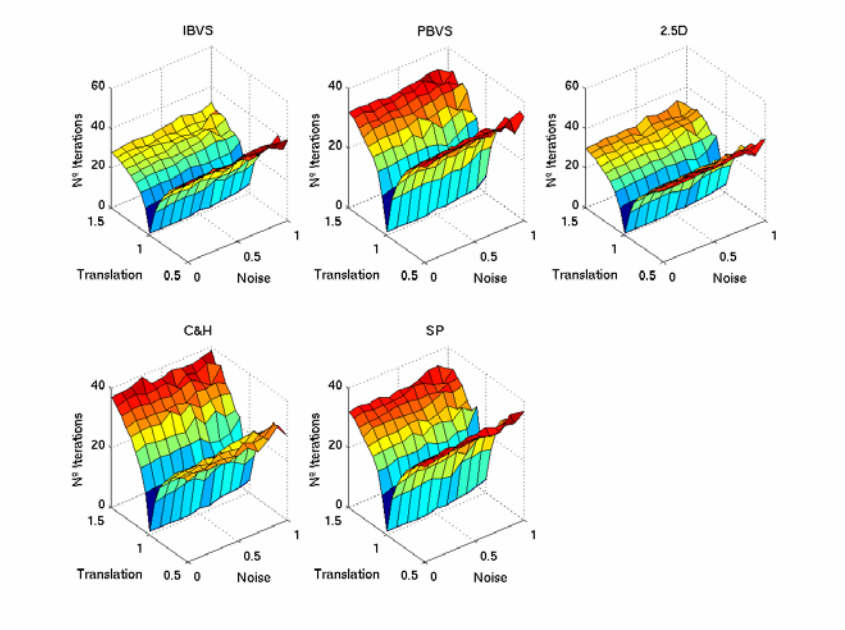


Figure 20 – Translation along the optical axis; number of iterations in function of the camera translation and image noise.

All graphics present just a little increase of the number of iterations with the increase of the image noise. In the translation axis appears important differences. PBVS and SP have a similar and symmetric performance. This is due to, in both methods, the translation to be controlled by the error in the 3D Cartesian space, and the norm of this error is the same for the two most extreme starting points (± 0.5 m). The IBVS method presents a non-symmetric performance along the translation axis; the algorithm takes more iterations to reach the goal when the camera start its movement closer to the target than when start it farther. The same occurs in 2.5D; this similarity in the performance of these two methods, happens because a part of the 2.5D Jacobian is equal of the IBVS Jacobian (Equations 51 and 65). The non-symmetric performance occurs because their interaction matrixes depend of the values of the depth, and different depth conduce to different convergence speed rates.

In the case of CH, the system reaches quicker the goal when the camera stars its displacement closer from the target. This happens because its function, that controls the displacement along the optical axis, is no linear (Equation 73); the error of the feature that controls the translation is bigger when the camera is closer from the target than when it is farther, for the same quantity of displacement.

4.2.2.3 Discussion

In this test was evaluated the performance of the visual servoing techniques for different initial positioning of the camera along its optical axis. All methods had similar performances; all of them could put the final error under one pixel when without noise, although these values grow linearly with the increase of the image noise. With exception of the CH, all methods had a symmetric performance along the translations axis, in the final pixels error. In the case of CH, its performance is better when the camera started its displacement closer to the target, performing in this case even better than all the other methods.

About the number of iterations, the CH was the fastest to reach the goal, when the camera starts its movement closer to the target, and the slowest, when the camera starts it farther, see Tables 4 and 5. This occurs due to its non-linear control law for the displacement along the z-axis. The PBVS and the SP had a symmetric performance along the translation axis in this feature. IBVS and 2,5D had a slower performance when the camera starts the movement closer to the target; this occurs because their interaction matrixes depend on the values of the depth, and different depths provoke different convergence speed rates.

About the convergence time, attending in Tables 4 and 5, the fastest method was the SP and the second the CH.

P_0 = [0, 0, 0.5, 0, 0, 0], η = 0.5

Method	FPE [pixel]	NI	T [ms]
IBVS	1.12	40.5	19
PBVS	1.01	34.4	16
2.5D	1.18	36.8	17
C.&H.	1.01	28.8	14
S.P.	1.06	34.3	13

Table 4 – Translation along the optical axis; performance of the methods with the camera starting in the position P_0=[0, 0, 0.5, 0, 0, 0] and image noise equal to 0.5.

P_0 = [0, 0, 1.5, 0, 0, 0], η = 0.5

Method	FPE [pixel]	NI	T [ms]
IBVS	1.08	29.8	18
PBVS	1.37	32.1	16
2.5D	1.30	30.3	17
C.&H.	1.31	36.2	15
S.P.	1.34	32.5	14

Table 5 – Translation along the optical axis; performance of the methods with the camera starting in the position P_0=[0, 0, 1.5, 0, 0, 0] and image noise equal to 0.5.

4.2.3 Rotation Around an Axis Coplanar with the Target Plane

Figures 21 and 22 present the trajectories of the points in the image plane for rotations around an axis coplanar with the target plane.

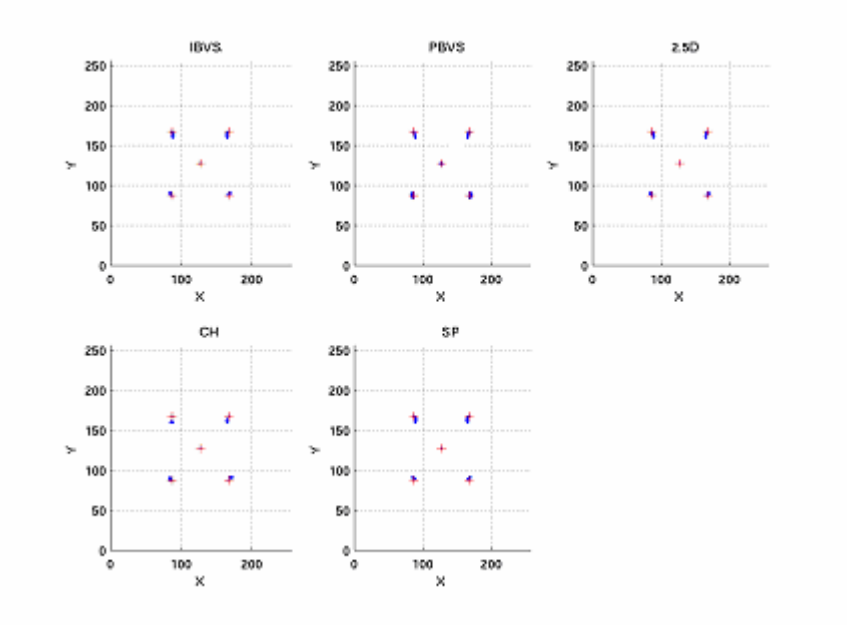


Figure 21 – Trajectory of the points in the image plane for a rotation of 30º around an axis coplanar with the target plane. The camera starts in the position P_0=[0,0,1,30,0,0] and there is not image noise.

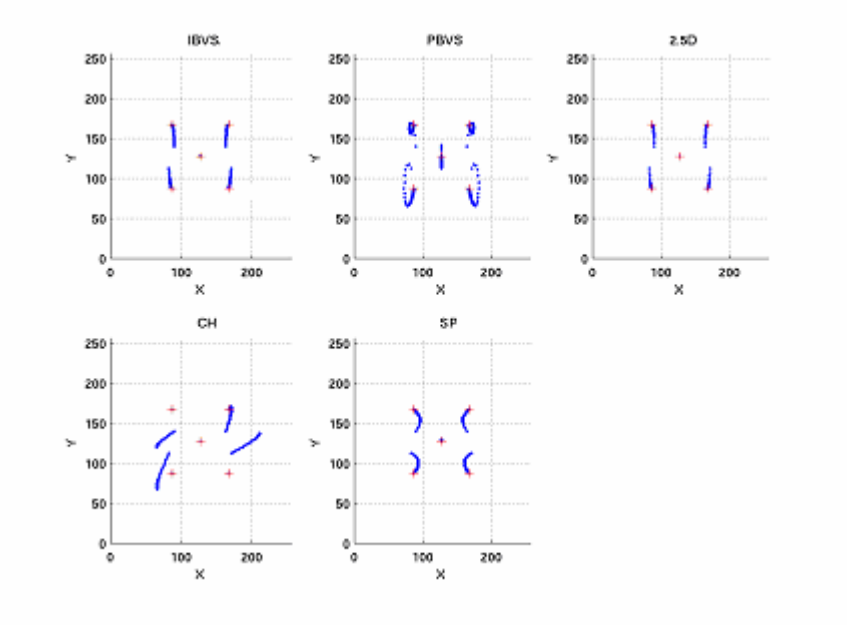


Figure 22 – Trajectory of the points in the image plane for a rotation of 70º around an axis coplanar with the target plane. The camera starts in the position P_0=[0,0,1,70,0,0]; there is not image noise.

4.2.3.1 Final Pixels Error

Figure 23 presents the final pixels error in function of the rotation around an axis coplanar with the target plane and the image noise.

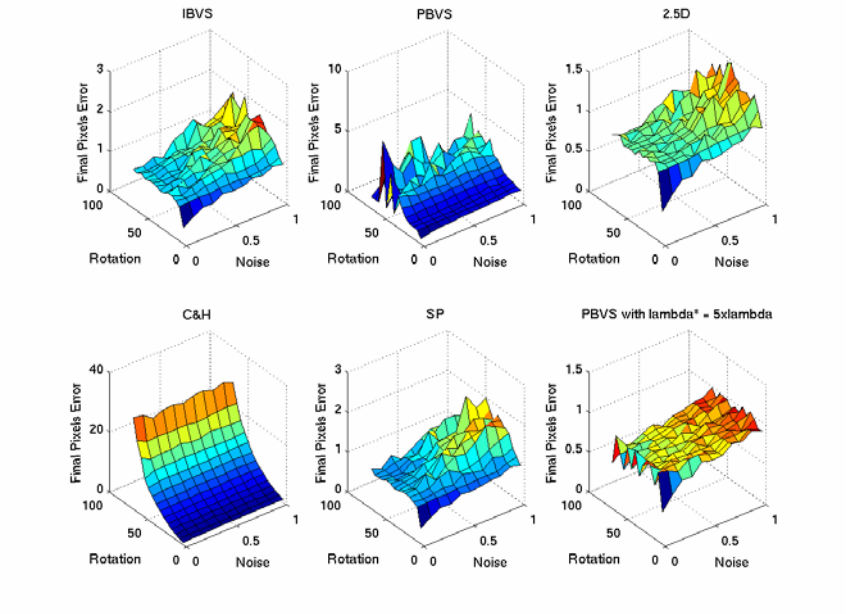


Figure 23 – Rotation around an axis coplanar with the target plane; final pixels error in function of the camera rotation and image noise.

In this test, IBVS, 2.5D and SP had a similar performance; the final pixels error depends mainly of the image noise and few of the rotation. In IBVS and SP the final error rise until 2 pixels, while in the 2.5D this value grows only until 1.5 pixels.

In CH, the final pixels error grows as a parabola with the levels of the rotation, and it is imperceptible the influence of the noise in the results.

PBVS (Figure 23 up center) presents a strange behavior. As this method is based in the homography matrix, its performance should be independent of the camera inclination. In the present test, happens that the method stays frequently blocked in local minimums. This occurs because, for this test, in PBVS the value of the step gain, λ, is too small in comparison of the stop criterion. For this specific situation it was tested to increase the value of λ 5 times and the result is a graphic showing dependence with the noise but no with the rotation, see Figure 23, down left. In graphic of PBVS, in Figure 23, there are a lot of high peaks; this could suggest a presence of a high level of outliers which could help to explain the results.

4.2.3.2 Number of iterations

Figure 24 presents the influence of the initial camera inclination and the noise in the number of the iterations of the methods.

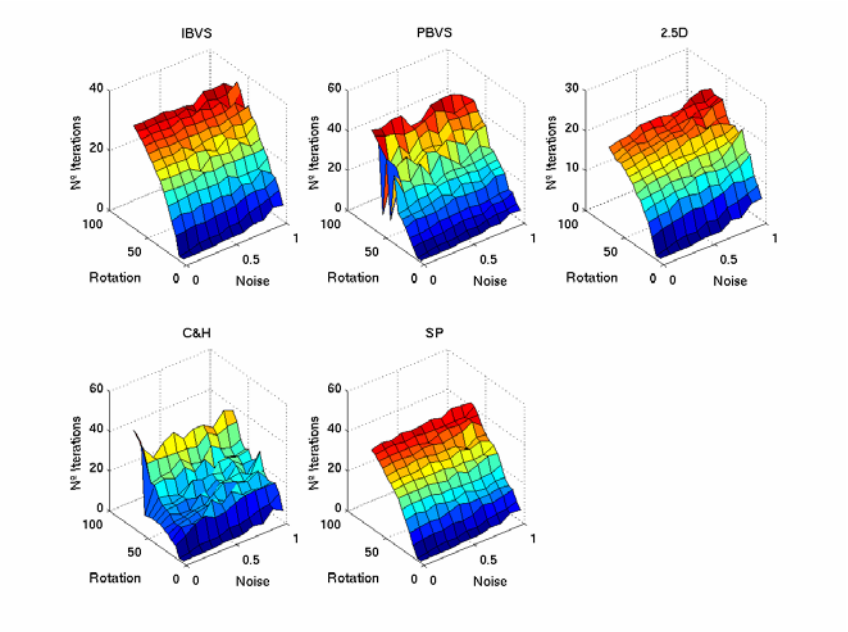


Figure 24 – Rotation around an axis coplanar with the target plane; number of iterations in function of the camera translation and image noise.

In IBVS, PBVS, 2,5D and SP the influence of the noise in the results seems to be very small. The 2.5D is the most susceptible of these to it. The number of iterations is much more influenced by the

level of the initial camera inclination. For these methods, at the maximum camera inclination, the PBVS took normally around 50 iterations to be halted, the SP 40, the IBVS 35 and the 2,5D around 20 iterations. In the PBVS graphic can be seen some negative peaks for some high values of rotation; these erroneous values are related with the problem referred in the former graphic, Figure 23.

In the CH the number of iterations grows immediately after the introduction of the firsts values of noise (0.1 pixels) from 10 iterations to around 15. But the CH's performance seems to be much more dependent of the values of the initial camera inclination: between 0 and 20º the number of iterations is quite low, between 20ª and 55º it is around 20, and after 55º it grow up abruptly at least to 40 iterations.

4.2.3.3 Maximum Feature Point Excursion

Figure 25 presents the evolution of the maximum distance of a point from the center of the image. The evaluated methods had different behaviors.

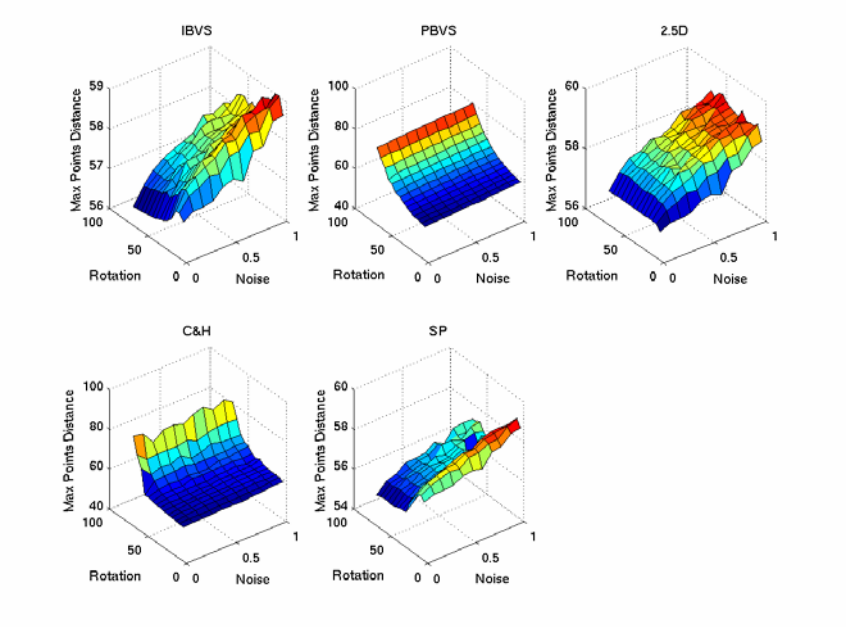


Figure 25 – Rotation around an axis coplanar with the target plane; maximum feature point excursion in function of the camera displacement and image noise.

The maximum feature point excursion in 2.5D grows with the increase of the noise from 56, until around 59 pixels and there is not any dependence with the level of initial camera inclination.

In PBVS the values grow as a parable, with the level of the initial camera inclination and these are almost independent of the noise.

The CH had a performance similar with PBVS. For small rotations, the values of the maximum feature point excursion keep quite lows, around 56 pixels (value of the desired situation). However, for an

initial camera inclination above around 55º, these values grow a lot. In these cases the system is not any more blocked by local minimums, and it becomes completely unstable. In this method the influence of the noise in this feature seems to be very low.

IBVS and SP have a strange behavior. In both cases the values rise a little with the increase of the noise, although the strange fact is that the maximum feature point excursion is higher for small rotations. This occurs because for low levels of initial camera inclination, the algorithm stops quickly; for high initial inclinations, the feature points error rises with inclination only in the yy direction but not in xx direction (see Figure 22); then, happens that while the system tries zeroing the values along the yy direction, the values of the errors in xx direction are also being reduced; then, when the system arrives close to the final position along the yy axis, the error along the xx axis is smaller conducing to a smaller final value of the maximum feature point excursion.

4.2.3.4 Discussion

As can be seen in Figure 22, for high initial camera angles, the features can become much skewed, so this is a hard test for the visual servoing techniques.

IBVS, 2.5D and SP despite of growing the final pixels error with the increase of the image noise, could perform correctly in this test. The 2.5D was the best of these, once it was the fastest in number of iterations and had the smallest final error, 1.5 pixels instead of 2 for the other two methods.

In number of iterations, the 2.5D was always the fastest method, and the PBVS the slowest.

PBVS could also perform correctly, although it had the highest value of the maximum feature point excursion. The high final pixels error of PBVS presented in Table 7 is due to the value of the step gain, λ, used, which is too small in comparison of the stop criterion. This value of λ was chosen in order to could perform satisfactorily in all the tests developed.

CH only had good results for small levels of initial camera inclination, until around 20º, performing in these cases well, even in the presence of image noise (the final pixels error is around 2.5 pixels for a initial inclination of 20º). After that level of initial inclination, it gets easily stopped in local minimums and has a high final error and maximum feature point excursion. For initial camera inclination above 55º the method becomes completely unstable (Figure 25), rising the final pixels error and the maximum feature point excursion, even faster.

In all tests the SP was the fastest method in simulation time and after it, the second was the CH.

P_0 = [0, 0, 1, 60, 0, 0], η = 0

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	0.99	31.0	56.47	15
PBVS	0.94	45.0	71.39	15
2.5D	0.98	18.0	57.39	12
C.&H.	18.70	45.0	71.39	12
S.P.	0.91	36.0	55.6	11

Table 6 – Rotation around an axis coplanar with the target plane; performance of the methods with the camera starting in the position P_0=[0, 0, 1, 60, 0, 0] and no image noise.

P_0 = [0, 0, 1, 60, 0, 0], η = 1.0

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	1.53	33.2	58.35	18
PBVS	3.35	39.8	72.60	17
2.5D	1.34	23.4	60.18	15
C.&H.	17.20	25.8	67.15	14
S.P.	1.47	38.3	57.36	13

Table 7– Rotation around an axis coplanar with the target plane; performance of the methods with the camera starting in the position P_0=[0, 0, 1, 60, 0, 0] and image noise equal to 1.

4.2.4 Translation Along an Axis Parallel to the Target Plane

Figure 26 presents the trajectories of the points in the image plane for translations along an axis parallel to the target plane.

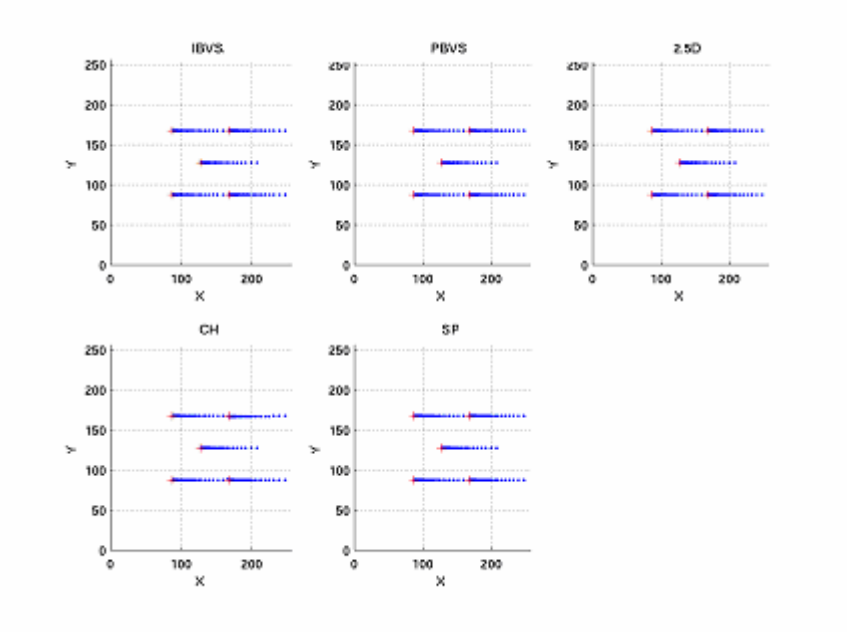


Figure 26 – Trajectory of the points in the image plane for a translation along an axis parallel to the target plane. The camera starts in the position P_0=[0.2,0,1,0,0,0] and there is not image noise.

4.2.4.1 Final Pixels Error

Figure 27 presents the final pixels error in function of initial camera positioning along an axis parallel to the target plane and the increase of the image noise. All the methods had a similar performance. Their performances are linearly dependent with the image noise and almost independent of the displacement.

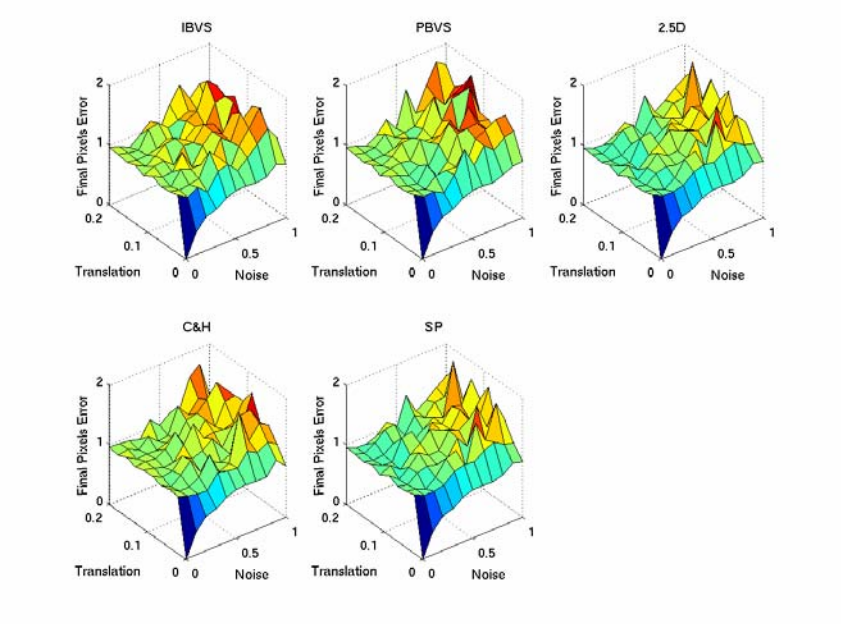


Figure 27 – Translation along an axis parallel to the target plane; final pixels error in function of the camera translation and image noise.

4.2.4.2 Number of iterations

The number of iterations, exposed in Figure 28, has almost the same values under the same circumstances for all methods. The level of the initial camera displacement is the only fact that affects the number of iterations necessary to the systems converges.

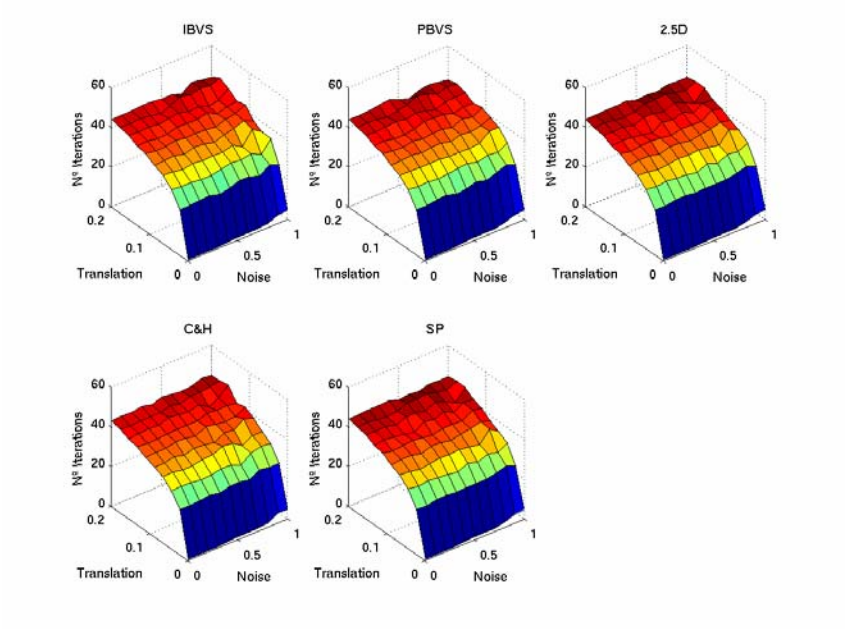


Figure 28 – Translation along an axis parallel to the target plane; number of iterations in function of the translation and image noise.

4.2.4.3 Discussion

For this test, all methods performed similarly and correctly. Figure 27 and Table 9 show that the final pixels error, in presence of noise, is a little higher in PBVS. It is also possible to see (Table 9) that the IBVS and CH were the methods that presented less final error in the presence of noise; this is due to they do not use the homography matrix in the computation of its control law. Attending in Tables 8 and 9, the SP was the fastest method in simulation time and after it the CH.

P_0 = [0.2, 0, 1, 0, 0, 0], η = 0.2

Method	FPE [pixel]	NI	T [ms]
IBVS	0.82	44.2	20
PBVS	0.84	44.3	18
2.5D	0.86	44.5	19
C.&H.	0.92	43	15
S.P.	0.86	44.5	14

Table 8 – Translation along an axis parallel to the target plane; performance of the methods with the camera starting in the position P_0=[0.2,0,1,0,0,0] and image noise equal to 0.2.

P_0 = [0.2, 0, 1, 0, 0, 0], η = 1.0

Method	FPE [pixel]	NI	T [ms]
IBVS	1.38	44.1	20
PBVS	1.65	42.8	18
2.5D	1.45	44.3	19
C.&H.	1.14	44.9	15
S.P.	1.45	44.3	14

Table 9 – Translation along an axis parallel to the target plane; performance of the methods with the camera starting in the position P_0=[0.2,0,1,0,0,0] and image noise equal to 1.

4.2.5 Generic Motions

4.2.5.1 Test 1

Tables 10 and 11 present tests with the camera starting in the position:

P_0 = [0.2, 0.2, 1.5, 0, 0, 0], η = 0

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	0.97	43	113.1	17
PBVS	0.99	47	113.1	15
2.5D	0.97	43	113.1	16
C.&H.	0.97	51	113.1	13
S.P.	0.98	43	113.1	11

Table 10 – Generic motion; performance of the methods with the camera starting in the position P_0=[0.2,0.2,1.5,0,0,0] and no image noise.

P_0 = [0.2, 0.2, 1.5, 0, 0, 0], η = 1

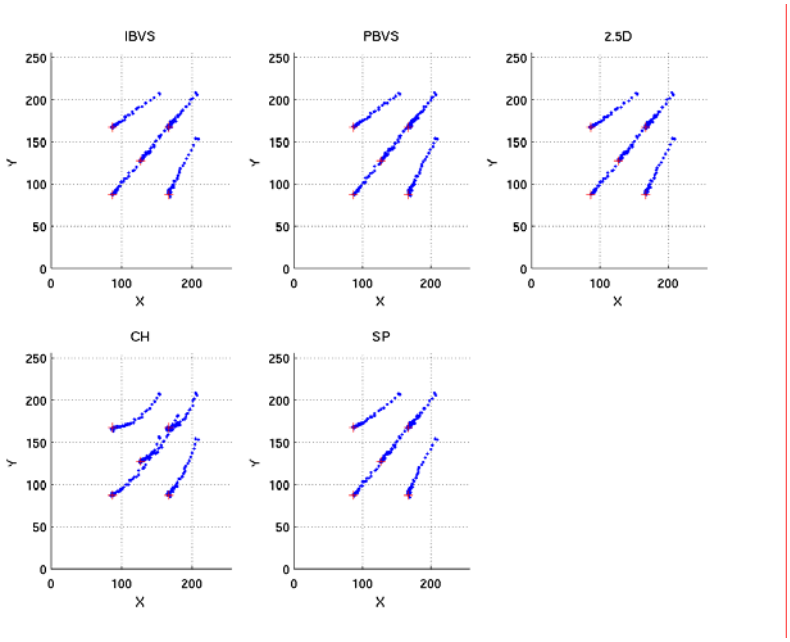


Figure 29 – Trajectory of the points in the image plane for a generic motion with the camera starting in the position P_0=[0.2,0.2,1.5,0,0,0] and image noise equal to 1.

Method	TE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	0.99	46	113.6	20
PBVS	0.96	49	113.6	20
2.5D	0.99	51	113.6	20
C.&H.	0.89	52	113.6	16
S.P.	0.75	52	113.6	14

Table 11 – Generic motion; performance of the methods with the camera starting in the position P_0=[0.2,0.2,1.5,0,0,0] and image noise equal to 1.

In this test the camera only needs to make a translation along the xx, yy and zz axis from the initial position to the desired one; no rotation is required. All methods performed correctly, even in the presence of the noise. The CH was the slowest in number of interactions and the IBVS the fastest. The SP was the fastest in execution time.

4.2.5.2 Test 2

Tables 12 and 13 present tests with the camera starting in the position:

P_0 = [0.2, 0, 1.5, 0, 0, 180], η = 0

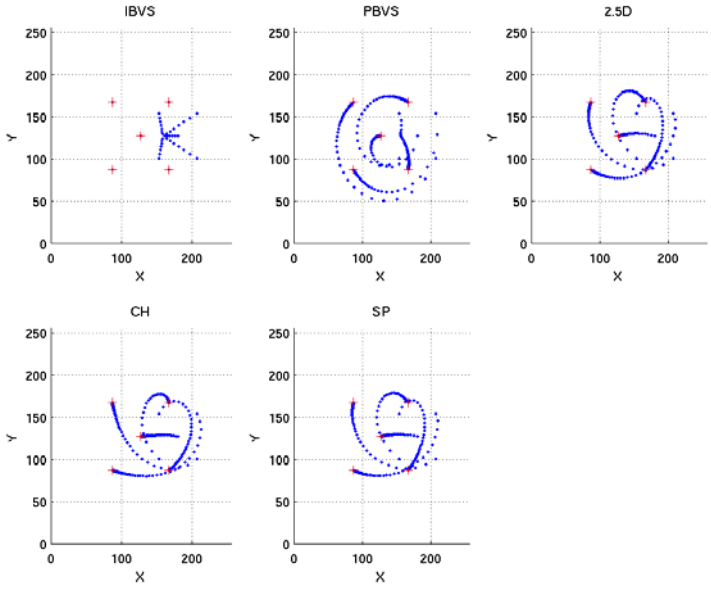


Figure 30 – Trajectory of the points in the image plane for a generic motion with the camera starting in the position P_0=[0.2,0,1.5,0,0,180] and no image noise.

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	63,40	7	84.3	10
PBVS	0.91	51	84.3	16
2.5D	0.96	50	85.1	17
C.&H.	0.97	55	86.7	14
S.P.	0.97	50	84.7	12

Table 12 – Generic motion; performance of the methods with the camera starting in the position P_0=[0.2,0,1.5,0,0,180] and no image noise.

P_0 = [0.2, 0, 1.5, 0, 0, 180], η = 1

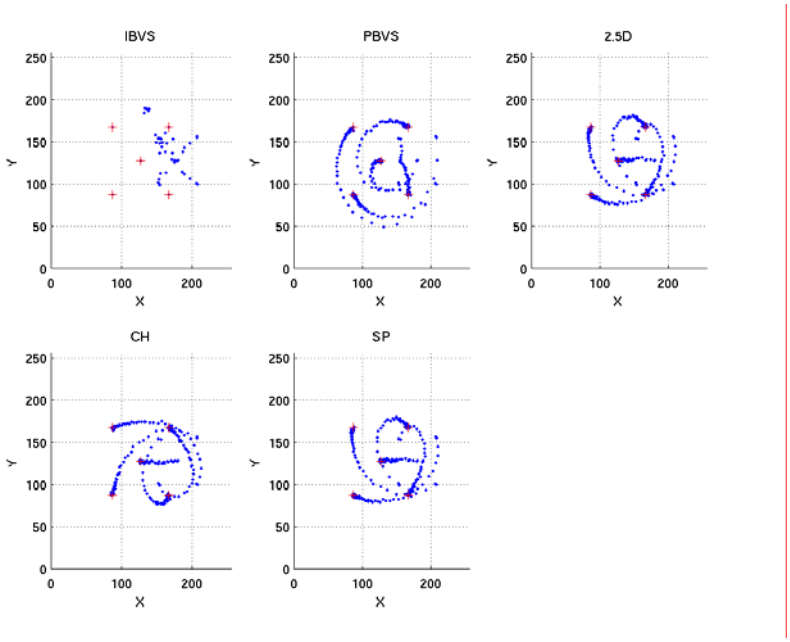


Figure 31 – Trajectory of the points in the image plane for a generic motion with the camera starting in the position P_0=[0.2,0,1.5,0,0,180] and image noise equal to 1.

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	61.33	7.2	84.82	13
PBVS	2.87	45.7	84.8	18
2.5D	2.87	45.4	85.8	19
C.&H.	0.97	58.3	87.6	16
S.P.	2.87	45.1	85.5	14

Table 13 – Generic motion; performance of the methods with the camera starting in the position P_0=[0.2,0,1.5,0,0,180] and image noise equal to 1.

This test, beyond the translation (the same as in Test 1), requires a rotation of 180º around the camera's optical axis. IBVS presented again its incapacity in making rotations of 180º around the optical axis. From it, it is possible to conclude that this problem is independent of the translation that the camera needs to make. Beyond IBVS, all methods, in the tests without noise, could reach the desired position. With noise, the final pixels error of PBVS, 2.5D and SP rose almost until 3 pixels. The method that best performed was the CH, which in the presence of noise, was the only one that could put the final error under the specified value, 1 pixel. This can prove its good performance in the

presence of the noise when the features are not skewed. The CH was also the slowest in number of iterations. Beyond IBVS, which does not converge in this test, the fastest method in execution time was the SP.

4.2.5.3 Test 3

Tables 14 and 15 present tests with the camera starting in the position:

P_0 = [0, 0, 1.5, 30, 0, 90], η = 0

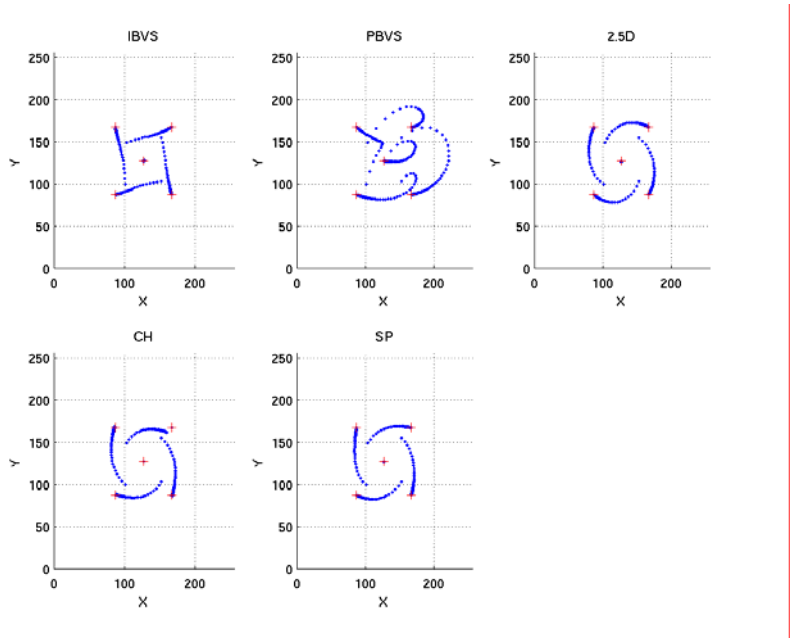


Figure 32 – Trajectory of the points in the image plane for a generic motion with the camera starting in the position P_0=[0,0,1.5,30,0,90] and no image noise.

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	0.98	40.9	55.6	16
PBVS	0.92	54	95.58	17
2.5D	0.98	43.2	56.3	16
C.&H.	2.43	53.6	57.5	14
S.P.	0.94	43.9	56.12	12

Table 14 – Generic motion; performance of the methods with the camera starting in the position P_0=[0,0,1.5,30,0,90] and no image noise.

P_0 = [0, 0, 1.5, 30, 0, 90], η = 1

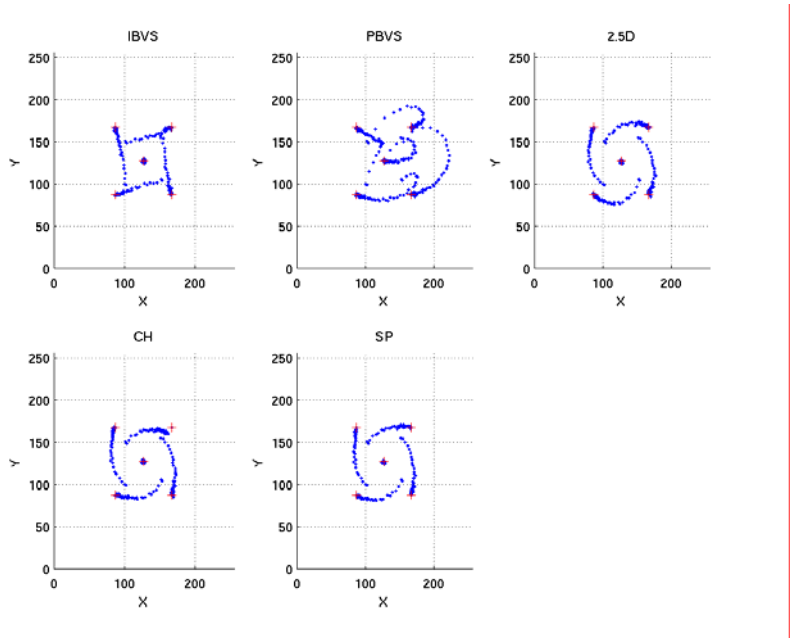


Figure 33 – Trajectory of the points in the image plane for a generic motion with the camera starting in the position P_0=[0,0,1.5,30,0,90] and image noise equal to 1.

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	0.91	46.3	57.3	20
PBVS	0.94	50.7	95.3	19
2.5D	1.78	41.4	58.42	18
C.&H.	3.23	40.3	58.23	14
S.P.	0.93	45.6	57.83	14

Table 15 – Generic motion; performance of the methods with the camera starting in the position P_0=[0,0,1.5,30,0,90] and image noise equal to 1.

Tables 14 and 15 present the results of a test which involves a translation and rotation along the optical axis of the camera, and a rotation around an axis coplanar with the target plane.

The CH method was the one which had more problems to reduce the final pixels error, even without the presence of noise; with the introduction of the image noise its final pixels error increased even more. The poor result of this method happens due to the distortion of the image features caused by the inclination of the camera.

In PBVS the maximum feature excursion value rose until 95 pixels, while for all the other methods it was almost constant, rising just a little with the increase of the noise. The SP was the fastest in convergence time. With noise, the CH was also as fast as SP, although this is due to its premature stopping in local minimums.

4.2.5.4 Test 4

Tables 16 and 17 present tests with the camera starting in the position:

P_0 = [0, 0, 1.5, 60, 0, 90], η = 0

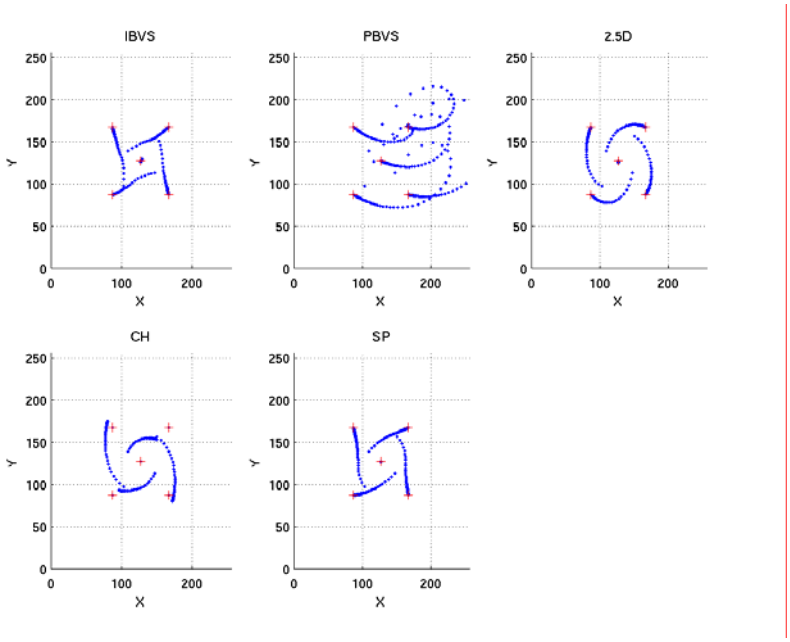


Figure 34 – Trajectory of the points in the image plane for a generic motion with the camera starting in the position P_0=[0,0,1.5,30,0,90] and no image noise.

Method	TE [pixel]	I	MPD [pixel]	T [ms]
IBVS	0.90	40.6	55.87	16
PBVS	0.92	60.0	161.366	19
2.5D	0.90	44.3	56.44	17
C.&H.	10.41	32.8	66.45	12
S.P.	0.99	47.6	55.7	11

Table 16 – Generic motion; performance of the methods with the camera starting in the position P_0=[0,0,1.5,60,0,90] and no image noise.

P_0 = [0, 0, 1.5, 60, 0, 90], η = 1

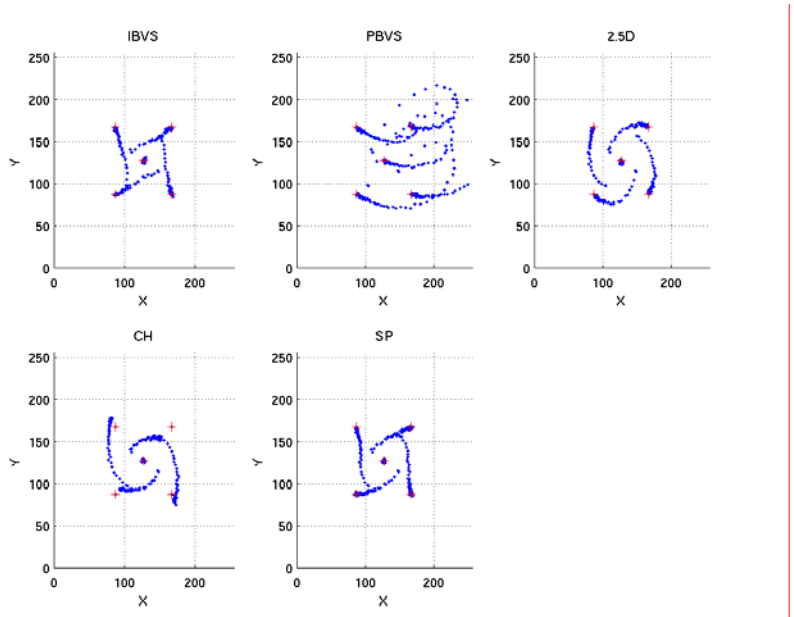


Figure 35 – Trajectory of the points in the image plane for a generic motion with the camera starting in the position P_0=[0,0,1.5,60,0,90] and image noise equal to 1.

Method	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	0.95	45.6	57.6	19
PBVS	2.14	58.9	161.01	20
2.5D	1.82	40.5	58.7	20
C.&H.	10.4	33.9	67.8	14
S.P.	0.83	52	57.45	14

Table 17 – Generic motion; performance of the methods with the camera starting in the position P_0=[0,0,1.5,60,0,90] and image noise equal to 1.

In this test, it is evaluated the same as in the previous one, but with a higher level of inclination around the axis coplanar with the target plane. So this is the hardest test effectuated because the features get more skewed.

It was possible to verify that in PBVS one of the points left the camera's field of view, Figure 35. This happened because there is not control over the features in the image plane. The CH method clearly couldn't reach its task. This method is which had the smallest number of iterations until converge because it stops quickly in local minimums. IBVS, 2.5D and SP could perform correctly in this test. The SP was the fastest in convergence time.

In the presence of noise, the IBVS was which performed best, minor final error, which was expected due to this method does not use the homography matrix to compute the control law.

4.2.5.5Discussion

The CH method had a very good performance when the camera is almost perpendicular with the target plane. When the camera makes a big angle with a perpendicular line of the target plane (more than around 20º) the method starts performing badly and this even more, in the presence of the noise. This bad performance happens because the CH method uses the area of the target as a feature and this becomes much skewed with the inclination of the camera in relation of the target plane.

In PBVS was possible to see that in the last test, one of the points left the camera's field of view. This occurs because in this method, there is not any control over the image plane features. In general the PBVS took more iterations than the other methods to reach the goal position.

In IBVS it was possible to observe that the problem of Chaumette Conundrum happens for rotation around the optical axis of the camera, independently of the translation that it needs to do.

The 2.5D and the SP were the only methods that had always acceptable results in all tests effectuated.

The SP was always the fastest method in execution time and after it, the second was the CH.

4.3 Synthesis of the Results

After all the tests had been covered, the main conclusions obtained from them are exposed and discussed.

Testing the *rotation around the optical axis,* it was possible to verify the problems of IBVS with this task. To reach goal position, the camera needs to retreat and after about 160 degrees of rotation required, the system enter in complete instability. It was also proved that this problem is independent of the translation that the camera needs to effectuate. In this test, the camera displacement for all the other methods was almost zero.

In the test of *translation along the optical axis* all the methods could reach their goals. The CH method, due to its non-linear function that controls the translation along the optical axis, has a non-symmetric performance along the translation axis; this performed poorer when the camera starts its displacement farther from the target because it uses the area as an image feature. With the camera farther from the target, the area of this is smaller which reduces the accuracy of this feature. All the other methods had a symmetric performance of the final pixels error along the translation axis.

For the *rotation around an axis coplanar with the target plane* IBVS, 2,5D and SP performed correctly and their results are just disturbed by the noise and almost nothing by the level of the rotation required. The PBVS also performed well, although the points tracked moved a lot in the image screen. In a generic motion test, one of the points inclusively left the camera's field of view. Generically the 2.5D was the method that performed best in this test.

For the *translation along an axis parallel to the target plane* all the methods had a similar performance and could reach correctly the goal. In this test the PBVS was which its performance was more affected by the presence of noise. IBVS and CH were the methods that presented less final pixels error in the presence of noise; this is due to they do not use the homography matrix in the computation of its control law.

In the tests performed, the number of iterations rose very slight with the increase of the image noise. This occurred because the value of lambda λ chosen, was too high to could be possible measure it with high accuracy.

CH performed correctly for rotations and translations around/along the optical axis, for which it was designed. When the camera is almost perpendicular with the target plane the CH performed well even in the presence of the image noise. However, for high levels of the camera rotation around an axis coplanar with the target plane, above around 20º, it performs poorly and normally does not converge; in these situations the CH seems also to be very sensible to the noise. Due to that, this method does not seem to be a good choice to be used for a generic application in visual servoing.

Along all the experiences effectuated, a constant conclusions was that the SP was always the fastest method in execution time and after it the CH. This happens because the SP has an easy mathematic description and it was possible to define directly the inverse Jacobian, instead of defining the jacobian and then invert it. The CH was the second fastest method in simulation time. This occurs because despite of being necessary to invert the CH Jacobian, it has only a dimension of [2x4].

To better understand the results, it was made one table, which summarizes the results obtained during this project - Table 18. This table was done based on the data of tables 3, 5, 7, 9, 11, 13, 15 and 17. To define the efficiency, presented in Table 18, it was defined that each algorithm success if the final pixels error is below 3 pixels.

Method	Efficiency	FPE [pixel]	NI	MPE [pixel]	T [ms]
IBVS	6 / 8	1,2	40,8	71,7	19,2
PBVS	6 / 8	1,5	45,1	88,1	18,2
2.5D	8 / 8	1,6	40,8	72,1	18,5
CH	5 / 8	1,1	40,4	86,6	15,4
SP	8 / 8	1,4	45,0	71.7	13,9

Table 18 – Table of Synthesis of the Results.

Looking to Table 18, the most relevant feature is that 2.5D and the SP were the only visual servoing methods that could always reach acceptable results, showing to be the most versatile methods to use in generic applications.

5 Conclusions

After the conclusion of this work, it is possible better understanding the behavior of some visual servoing methods and knowing more about its strengths and weaknesses.

The ViSP proved to be a very useful, flexible and user-friendly tool. Without it, it would not have been possible go so far in the investigation.

About the methods evaluated, the image based visual servoing had problems with rotations around the optical axis; the system becomes completely unstable after levels of rotation close 180º, and this behavior is independent of the translation that the camera needs to do.

The Corke & Hutchinson method performed correctly in the specific task for which it was designed, rotation around the optical axis, executing also perfectly for pure translation through the optical axis. However, for generic motions, particularly when those involve rotation around axis coplanar with the target plane, it performs poorly and normally does not converge. When the camera is almost perpendicular with the target plane it works very well, even in the presence of the noise. However when the camera starts its displacement in an initial position closer to be coplanar with the target plane, its performance in the presence of noise gets even worst. In this sense, this seems to be a weak method to be used in a generic visual servoing application.

The position based visual servoing had also a good behavior, although for generic motion with large rotations, it could be seen that the points tracked can leave the camera's field of view.

The 2.5D visual servoing performed correctly in all tests and it was the most versatile method. The shortest path visual servoing worked also correctly in all tests, and had showed a very similar behavior to 2.5D visual servoing.

In matter of execution time, the shortest path visual servoing was always the fastest method followed by the Corke & Hutchinson visual servoing.

After the development of this work it could be expected that the visual servoing theory has been improved, in the direction of building a more efficient and robust image based control system.

6 Future Work

To go further in the evaluation of visual servoing techniques, other kind of conditions that can affect the performance of the system can be evaluated. These include for example, errors in the camera calibration, errors in the robot kinematics and different kinds of depth estimation.

To get a better knowledge about the methods evaluated, the next steps could be for example implementing, and testing them in a real system. The ViSP is oriented to do such implementations without the need to make big modifications in the code. In this case, it would be possible to see the evolution of the systems but would not be easy to evaluate the influence of the image noise in those real systems, because it is hard to simulate it in a physic system. In the case of making comparisons tests in a real system, it would be more interesting to evaluate the influence of previously referred errors, which could affect the performance of a visual servoing system.

Literature References

[1] – Malis, E., Chaumette, F., & Boudet, S., Apr. 1999, “2-1/2-D visual servoing”, IEEE Transactions on Robotics and Automation, vol. 15, no. 2, pp. 238–250.

[2] – Gans, N. R., Hutchinson, S. A., & Corke, P. I., 2003, “Performance tests for visual servo control systems, with application to partitioned approaches to visual servo control” Int. J. Robot. Res., vol. 22, no. 10, pp. 955–981.

[3] – Mansard, N. and Chaumette, F., Nov. 2004, “Tasks sequencing for visual servoing” in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS’04),Sendai, Japan,.

[4] – Mansard, N. and Chaumette, F., Feb. 2007, “Task sequencing for high level sensor-based control”, IEEE Transactions on Robotics.

[5] – Kyrki, V., Kragic, D., and Christensen, H., Oct. 2004, “New shortest-path approaches to visual servoing” in IEEE Int. Conf. on Intelligent Robots and Systems, Sendai, Japan, pp. 349–355.

[6] – Miller, T., Aug. 1989, “Real-Time application of neural networks for sensor-based control of robots with vision” in IEEE Transactions on systems, man, and cybernetics, vol. 19, no. 4.

[7] – Corke, P. I. & Hutchinson S. A., Aug. 2001, “A new partitioned approach to image-based visual servo control” IEEE Transactions on Robotics and Automation, vol. 17, no. 4, pp. 507–515.

[8] – Mansard, N., Lopes, M., Santos-Victor, J., & Chaummette, F., 2006, “Jacobian learning methods for sequencing visual servoing” in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS’06).

[9] – Tell, D., 2002, “Wide Baseline Matching with applications to Visual Servoing”, PhD thesis, CVAP, NADA, Royal Institute of Technology, KTH.

[10] – Malis, E., Nov. 1998, “Contributions à la modélisation et à la commande en asservissement visuel”, PhD thesis, Université of Rennes I, IRISA.

[11] – Faugeras, O. & Lustman, F., 1988, “Motion and structure from motion in a piecewise planar environment”, International Journal of Pattern Recognition and Artificial Intelligence 2, no. 3, 485{508}.

[12] – Gans, N. & Hutchinson, S., 2002, “A switching approach to visual servo control” Proc. 17th IEEE International Symposium on Intelligent Control, Vancouver, Canada, pp. 770-760

[13] – Liberzon, D. & Morse, A., 1999, “Basic problems in stability and design of switched systems” in IEEE Control Systems Magazine 19.

[14] – Branicky, M., 1998, “Multiple Lyapunov functions and other analysis tools for switched and hybrid systems” in IEEE Trans. Automat. Control.

[15] – Tarabanis, K.A., Allen, P.K. & Tsai, R.Y., 1995, "A survey of sensor planning in computer vision ", in IEEE Transactions on Robotics and Automation 11(1), 86-104.

[16] – Hosoda, K. & Asada, M., Sep 1994, “Versatile visual servoing without knowledge of true jacobian”, in IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994 (IROS ’94), pp.186–193, München.

[17] – Gans N.R., Corker P.I., & Hutchinson, S.A., 2001, “Comparison of Robustness and Performance of Partitioned Image Based Visual Servo Systems” Proc. of Australian Conference on Robotics and Automation, Sydney.

[18] – Marchand, E., Spindler, F. & Chaumette, F., Dec. 2005, “ViSP for visual servoing: a generic software platform with a wide class of robot control skills”. IEEE Robotics and Automation Magazine, Special Issue on "Software Packages for Vision-Based Control of Motion", P. Oh, D. Burschka (Eds.), 12(4):40-52.

[19] – Marchand, E., Mai. 1999, “ViSP: A Software Environment for Eye-in-Hand Visual Servoing”, in IEEE Int. Conf. on Robotics and Automation, ICRA'99, Volume 4, Pages 3224-3229, Detroit, Michigan.

[20] – Hutchinson, S., Hager, G. & Corke, P., 1996, “A tutorial on visual servo control”, IEEE Transactions on Robotics and Automation 12(5), 651-670.

[21] – Chaumette, F., 1998, “Potencial problems of stability and convergence in image-based and position-based visual servoing” in the Confluence of Vision and Control, D. Kriegma, G. Hager, and S. Morse. Eds. Berlin, Germany: Springer - Verlag, vol. 237, pp 66-78.

[22] – Haralick, R.M., Lee, D. Ottenburg, K. & Nolle, M., 1991, “Analysis and solutions of the three point perspective pose estimation problem” in Proc. IEEE Conf. Computers Vision Pattern Recognition, pp. 592-598.

[23] – Liu, Y., Huang, T.S. & Faugeras, O. D., 1990, “Determination of camera location from 2-D to 3-D line and point correspondences” in IEEE Trans: Pat. Anal. Machine Intell., no. 1, pp. 28-37.

[24] – Feddema, J.T., Lee, C.S.G. & Mitchell, O.R., Feb 1991, “Weighted selection of image features for resolved rate visual feedback control”, IEEE Trans. Robot. Automat., vol. 7, pp. 31-47.

[25] – Vincze, M., Ayromlou, M. & Kubinger, 1999, W., “Improving the robustness of image-based tracking to control 3D robot motions”, in Proceedings of the international Conference on Image Analysis and Processing, pp. 274-279.

[26] – Bray, A. J., 1990, “Tracking objects using image disparities”, Image and Vision Computing 8(1), 4-9.

[27] – Schick, J. & Dickmanns, E.D., 1991, “Simultaneous estimation of 3D shape and motion of objects by computer vision”, in Proceedings of the IEEE Workshop on Visual Motion, pp. 256-261.

[28] – Faugeras, O., 1993 “Three-Dimensional Computer Vision: A Geometric Viewpoint”, The MIT Press.

[29] – Wilson, W., Hulls, C. W. & Janari-Sharifi, F., 2000, "Robust image processing and position-based visual servoing", in M. Vincze & G. Hager, eds, “Robust Vision for Manipulation”, Spie/IEEE Series, pp. 163-220.

[30] – Lopes, M., Bernardino, A., and Santos-Victor, J., April 2005, “A developmental roadmap for task learning by imitation in humanoid robots”. In Y. Demiris, editor, AISB - Third Int. Symp. on Imitation in Animals and Artifacts, Hatfield, UK.

[31] – Kragic, D., 2001, “Visual Servoing for Manipulation: Robustness and Integration”, PhD thesis, CVAP, NADA, Royal Institute of Technology, KTH.

[32] – Malis, E., Benhimane. S., July 2005, “A unified approach to visual tracking and servoing”, in: Robotics and Autonomous Systems, vol. 52, n^{o} 1, p. 39-52.

[33] –Hager, G. & Belhumeur, P., Oct. 1998, “Efficient region tracking with parametric models of geometry and illumination”, IEEE Trans. Pattern Anal. Machine Intell., vol. 20, no. 10, pp. 1025–1039.

[34] – Gonzalez, R., Woods, R., 2002, “Digital Image Processing”, second edition, pp 587-591.

[35] – Hough, P., 1962, “A method and means for recognizing complex patterns”, U.S. Patent, Number 3 069 654.

[36] – Chesi, G., Hashimoto, K., Prattichizzo, D., & Vicino, A., Sep 2003, “A switching control law for keeping features in the field of view in eye-in-hand visual servoing”. IEEE Int. Conf. on Robotics and Automation (ICRA’03), 3:3929–3934.

[37] – Mansard, N. & Chaumette, F., 2005, “A new redundancy formalism for avoidance in visual servoing” in IEEE/RSJ International Conference on Intelligent Robots and Systems.

.

Appendix A Hough Transform

The Hough Transform [34], [35] allows finding global relation between the pixels, i.e., straight lines or curves. This method is particular useful to find lines when there are partial occlusions in the image or this is very noisy. For example, attending in Figure 36-a) and after had been extracted the list of point, potentially belonging to one edge Figure 36-b, for example computing its gradient (second order derivative of the image) it is desired to know which of these lie in a straight line.

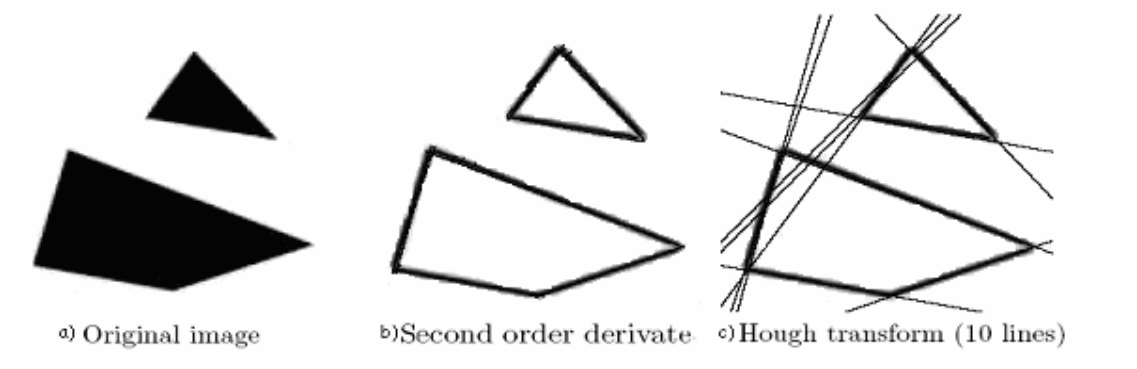


Figure 36 - Application example of the Hough Transform

For each point, (x_i, y_i), (Figure 36-b) an infinitive number of lines pass through it; all of these lines are defined as $y_i = a \cdot x_i + b$. Solving the equation in order to b, it is obtained $b = -a \cdot x_i + y_i$. This equation means that each point define one line (a_i, b_i) in the (a,b) parameters space, see Figure 37-b)

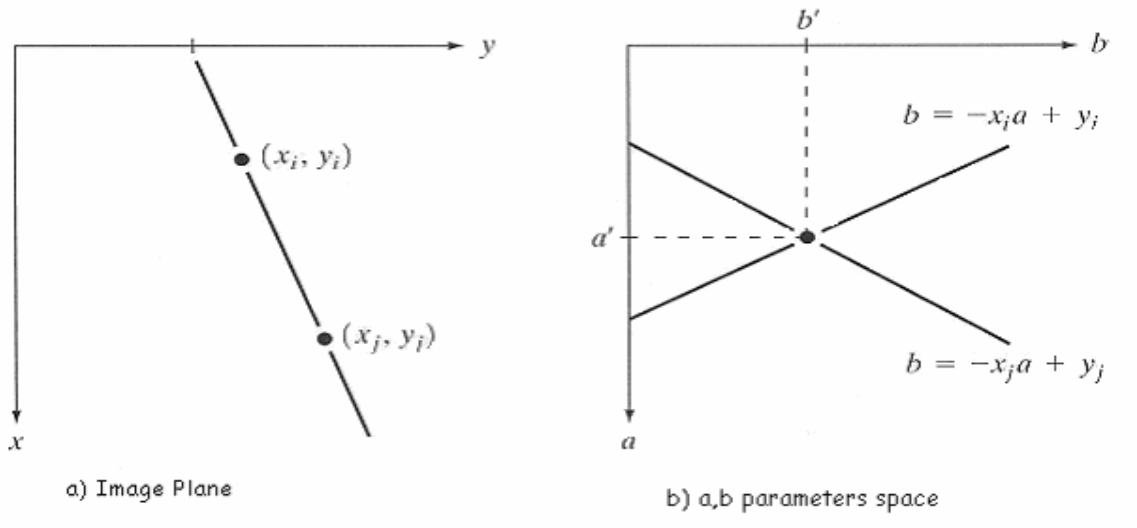


Figure 37 – Hough Transform: a) Image space; b) *a*, *b* parameters space.

Then, is defined a matrix accumulator, and for each point are stored in that accumulator the possible values that the parameters (a,b) can take, ranging the a values from the minimum to the maximum value that it can takes. As can be seen in Figure 37-b), the lines in the parameters space of two points which belong to the some line in the image will be intercepted.

Using this approach for all points (Figure 36-b)), each edge, in the image provokes one local maximum in the parameters accumulator. The number of the edge detected is equal to the number of local maximums in the accumulator.

This approach has one problem: for a vertical line the value of a is infinite. To avoid it, instead of use the (a,b) parameters to define one line, is used the transformation coordinates:

$$x \cdot cox\theta + y \cdot \sin\theta = \rho$$

And the accumulator is defined in the (ρ,θ) space, see Figure 38.

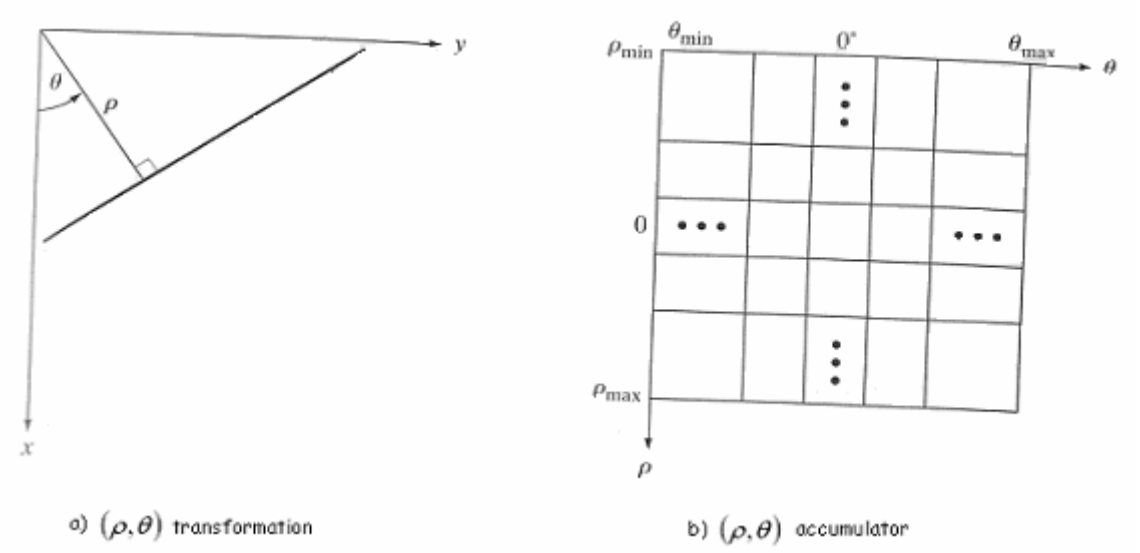


Figure 38 – Hough Transform: a) θ, ρ transformation; b) θ, ρ accumulator.

The former explication was done for straight lines, although it would be extended to geometric figures with several parameters, for example, for a circle, the transformation coordinates would be:

$$(x-c_1)^2 + (y-c_2)^2 = c_3$$

In this case the dimension of the accumulator would be 3 instead of 2. An example of a final result of the application of this method can be seen in Figure 36-a).

Appendix B Visual Servoing Platform - ViSP

To make research in the field of visual servoing it is always necessary to start by building all the tools related with that; each robot has its own Jacobians and speciation; this requires a lot of time and resources. The VIsual servoing Plataform, ViSP, is a library built in the C++ object-oriented programming language and has as its firth goal to create a generic tool that can be easily used for a fast prototyping of visual servoing tasks.

ViSP is built to facilitate the development of research in visual servoing. It is done to could be easily used and developed by the end-user and to be easily adapted to any machine (robot).

The control law used is similar with the exposed in Chapter 3.2:

$$\dot{s} = L_s \cdot v$$

to put the frame work in the end-effector terminal, and in the joints space, it is used the relation

$$J_q = L_s \cdot {}^{e}V_c \cdot J_R$$

the control law is given by

$$\dot{q} = -\lambda \cdot J_s^{+} \cdot (s - s^{*}) - \frac{\partial \hat{s}}{\partial t}$$

In which $\frac{\partial \hat{s}}{\partial t}$ is the estimation of the target displacement, which it is zero for a static target.

When some degrees of freedom of the robot are not been used ViSP allows also specifying a second task, task sequencing using a redundant formalism.

In that case the control law, for an static target is given by

$$\dot{q} = -\lambda \cdot \left(W^{+}W \cdot \hat{J}_q^{+} \cdot (s - s^{*}) + \left(I - W^{+}W \right) \cdot e_2 \right)$$

or in the Cartesian space, in the camera frame

$$v = -\lambda \cdot \left(W^{+}W \cdot \hat{L}_s^{+} \cdot (s - s^{*}) + \left(I - W^{+}W \right) \cdot e_2 \right)$$

ViSP allows also the two camera position approaches: eye-in-hand and eye-to-hand. It has already incorporated several visual features to track like 2-D points, 2-D straight line, 2-D ellipses, 3-D points, 3-D straight lines or the θu feature. The contour tracking in the 2D image plane space is done by the *moving edges algorithm*, which allows a good performance in real time. It has also several pose-estimation algorithms like the proposed by Dementhon, Lagrange and Lowe, which can be used in PBVS and some hybrid methods.

These libraries have also the possibility to estimate the homography matrix and the camera displacement. It uses the Hager Belhumeur algorithm [33] for matching of image templates, and other useful algorithms which can be useful for visual servoing.

ViSP has also the possibility to simulate a virtual robot specifying its own joints limits, Jacobians, etc… Then, to implement it in a real robot, just few changes in the code should be done to put the system working.

The ViSP architecture is presented in Figure 39 and is divided in three blocks:

Visual Servoing Block, in which are approached the visual features, visual servoing control laws and robot controller;

Image Processing and Computer Vision Block, where are approach the image processing and tracking algorithms and other computer vision algorithms.

Visualization Block, which provides some visualization tools for the real robots and simulations.

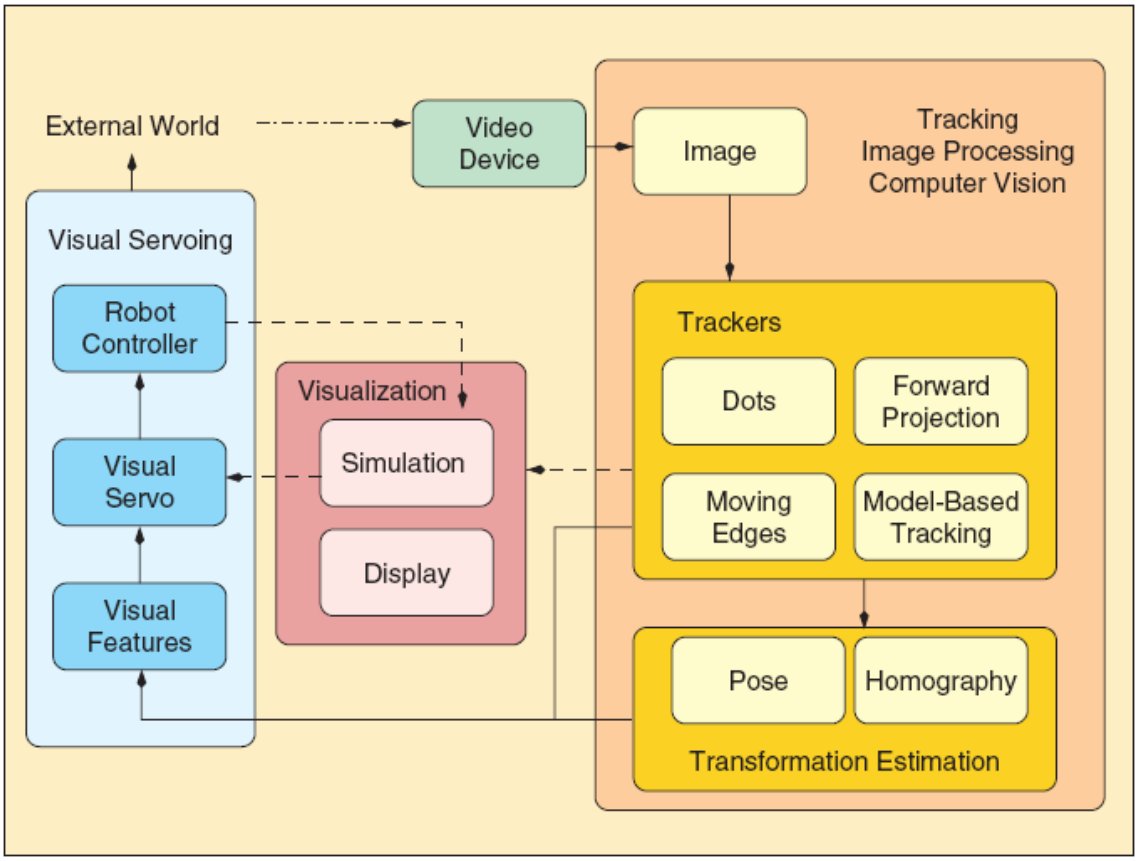


Figure 39 – ViSP software architecture.

A more complete description of VISP is presented in [18] and [19].