A PROBABILISTIC ROADMAP BASED PATH PLANNING FOR VISUAL SERVO OF ROBOTIC MANIPULATORS

FARID ARVANI

A Probabilistic Roadmap based Path Planning for Visual Servo of Robotic Manipulators

© Farid Arvani, B.Sc (Hons)

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland
December, 2008
St. John's, Newfoundland, Canada

Abstract

Vision feedback is a competent control technique for a large class of applications but they suffer from several imperfections. The well-known image-based visual servo (IBVS) methods regulate error in the image space i.e. the controller compares the current view of the target against the reference view and generates an error signal at the sampling rate of the vision system.

Contrary to position-based visual servo (PBVS), which regulates error in Cartesian space, IBVS ensures a local stability and convergence in the presence of modeling error and noise perturbations since the control loop is directly closed in the image space. However, sometimes (and specifically) when the initial and desired configurations are distant, the camera trajectory induced by IBVS is neither physically valid nor optimal due to the nonlinearity and singularities in the relation from image space to the workspace which can cause the target to leave the field of view. Furthermore, introducing constraints such that the target remains in the camera field of view and/or such that the robot avoids its joint limits during servoing is not trivial in classical PBVS and IBVS control techniques. When the displacement to realize is large, this incapability leads to the failure of servoing process.

This research presents a method to resolve the problems associated with classical servo control. Visual servoing control solutions are local feedback control schemes and thus require the definition of intermediate subgoals at the task planning level. This work introduces and details a trajectory planning scheme in order to achieve more robust visual servoing through the introduction of subgoal images. This ensures that the error signal is kept small since the current measurement always remains close to the desired value so that one can exploit the local stability of the IBVS control solution. The proposed method is based on Probabilistic Roadmaps (PRM) and its flexible

platform is used to introduce desired constraints such as visibility constraint, joint limit constraint, obstacle avoidance constraint, and occlusion avoidance constraint to the generated path at the task planning level. It is noteworthy that visibility constraint is intended to keep the target in the camera field of view (FOV). Joint limit constraint restricts the manipulator to avoid its joint limits. Obstacle avoidance and occlusion avoidance constraints ensure that the generated path is collision- and occlusion-free. One of the advantages of the proposed method is that targets are not required to have 3D models. However the method requires a 3D model of the obstacles to avoid obstacle collision and occlusion.

The proposed method plans the camera trajectory using PRM and then deduces the corresponding trajectories in the image plane which is a discrete geometric trajectory of the target in the image plane. A continuous and differentiable cubic spline presentation of the feature trajectories in the image plane is computed to be used as a time-varying reference to pure IBVS loop. Off-line path planning is performed using the kinematics of a 5-DOF robot arm to confirm the validity of the approach. Simulation of different IBVS scenarios is provided to demonstrate the performance of the proposed method.

Acknowledgments

I would like to express my sincere gratitude to my supervisors Dr. George Mann, Dr. Andrew Fisher and Dr. Raymond Gosine for their constant guidance, encouragement and financial support, without which I could not have completed this program. They have not only allowed me the freedom to explore new fields but also helped me fraternally in all the hardship and the strife. I appreciate all the care and unwavering support from my supervisors.

Also I would like to thank the ACOA (Atlantic Innovation Fund) and Memorial University of Newfoundland for providing financial assistance for this research.

I would thank every one of my friends and colleagues at the Intelligent Systems Lab (ISLab) and Instrumentation Control and Automation Centre (INCA), specially, Dilan, Eihaz, Rajib, Momotaz, Jeffery and Nusrat for their friendship during my stay from whom I learnt so much.

Finally, I would like to thank my wife for love, respect and support. Without her company, life would have never been so exciting; to whom I dedicate this thesis.

Contents

List of Tables

List of Figures

List of Symbols

$\mathcal{F}_t$	: the coordinate system attached to the target
$\mathcal{F}_o$	: the coordinate system attached to the obstacle
$\mathcal{F}_i$	: the coordinate system attached to the camera in its initial position
$\mathcal{F}_*$	: the coordinate system attached to the camera in its desired position
$\mathcal{F}_k$	: the coordinate system attached to the camera in its current position
$\mathcal{F}_b$	: the coordinate system attached to the base of the manipulator
$\mathcal{F}_c$	: the coordinate system attached to camera as the end-effector
n	: number of target features
$\mathbf{m}$	: normalized coordinates of a 3D point $\mathbf{M} = [X\ Y\ Z\ 1]^T$
$\mathbf{p} = [\tilde{\mathbf{p}}\ 1]^T$	: image coordinates of a normalized point $\mathbf{m}$ (pixels)
$\mathbf{C}$	: non-singular matrix of camera intrinsic parameters
$u_0,\ v_0$	: pixel coordinates of the camera principle point
α_u, α_v	: focal length in horizontal and vertical directions
λ	: focal length of the camera lens
${}^k\mathbf{T}_l$	: homogenous transformation $({}^k\mathbf{R}_l, {}^k\mathbf{t}_l)$ of frame l with respect to frame k
${}^i\mathbf{H}_*$	: Euclidean homography between frame $\mathcal{F}_i$ and $\mathcal{F}_*$
${}^i\mathbf{G}_*$	: projective homography between frame $\mathcal{F}_i$ and $\mathcal{F}_*$
$\mathbf{s}(t)$	: feature pixel coordinates in image plane at time t
$\mathbf{s}^*(t)$	: desired feature pixel coordinates in image plane at time t
$\mathbf{s}^*$	: fixed feature pixel coordinates in $\mathcal{F}_*$
$\mathbf{s}^i$	: feature pixel coordinates in $\mathcal{F}_i$
$\dot{\mathbf{r}}$	: camera velocity expressed in $\mathcal{F}_c$

$\mathbf{v}^T$: instantaneous linear velocity of camera expressed in $\mathcal{F}_c$

ω^T : instantaneous angular velocity of camera expressed in $\mathcal{F}_c$

$\mathbf{I}$: interaction matrix/image Jacobian

$\mathbf{I}^+$: generalized inverse of $\mathbf{I}$

κ : proportional control law parameter

Z^* : feature depth in $\mathcal{F}_*$

$\mathcal{Q}$: configuration space

$\mathcal{Q}_{free}$: collision-free subset of Q

$\mathcal{W}$: workspace

Γ : discrete sequence of intermediate camera poses

Γ_k : k-th element of Γ

Θ : discrete sequence of intermediate configurations in joint space

Υ : discrete target trajectory in image space

Ψ : discrete depth information of target features

$\mathbf{s}_i$: i-th element of Υ

$\mathbf{p}_k^j$: image coordinates of j-th feature in $\mathcal{F}_k$

Z_k^j : depth of j-th feature in $\mathcal{F}_k$

$\mathbf{n}^i$: normal to target plane in the coordinate system of $\mathcal{F}_i$

d^i : distance of the target feature to the origin of $\mathcal{F}_i$

d^* : distance of the target feature to the origin of $\mathcal{F}_*$

q_k : robot configuration in joint space corresponding to k-th camera pose

q_{init} : robot configuration corresponding to initial camera pose

q_{goal} : robot configuration corresponding to desired camera pose

ρ_k^j : depth ratio between Z_k^j and d^*

$\mathbf{M}_o^j$: j-th vertex of the obstacle OBB

C^0 : convex set of reconstructed target

C^1 : convex set of projected obstacle

$(V_i^0)_{i=0}^{N_0-1}$: counterclockwise ordered elements of C^0 $(N_0 = n)$

$(V_i^1)_{i=0}^{N_1-1}$: counterclockwise ordered elements of C^1 $(N_1 = 8\times$number of obstacle$)$

$\mathbf{B}$: normal direction vector

$G = (V, E)$: roadmap with nodes V and edges E

N : number of nodes in the roadmap

k : number of closest neighbors to a configuration

NN_q : nearest neighbors of q

$D_{\mathcal{Q}}(q_n, q_m)$: weighted distance between q_n and q_m in configuration space

$D_{\mathcal{W}}(q_n, q_m)$: weighted distance between q_n and q_m in workspace

Δ : PRM local planner

Q_n : quaternion representation of $\mathbf{R}_n$

δ : number of discretizations used by the local planner

${}^i\delta_{i+1}$: number of discretizations for the edge (q_i, q_{i+1}) used by trajectory planner

ζ : distance parameter

$\mathcal{T}$: timing parameter

superscripts

$\widehat{\ }$: estimated or approximated value of the parameter

List of Abbreviations

AABB	:	Axis-aligned Bounding Box
C-space	:	Configuration Space
CAD	:	Computer-aided Design
FOV	:	Field of View
IBVS	:	Image Based Visual Servoing
NF	:	Navigation Functions
NN	:	Nearest Neighbor
OBB	:	Oriented Bounding Box
PBVS	:	Position Based Visual Servoing
PRM	:	Probabilistic Roadmap planners
RPP	:	Randomized Path Planner
RRT	:	Rapidly-exploring Random Trees
SSE	:	Sum of Squared Error
UAV	:	Unmanned Aerial Vehicle

Chapter 1

Introduction

About this chapter: This chapter introduces the problem addressed in this thesis. A list of contributions of the thesis to path planning for visual servoing is provided. Finally, thesis organization is detailed.

1.1 Introduction

During recent years, the number of robots operating in manufacturing sites and factories has astonishingly increased. Robot systems are employed for a variety of tasks ranging from performing medical surgery to the task of assembling a car. Until recently, use of robot manipulators has had shortcomings where the work environment and object placement had to be controlled accurately. This limitation was due to the inherent lack of sensory capabilities of the robot systems. The significant motivation to improve autonomy for robot systems has led researchers to investigate the integration of vision system into robotic systems. Although sensor integration has long been acknowledged to be fundamental to increasing the versatility and capabilities of the robotic systems, specialized and costly hardware required to carry out video captur-

ing and image processing has been prohibitive. With the advent of new technologies, this barrier has been removed and has resulted in various commercial robot systems equipped with off-the-shelf hardware.

Vision is a beneficial robotic sensor since it mimics the human sense of vision and allows for non-contact measurement of the environment; e.g. integration of vision with robotic systems makes manipulation of objects easier without the exact and/or prior knowledge of the model of the environment and object. Robot control using visual information has been an active research field for decades. Visual sensing and manipulation were combined in an open-loop fashion (i.e. *look then move*) in the early works to correct the position of a robot to increase task accuracy [1]. The accuracy of this operation depends directly on the accuracy of the visual sensor and the robot end-effector. Machine vision can provide closed-loop position control for a robot end-effector which is referred to as *visual servoing.* [1]. Visual servoing is the fusion of results from many basic areas including high-speed image processing, kinematics, dynamics, control theory, and real-time computing. A task in visual servoing is to control a robot to manipulate its environment using vision as opposed to just observing the environment performed in active vision, motion from motion and structure from motion.

Vision-based robot control using an eye-in-hand system can be classified into two groups: position-based visual servo (PBVS) and image-based visual servo (IBVS) systems [2], [3], [4], [5]. In a PBVS control system, the input is computed in the three-dimensional (3-D) workspace [6], which makes the camera a virtual 3D Cartesian sensor (for this reason, this approach is sometimes called 3-D visual servoing). The pose of the target with respect to the camera is estimated from image features corresponding to the perspective projection of the target in the image. Numerous methods exist to recover the pose of an object which are all based on the knowledge

of a perfect geometric model of the object and necessitate a calibrated camera to obtain unbiased results (e.g. [7]). On the other hand, in an image-based control system, the input is computed in the 2-D image space [8]. IBVS controllers have gained more popularity due to the shortcomings of the PBVS method. Any error in calibration of the lens camera system causes errors in 3D reconstruction and thus leads to erratic task execution. In addition, since the control law for PBVS is expressed in workspace, there is no mechanism to regulate the error in the image space and the target features may exit the field of view (FOV).

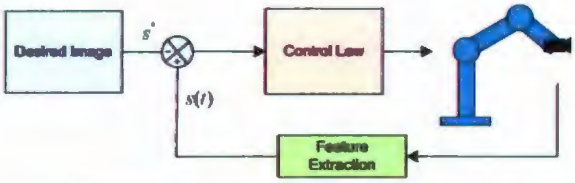
1.2 Problem statement

Although the IBVS approach reduces computational delay and eliminates errors due to sensor modeling and camera calibration [3], it poses a significant difficulty on controller design since the overall plant is nonlinear and highly coupled leading to non convergence issues of the controller [9]. The introduction of kinematic and dynamic constraints on the induced image and camera trajectories is also nontrivial [10]. The following is a description of the problems associated with IBVS :

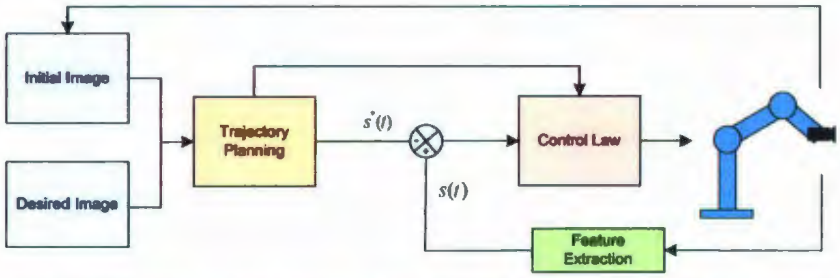
- Sometimes when the camera displacement to realize is large, camera trajectory induced by IBVS is physically invalid and/or non-optimal due to the nonlinearity and potential singularities in the relation form the image space to the workspace [11, 12].

- Introducing constraints such that the target remains in the camera FOV and/or such that the robot avoids its joint limits during servoing is not trivial in classical PBVS and IBVS control techniques [10, 11].

- During visual servo, feature correspondences may be lost due to occlusion in



1.1.1: Classical approach to image-based visual servoing



1.1.2: Proposed image-based visual servoing using a path planning scheme

Figure 1.1: Comparison of classical IBVS and proposed method for robust IBVS using path planning

the image space which leads to the failure of the visual servoing process.

This research will focus on the development of a method to provide more robustness to visual servoing task. It describes a flexible and extendible path planning scheme that provides the possibility to control the camera motion with different constraints so that the target does not exit FOV. The visual servoing task is carried out using a tracker control scheme where a trajectory generator is developed to form a path between the initial and desired poses. Additionally, this thesis entails the development of a method to avoid obstacles and occlusion during visual servo which are two problems in classical visual servo aside from the convergence problems. Figure 1.1 illustrates the difference between the classical IBVS and the proposed method.

1.3 Research objectives

The main focus of this research is to develop an off-line path planning scheme to achieve more robust visual servoing. Furthermore, introduction of constraints into the camera and image trajectories induced by visual servo control law is not a trivial task. Thus the purpose of the trajectory planning for visual servoing is to find a series of intermediate feature images that takes the initial image to the goal image while taking the required constraints into account, namely visibility constraint and occlusion avoidance constraint, avoiding the obstacle and robot joint limits. The main focus of this research is to design a trajectory-generator such that

- keeps target features in FOV all the time,
- robot avoids its joint limits, and
- the generated path is not in collision and not occluded with any obstacles in the workspace.

Additionally, the developed path planning scheme is intended to be flexible and easily extendable to accommodate other kinematic and dynamic constraints. A detailed explanation of the requirements and the problem is presented in Chapter 3, Section 3.1.2.

1.4 Contributions of the thesis

The proposed method builds a flexible platform using probabilistic roadmaps for path planning for visual servoing by generating a trajectory that is further fed into a trajectory following visual servo controller. This thesis develops the required constraints for probabilistic roadmap to impose on the generated path. One of the advantages

of the method is that it does not require a 3D model of the target. Vision-based occlusion avoidance, obstacle avoidance and circumvention will be provided with the aid of visual information available from the eye-in-hand configuration.

The resulting contributions of this thesis can be highlighted as follows:

1. Development of a visual path planning scheme using probabilistic roadmaps that makes classical visual servoing robust that relies only on two given initial and desired images.

2. Development and implementation of two constraints for PRM visual path planning :

   Visibility Constraint This constraint will force the target features to remain in FOV of the camera.

   Occlusion Constraint Visual occlusion avoidance algorithm is developed in PRM framework that avoids occlusion of the features by any obstacles.

3. Successful results are demonstrated to validate the proposed path planning scheme.

1.5 Thesis organization

Chapter 2 provides the fundamental concepts necessary for the research performed in this thesis. This chapter reviews manipulator kinematics, IBVS control and some topics from computer vision.

Chapter 3 develops the machinery required for path planning and then describes the proposed PRM-based path planning scheme designed using the tools developed earlier.

Chapter 4 shows the off-line experiments conducted to verify the proposed method. Advantages and disadvantages of the method are also discussed in this chapter.

Chapter 5 provides concluding discussion about this research and presents some future works.

Chapter 2

Background

About this chapter: This chapter will first review some basic concepts from computer vision and robot kinematics. Then the fundamentals of image-based visual servoing (IBVS) are introduced. A discussion on the performance of different IBVS control schemes is presented.

2.1 Fundamentals and concepts

This section presents an overview of some of the fundamentals of computer vision. A brief explanation of the kinematics of Catalyst5 robot arm used in this research will follow.

2.1.1 Lens and camera modeling

To control the robot using visual information, it is mandatory to understand the geometric aspects of the imaging process. Each camera is equipped with a lens that forms a 2D projection of the scene on the image plane where the sensor is located.

Among several projection models for vision system modeling, *perspective projec-*

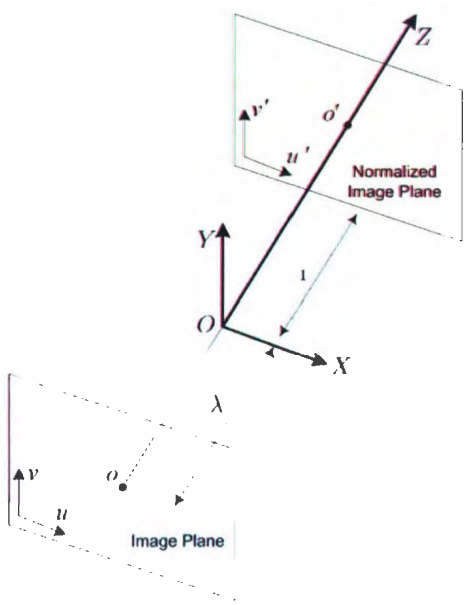


Figure 2.1: Camera coordinate system and image plane

tion is employed to model the projective geometry of the camera as it is widely used in visual servoing. A camera coordinate system is assigned such that the x-axis and y-axis form a basis for the image plane. The z-axis is considered perpendicular to the image plane along the optic axis. The origin of the image plane is located at distance λ behind the camera coordinate system, where λ is the focal length of the camera lens (Figure 2.1).

Using the perspective projection model, a point $\mathbf{M} = [\widetilde{\mathbf{M}}\ 1]^T = [X\ Y\ Z\ 1]^T$ whose coordinates are expressed with respect to the camera coordinate system, will be projected onto the image plane with coordinates $\mathbf{m} = [x\ y\ 1]^T = [\frac{X}{Z}\ \frac{Y}{Z}\ 1]^T$. The corresponding coordinates in pixel on the image plane will be denoted $\mathbf{p} = [\tilde{\mathbf{p}}\ 1]^T = [u\ v\ 1]^T$, which will be related to each other through

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \widetilde{\mathbf{C}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.1}$$

where

$$\widetilde{\mathbf{C}} = \begin{bmatrix} -\lambda K_u & 0 & u_0 & 0 \\ 0 & -\lambda K_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

The points such that $S = 0$ are called points at infinity of the image plane. If $S \neq 0$, then $u = \dfrac{U}{S}$ and $v = \dfrac{V}{S}$. By expressing the quantities X, Y, Z and λ in meters and u and v in pixel units, then the equations

$$u = \frac{U}{S} = -\lambda K_u \frac{X}{Z} + u_0 \tag{2.3}$$

$$v = \frac{V}{S} = -\lambda K_v \frac{Y}{Z} + v_0 \tag{2.4}$$

show that K_u and K_v are expressed in $pixel \times m^{-1}$. The quantities $1/K_u$ and $1/K_v$ can be interpreted as the size of the horizontal and vertical pixels in meters, respectively. The parameters u_0 and v_0 are the pixel coordinates of the principle point.

Matrix $\mathbf{C}$ is a nonsingular matrix containing the intrinsic parameters of the camera lens system and can be rewritten as

$$\mathbf{C} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

where $\alpha_u = -\lambda K_u$ and $\alpha_v = -\lambda K_v$ are expressed in pixels. These parameters can

be interpreted as the focal length in horizontal and vertical directions, respectively. According to this projection matrix, equation (2.1) can be expressed as

$$\mathbf{p} = \mathbf{C}\mathbf{m} \tag{2.6}$$

2.1.2 Manipulator kinematics

A serial manipulator consists of a series of links connected by means of kinematic pairs or joints. The whole structure forms an open kinematic chain where one end of it is constrained to a base and an end-effector is connected to the other end of it. The aim of this section is to derive the *forward kinematics* i.e. compute the position and orientation of the end-effector with respect to the manipulator base as a function of the joint variables.

Denavit-Hartenberg (D-H) convention [13] is adopted to define and compute the coordinate transformation between the links of a manipulator with five revolute joints. The coordinate transformation between i-th coordinate frame and $(i-1)$-th for two successive joints is given by

$$^{i-1}T_i = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & -c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

where c and s denote, respectively, *cos* and *sin* of the specified angles. Using the coordinate systems established in figure 2.2, the corresponding D-H link parameters are listed in table 2.1 where $a_2 = a_3 = d_1 = 254$ and $d_5 = 50.8$.

The transformation matrices of robot links are obtained using D-H transformation

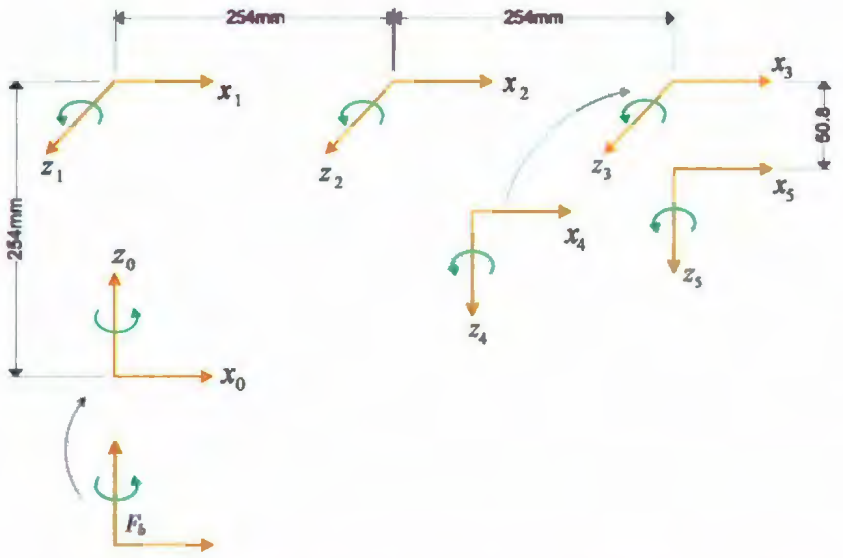


Figure 2.2: Coordinate System of Catalyst 5-DOF manipulator

Table 2.1: D-H parameters of Catalyst 5-DOF manipulator

Joints	α_i	a_i	d_i	θ_i
1	$\pi/2$	0	d_1	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3
4	$\pi/2$	0	0	θ_4
5	0	0	d_2	θ_5

matrix (2.7) and are then used to compute the overall transformation matrix between the base coordinate system $\mathcal{F}_b$ and the coordinate system of the camera attached to the end-effector $\mathcal{F}_c$ or simply

$$^0\mathbf{T}_c = {}^0\mathbf{T}_1\,{}^1\mathbf{T}_2\,{}^2\mathbf{T}_3\,{}^3\mathbf{T}_4\,{}^4\mathbf{T}_5\,{}^5\mathbf{T}_c \tag{2.8}$$

$^5\mathbf{T}_c$ is transformation matrix between the 5th frame and the camera coordinate system which should be obtained through an eye-hand calibration procedure. In the rest of the thesis, the following notations will be used to denote forward and inverse

kinematics.

$$\begin{cases} (\mathbf{R}, \mathbf{t}) = \text{ForwardKinematics}(q) = \mathcal{FK}(q) \\ q = \text{InverseKinematics}(\mathbf{R}, \mathbf{t}) = \mathcal{IK}(\mathbf{R}, \mathbf{t}) \end{cases} \tag{2.9}$$

where q is the joint angle vector $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$. $\mathbf{R}$ and $\mathbf{t}$ denote, respectively, the rotational and translational components of ${}^0\mathbf{T}_c$ corresponding to q .

2.1.3 Projective homography

There exists an analytic transformation from one image frame coordinates to the other image frame coordinates, if the tokens on the image plane that have correspondences are produced by visual features situated in a plane. This analytic transformation is a *collineation* between the two image planes considered as projective planes and is a function of the rotation and translation of the plane parameters between two image frames [14].

Figure 2.3 illustrates the geometry of the constraint between two images. The origin of the camera at its initial position is at $\mathcal{F}_i$ in figure 2.3. The position and orientation of the second camera position with respect to the first one is defined by the translation vector ${}^i\mathbf{t}_*$ and the rotation matrix ${}^i\mathbf{R}_*$.

Let $\mathbf{M} = [\widetilde{\mathbf{M}}\ 1]^T = [X\ Y\ Z\ 1]^T$ be a point on a plane II. The coordinates $\mathbf{M}_1$ and $\mathbf{M}_2$ of $\mathbf{M}$ in the two coordinate systems of the camera are related by

$$\mathbf{M}_1 = {}^i\mathbf{H}_*\mathbf{M}_2 \tag{2.10}$$

where ${}^i\mathbf{H}_*$ denotes the Euclidean homography.

Since the Euclidean position of $\mathbf{M}$ in two camera frames cannot be directly measured, equation (2.10) has to be related to the measurable image-space coordinates

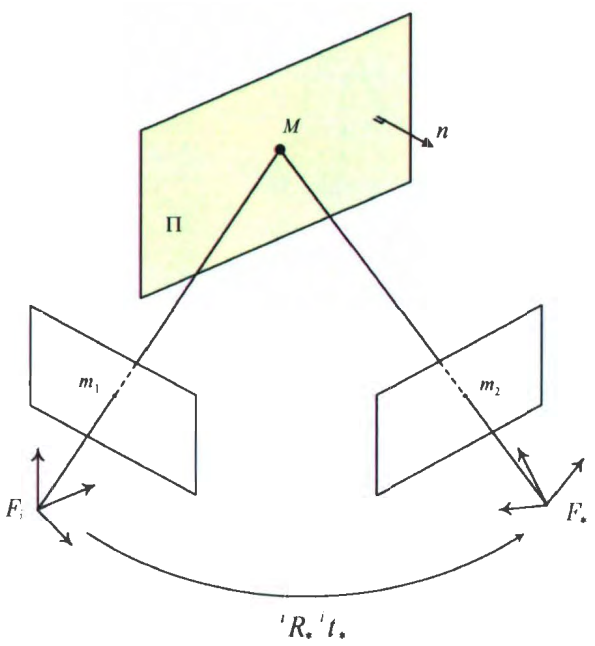


Figure 2.3: Geometry of the homography looking at a plane

$\mathbf{m} = [x\ y\ 1]^T$, given by

$$\alpha \mathbf{m}_1 = {}^i\mathbf{H}_* \mathbf{m}_2 \tag{2.11}$$

where $\alpha = \dfrac{Z_1}{Z_2}$ is the depth ratio of $\mathbf{M}$ between the first and the second camera coordinate systems. Using equation (2.6) and substituting

$$\mathbf{p_1} = \mathbf{C}\mathbf{m_1}\ ,\ \mathbf{p_2} = \mathbf{C}\mathbf{m_2} \tag{2.12}$$

into equation (2.11), the homography relationship can be expressed as

$$\alpha \mathbf{p_1} = {}^i\mathbf{G}_* \mathbf{p_2} \tag{2.13}$$

where ${}^i\mathbf{G}_* \in \mathbb{R}^{3\times3}$, given by

$${}^i\mathbf{G}_* = \mathbf{C}\,{}^i\mathbf{H}_*\mathbf{C}^+ \tag{2.14}$$

denotes the projective homography.

In practice, ${}^{i}\mathbf{G}_{*}$ is computed from the target feature correspondences. If the target is planar, ${}^{i}\mathbf{G}_{*}$ can be estimated solving a linear system using at least four points of this plane [15, 16]. On the other hand, if the target is non-planar, the estimation of ${}^{i}\mathbf{G}_{*}$ becomes a nonlinear problem. Then three points can be used to define plane Π and five supplementary points not belonging to Π are needed to estimate ${}^{i}\mathbf{G}_{*}$. Linearized algorithm presented in [16] can be used to estimate the homography matrix in real-time. Classical linearized methods for computing the camera displacement based on the epipolar geometry can be found in [17] and [18] but near the convergence of the visual servo control where the current and desired images become similar, the epipolar base line becomes smaller and thus the epipolar geometry becomes degenerate and the estimate of the partial pose between the two views is biased with error [11].

Using equation (2.14), the Euclidean homography ${}^{i}\mathbf{H}_{*}$ of plane Π is obtained from

$$ {}^{i}\mathbf{H}_{*} = \mathbf{C}^{-1}\,{}^{i}\mathbf{G}_{*}\mathbf{C} \tag{2.15} $$

and can be decomposed into a rotation matrix and a rank-1 matrix [14, 15] of the form

$$ {}^{i}\mathbf{H}_{*} = {}^{i}\mathbf{R}_{*} + \frac{{}^{i}\mathbf{t}_{*}\mathbf{n}^{*T}}{d^{*}} \tag{2.16} $$

where plane Π is defined by its normal $\mathbf{n}^{*}$ expressed in the coordinate system of $\mathcal{F}_{*}$, and its distance d^{*} to the origin of $\mathcal{F}_{*}$.

2.2 Image-based visual servoing

Visual servo control refers to the use of computer vision data to control the motion of a robot. The vision data is usually acquired from a camera that is mounted directly

on a robot manipulator where motion of the robot induces camera motion, or the camera can be fixed in the workspace. The mathematical underpinning of all these different configurations is similar and in this review the focus is primarily on the former, so-called eye-in-hand case.

Visual servo control relies on techniques from several research fields such as image processing, computer vision, control theory, kinematics and dynamics, mainly. Since the main purpose of this work is to develop a robust path planning technique for IBVS, this section will focus primarily on issues related to control and stability issues that are uniquely relevant to the study of IBVS control schemes. IBVS control solutions regulate error in the image space and thus ensure a local stability and convergence in the presence of modeling error and noise perturbations. A review of classical image-based visual servo will be given and a brief discussion on the performance of IBVS is provided.

2.2.1 Background

The purpose of IBVS control schemes is to minimize an error $\mathbf{e}(t)$, which is defined by

$$\mathbf{e}(t) = \mathbf{s}(t) - \mathbf{s}^*(t) \tag{2.17}$$

$\mathbf{s}(t)$ corresponds to visual features obtained from image measurements. For a fiducial point target, $\mathbf{s}(t)$ represents the image coordinates of interest points or for a circular target, $\mathbf{s}(t)$ contains the image coordinates of the centroid of the object. These calculations are performed with some additional knowledge about the system such as coarse camera intrinsic parameters. The vector $\mathbf{s}^*(t)$ contains the desired values of the features. In case of a fixed desired pose, $\mathbf{s}^*(t)$ is a fixed vector, denoted as $\mathbf{s}^*$. For a camera mounted on a manipulator end-effector (eye-in-hand configuration) changes

in $\mathbf{s}(t)$ depend only on camera motion, if the target is assumed motionless.

To design an IBVS controller, the relationship between $\mathbf{s}$ and the camera velocity is required. Let $\mathbf{s} = (u, v)$ be the current value of visual features observed by the camera and $\mathbf{s}^*$ be the desired value of $\mathbf{s}$ to be reached in the image space and $\dot{\mathbf{r}} = [\mathbf{v}^T, \omega^T]^T$ represent the camera velocity with $\mathbf{v}^T$ being the instantaneous linear velocity of the origin of the camera frame and ω^T being the instantaneous angular velocity of the camera frame. This relationship is given by

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{s}}{\partial \mathbf{r}}(\mathbf{s}, Z, \mathbf{C})\, \dot{\mathbf{r}} = \mathbf{I}(\mathbf{s}, Z, \mathbf{C})\, \dot{\mathbf{r}} \tag{2.18}$$

in which $\mathbf{I} \in \mathbb{R}^{2n \times 6}$ is the *interaction matrix* or *image Jacobian* associated with $\mathbf{s}$ for n features and can be derived [1] by using the perspective lens camera model and motion dynamics for a mounted camera as

$$\mathbf{I}(\mathbf{s}, Z, \mathbf{C}) = \begin{bmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{bmatrix} \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -1-x^2 & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{2.19}$$

where x and y are given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_u} & 0 \\ 0 & \frac{1}{\alpha_v} \end{bmatrix} \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} \tag{2.20}$$

In case of eye-to-hand configuration, $\mathbf{I} = -\mathbf{I}$.

Using equations (2.17) and (2.18), the relationship between camera velocity and the time variation of the error is derived as

$$\dot{\mathbf{e}} = \dot{\mathbf{s}}(t) - \dot{\mathbf{s}}^*(t) = \mathbf{I}\,\dot{\mathbf{r}} + \frac{\partial \mathbf{e}}{\partial t} - \frac{\partial \mathbf{s}^*(t)}{\partial t} \tag{2.21}$$

where the term $\frac{\partial \mathbf{e}}{\partial t}$ is the time variation of $\mathbf{e}$ due to target motion and $\frac{\partial \mathbf{s}^*(t)}{\partial t}$ denotes the time-varying reference image features.

The existing control schemes compute the camera velocity sent to the robot controller [9, 19, 20] with the form

$$\dot{\mathbf{r}} = \mathbf{I}^+ \dot{\mathbf{e}} \qquad \text{or} \qquad \dot{\mathbf{r}} = f(\mathbf{I}^+ (\mathbf{s} - \mathbf{s}^*)) \tag{2.22}$$

since most of the commercial robotic systems accept velocity input. If joint limits avoidance and singularity avoidance is required, the robot Jacobian is used to map the camera velocity directly to the robot joints velocity. In equations (2.22), function f can be any control scheme from a simple proportional gain [21] to optimal control in state space [22] or nonlinear control [23].

In order to ensure an exponential decoupled decrease in error (e.g. $\dot{\mathbf{e}} = -\kappa \mathbf{e}$) and using (2.22) (assuming that the target is motionless), the control law is expressed as

$$\dot{\mathbf{r}} = -\kappa \, \mathbf{I}^+ \, \mathbf{e} \tag{2.23}$$

where $\mathbf{I}^+$ represents the Moore-Penrose pseudoinverse of $\mathbf{I}$ and $\dot{\mathbf{r}}$ denotes the input to the robot controller.

In real visual servo systems, $\mathbf{I}$ or $\mathbf{I}^+$ cannot be perfectly determined. So an approximation or an estimation of one of these two matrices must be realized. In the sequel, both the pseudoinverse of the approximation of the interaction matrix and the approximation of the pseudoinverse of the interaction matrix will be denoted by $\widehat{\mathbf{I}}^+$. Using this notation, the control law is in fact:

$$\dot{\mathbf{r}} = -\kappa \, \widehat{\mathbf{I}}^+ \, \mathbf{e} \tag{2.24}$$

There are several choices available for constructing $\widehat{\mathbf{I}}^{+}$:

- $\widehat{\mathbf{I}}^{+} = \widehat{\mathbf{I}}^{+}(t)$: In this case, the interaction matrix is numerically estimated using either an off-line learning step or an on-line estimation step during visual servoing without using the analytical form given by (2.19). Neural networks have been used to perform the estimation [24]. The Broyden update rule can be used for on-line iterative estimation of $\widehat{\mathbf{I}}^{+}$ [25]. The main advantage of using such numerical estimations is that it avoids all the modeling of camera and robot. It is particularly useful when using features whose interaction matrix is not available in analytical form.

- $\widehat{\mathbf{I}}^{+} = \widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$: In this case the interaction matrix is updated at each iteration using the depth information $\widehat{Z}$ of each feature point [10,22]. Depth information plays an important role in the convergence of the control scheme. $\widehat{Z}$ can be obtained either from the 3D model of the target or the camera motion. Although the image trajectories of points are straight, camera motion is shown to be far from straight due to the condition number of $\widehat{\mathbf{I}}^{+}$, leading to local minima and approaching of visual servo task singularities.

- $\widehat{\mathbf{I}}^{+} = \widehat{\mathbf{I}}^{+}(\mathbf{s}^{*}, \widehat{Z^{*}}, \widehat{\mathbf{C}})$: $\widehat{\mathbf{I}}^{+}$ is constant and is equal to $\widehat{\mathbf{I}}^{+}$ at $\mathbf{s}^{*}$ where $\mathbf{e} = 0$ [9,26]. In this case, no time-variant 3D parameter including depth estimation is performed during visual servo. It is shown that this method provides more satisfactory results than using the previous method. However the behavior of the control scheme (computed camera velocity and the 3-D trajectory of the camera) could be inadequate when the camera displacement to realize is quite large, although it might converge [27].

- $\widehat{\mathbf{I}}^{+} = 1/2\left(\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}}) + \widehat{\mathbf{I}}^{+}(\mathbf{s}^{*}, \widehat{Z^{*}}, \widehat{\mathbf{C}})\right)$: It is reported that this choice provides

good performance in practice [27, 28]. In this case, the camera velocity components do not include large oscillations and provide a smooth trajectory both in the image and in 3-D. However since $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$ is involved in this method, the current depth of the features must be estimated.

The velocity screw of the camera $\dot{\mathbf{r}}$ obtained from equation (2.24) is expressed in the coordinate system of the camera. The equivalent velocity screw in base coordinates of the robot arm is obtained [13] by

$$\begin{bmatrix} {}^{b}\mathbf{v}_{b} \\ {}^{b}\omega_{b} \end{bmatrix} = \begin{bmatrix} {}^{b}\mathbf{R}_{c} & -{}^{b}\mathbf{R}_{c}\,{}^{b}\mathbf{t}_{c}\times \\ 0 & {}^{b}\mathbf{R}_{c} \end{bmatrix} \begin{bmatrix} {}^{c}\mathbf{v}_{c} \\ {}^{c}\omega_{c} \end{bmatrix} \tag{2.25}$$

where $\mathbf{t}\times$ is a skew-symmetric matrix operator obtained from

$$\mathbf{t}\times = \begin{bmatrix} 0 & -\mathbf{t}_{z} & \mathbf{t}_{y} \\ \mathbf{t}_{z} & 0 & -\mathbf{t}_{x} \\ -\mathbf{t}_{y} & \mathbf{t}_{x} & 0 \end{bmatrix} \tag{2.26}$$

IBVS is known to perform satisfactory in the presence of important intrinsic and extrinsic parameter calibration errors. However it is known that it suffers from stability and convergence problems [9, 10]. $\mathbf{I}$ and thus $\widehat{\mathbf{I}}^{+}$ may become singular leading to an unstable behavior. In addition, due to the existence of unrealizable image motions, local minima may be reached. In [9], it is shown that the control scheme employing $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$ may reach local minima, while employing $\widehat{\mathbf{I}}^{+}(\mathbf{s}^{*}, \widehat{Z^{*}}, \widehat{\mathbf{C}})$ allows the system to avoid local minima if it computes image motions which are not unrealizable. Therefore $\widehat{\mathbf{I}}^{+}(\mathbf{s}^{*}, \widehat{Z^{*}}, \widehat{\mathbf{C}})$ is sometimes more interesting to use than $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$, specifically when visual servoing is performed in the vicinity of desired robot pose although there is no proof.

A simple simulation demonstrates this issue. Figures 2.4 and 2.5 show, respectively, the convergence of the visual servoing task using $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$ and $\widehat{\mathbf{I}}^{+}(\mathbf{s}^{*}, \widehat{Z^{*}}, \widehat{\mathbf{C}})$. The simulation of visual servo using $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$ converges in virtue of the assumption of a wide FOV. It is important to note the boundary of FOV (dash-dotted line) of the camera in Figure 2.4.4. It is obvious that the visual servo using $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$ does not converge in real world since the target exits FOV. However visual servoing with $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$ converges and results in a smoother performance.

This property is used to develop a path planning scheme in Chapter 3 by introducing subgoal images to the visual servoing control scheme using $\widehat{\mathbf{I}}^{+}(\mathbf{s}^{*}, \widehat{Z^{*}}, \widehat{\mathbf{C}})$ at each subgoal so that it takes the initial image to the desired one such that the error remains small all the time.

2.2.2 Local stability analysis

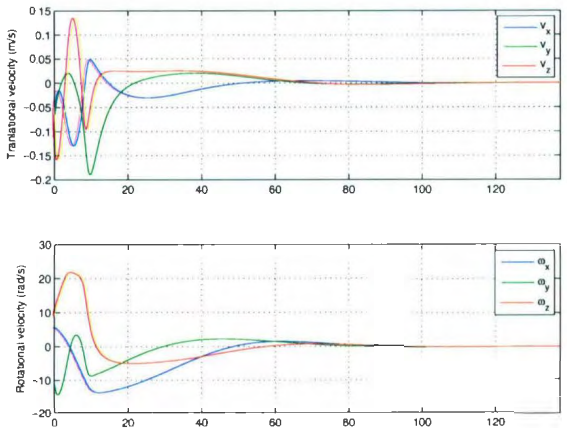
This section reviews the stability of the closed-loop visual servo systems using Lyapunov analysis [27]. Let's choose the squared error norm as the candidate Lyapunov function defined by $\mathfrak{L} = 1/2\|\mathbf{e}(t)\|^{2}$ whose derivative is given by

$$\dot{\mathfrak{L}} = \mathbf{e}^{T}\dot{\mathbf{e}} = -\kappa \mathbf{e}^{T}\mathbf{I}\widehat{\mathbf{I}}^{+}\mathbf{e} \tag{2.27}$$

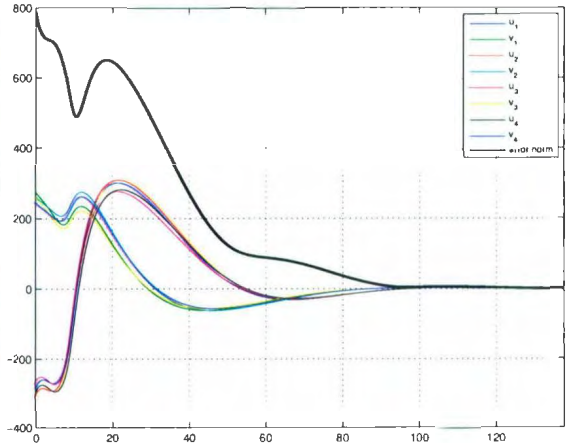
Therefore a sufficient condition to ensure the global stability is given by

$$\mathbf{I}\widehat{\mathbf{I}}^{+} > 0 \tag{2.28}$$

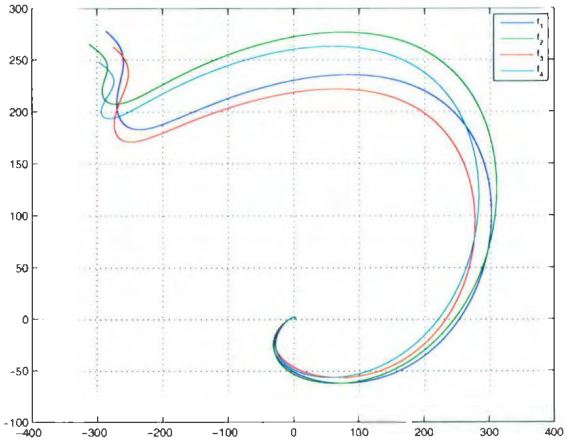
In practice, for most of visual servo control schemes condition (2.28) can never be ensured since $\mathbf{I}\widehat{\mathbf{I}}^{+} \in \mathbb{R}^{2n\times 2n}$ is at most a matrix of rank 6 i.e. $\mathbf{I}\widehat{\mathbf{I}}^{+}$ has a null space that cannot be easily determined.



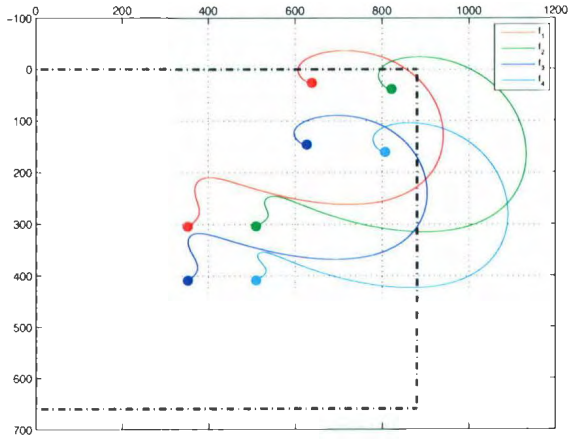
2.4.1: Camera velocities (rad/s and m/s)



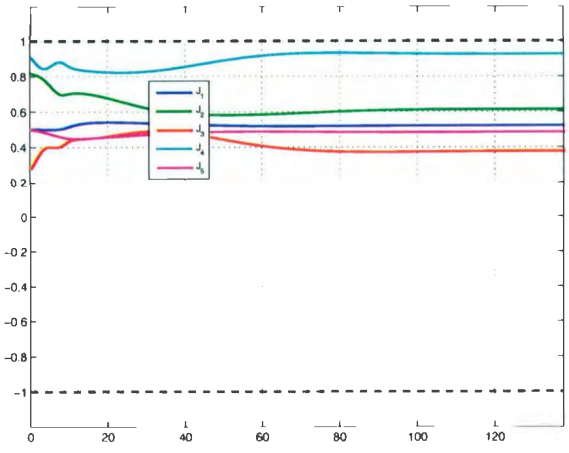
2.4.2: Error ($\mathbf{s}(t)$ - $\mathbf{s}^*$)



2.4.3: Error trajectory ($\mathbf{s}(t) - \mathbf{s}^*$) in pixels in image space

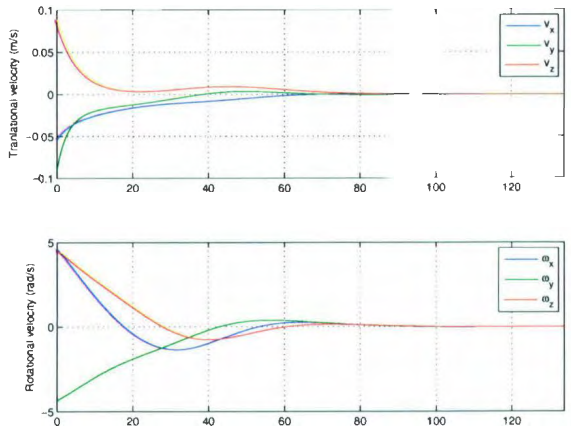


2.4.4: Target trajectory $\mathbf{s}(t)$ in image space

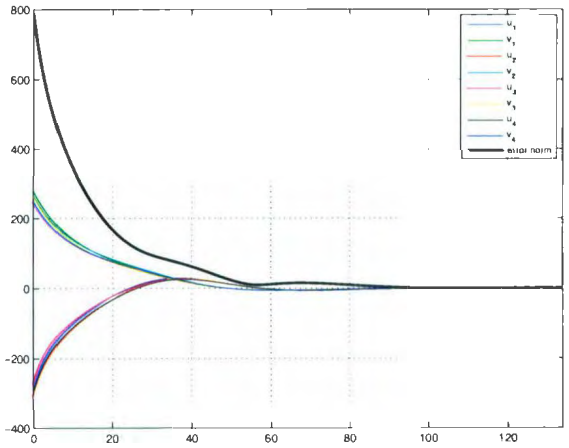


2.4.5: Normalized trajectories in joint space

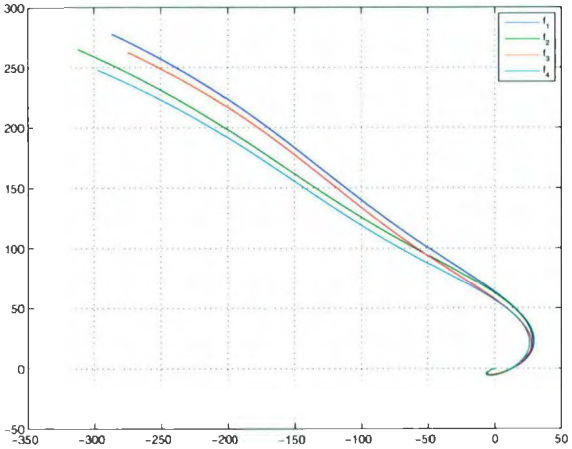
Figure 2.4: Visual Servoing using $\widehat{\mathbf{I}}^{+}(\mathbf{s}, \widehat{Z}, \widehat{\mathbf{C}})$



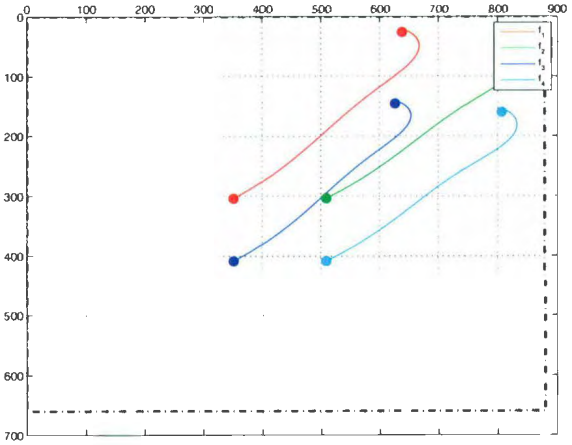
2.5.1: Camera velocities (rad/s and m/s)



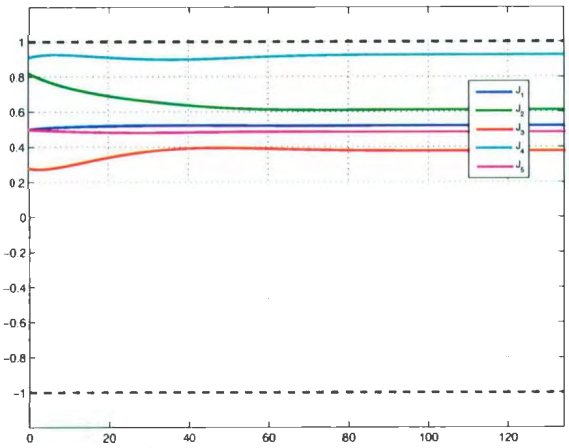
2.5.2: Error ($\mathbf{s}(t)$ - $\mathbf{s}^*$)



2.5.3: Error trajectory ($\mathbf{s}(t) - \mathbf{s}^*$) in pixels in image space



2.5.4: Target trajectory $\mathbf{s}(t)$ in image space



2.5.5: Normalized trajectories in joint space

Figure 2.5: Visual Servoing using $\widehat{\mathbf{I}}^{+}(\mathbf{s}^*, \widehat{Z^*}, \widehat{\mathbf{C}})$

Using control law (2.24), each component of $\mathbf{e}$ is ensured a exponential decrease with the same convergence speed, causing straight-line trajectories to be realized in the image space. It is however reported that the error reached may not be exactly zero, and it is obvious that the system has been attracted to a local minimum far away from the desired configuration. To guarantee the local asymptotic stability, a new error $\mathbf{e}' = \widehat{\mathbf{I}}^{+}\mathbf{e}$ is defined. Taking derivative from both sides and using (2.21), the following equation is obtained

$$\dot{\mathbf{e}}' = \widehat{\mathbf{I}}^{+}\dot{\mathbf{e}} + \dot{\widehat{\mathbf{I}}}^{+}\mathbf{e} = (\widehat{\mathbf{I}}^{+}\mathbf{I} + \mathbf{O})\dot{\mathbf{r}} \tag{2.29}$$

where $\mathbf{O} \in \mathbb{R}^{6\times 6}$ is zero when $\mathbf{e} = 0$ independent of $\widehat{\mathbf{I}}^{+}$. Substituting the control scheme (2.24), equation (2.29) is rewritten as

$$\dot{\mathbf{e}}' = -\kappa(\widehat{\mathbf{I}}^{+}\mathbf{I} + \mathbf{O})\mathbf{e}' \tag{2.30}$$

which is known to be locally asymptotically stable in a neighborhood of $\mathbf{e} = \mathbf{e}^{*} = 0$ if condition (2.28) is ensured. Since the local asymptotic stability is of interest, only the linearized system $\dot{\mathbf{e}}' = -\kappa\widehat{\mathbf{I}}^{+}\mathbf{I}\mathbf{e}'$ has to be considered. If $\widehat{\mathbf{I}}^{+}$ and $\mathbf{I}$ are of rank 6 and the approximation involved in $\widehat{\mathbf{I}}^{+}$ is not coarse then condition (2.28) is ensured.

In order to end the demonstration of local stability, it has to be shown that there does not exist any configuration $\mathbf{e} \neq \mathbf{e}^{*}$ such that $\mathbf{e} \in \mathrm{Ker}\,\widehat{\mathbf{I}}^{+}$ in a small neighborhood of $\mathbf{e}^{*}$ and its corresponding pose $\mathbf{r}^{*}$.

Configurations where $\dot{\mathbf{r}} = 0$ and $\mathbf{e} \neq \mathbf{e}^{*}$ correspond to local minima. If such a pose $\mathbf{r}$ would exist, it is possible to restrict the neighborhood around $\mathbf{r}^{*}$ so that there exists a camera velocity $\dot{\mathbf{r}}$ to reach $\mathbf{r}^{*}$ from $\mathbf{r}$ which would imply a variation of the error $\dot{\mathbf{e}} = \mathbf{I}\,\dot{\mathbf{r}}$. However, such a variation cannot belong to $\mathrm{Ker}\,\widehat{\mathbf{I}}^{+}$ since $\mathbf{I}\widehat{\mathbf{I}}^{+} > 0$.

Therefore, $\dot{\mathbf{r}} = 0$ if and only if $\dot{\mathbf{e}} = 0$ which implies $\mathbf{e} = \mathbf{e}^*$, in a neighborhood of $\mathbf{r}^*$.

Even though local asymptotic stability can be ensured when $n > 6$, global asymptotic stability cannot be guaranteed. There may exist local minima corresponding to configurations where $\mathbf{e} \in \text{Ker}\, \widehat{\mathbf{I}}^+$ which is outside of the stability neighborhood mentioned above. Although the convergence neighborhood is quite large in practice, there is no mechanism to determine the size of it.

Singularities of the Jacobian will force the system to induce unexpected robot trajectories which might be due to the singularities in the interaction matrix or the robot Jacobian.

2.2.3 Kinematic visual feedback control

The control law (2.24) produces a pose rate control signal that, for a position-controlled robot, must be integrated to determine the robot joint angle control signal. The integration can be performed in joint space or workspace.

The Cartesian velocity control signal can be resolved to joint velocity control signal and integrated

$$\mathbf{q} = \int \mathbf{J}^+(\dot{\mathbf{r}})\, dt \tag{2.31}$$

where $\mathbf{J}^+$ denotes the generalized inverse of the robot Jacobian $\mathbf{J}$.

The resolved velocity control is less robust since numerical errors in the computed robot Jacobian result in a workspace velocity slightly different from that demanded, causing the robot's pose to drift slowly with time. Since the performance of *strong* kinematic control using inverse kinematics overweighs that of the above resolved-rate control through robot Jacobian (i.e. *weak* kinematic control) due to the mentioned incapability, integration in workspace is preferred [29]. In this control technique, desired workspace velocity is integrated and the corresponding joint positions are

obtained using inverse kinematics

$$\mathbf{q} = \mathcal{IK}(\int \dot{\mathbf{r}}\ dt) \tag{2.32}$$

2.3 Summary

Some fundamental concepts from vision system modeling are presented. The projective homography i.e. relation between the correspondences in different views is studied. The formulation to compute and decompose this relation into camera transformation is reviewed. The basics of IBVS control are presented in this chapter and various choices of the interaction matrix for IBVS with a brief discussion on the advantages and disadvantages of each method is provided and it is shown that visual servoing using a constant interaction matrix at the desired pose sometimes provides more robustness to control solution. This chapter also reviews the proof of the local stability of IBVS which holds when the error to regulate is small. It is also explained that integration of the velocity screw in workspace rather than in joint space provides a *stronger* kinematic control.

Chapter 3

Path-planning for visual servoing

About this chapter: This chapter introduces a path planning technique to improve the robustness of the pure image-based visual servoing (IBVS) control scheme. A thorough review of the state-of-the-art techniques used for robust IBVS control and robot motion planning using sampling-based methods will be presented. First the definitions and the formulations for partial target reconstruction and occlusion avoidance are developed which will then be used to design a visual servoing trajectory planner based on probabilistic roadmaps (PRM).

3.1 Introduction

IBVS is a popular vision feedback control loop technique which measures the error signal directly in sensor space and maps to workspace. Since the error is regulated directly in image space, stability and convergence in the presence of modeling error and noise perturbations is ensured locally [26]. However, sometimes (and specifically) when the initial and desired configurations are distant, the camera trajectory induced by IBVS control scheme is neither physically valid nor optimal due to the nonlinearity

and singularities in the relation from image space to the workspace expressed in equation (2.18) [9]. More precisely, the control scheme can cause excessive control action and transient response which can cause the target to leave the FOV. Visual servoing control solutions are local feedback control schemes and thus require the definition of intermediate subgoals in the sensor space at the task planning level [11]. Therefore, this chapter proposes a path planning solution that uses the local stability of IBVS (discussed in Chapter 2) by specifying sufficient trajectories to be followed in the image space. If the initial error is too large, a reference trajectory including a sequence of images can be designed. The initial error is sampled so that at each iteration of the control scheme, the error to regulate remains small; thus exploiting the local stability of IBVS. In other words, the approach uses the path to improve the robustness of pure IBVS such that the error is small enough for the local stability to hold. One of the other shortcomings associated with classical PBVS and IBVS control techniques is the nontriviality of accommodating constraints such that the target remains in the camera FOV (i.e. visibility constraint) or such that the robot avoids its joint limits during servoing. When the displacement to realize is large, this incapability leads to the failure of servoing process [10, 11]. The proposed method utilizes the flexible platform provided by PRM to introduce visibility, joint limit, obstacle avoidance and occlusion constraints.

3.1.1 Related Work

In the following, a detailed review of the state-of-the-art techniques used to improve the performance and robustness of IBVS is provided. Then, a review on the application of the sampling-based methods (e.g. PRM) to manipulation planning and robot motion is presented.

Robust visual servoing

Visual servo control system design is one of the methods used to enhance the pure IBVS by partitioning the visual servo dynamics or by exploiting the advantages of IBVS and PBVS to investigate hybrid approaches. Most of these strategies ensure the local stability of the controller and fail when the displacement is large. Introducing subgoals to the visual servo controller using path planning schemes leads to more robust results by using an IBVS control scheme as a local controller. Potential function approach has been extensively utilized to perform trajectory planning in image space. Homography interpolation is another method that some researchers have investigated with some reported benefits.

In IBVS, the control loop is directly closed in the image space which ensures a local stability and convergence. Although the method regulates the error in the presence of modelling error and noise perturbations, the control scheme may yield excessive control action and transient response which can cause the target to leave the FOV and cause unspecified behavior.

Different characteristics of IBVS and PBVS have motivated several researchers to investigate hybrid approaches to improve the global stability of the control solution. Malis et al. [30] have proposed a homography-based globally stabilizing control scheme called 2 $\frac{1}{2}$-D visual servoing. This method decomposes the translational and rotational components of homography matrix and therefore performs the partial reconstruction of the target to extract the Cartesian component of the error function. It employs some advantages of IBVS and PBVS approaches to develop a control scheme that does not require the accurate geometric model of the environment or the target. The potential singularities in the interaction matrix are also eliminated since the image Jacobian matrix is designed to be triangular for homography-based

visual servo. Motivated by the advantages of the homography-based technique, various error regulation controllers for robot manipulators have been developed which are intended to improve the behavior of basic IBVS and PBVS. In [31], a Lyapunov-based homography-based adaptive control strategy is employed to actively compensate for the lack of unknown depth measurements in order for a robot end-effector to track a desired workspace trajectory as determined by a sequence of images for camera-in-hand and fixed-camera configurations. A visual servo tracking controller is developed for a monocular camera system mounted on a Unmanned Aerial Vehicle (UAV) to track a leading UAV with a fixed relative position and orientation [32]. In this work, reference desired feature points on the leading UAV are provided from a prerecorded desired image set. An iterative learning control scheme for robot planar motion visual servo with an arbitrarily mounted camera is presented in [33] which is, however, valid when the image plane and the motion plane are parallel with a constant but unknown image Jacobian matrix and uses an iterative learning control law with a Nussbaum learning gain to perform trajectory tracking in the presence of camera calibration errors. In another similar work [34], a visual servo tracking controller is developed for an underactuated wheeled mobile robot (WMR) subject to nonholonomic motion constraints with a monocular camera system mounted on it. Again, a prerecorded image sequence (e.g., a video) of three target points is used to define a desired trajectory for the WMR which is compared with the live features to create the error vector. While homography-based approaches exploit the benefits of IBVS and PBVS, a common problem with all the aforementioned approaches is the inability to achieve the control objective while ensuring a specific constraint. Introducing constraints in the realized trajectory such that the target remains in the camera FOV or such that the robot avoids its joint limits is not trivial. Without appropriate measures to account for these constraints, the object can exit the FOV or the robot may reach its joint

limits [11].

In IBVS control solution, where control is effected with respect to the image, there is no direct control over the Cartesian velocities of the robot end-effector. As a result, the robot executes trajectories that are desirable in the image, but which can be contorted in Cartesian space. To overcome this problem, Corke and Hutchinson [35] have proposed a method which decouples the z-axis rotational and translational components of the control from the remaining degrees of freedom. In [36], a method is proposed to avoid the joint limits by using a shortest-path approach which is predictable and its generated minimized straight line trajectory avoids trying to move outside the robot workspace in most cases, although there is no direct control on it. In [37], Gans and Hutchinson developed a strategy that switches between an IBVS and a PBVS controller to ensure asymptotic stability of the position and orientation (i.e. pose) in the Euclidean and image space. There is the possibility of feature points leaving the image plane in all hybrid schemes and a direct solution is required to keep the target in the FOV. However, for 6-DOF visual servoing, Malis et al. [38] guarantee that a single feature point remains within the FOV. Morel et al. [39] extend this idea by decoupling the translational motion of a custom-designed feature vector, a circle containing all the feature points, from the rotational motion of the camera to guarantee that all feature points remain within the FOV. However, in addition to the mentioned imperfections, partitioning of the control often causes the Cartesian path to become more complex, which might result in operation close to the joint limits. When the displacement to realize is large, the aforementioned deficiencies often lead to the failure of the servoing process [32].

To address the issues mentioned above to improve the global stability of visual servo, a definition of intermediate subgoals in the sensor space at the task-planning level is required. For instance, in [34], a prerecorded sequence of subgoal features

in the image plane is manually provided to the control scheme which regulates the changing error while the object is moving towards the desired goal. In [20, 40], the null space of the Jacobian is exploited to introduce other constraints such as joint limit avoidance and visibility constraints. However, if all the robot degrees of freedom are used to realize the task, the null space can not be exploited to perform secondary tasks and therefore this method is not beneficial. An alignment task using intermediate views of an object is presented in [41] which employs image morphing. A path planning for a straight-line robot translation observed by an uncalibrated stereo-rig system is performed in [42] using interpolation and homography.

Several recent papers use potential functions and navigation functions for path planning to basically introduce constraints and address the FOV problem. In [35], keeping the target in FOV is considered as a collision avoidance problem in image space and employs potential field techniques to repel the feature points from the image plane boundary. Mezouar et al. [43] use the approach of image-based path planning and local visual servoing along the intermediate subgoals on the path to avoid mechanical limits and visibility obstacles. In another work, Mezouar et al. [11, 44] developed a trajectory planning scheme which generates subgoal features for a basic visual servo control solution to follow. Potential functions are employed to perform the path planning and to introduce visibility and joint limit constraints. The proposed method does not require the 3D model of the target and an exact camera-intrinsic parameters.

Local minima associated with traditional potential functions may exist [45]. A basic strategy to take out of potential local minima is to execute a random motion by favoring the repulsive force. Obviously, reaching the global minimum is not guaranteed. To ensure such a property, specialized potential functions free of stable local minima called navigation functions (NF) are constructed which is originally proposed

by Koditschek and Rimon [46,47].

In a series of papers [48-50], Cowan et al. employ navigation functions to introduce attractive or repulsive potential to the desired pose, objects and obstacles accordingly. In [48], a globally stabilizing method using navigation function is proposed that guarantees visibility. A trajectory planner is described in [51] for stereo vision system using navigation functions and applied to obstacle avoidance. For nonholonomic mobile robots, Zhang and Ostrowski [52] adopt path planning to find kinematic trajectories that keep features within the FOV. In [53], Cowan et al. developed a hybrid position/image-space controller that forces a manipulator to a desired pose while ensuring the object to remain visible through navigation functions and by avoiding pitfalls such as self-occlusion. An image-space based follow-the-leader application for mobile robots was developed in [54] that exploits an image space navigation function. Specifically, an input/output feedback linearization technique is applied to the mobile robot kinematic model. An NF-based approach to the follow-the-leader problem for a group of fully actuated holonomic mobile robots is considered in [55] where configuration-based constraints are developed to ensure the robot edges remain in the sight of an omnidirectional camera. A Lyapunov-based analysis is provided in [55] to ensure that the NF decreases to the goal position.

Potential and navigation functions, are path planning techniques that incrementally *explore* free space while searching for a path. These path planning algorithms maneuver through free space without constructing the configuration space and can be applied to a large class of robots since they apply to a more general class of configuration spaces, including those that are multidimensional and non-Euclidean. However navigation planners based on this technique have disadvantages. Navigation functions do not suffer from the local minima problem associated with potential functions and provide the machinery to apply potential functions to second-order plants, while still

ensuring obstacle avoidance with convergence guarantees and no need for intermediate trajectories. However, constructing such a navigation function requires the complete knowledge of the space topology and the object model, and many advantages of the approaches based on potential functions such as robustness with respect to modeling errors and application to an object with an unknown CAD model will be lost [11].

In [56], a trajectory generator for a visual servoing system using stereo vision is proposed to make the system accomplish obstacle avoidance tasks in unknown environments. Using the epipolar constraint, the proposed scheme can generate trajectories for the visual servoing system on the 2D image planes to avoid obstacles without reconstructing 3D geometry.

Sampling-based path planning for manipulators

Different roadmap based planners such as visibility graphs and Generalized Voronoi Diagrams (GVD) build maps in the free configuration space. Each of these methods relies on an explicit representation of the geometry of the configuration space. Therefore these planners become impractical as the dimension of the configuration space increases [57].

In recent years, a number of sampling-based motion planning algorithms such as probabilistic roadmap planners (PRMs) [58,59], Randomized Path Planner (RPP) [60,61], and Rapidly-exploring Random Trees (RRTs) [62] have been introduced which have had considerable success in solving motion planning problems, specifically with many degrees of freedom (DOFs) [58,63]. Path planning methods can be categorized either as *single query planning* or *multiple query planning* methods. Single query planning methods compute one path for the environment fast and without preprocessing. However multiple query planning methods compute many paths for the same environment and thus the environment model can be preprocessed. PRM is one of

the most powerful and capable multiple query planning methods. Sampling-based methods utilize a variety of strategies to generate samples (i.e. collision-free configurations of the robot) and then connect the samples with paths to obtain solutions to path-planning problems [58]. Unlike earlier planners which rely on the explicit representation of the obstacle boundary in the configuration space, sampling-based methods use a collision detector as they search the configuration space. Sampling-based methods are used to address problems that extend beyond the classic path planning where dimensionality is an issue. Sampling-based methods have been applied to various research fields ranging from computer animation of human figures, centralized and decoupled planning of multiple robots to manipulation planning and assembly planning [57].

PRM checks if a single robot configuration is in collision-free space, $\mathcal{Q}_{free}$, achieved through collision detection. A configuration q is collision-free, if the robot placed at q has null intersection with the obstacles in the workspace. The free space $\mathcal{Q}_{free}$ is the set of free configurations. PRM uses collision-free configurations to create a roadmap in $\mathcal{Q}_{free}$. After the roadmap has been generated, planning queries can be answered by connecting the user-defined initial and goal configurations to the roadmap, and by using the roadmap to solve the path-planning problem at hand [58].

PRM planners are capable of dealing with robots with many degrees of freedom and with many different constraints despite their simplicity. In [64], a sampling-based planner is developed that imposes kinodynamic constraints on the path. In [65,66], a path planner for closed kinematic chains is presented that takes closed-loop kinematics into account. A fast planner is designed for vertically-climbing robots in [67] which relies on an efficient test of the quasi-static equilibrium of the robot. A sampling-based planner is used to perform reconfiguration planning of self-configuring modules using appropriate reconfigurable constraints for reconfigurable robots [68,69]. Lamiraux et

al. describe a path planning scheme for elastic objects under energy constraints using the principles of elasticity theory [70]. In [71], a planner is developed that ensures the planned path to be compliant to a desired contact constraint. In [59,72], the shortest path for the robot is designed using a visibility constraint such that each point on boundary of the workspace is visible from some point on the path. In this thesis, however, visibility constraint is meant to keep the target in FOV.

Motion planning for manipulators typically involves the finding of a collision-free path from an initial configuration of the robot to a goal configuration of the robot. The multiple movers problem deals with path planning for many robots [73]. A collision-free path in this case implies that at every step, there is no collision between a robot and an obstacle or between any two robots. Centralized planning is a solution to this problem that considers the different robots as a single multi-body robot and represents C-space, $\mathcal{Q}$, as the Cartesian product of the configuration spaces of all the robots where the dimensionality of $\mathcal{Q}$ is equal to the total number of degrees of freedom of all the robots. It is obvious that the *curse of dimensionality* causes some difficulty due to the high dimensionality of $\mathcal{Q}$. Coordination of the robots is trivially achieved since a collision-free configuration in $\mathcal{Q}$ describes the configuration of each individual robot and ensures that no robot is in collision with some obstacle or some other robot. This solution is applied to a workspace where six robots cooperate on a welding task [74]. Another solution is a two-phase decoupled planning where collision-free paths are initially computed for each robot individually, without taking into account the other robots but simply considering the obstacles of the workspace. In the second stage, coordination is achieved by computing the relative velocities of the robots along their individual paths that will avoid collision among them [73]. Decoupled planning does not increase the dimensionality of the configuration space but it is incomplete, even when the algorithms used in both of its stages are complete [57].

In [75], the preprocessing stage creates a representation of the configuration space that can be easily modified in real time to account for changes in the environment which facilitates real-time planning. The mapping from workspace cells to the graph is encoded so that when the environment changes, appropriate modifications to the graph is made, and plans can be generated by searching the modified graph. In another work, a path planner for robots operating in dynamically changing environments with both static and moving obstacles is developed [76]. It combines the lazy-evaluation mechanisms with a single-query technique as local planner in order to rapidly update the roadmap according to the dynamic changes.

As mentioned previously, an important feature of sampling-based planners is that they do not attempt to explicitly construct the boundaries of the configuration space obstacles. Instead, they check whether a given configuration of the robot is in collision with the obstacles or not. Efficient collision detection procedures ease the implementation of sampling-based planners and increase the range of their applicability.

Collision Checking in 2-D Workspace The 2D workspace allows for very fast collision checking techniques. Collision checking in 2D workspace in case of a multi-link robot is performed by precomputing a C-obstacle bitmap representing the obstacle in the 3D configuration space for each link [58, 59, 77]. Since each link is free to translate and rotate, C-obstacle bitmap constitutes a 3D space. Then planner checks each link against its C-space bitmap. Since the computation of any bitmaps needed for collision checking is performed only once prior to the learning phase, the collision checking is fast. If the link and the obstacles are modeled as collections of possibly overlapping convex polygons, the construction of a 2D bitmap can be done [78]. Each 2D bitmap may also be computed using the FFT-based method whose complexity depends only

on the size of the bitmap. Apparently this technique requires the construction of a 6D bitmap for 3D workspaces which renders this technique impractical for 3D workspaces.

Collision Checking in 3-D Workspace There are many existing libraries and techniques (e.g. GJK, SOLID, V-Clip, I-Collide and V-Collide) for collision detection and measurement. A fundamental and effective method represents objects by hierarchy of objects of simple shapes (e.g., spheres, parallelepipeds) and eventually reduces collision detection to collision checking/distance computation between two objects of basic shape (e.g., two convex polyhedra) [79, 80]. The principles are as follows:

1. Triangulate the boundary of each object.
2. Represent each object by a binary tree of spheres, such that:
   - The sphere at each node contains the spheres at each of the two children of it.
   - The tree is approximately balanced.
3. To check for collision or compute the distance between two objects, traverse their sphere trees concurrently. When leaf nodes are reached, check collision or compute distance between two triangles.

Collision detection is performed in the preprocessing or query phase of the path planning and discards the configurations with collisions.

3.1.2 Objective

The main purpose of the trajectory planning for visual servoing is to find a series of feature images that takes the initial image to the goal image. This is achieved

through occlusion detection and introducing visibility constraint by eliminating the milestones (configurations) with occlusion and the configurations with features out of FOV.

This work investigates path planning problem for robust visual servoing in PRM framework. In this proposal, visual servo path planning will be accessible with the existence of obstacles. Vision-based occlusion avoidance, obstacle avoidance and circumvention will be provided with the aid of visual information available from the eye-in-hand configuration.

The aim of this research is to design a trajectory-generator that generates a continuous and differentiable curve $\mathbf{s}^*(t) = [\tilde{p}^1_{(t)} \ \ldots \ \tilde{p}^n_{(t)}]$ between the initial features $\mathbf{s}^i = [\tilde{p}^1_i \ \ldots \ \tilde{p}^n_i]$ and the desired features $\mathbf{s}^* = [\tilde{p}^1_* \ \ldots \ \tilde{p}^n_*]$ such that $\mathbf{s}^*(0) = \mathbf{s}^i$ and $\mathbf{s}^*(t_f) = \mathbf{s}^*$ where t_0 and t_f denote initial and final times, respectively. First, a discrete sequence of K intermediate camera poses $\Gamma = \{(R_i, t_i) \mid i \in 1 \ldots K\}$ and the robot trajectory in the joint space $\Theta = \{\mathbf{q}_i \mid i \in 1 \ldots K\}$ are generated using a PRM planner. Then, the discrete object trajectory in the image $\Upsilon = \{\mathbf{s}_i \mid i \in 1 \ldots K\}$ is obtained from Γ. Finally, a continuous and differentiable geometric path in the image is generated to be tracked using an IBVS control law.

3.2 Scaled partial 3D reconstruction

Using the homography concept introduced in Section 2.1.3 of Chapter 2, the resulting image points $\mathbf{p}^j_i$ in the initial camera frame $\mathcal{F}_i$ of a point $\mathbf{M}_i$ on a plane Π, are related to the corresponding image points $\mathbf{p}^j_*$ in the desired camera frame $\mathcal{F}_*$, by a projective homography such that

$$\alpha^j_i \mathbf{p}^j_i = {}^i\mathbf{G}_* \mathbf{p}^j_* \tag{3.1}$$

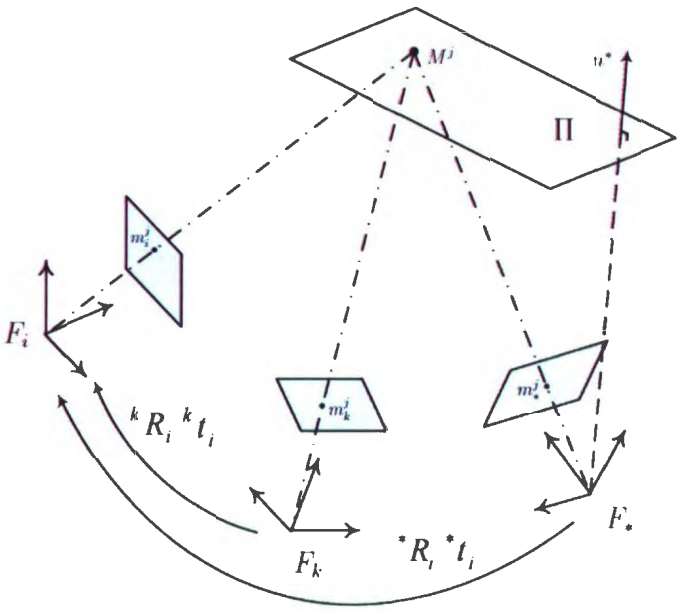


Figure 3.1: Geometry of the homography between views

where α_i^j is a positive scaling factor. Figure 3.1 illustrates the geometry of the constraint between different camera frames.

The homography between initial and current camera poses ${}^k\mathbf{H}_i$ can be expressed and written in terms of the known parameters as

$$ {}^k\mathbf{H}_i = {}^k\mathbf{R}_i + \frac{{}^k\mathbf{t}_i {}^i\mathbf{R}_* \mathbf{n}^{*T}}{d^* \det(\mathbf{H}^*)} \tag{3.2} $$

where plane Π is expressed with the normal $\mathbf{n}^i$ in the coordinate system of $\mathcal{F}_i$, and its distance d^i to the origin of $\mathcal{F}_i$. The rotation matrix ${}^k\mathbf{R}_i$ and the translation vector ${}^k\mathbf{t}_i$ denote the position and orientation of the initial camera pose with respect to the current one. It is important to notice that $r = d^i/d^* = \det(\mathbf{H})$ [30].

The distances d^i and d^* are unknown, but the ratio $\rho_k^j = \dfrac{Z_k^j}{d^*}$ can easily be estimated using the method developed in [16,81]. Taking note of $\mathbf{n}^i = {}^i\mathbf{R}_* \mathbf{n}^*$ [30], depth

ratio at frame $\mathcal{F}_k$ is obtained as

$$\rho_k^j = \frac{Z_k^j}{d^*} = \frac{1 + \mathbf{n}^{*Ti}\mathbf{R}_*^{Tk}\mathbf{R}_i^T({}^k\mathbf{R}_i{}^i\mathbf{t}_* + {}^k\mathbf{t}_i)}{\mathbf{n}^{*Ti}\mathbf{R}_*^{Tk}\mathbf{R}_i^T m_k^j} \tag{3.3}$$

Computing this ratio can be easily extended for targets not on Π i.e. nonplanar targets [16, 30]. These parameters are used in the design of the control scheme and the path planning scheme.

3.3 Target trajectory in image space

The homography matrix ${}^k\mathbf{G}_i$ of plane Π relating the current and desired images can be computed from the transformation matrix obtained from the current robot configuration q_k using forward kinematics.

The current camera position corresponding to robot configuration q_k with respect to the initial camera pose i.e. ${}^k\mathbf{T}_i = ({}^k\mathbf{R}_i, {}^k\mathbf{t}_i)$ is obtained from

$$\begin{cases} {}^k\mathbf{R}_i = {}^b\mathbf{R}_k^{Tb}\mathbf{R}_i^T \\ {}^k\mathbf{t}_i = {}^b\mathbf{R}_k^T({}^b\mathbf{t}_i - {}^b\mathbf{t}_k) \end{cases} \tag{3.4}$$

where ${}^b\mathbf{R}_i^T$ and ${}^b\mathbf{R}_k^T$ are given by forward kinematics using the known initial and current robot configurations, respectively.

Using equations (3.2) , (3.4) and (2.14), the projective homography is implemented as

$${}^k\mathbf{G}_i = \mathbf{C}({}^k\mathbf{R}_i + \frac{{}^k\mathbf{t}_i{}^i\mathbf{R}_*\mathbf{n}^{*T}}{d^*\det(\mathbf{H}^*)})\mathbf{C}^+ \tag{3.5}$$

Using equations (3.1) and (3.5), the image coordinates of the points $\mathbf{M}^j$ of the

target at configuration q_k are given by

$$\alpha_k^j \mathbf{p}_k^j = [\alpha_k^j u_k^j \ \alpha_k^j v_k^j \ \alpha_k^j] = {}^k\mathbf{G}_i p_i^j \tag{3.6}$$

At each robot configuration q_k, $\mathbf{p}_k^j$ is computed using equations (3.5) and (3.6); thus the target is partially reconstructed in the image plane. This useful feature is used to design the path planner.

3.4 Visual occlusion avoidance

In order to ensure occlusion in trajectory planning for visual servoing, a collision detection is performed in the image plane between the reconstructed visual target (see details in Section 3.3) and projected obstacles.

To project an obstacle onto the image plane in different frames, the geometry of the obstacle should be known. As the complexity of the geometry of the obstacle increases, more points on the obstacle need sampling and projecting such that the projection represents the object properly in the image plane. This task is not trivial and requires complex algorithms to calculate the necessary points according to the camera point of view.

To alleviate the problems associated with sampling, the *bounding volume* concept is borrowed from computational geometry. In computer graphics, bounding volumes are used in ray-intersection tests, and in many rendering algorithms, they are used for *viewing frustum* tests. If the ray or viewing frustum does not intersect the bounding volume, it cannot intersect the object contained in the volume. These intersection tests produce a list of objects that must be rendered. In collision detection, when two bounding volumes do not intersect, then the contained objects cannot collide, either.

Testing against a bounding volume is typically much faster than testing against the object itself, because of the bounding volume's simpler geometry. This is because an object is typically composed of polygons or data structures that are reduced to polygonal approximations. In either case, it is computationally wasteful to test each polygon for collision detection if the objects are not colliding.

To obtain bounding volumes of complex objects, a common way is to break the objects down using bounding volume hierarchies e.g. OBB-trees [80, 82]. The basic idea behind this is to organize a object in a tree-like structure where the root comprises the whole object and each leaf contains a smaller subpart. There are various convex bounding volumes, among which a bounding box appears beneficial to this work. A bounding box is a cuboid containing the object. In dynamical simulation, bounding boxes are preferred to other shapes of bounding volume such as bounding spheres or cylinders for objects that are roughly cuboid in shape when the intersection test needs to be fairly accurate. The benefit is obvious, for example, for objects that rest upon other, such as an object resting on the ground; a bounding sphere would show the object as possibly intersecting with the ground, which then would need to be rejected by a more expensive test of the actual model of the object; a bounding box immediately shows the object as not intersecting with the ground, saving the more expensive test.

In many applications, the bounding box is aligned with the axes of the co-ordinate system, and it is known as an *axis-aligned bounding box* (AABB). To distinguish the general case from an AABB, an arbitrary bounding box is called an *oriented bounding box* (OBB). AABBs are much simpler to test for intersection than OBBs, but have the disadvantage that when the model is rotated they cannot be simply rotated with it, but need to be recomputed. In an ideal case, the OBB would be oriented such that it encloses an object as tightly as possible. In other words, the bounding box is the

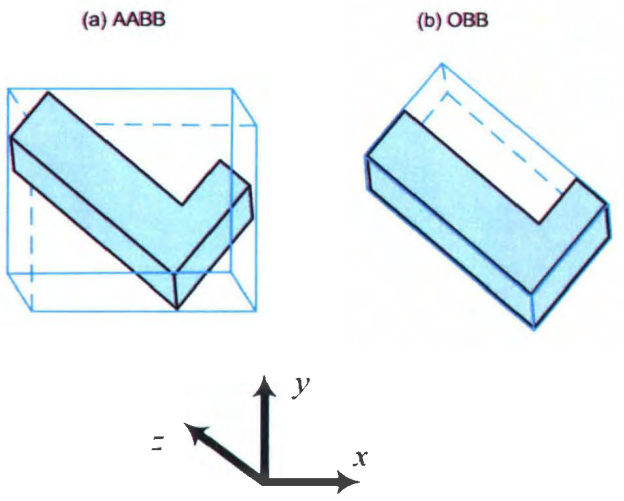


Figure 3.2: Difference between AABB and OBB

smallest possible bounding box of arbitrary orientation that can enclose the geometry in question. When compared with AABBs, OBBs generally allow geometries to be bounded more tightly with a fewer number of boxes. The difference between AABB and OBB is illustrated in figure 3.2.

Let $\mathbf{M}_o^j$ be a vertex of the formed OBB of the obstacle expressed in the coordinate system of the obstacle $\mathcal{F}_o$. Given the transformation matrix from the base frame $\mathcal{F}_b$ to the obstacle frame $\mathcal{F}_o$, projection of vertex $\mathbf{M}_o^j$ can be easily achieved through reformulation of the transformation matrix between the obstacle frame $\mathcal{F}_o$ and $\mathcal{F}_k$, frame attached to the camera at the current pose. Let ${}^b\mathbf{T}_o = ({}^b\mathbf{R}_o, {}^b\mathbf{t}_o)$ be the transformation matrix between $\mathcal{F}_b$ and $\mathcal{F}_o$. The mechanics of the problem is depicted in figure 3.3. The transformation matrix ${}^k\mathbf{T}_o$ with ${}^k\mathbf{R}_o$ and ${}^k\mathbf{t}_o$ representing the rotation and translation components between $\mathcal{F}_o$ and $\mathcal{F}_k$ is given by

$$\begin{cases} {}^k\mathbf{R}_o = {}^b\mathbf{R}_k{}^T {}^b\mathbf{R}_o \\ {}^k\mathbf{t}_o = -{}^b\mathbf{R}_k{}^T({}^b\mathbf{t}_k - {}^b\mathbf{t}_o) \end{cases} \tag{3.7}$$

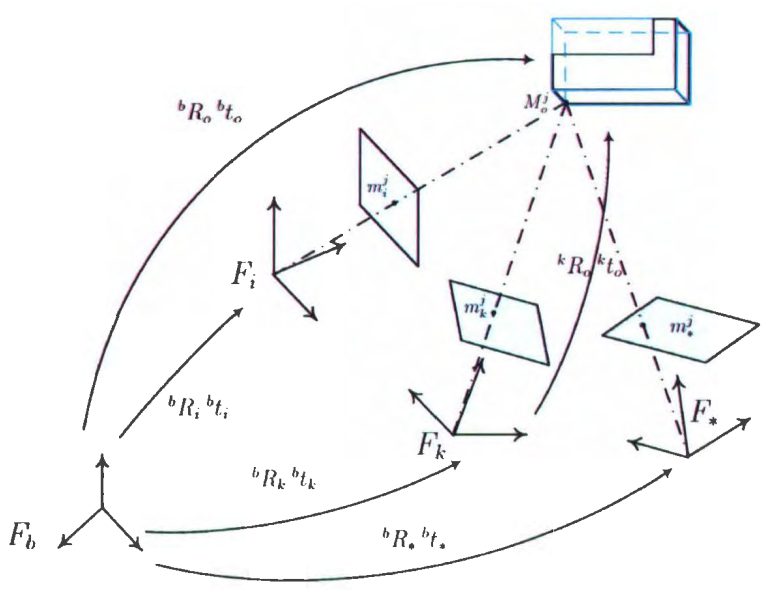


Figure 3.3: Computing the obstacle projections in different camera frames

governing the relation

$$\mathbf{M}_k^j = [X_k^j\ Y_k^j\ Z_k^j]^T = [{}^k\mathbf{R}_o\ {}^k\mathbf{t}_o]\mathbf{M}_o^j \tag{3.8}$$

where $\mathbf{M}_k^j$ are the coordinates of $\mathbf{M}_o^j$ expressed in $\mathcal{F}_k$.

To perform occlusion avoidance, the vertices of the obstacle OBB, $\mathbf{M}_o^j$ with $j \in \{1..8\}$ are projected on the image plane at frame $\mathcal{F}_k$. The projection is obtained using the classical assumption that the camera performs a perfect perspective transformation with respect to the camera optical center (pinhole model). Using kinematic relations (3.7) and (3.8), $\mathbf{p}_k^j$ is easily obtained from

$$r_k^j\mathbf{p}_k^j = [r_k^j u_k^j\ r_k^j v_k^j\ r_k^j]^T = \mathbf{C}[{}^k\mathbf{R}_o\ {}^k\mathbf{t}_o]\mathbf{M}_o^j \tag{3.9}$$

by dividing $r_k^j\mathbf{p}_k^j$ by the last component r_k^j. The target is projected on the image plane through scaled 3D reconstruction explained in Section 3.3. Since the projections

of the target and the obstacle are set of points on the image plane, the problem narrows down to collision checking of two convex hulls formed by these two sets. In mathematics, the convex hull or convex envelope for a compact[1] set of points C is the minimal convex set[2] containing C. In other words, any subset C of the vector space is contained within a smallest convex set (called the convex hull of C), namely the intersection of all convex sets containing C. In computational geometry, it is common to use the term *convex hull* for the boundary of the minimal convex set containing a given non-empty finite set of points in the plane. The implementation of the Quickhull Algorithm[3] in MATLAB is used to form the convex hulls of the set of projection points [82,83]. Let C_j with $j \in \{0,1\}$ be the convex sets, respectively of the target and obstacle projection points on the image plane, with vertices $(V_i^j)_{i=0}^{N_j}$ [1] ordered counterclockwise. It is important to note that $N_1 = 8\times$(number of obstacles) and $N_0 = n$. In the sequel, two fast tests for nonintersection of convex hulls are presented which are used to ensure occlusion avoidance.

3.4.1 Method 1: Separating axis theorem

For objects lying in a 2-dimensional space, if there exists a line for which the intervals of projection of the two objects onto that line do not intersect, then the objects do not intersect. Such a line is called a *separating line* or, more commonly, a *separating axis*. The translation of a separating line is also a separating line, so it is sufficient to consider lines that contain the origin. Given a line with unit-length direction $\mathbf{B}$ passing through the origin, the projection of a convex set C onto the line is the

[1] A set is compact if it is closed and bounded. To illustrate in one dimension, the interval $[0,1]$ is closed and bounded, so it is compact. The interval $[0,1)$ is not compact since it is is bounded, but not closed. The interval $[0,\infty)$ is closed, but not bounded, so it is not compact.

[2] A set is convex if given any two points P and Q in the set, the line segment $(1-t)P + tQ$ for $t \in [0,1]$ is also in the set.

[3] Qhull, http://www.qhull.org

interval

$$I = [\lambda_{min}(\mathbf{B}), \lambda_{max}(\mathbf{B})] = [\min\{\mathbf{B} \cdot V : V \in C\} \ , \ \max\{\mathbf{B} \cdot V : V \in C\}] \qquad (3.10)$$

Two compact convex sets C^0 and C^1 are separated if there exists a direction $\mathbf{B}$ such that the projection intervals I_0 and I_1 do not intersect. Specifically, they do not intersect when

$$\lambda^0_{min}(\mathbf{B}) > \lambda^1_{max}(\mathbf{B}) \ \text{ or } \ \lambda^0_{max}(\mathbf{B}) < \lambda^1_{min}(\mathbf{B}) \qquad (3.11)$$

The superscript corresponds to the index of the convex set. The comparison results are invariant to changes in length of the vector since

$$\lambda_{min}(t\mathbf{B}) = t\lambda_{min}(\mathbf{B}) \ \text{ and } \ \lambda_{max}(t\mathbf{B}) = t\lambda_{max}(\mathbf{B}) \text{ for } t \in \mathbb{R} \qquad (3.12)$$

and similarly the boolean value of the pair of comparisons is also invariant when $\mathbf{B}$ is replaced by the opposite direction $-\mathbf{B}$ ($t = -1$). When $\mathbf{B}$ is not unit length, the intervals obtained for the separating axis tests are not the projections of the object onto the line, rather they are constant scaled versions of the projection intervals. The *Normal direction vector* denotes the perpendicular direction to the separating axis, a direction that is not necessarily unit length. Given an edge (u, v), an outward pointing normal direction is obtained from $(u, v)^{\perp} = (v, -u)$. Figure 3.4 shows two nonintersecting polygons that are separated along a normal direction vector. The corresponding edge to the normal direction vector is annotated. It is obvious that the direction of the annotated edge is the separating axis.

For a pair of convex polygons in 2D, only a finite set of separating axis needs to be considered for separation tests. That set includes the normal direction vectors to the edges of both polygons. Since the number of vertices is limited to a definite small

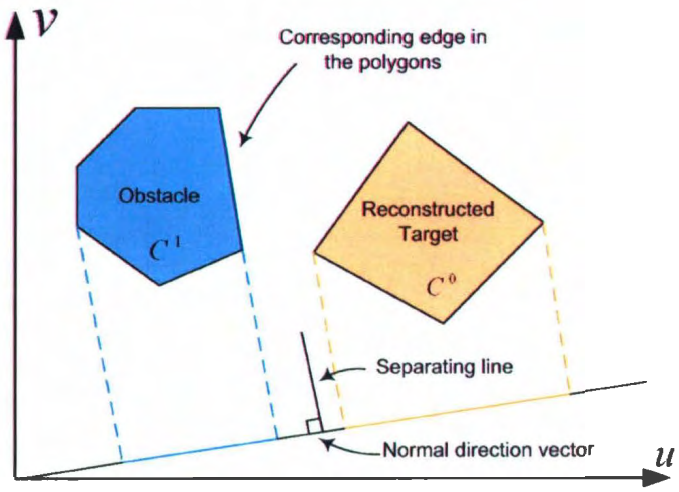


Figure 3.4: Nonintersecting convex hulls: obstacle projection and reconstructed target

number ($\leq 8m + n$ for m obstacles), the direct implementation will be employed in which for a separation test for direction $\mathbf{B}$ computes the extreme values of the projection and compares them. That is, compute

$$\lambda^j_{min}(\mathbf{B}) = \min_{0 \leq i < N_j - 1}\{\mathbf{B} \cdot V^j_i\} \tag{3.13}$$

$$\lambda^j_{max}(\mathbf{B}) = \max_{0 \leq i < N_j - 1}\{\mathbf{B} \cdot V^j_i\} \tag{3.14}$$

and test the inequalities in Equation (3.11). If there exists a direction for which the intervals of projection of the target and the obstacle onto that line do not overlap, then it is simply concluded that there is no occlusion at configuration q_k.

3.4.2 Method 2: Geometric verification

If it is of interest to check whether target features are inside the obstacle projection convex hull regardless of the geometry of the target, geometric verification is used. It is important to note that the previous method tests whether the convex hull of the

target is overlapping the obstacle convex hull in image plane. But this method checks only for the intersection of the target features with the obstacle convex hull in the image plane regardless of the target geometry (convex hull).

A simple concept from geometry is used to perform the intersection detection. Given any three points on the plane (x_0, y_0), (x_1, y_1), and (x_2, y_2), the area of the triangle determined by them is given by

$$Area = \frac{1}{2}\begin{vmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} \tag{3.15}$$

and is positive if the three points are taken in a counter-clockwise orientation, and negative otherwise. Figure 3.5 illustrates the concept for a target feature, V_i^0. The convex hull formed by the obstacle projection, C^1 is the blue area. For a target feature V_i^0 to be inside the obstacle convex hull C^1 in image plane, the computed area should be positive for all the triangles formed by any two successive vertices of C^1 and V_i^0 as one traces around in a counter-clockwise direction from V_0^1 to $V_{N_1-1}^1$ and back to V_0^1 of the obstacle convex set. To check whether C^0 is in collision with C^1, all vertices of C^0 ,$\{V_i^0 \mid i = 0 \cdots N_0\}$, are tested for collision using the same procedure.

3.5 Probabilistic Roadmaps

PRM is a powerful and versatile sampling-based planner which can be used to solve high-dimensional problems. PRM divides the planning task into two phases: the preprocessing or learning phase, during which a roadmap is constructed in $\mathcal{Q}$; and the query phase, during which user-defined query configurations are connected with the roadmap precomputed in the previous phase. In this section, PRM is utilized as

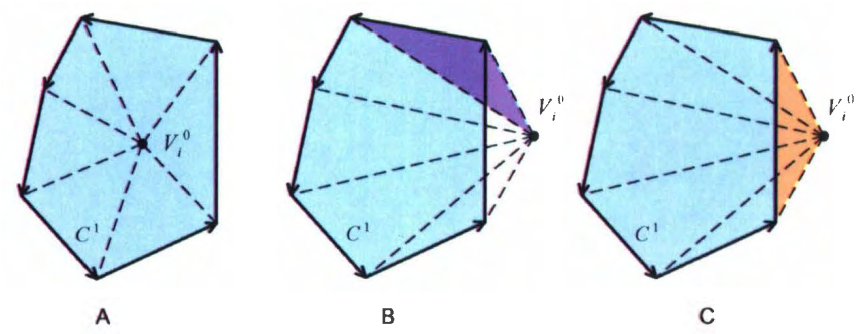


Figure 3.5: Geometric verification for nonintersection of convex sets (A)Target feature resides in C^1: All areas> 0 (B,C)Target feature is outside C^1: All areas> 0 (e.g. purple region in B) except one area (i.e. Orange region in C)

the path planning scheme for visual servoing. PRM will find a path from the initial configuration to the desired configuration so that the target is always in the FOV of the vision system mounted on the robot arm. In the following, a brief introduction to the two phases of PRM path planning is provided.

3.5.1 Roadmap construction

The PRM algorithm first constructs a roadmap in a probabilistic way for a given configuration space. A set of collision-free robot configurations V is chosen by a method over $\mathcal{Q}$. The generation of these configurations is basically performed randomly from a uniform distribution. The roadmap is represented by an undirected graph $G = (V, E)$. The edges in E correspond to paths between nodes in V; an edge (q', q'') corresponds to a collision-free path connecting configurations q' and q''. These paths, which are referred to as local paths, are computed by a local planner. In its simplest form, the local planner connects two configurations by the straight line in $\mathcal{Q}$, if such a line exists. Since the construction of the roadmap is computationally expensive, roadmap is build and stored in the preprocessing phase to be used in the query phase.

Initially, the graph $G = (V, E)$ is empty; then, repeatedly, a configuration is sampled from $\mathcal{Q}$. The sampling is performed according to a uniform random distribution on $\mathcal{Q}$. The nodes of the roadmap constitute a uniform random sampling of $\mathcal{Q}$. To obtain a configuration, each rotational degree of freedom of the robot is sampled from the interval of allowed values of the corresponding degree of freedom (i.e. joint limits of the manipulator) using the uniform probability distribution over this interval. If the configuration is collision-free, it is added to the roadmap. The process is repeated until N collision-free configurations have been sampled. For every node $q \in V$, according to the metric distance function explained in Section 3.5.2, a set NN_q of k closest neighbors to the configuration q is chosen from V. In order to determine the set NN_q of nearest neighbors to a configuration q, many data structures have been proposed in the field of computational geometry that deal with the problem of efficiently calculating the closest neighbors to a point in a d-dimensional space. A capable and efficient method is the kd-tree data structure [82] (See Appendix A). The local planner is called to connect q to each node $q' \in NN_q$. Whenever the local planner succeeds in connecting q to q', the edge (q, q') is added to the roadmap. The algorithm to construct the roadmap is outlined in Algorithm 1.

Algorithm 1 checks each configuration for collision rather than applying lazy collision-checking strategy proposed in [74,84] where it postpones collision tests along connections in the roadmap until they are absolutely needed. There are several reasons to postpone collision tests. Checking collision consumes a lot of computations and most connections is not included in the final path and furthermore the collision test for a connection is the most expensive when there is no collision; and finally the probability that a short connection is collision-free is large [84]. However visibility and occlusion avoidance constraints are computationally expensive, thus some of the computational burden is carried to the preprocessing phase by performing the

Algorithm 1 CONSTRUCTROADMAP(N, k, D)

Require: number of nodes to put in the roadmap: N,
number of closest neighbors to examine for each configuration: k,
maximum search range: D
Ensure: A roadmap $G = (V, E)$.

```
1: V ← ∅
2: E ← ∅
3: while |V| < N do
4:     repeat
5:         q ← a random configuration from Q
6:     until q is collision-free
7:     V ← V ∪ {q}
8: end while
9: for all q ∈ V do
10:    NN_q ← k nearest neighbors of q in range D queried from kd-tree
11:    for all q' ∈ NN_q do
12:        E ← E ∪ {(q, q')}
13:    end for
14: end for
```

collision checking in the preprocessing phase, despite the obvious advantages of lazy collision-checking strategy.

In the query phase, the roadmap is used to solve user-specified queries. Given an initial configuration q_{init} and a goal configuration q_{goal}, the method first tries to calculate the k closest neighbors for the query points; the local planner then attempts to connect q_{init} and q_{goal} to them. Assume that q_{init} and q_{goal} are connected to only two nodes q' and q'', respectively, in V. As soon as they are connected to the same component, A* algorithm is run to search the graph G for a sequence of edges in E connecting q' and q''. Finally, the planner transforms this sequence into a feasible path for the robot by recomputing the corresponding local paths and concatenating them. Local paths can be stored in the roadmap but this would increase the storage requirements of the roadmap. If the local planner is very fast, local paths can be calculated in the query phase without the need for storing them in

the preprocessing phase. The roadmap can be further augmented to capture the connectivity of $\mathcal{Q}$. Although the preprocessing phase is usually performed before any path-planning query, the two phases can also be interwoven [57]. It is reasonable to spend a considerable amount of time in the learning phase if the roadmap will be used to solve many queries. More application specific details will be provided in Section 3.6.

Sampling strategy: uniform sampling

Several node-sampling strategies have been developed over the years for PRM. For many path-planning problems, a surprisingly large number of general sampling schemes will provide reasonable results [57].

The uniform random sampling used in early work in PRM is the easiest sampling scheme to implement [58]. As a random sampling method, it has the advantage that, in theory, a malicious opponent cannot defeat the planner by constructing carefully crafted inputs. It has the disadvantage, however, that, in difficult planning examples, the running time of PRM might vary across different runs. Nevertheless, random sampling works well in many practical cases involving robots with a large number of degrees of freedom.

For some difficult problems, uniform random sampling shows poor performance and proves inappropriate as in the case of narrow passage problem [57]; thus more robust and efficient sampling strategy has to adopted [85, 86].

In this work, the choices for the sampling and connection strategies of PRM are reduced to a minimum to concentrate on the main purpose of the research. The emphasis here is to describe a planner that is easy to implement and works well for visual path planning. Further implementation of more advanced sampling techniques and sophisticated collision detection methods is easily achievable.

3.5.2 Configuration space and its metric

The configuration of a robot system is a complete specification of the position of every point of that robot system. The *configuration space*, or *C-space*, of the robot system is the space of all possible configurations of the system. Thus a configuration is simply a point in this abstract configuration space. $\mathcal{Q}$ and q denote, respectively, C-space and a configuration in C-space. The number of degrees of freedom of a robot system is the dimension of the C-space, or the minimum number of parameters needed to specify a configuration.

A rigid robot manipulator with five joints has five degrees of freedom. Therefore its C-space is fully defined by five parameters $q = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$. Each joint angle θ_i corresponds to a point on the unit circle S^1, and thus the C-space is $S^1 \times S^1 \times S^1 \times S^1 \times S^1 = \mathbb{T}^5$ which is a five-dimensional torus. It is common to picture a torus as its surface since a $\mathbb{T}^5$ torus has a natural embedding in $\mathbb{R}^6$ as a circle S^1 has a natural embedding in $\mathbb{R}^2$. By cutting this five-dimensional torus along the $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = 0$, $\theta_4 = 0$ and $\theta_5 = 0$ curves, one can flatten the torus onto the hyperplane in $\mathbb{R}^6$. The points on S^1 are identified by points in the interval $[0, 2\pi) \subset \mathbb{R}$ using this hyperplanar representation. Although this representation covers all points in S^1, the interval $[0, 2\pi)$, being a subset of the real line, does not naturally wrap around like S^1, so there is a discontinuity in the representation since S^1 is topologically different from any interval of $\mathbb{R}$.

A metric has to be defined for the C-space manifold. The workspace region swept by the robot can be defined as a measure of metric. Intuitively, minimizing the swept volume will minimize the chance of collision with the obstacles. An exact computation of swept areas or volumes is disreputably difficult, which is why heuristic metrics generally attempt to approximate the swept-volume metric [57, 62, 87].

The technique proposed in [77] is employed and instead of the expensive theoretical calculation of the swept-region, an approximate method can be constructed as follows. Since $\mathbb{T}^5$ can be embedded in Euclidean $\mathbb{R}^6$, the robot's configurations q_n and q_m can be mapped to points in a Euclidean space and the Euclidean distance $D_{\mathcal{Q}}(q_n, q_m)$ in C-space between them can be used,

$$D_{\mathcal{Q}}(q_n, q_m) = \left(\sum_{i=1}^{5} w_i |q_n^i - q_m^i|^2 \right)^{1/2} \tag{3.16}$$

where w is a weight vector that gives higher weights to the joints closer to the robot base since they have more effect on the motion range of the robot end-effector [59]. Considering the swept-volume metric, it is noteworthy that the embedding does not take into account obstacles. So even when two configurations are close to one another, connecting them may be impossible due to obstacles.

3.5.3 Local planner

In order to find a collision-free path between nodes in V, a local planner is used that tests the path between configurations for collision and other constraints and associate the path with corresponding edge in E, if appropriate.

Let Δ be the local planner that takes two inputs q' and q'' and returns either a collision-free path from q' to q'' or NIL, if it cannot find such a path.

The Local planner has a significant role in preprocessing and query phase. While constructing the roadmap in the preprocessing phase, the local planner tries to connect two neighbor nodes in V with a path and checks it for collision and adds the edge between them to E. The choice of the local planner also affects the query phase. It is important to be able to connect any given q_{init} and q_{goal} configurations to the roadmap or to detect very quickly that no such connection is possible. This requires

that the roadmap be dense enough in order to easily connect q_{init} and q_{goal} to it. There is a tradeoff between the time spent in each individual call of the planner and the number of calls.

Since Δ is a deterministic local planner, it will always return the same path between two configurations and the roadmap does not have to store the local path between two configurations in the corresponding edge. The path can be recomputed if needed in the query phase. On the other hand, if a nondeterministic local planner was used, the roadmap would have to store the local path computed by Δ with each edge which then would have increased the storage requirements of the roadmap.

A simple and popular planner connects any two given configurations by a straight-line segment in $\mathcal{Q}_{free}$ and checks this line segment for collision. Kavraki et al. [59,77] introduce a fast and efficient local planner for articulated robots which can be easily implemented but it is beneficial to use a workspace planner in this work for two reasons:

Visibility constraint In order to take the visibility constraint into account during path planning, straight lines in the workspace are required. If the target is visible on two nodes of G, q' and q'', then the target will be within FOV on the straight line connecting q' to q''; i.e. to ensure that the target remains in FOV, a path between q' and q'' in workspace is required.

Differentiable trajectory Visual servoing control scheme requires the first derivative of the path to be tracked. To generate a C^2 path in image space, a 3D path should be designed such that the control points on the path are equidistant (i.e. the distance between two discrete intermediate camera poses Γ'_k and Γ'_{k+1} should be constant) which requires the capability to measure distances in workspace. Section 3.7 introduces the generation of $\mathcal{C}^2$ trajectory in detail.

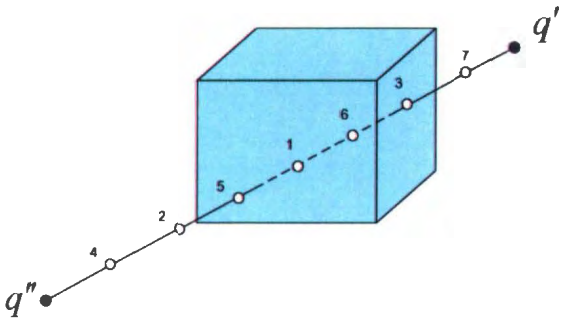


Figure 3.6: Subdivision collision checking in $\mathcal{W}$

The designed local planner Δ is a deterministic and symmetric workspace planner. It is noteworthy that the planning in the workspace requires more time than configuration space planning and that PRM construction is performed in C-space, however PRM local planning is carried out in workspace.

Given any two configurations q' and q'', local planner Δ will connect them by a straight-line segment in workspace $\mathcal{W}$ using the metric introduced in next subsection. This line segment is a discretized line constructed with δ configurations $\{q_1 \cdots q_\delta\}$ where $q' = q_1$ and $q'' = q_\delta$. The subdivision collision-checking algorithm is then used to test the line segment for collision [57]. Subdivision collision checking cuts down the length of the local path. In subdivision collision checking, the middle point q_m of the discretized line in $\mathcal{W}$ between q' and q'' is first checked for collision. Then the algorithm recurses on the discretized lines between (q', q_m) and (q_m, q''). The recursion halts when a collision is found. If none of the intermediate configurations yields collision, the path is considered collision-free [57]. Figure 3.6 depicts the subdivision collision checking algorithm for a sample path in $\mathcal{W}$.

The number of discretizations over the straight line between q' and q'' is determined by a parameter, δ, in discretization algorithm explained in the next subsection. In general, the value of δ needs to be large enough to guarantee that all collisions

are found. Although δ is assumed constant for simplicity, it is also possible to use an adaptive subdivision collision-checking algorithm that dynamically adjusts δ [74]. Furthermore, the method proposed in [74] always finds a collision when a collision exists, whereas the above discretization technique may miss a collision if δ is small. This value is dependent on the size of the obstacle in workspace. More precisely, the distance between any two consecutive configurations in $\{q_1 \cdots q_k\}$ should be less than the size of the obstacle in every dimension.

3.5.4 Workspace distance metric

An articulated robot arm moves in a three-dimensional Euclidean space $\mathbb{R}^3$ which is referred to as the workspace $\mathcal{W}$. The different physical locations of the end-effector as a rigid body lie in a non-Euclidean 6-manifold due to *topological nontriviality*. The Euclidean space is *simply-connected* by virtue of the *shrinkability* property. The robot workspace is, on the other hand, *multiply-connected* because of the existence of non-shrinkable loops [88]. There are three dimensions (degrees of freedom) in the position of the center of gravity and three more in the rotational orientation of the body. Thus workspace metric cannot be expressed using Euclidean geometry.

It is known that in $SE(3)$ there is no Ad-invariant Riemannian metric, which implies that there is no natural way of transporting vector fields between points of $SE(3)$ and that there is no natural concept of distance on $SE(3)$ [89]. The two most common approaches to tackle this obstruction are

- Ad-invariant pseudo-Riemannian structure
- Double geodesic.

Figure 3.7 illustrates the two possible metrics for $SE(3)$. Either choice has advantages and disadvantages, according to the task in mind. In the left-hand side

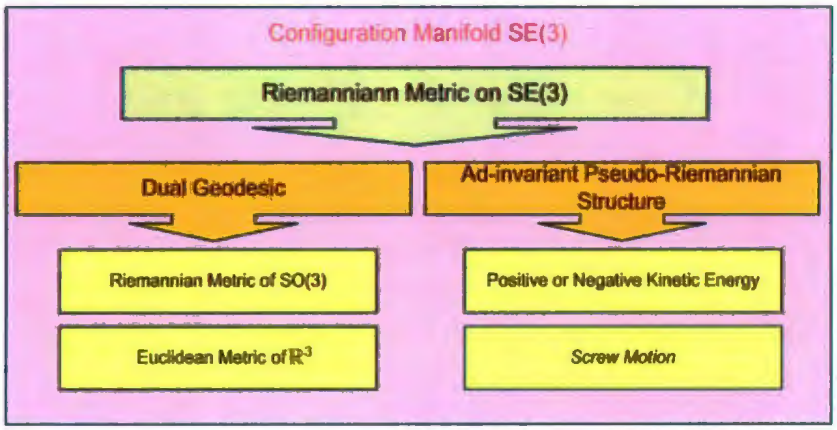


Figure 3.7: Metric structures for $SE(3)$: Dual geodesics and Ad-invariant pseudo-Riemannian structure

case, the group structure of $SE(3)$ is considered the Cartesian product of two distinct groups (rotations and translations) $SO(3) \times \mathbb{R}^3$. However the right-hand case consists in choosing an inner product which is non-degenerate but can assume both negative and positive values. This corresponds to having curves with both negative and positive energy and gives as geodesics the so-called screw motions.

The local planner in this research is based on the dual geodesic approach where the group structure of $SE(3)$ is considered separately by bi-invariant metric of $SO(3)$ and the Euclidean metric of $\mathbb{R}^3$. Every configuration q_k in V is mapped to ${}^b\mathbf{T}_k = ({}^b\mathbf{R}_k, {}^b\mathbf{t}_k)$ using forward kinematics. Therefore the workspace distance between any two configuration can be obtained using a weighted metric [90] given by

$$D_{\mathcal{W}}(q_n, q_m) = w_t||\mathbf{t}_n - \mathbf{t}_m|| + w_r f(\mathbf{R}_n, \mathbf{R}_m) \tag{3.17}$$

where the translation component $||\mathbf{t}_n - \mathbf{t}_m||$ is obtained using a standard Euclidean norm, and the positive scalar function $f(\mathbf{R}_n, \mathbf{R}_m)$ gives an approximate measure of the distance between the rotations $\mathbf{R}_n, \mathbf{R}_m \in SO(3)$. Due to the incapabilities of Euler

angles to represent rotational components, unit quaternions are used to represent rotations. The rotation distance is scaled relative to the translation distance via the weights w_t and w_r. Determining the proper weight values is a difficulty of this method. It is shown that the relative importance of the rotation component decreases as the planning queries become harder [87].

There are multiple sets of Euler angles which can yield the same rotation causing ambiguity due to the interdependence of the rotations. In addition, when the axes of two of the three Euler angles needed to compensate for rotations in three dimensional space are driven to the same direction, a degree of freedom will be lost. This problem is called *gimbal lock*. More importantly, Euler angles have serious problems in rotation presentation in the context of path planning, namely, in interpolation and distance metrics. The measure of distance between Euler angles does not correctly handle multiple representations of the same rotation. Two sets of Euler angles with relatively large differences in individual angle values may actually map to very similar or identical rotations in $SO(3)$. The implication is a relatively large swept-volume due to the wrong interpolated values. These problems along with the difficulty in defining metrics generally makes Euler angles a poor choice for representing the rotation component of $SE(3)$ in path planning applications.

Quaternions are used to parameterize rotations in three dimensions, inspired by axis-angle parameterization of 3D rotations. Any arbitrary orientation in three dimensions could be achieved by a single rotation α about an axis $\nu = (\nu_x, \nu_y, \nu_z)$. The corresponding unit quaternion is given by

$$Q = (w, x, y, z) = \left(\cos(\frac{\alpha}{2}), \nu_x \cos(\frac{\alpha}{2}), \nu_y \cos(\frac{\alpha}{2}), \nu_z \cos(\frac{\alpha}{2})\right) \tag{3.18}$$

with the property that $||Q|| = 1$.

In the context of path planning, unit quaternions are an excellent choice for representing rotations since it is relatively easy to define methods for interpolation, and distance measure between quaternion rotations. Unlike Euler angles, it is possible to derive a geodesic metric for unit quaternion representations of $SO(3)$. The *great circle arc* on the 4D unit sphere between two unit quaternions defines a geodesic path for interpolating two rotations [90]. Given two unit quaternions $Q_n = (w_n, x_n, y_n, z_n)$ and $Q_m = (w_m, x_m, y_m, z_m)$, the weighted rotation distance component of (3.17) is given by

$$f(\mathbf{R}_n, \mathbf{R}_m) = 1 - ||Q_n \cdot Q_m|| \tag{3.19}$$

where $Q_n \cdot Q_m = w_n w_m + x_n x_m + y_n y_m + z_n z_m$ is the inner product of two quaternions. The angle formed by this pair of quaternions is related to the inner product by its cosine

$$\varphi = \arccos(Q_n \cdot Q_m) \tag{3.20}$$

The ability to smoothly interpolate between two rotations Q_n and Q_m in $SO(3)$ along the great-circle arc is one of the great advantages of using quaternions. The geodesic for a $4D$ unit sphere is the great-circle arc. Points along this curve are the smoothly-varying intermediate rotations in $SO(3)$ that connect the two rotations Q_n and Q_m. These intermediate rotations can easily be obtained by linearly interpolating two unit quaternions as points in $\mathbb{R}^4$ and projecting the generated quaternions onto the $4D$ unit sphere. *Spherical linear interpolation* is used to perform interpolation between two unit quaternions [90], illustrated in Algorithm 2. First the inner product is computed for two unit quaternions; if the rotations are very close ($f(\mathbf{R}_n, \mathbf{R}_m) < \epsilon$), then linear interpolation is performed. Otherwise, spherical linear interpolation is used to compute evenly-distributed intermediate points along the geodesic arc. Finally, the intermediate quaternions are normalized to prevent numerical drift resulting

Algorithm 2 PATHDISCRETIZE(q', q'', δ)

Require: q': start configuration
q'': end configuration
δ: the number of discretizations
Ensure: Discretized straight path $\mathfrak{D}$ between q' and q'' in $\mathcal{W}$

1: $(Q', \mathbf{t}') \leftarrow$ FORWARDKINEMATICS(q')
2: $(Q'', \mathbf{t}'') \leftarrow$ FORWARDKINEMATICS(q'')
3: $\eta = Q' \cdot Q''$
4: **if** $\eta < 0$ **then**
5: $\quad Q' = -Q'$, $\eta = -\eta$
6: **end if**
7: **for** $i = 0$ to 1 with increments of $\frac{1}{\delta}$ **do**
8: $\quad$ **if** $|1 - \eta| < \epsilon$ **then**
9: $\quad\quad r = 1 - i$, $s = i$
10: $\quad$ **else**
11: $\quad\quad \varphi = \arccos(\eta)$, $\tau = \frac{1}{\sin(\varphi)}$
12: $\quad\quad r = \sin((1 - i) * \varphi) * \tau$
13: $\quad\quad s = \sin(i * \varphi) * \tau$
14: $\quad$ **end if**
15: $\quad Q = rQ' + sQ''$ {rotational component}
16: $\quad Q = \frac{Q}{\|Q\|}$
17: $\quad \mathbf{t} = (1 - i)\mathbf{t}' + i\mathbf{t}''$ {translational component}
18: $\quad \mathfrak{D} \leftarrow \mathfrak{D} \cup (Q, \mathbf{t})$
19: **end for**

from floating-point approximation errors. A step size parameter, δ, determines the density of the generated intermediate rotations in spherical linear interpolation algorithm. Formulations of angle-axis and unit quaternions [91] are used to convert rotation matrix to and from unit quaternions.

3.6 Queries and postprocessing queries

The main purpose of the trajectory planning for visual servoing is to find a series of feature images that takes the initial image to the goal image without any occlusion by obstacles or collision of the robot with obstacles.

Since the initial configuration of the robot and thus q_i (i.e. the configuration corresponding to $\mathcal{F}_i$) is known, q_* (i.e. the configuration corresponding to desired image) can be computed using the obtained camera displacement information in Chapter 2, Section 2.1.3. PRM is employed then to find a path from q_i to q_* such that it is collision-free, occlusion-free and such that the target remains in FOV.

During the query phase, path between arbitrary input configurations $q_{init} = q_i$ and $q_{goal} = q_*$ is searched using the roadmap constructed in the preprocessing phase. Algorithm 3 illustrates this process. A fast and inexpensive algorithm is required to connect q_{init} and q_{goal} to the roadmap. The same strategy employed in Algorithm 1 is used to connect q_{init} to the roadmap. k nearest nodes in the roadmap in order of increasing distance from q_{init}, is obtained using kd-tree (See Appendix A, Algorithms 6 and 7) and local planner Δ tries to connect q_{init} to each of them until one connection succeeds. The same procedure is used to connect q_{goal} to the roadmap. If the connection of q_{init} and q_{goal} to the roadmap is successful, the shortest path between q_{init} and q_{goal} is found on the roadmap using the constrained A*, detailed in Algorithm 4.

The number of closest neighbors (k) and maximum search range (D) considered in Algorithm 1 can be different from the one in Algorithm 3. Subroutine FINDPATH in this algorithm requires more explanation. A concise illustration of the details of this function is provided in Algorithm 4.

3.6.1 A* graph search

A* is a best-first, graph search algorithm that finds the least-cost path from a given initial node to goal node in a graph. The input for A* is a graph G, the initial and goal nodes. The nodes correspond to the robot configurations and edges correspond

Algorithm 3 QUERYROADMAP($q_{init}, q_{goal}, k, D, G$)

Require: q_{init}: the initial configuration
q_{goal}: the goal configuration
k: the number of closest neighbors
D: maximum search range
$G = (V, E)$: the constructed roadmap
Ensure: A path from q_{init} to q_{goal} or failure.

```
1: NN_{q_init} ← k closest neighbors of q_init from V in range D queried from kd-tree
2: NN_{q_goal} ← k closest neighbors of q_goal from V in range D queried from kd-tree.
3: V ← {q_init, q_goal} ∪ V.
4: for all q ∈ {q_init, q_goal} do
5:     q' ← the closest neighbor of q in NN_q.
6:     repeat
7:         if PRMCONSTRAINT(q, q') then
8:             E ← {(q, q')} ∪ E
9:         else
10:            q' ← next closest neighbor of q in NN_q
11:        end if
12:    until a connection was successful or the set NN_q is empty
13: end for
14: P ←FINDPATH(q_init, q_goal, G)
15: if P is not empty then
16:    return P
17: else
18:    return ∅
19: end if
```

to adjacent nodes and have values corresponding to the cost required to traverse between the adjacent nodes. Here A* is used in the query phase of the path planning to return the shortest path.

Algorithm 4 explains the details of the subroutine FINDPATH (i.e. A* search algorithm used in Algorithm 3 to find the shortest path). The explicit path through the graph is represented by a series of back pointers. A back pointer represents the immediate history of the expansion process. Thus the output of the A* algorithm is a back-pointer path, which is a sequence of nodes starting from the goal and going back to the start.

The A* search has a priority queue which contains a list of nodes sorted by priority denoted O in Algorithm 4, which is determined by the sum of the distance traveled in the graph thus far from the start node, and the heuristic. The processed nodes in O are put in a closed set C. The set of nodes adjacent to current node x is denoted $\textsc{Star}(x)$. The length of edge connecting nodes x_1 and x_2, $d(x_1, x_2)$, is obtained from equation (3.17). The path-cost function, $g(x)$, computes the total length of a backpointer path from current node x to q_{init}. The heuristic-cost function, $h(x)$, provides the estimated cost of straight-line path from current node x to goal node q_{goal}. Since $h(x)$ must be an admissible heuristic, it must not overestimate the distance to the goal. Finally $f(x) = g(x) + h(x)$ is the estimated cost of shortest path from q_{init} to q_{goal} via x. The order in which the search visits nodes in the graph is determined by $f(x)$. If $f(x) = h(x)$, then the search becomes a greedy search since the algorithm is only considering what it *believes* is the best path to the goal from the current node. When $f(x) = g(x)$, the algorithm becomes Dijkstra's algorithm because it is not using any heuristic function and grows a path that is shortest from the start until it encounters the goal [57].

A* is complete in the sense that it will always find a solution if there is one. In order to ensure optimality, all acyclic paths are explored to guarantee that the lowest cost path is found. This searching technique makes A* also optimal. A* will produce an optimal path if its heuristic is optimistic. The returned path may not be a smooth, short and efficient path and thus can be improved by running a postprocessing algorithm [45, 57].

Algorithm 4 FINDPATH(q_{init}, q_{goal}, G)

Require: q_{init}: the initial configuration
q_{goal}: the goal configuration
$G = (V, E)$: the roadmap
Ensure: A path from q_{init} to q_{goal} or failure.

```
1: O ← q_init, C ← ∅, g ← 0, f ← 0
2: repeat
3:     n_best ← the node in O with lowest f score
4:     if n_best = q_goal then
5:         return constructed path
6:     end if
7:     O ← O − {n_best}
8:     C ← C ∪ {n_best}
9:     S ← STAR(n_best) - STAR(n_best) ∩ C
10:    for all x ∈ S do
11:        if PRMCONSTRAINT(x, n_best) = PASSED then
12:            if x ∉ O then
13:                O ← O ∪ {x}
14:            else if g(n_best) + d(n_best, x) < g(x) then
15:                Update x's backpointer to point to n_best
16:            end if
17:        end if
18:    end for
19: until O is empty
```

3.6.2 Visibility and occlusion avoidance constraints

While algorithm 4 is checking the neighborhood of a configuration for possible low-cost path, it also checks the local path for visibility constraint and performs obstacle detection using subroutine PRMCONSTRAINT(q_1, q_2) which employs the formulations developed in Sections 3.2, 3.3 and 3.4. This subroutine checks the input configurations for visibility and obstacle occlusion constraints. Algorithm 5 illustrates this subroutine in detail.

To ensure that the path between $(\mathbf{R}_m, \mathbf{t}_m)$ and $(\mathbf{R}_n, \mathbf{t}_n)$ is collision-free and occlusion-free, another requirement is added to the local planner in Section 3.5.3. At each step,

Algorithm 5 PRMCONSTRAINT(q_m, q_n)

Require: q_m, q_n: two configurations
Ensure: Check for Visibility, Collision and Occlusion: PASSED or FAILURE.

```
(R_m, t_m) ← FORWARDKINEMATICS(q_m)
(R_n, t_n) ← FORWARDKINEMATICS(q_n)
Compute p_m^j and p_n^j using equation (3.6)
if (p_m^j and p_n^j ∈ FOV) then
    if path between (R_m, t_m) and (R_n, t_n) is collision-free and occlusion-free then
        return PASSED
    end if
end if
return FAILURE
```

local planner checks for occlusion in addition to collision detection. More precisely, the local planner Δ will connect q_m to q_n by a straight-line segment in workspace $\mathcal{W}$. The subdivision collision-checking algorithm is then used to test the line segment for obstacle occlusion in addition to collision detection. Figure 3.6 depicts the subdivision occlusion checking algorithm for the path between q_m and q_n in $\mathcal{W}$ for one of the features of the target. First, the middle point q' of the discretized line in $\mathcal{W}$ between q_m and q_n is checked for occlusion using the machinery developed in Section 3.4. Then the algorithm recurses on the discretized lines between (q_m, q') and (q', q_n). The recursion halts when an occlusion is found. If none of the intermediate configurations yields occlusion, the path is considered occlusion-free.

3.7 $\mathcal{C}^2$ trajectory planning in image space

In the previous sections, a discrete image trajectory is generated for each of the target features. Cubic B-spline interpolation [92] is used in this section to design continuous and $\mathcal{C}^2$ differentiable image trajectories, as a requirement of the visual servoing control solution (see equation (2.21)).

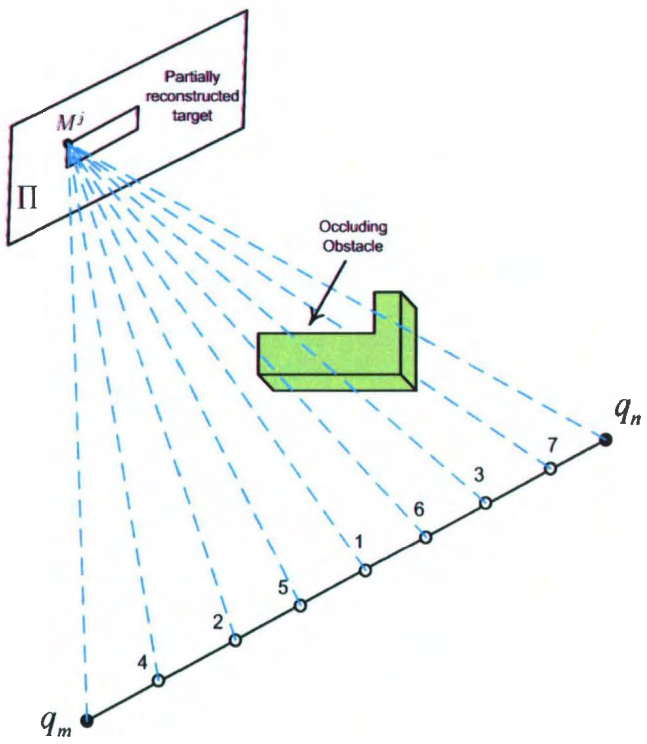


Figure 3.8: Subdivision occlusion detection for a feature in $\mathcal{W}$

The path generated by the query to PRM is reported by $\Theta' = \{\mathbf{q}_i \mid i \in 1 \ldots L\}$ in the joint space which is used to compute $\Gamma' = \{(R_i, t_i) \mid i \in 1 \ldots L\}$ using forward kinematics. The corresponding discrete object trajectory in the image space is denoted $\Upsilon' = \{\mathbf{s}_i \mid i \in 1 \ldots L\}$.

The step parameter δ in local planner in Algorithm 2 is chosen differently while performing the path planning than while generating final trajectory. The local planner requires a step value that ensures that there is no obstacle between the steps and so that the constraints are checked flawlessly. This will reduce the computations required to carry out primary path planning. However during feature trajectory generation, a sufficiently large δ is used so that the camera poses are dense enough to ensure that the local minima are not reached between two camera poses. These local

minima correspond to physically invalid camera positions [11]. Using this method, new discrete sets $\Theta = \{\mathbf{q}_i \mid i \in 1 \ldots K\}$ in the joint space and corresponding path in workspace $\Gamma = \{(R_i, t_i) \mid i \in 1 \ldots K\}$ are generated where K is larger than L depending on the value of step parameter. The discrete object trajectory in the image plane $\Upsilon = \{\mathbf{s}_i \mid i \in 1 \ldots K\}$ is computed from Γ.

The distance between any successive camera positions (R_i, t_i) and (R_{i+1}, t_{i+1}) is not a constant value. To compensate for distance variation between configurations, a constant distance parameter ζ is introduced. The distance between any successive configurations on the designed path is computed using the metric function in equation (3.17) in Section 3.5.4. The distance parameter ζ is used to subdivide this distance in order to obtain a specific step parameter δ for the path between camera positions (R_i, t_i) and (R_{i+1}, t_{i+1}). This step parameter is denoted ${}^i\delta_{i+1}$ and is given by

$$ {}^i\delta_{i+1} = \frac{D_{\mathcal{W}}(q_i, q_{i+1})}{\zeta} \tag{3.21} $$

where q_i and q_{i+1} are the joint space configurations from $\Theta = \{\mathbf{q}_i \mid i \in 1 \ldots K\}$ corresponding to (R_i, t_i) and (R_{i+1}, t_{i+1}) from $\Gamma = \{(R_i, t_i) \mid i \in 1 \ldots K\}$. It is important to note that there is bijective relation between Γ and Θ. The number of elements in the new sets K is given by

$$ K = \sum_{i=1}^{L-1} {}^i\delta_{i+1} + 1 \tag{3.22} $$

A decrease in ζ will make camera poses in the final trajectory denser. It is worth mentioning that δ is variable in final trajectory generation and constant in the primary discrete path planning in Section 3.5.3. Although the distance between Γ_i and Γ_{i+1} is constant, the points in image space Υ_i and Υ_{i+1} are not equidistant.

Given the discrete data points $\Upsilon = \{\mathbf{s}_i \mid i \in 1 \ldots K\}$ and the timing parameters $\mathcal{T} = \{\mathbf{t}_i \mid i \in 1 \ldots K\}$, a cubic B-spline $\mathbf{s}^*(t)$ is computed such that $\mathbf{s}^*(t_i) = \mathbf{s}_i$. Obviously, the timing parameter set $\mathcal{T}$ is not provided in practice. In order to efficiently control the camera velocity, the time values are chosen spaced proportional to ζ which is the distance between camera positions in Γ; thus the time between two following image features in Υ (i.e. $t_{i+1} - t_i$) is constant. This difference $\Delta t = t_{i+1} - t_i = T$ can be chosen as the sampling rate T of the vision system [11].

The desired image trajectory of the features $\mathbf{s}^*(t)$ has the property that $\mathbf{s}^*(0) = \mathbf{s}'$ and $\mathbf{s}^*(t_f) = \mathbf{s}^*$. Since there is no information about the end point derivatives, **not-a-knot** condition is used which makes the first and the last interior knots inactive [93]. The B-spline interpolation equation for $\mathcal{C}^2$ function $\mathbf{s}^*(t)$ is given by

$$\mathbf{s}^*(t) = A_i t^3 + B_i t^2 + C_i t + D_i t \tag{3.23}$$

for the interval $(i-1)T \leq t \leq iT$. A_i, B_i, C_i and D_i are $2n \times 2n$ diagonal spline coefficient matrices obtained from $\mathcal{T}$ and Υ.

The depth ratio set of features Ψ is estimated using scaled partial reconstruction and image features using equation (3.3) at each node of the path. Similarly, a continuous function $\Psi(t)$ should be computed for the depth information, given the discrete set $\Psi = \{\rho_k^j | j = 1 \cdots n \ , k = 1 \cdots K\}$ (with n and K being, respectively, the number of features and the number of intermediate camera poses) and the timing parameter set $\mathcal{T}$. The B-spline interpolation function $\Psi(t)$ is given by

$$\Psi(t) = E_i t^3 + F_i t^2 + G_i t + H_i t \tag{3.24}$$

for the interval $(i-1)T \leq t \leq iT$. E_i, F_i, G_i and H_i are $n \times n$ diagonal spline

coefficient matrices obtained from $\mathcal{T}$ and Ψ.

3.8 Feature trajectory tracking

The feature trajectories $\mathbf{s}^*(t)$ that take the initial configuration to the desired pose while ensuring the desired constraints, are computed using the path planning solution in the previous sections using spline interpolation.

When the induced error $\mathbf{s} - \mathbf{s}^*$ is large, modeling errors may have greater effect on the performance and even the robustness of the visual servo. Coupling path planning with trajectory following improves the robustness of the visual servo significantly with respect to modeling errors and incapacity of the control scheme.

Once the intermediate subgoal image trajectory $\mathbf{s}^*(t)$ is designed, the visual servo control scheme introduced in Chapter 2 is modified to take into account for the time-varying reference feature set so that the error $\mathbf{s}(t) - \mathbf{s}^*(t)$ remains small during visual servo. It is noteworthy that the interaction matrix (relation 2.19 in Chapter 2) depends on the estimated parameter $\widehat{d^*}$ through homography formulations and on the depth of the target Z.

A depth vector $\mathbf{Z}$ for the features on the designed trajectory is required but cannot be computed explicitly. Instead the depth ratio $\rho_k^j = \frac{Z_k^j}{d^*}$ will be used to rewrite the dynamics of the visual servo. The interaction matrix (2.19) is rewritten to take this ratio into account:

$$\mathbf{I}(\mathbf{s}^*(t), \Psi(t), \widehat{\mathbf{C}}, \widehat{d^*}) = \begin{bmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{bmatrix} \left[\frac{1}{\widehat{d^*}} \mathbf{A}(\mathbf{s}^*(t), \Psi(t)) \quad \mathbf{B}(\mathbf{s}^*(t)) \right] \tag{3.25}$$

$\Psi(t)$ is the depth ratio function, computed using Ψ and $\mathcal{T}$ by spline interpolation in the previous section. $\widehat{\mathbf{C}}$ and $\widehat{d^*}$ are the estimated value of $\mathbf{C}$ and d^*, respectively.

$\mathbf{A}$ and $\mathbf{B}$ are two $2n \times 3$ matrices given by

$$\mathbf{A} = \begin{bmatrix} -\frac{1}{\rho_k^j} & 0 & \frac{x_k^j}{\rho_k^j} \\ 0 & -\frac{1}{\rho_k^j} & \frac{y_k^j}{\rho_k^j} \end{bmatrix} \tag{3.26}$$

$$\mathbf{B} = \begin{bmatrix} x_k^j y_k^j & -1 - {x_k^j}^2 & y_k^j \\ 1 + {y_k^j}^2 & -x_k^j y_k^j & -x_k^j \end{bmatrix} \tag{3.27}$$

To track the image trajectories using an image-based control scheme, the error term of the visual servo control scheme (2.21) is

$$\dot{\mathbf{e}} = \dot{\mathbf{s}} = \mathbf{I}\,\dot{\mathbf{r}} - \frac{\partial \mathbf{s}^*}{\partial t} \tag{3.28}$$

assuming that the target is motionless. An exponential decoupled decay of $\mathbf{e}$ to zero (e.g. $\dot{\mathbf{e}} = -\kappa \mathbf{e}$) is desired; thus the corresponding control law is obtained using equation (3.28)

$$\dot{\mathbf{r}} = -\kappa\, \widehat{\mathbf{I}}^{+}\, \mathbf{e} + \widehat{\mathbf{I}}^{+}\, \frac{\partial \mathbf{s}^*}{\partial t} \tag{3.29}$$

where $\widehat{\mathbf{I}}^{+}\, \frac{\partial \mathbf{s}^*}{\partial t}$ compensates for the tracking error. It is noteworthy that $\frac{\partial \mathbf{s}^*}{\partial t}$ is easily computed from equation (3.23) as

$$\frac{\partial \mathbf{s}^*}{\partial t} = 3A_i t^2 + 2B_i t + C_i \tag{3.30}$$

for the interval $(i-1)T \leq t \leq iT$.

3.9 Summary

The path planning algorithm, presented in this chapter, is a flexible PRM-based planning method which fulfills the necessary requirements of a visual path planner. A sampling-based (Probabilistic Roadmap) planning method is used to perform visually constrained path planning for manipulators. The algorithm is using a visibility constraint to keep the target features in the camera FOV. An occlusion avoidance constraint is developed and used by the path generator to design occlusion-free paths. PRM also helps to generate a path such that the robot joint trajectories are within kinematic range of the joints. The algorithm also ensures that the generated path is not in collision with the environment. Since it is assumed that the target does not have a 3D model, a homography based method is used to partially reconstruct the target. If the target model is provided, the path planning can be easily performed by projecting the target on the image plane, similar to obstacle projection. Cubic B-splines are used to generate differentiable image trajectories which are used for the visual servo as a time-varying reference in the image plane. The classical IBVS control law is modified to be able to track the generated path. As will be shown in the next chapter, the control law is robust with respect to modeling errors and noise perturbations since the designed path introduces subgoal image features and keeps the error small such that the local stability of the visual servo control holds.

Chapter 4

Results

About this chapter: This chapter describes the simulation results to validate the proposed path planning scheme. A brief discussion on the equipment used to carry out the simulations and off-line experiments is provided. This follows the description and analysis of the results obtained from various simulations.

4.1 Robot and Vision System

The CRS A255 articulated arms (Catalyst 5) is used world-wide in applications ranging from automated laboratory tasks, automotive assembly and repetitive product testing. The robot arm is supported by the C500C controller and the RAPL3 programming language for task planning and coordination. The fixed architecture controller assures proper operation while it prevents one from implementing other dynamics and kinematics feedback controllers. In order to benefit an open architecture controller, the MultiQ ISA board interfaces the robot and the C500C amplifiers to PC. WinCon software allows one to run customized controllers using Simulink diagrams. One switch allows to take over the fixed control strategy.

A general review of the features of equipment is as follows:

Robot & C500C Controller Articulated 5 DOF robot. Encoder feedback, servo motors, servo gripper, $\pm 0.05mm$ repeatability. Controller with 6 PWM servo amplifiers and processors which incorporate all the control algorithms required to perform the desired tasks.

RAPL-3 Programming language for continuous path, joint interpolation, point-to-point relative motions. A straight-line plus on-line path planner to blend commanded motions in joint or straight-line mode.

WinCon Client/Server realtime control from Simulink diagram and via the INTERNET. Standalone control with control panels. Realtime tuning and plotting. 1 kHz sampling rate for 5DOF system

MultiQ board 8 A/D, 8 D/A, 8 Encoder inputs, 8 DIO, 3 Clocks.

4.1.1 Vision System

A Pantex A102fc camera equipped with a monofocal iris lens is mounted on the end-effector of the robot manipulator using a custom made bracket (see Figure 4.2). The calibration of the intrinsic parameters of the camera is performed using standard camera calibration techniques[1].

Eye-Hand Calibration

Robot eye-to-hand calibration is the process of determining the transformation between the end-effector coordinate system and the camera coordinate system. Generally, a number of movements of the robot arm and the corresponding changes in

[1]MATLAB toolbox for camera calibration, http://www.vision.caltech.edu/bouguetj/calibdoc/

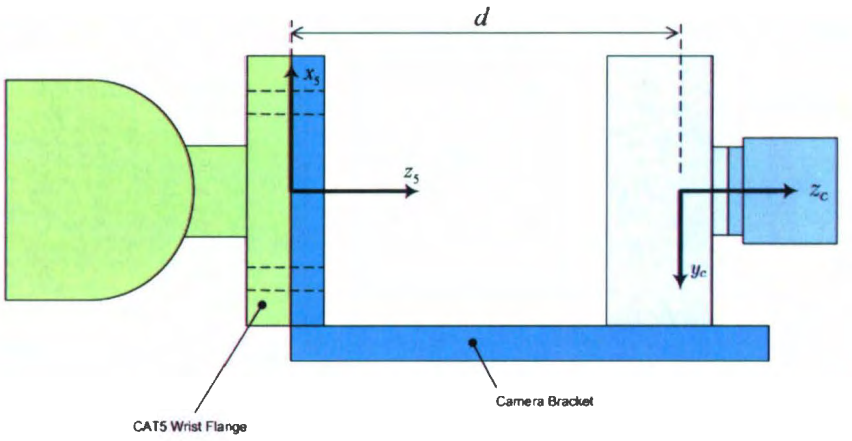


Figure 4.1: Details of camera mounting (coordinate systems)

image plane coordinates of a fixed object are required. Tsai's method relies on use of the planar calibration target, for instance. An algorithm is then applied to determine the camera transformation [29].

Instead of performing the tedious calibration task, a more pragmatic approach is used to determine the transformation from the known geometry of the camera, lens (figure 4.3) and robot arm. The location of the CCD sensor plane within the camera is not directly measurable. However the lens manufacturer's data shows that the focal point of the lens is $17.526\,mm$ behind the mating surface depicted in figure 4.3. This distance is called the Flange Focal Distance (FFD) and has the same value for C-Mount lenses. The plane of an equivalent simple lens will be located the focal length in front of the photo-sensitive surface of the sensor. From this data, the distance d in figure 4.1 can be inferred as $77.424 + \lambda$.

The coordinate frame of the camera is also shown in figure 4.1. The transformation

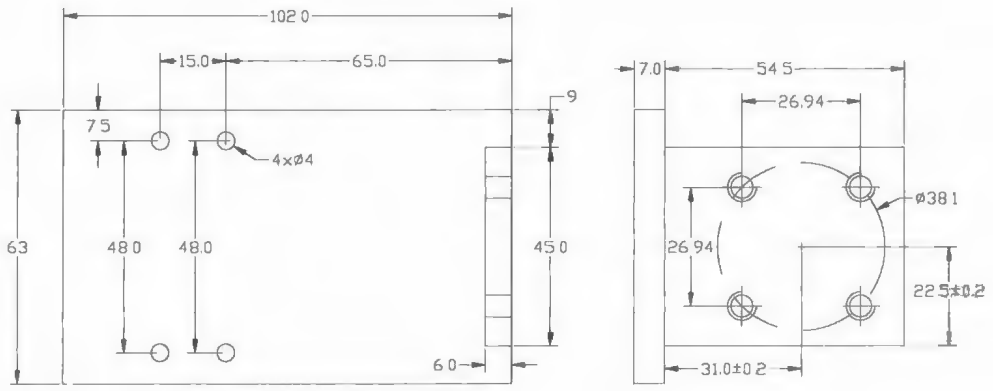


Figure 4.2: Camera mounting bracket

can be expressed in terms of rotations and translations as

$$ {}^{5}\mathbf{T}_{c} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 77.424 + \lambda \\ 1 & 0 & 0 & 1 \end{bmatrix} \tag{4.1} $$

where λ is the focal length and is determined by the camera intrinsic parameter calibration. The rotation component is the direction cosine representation of the rotation between the fifth frame and the camera frame.

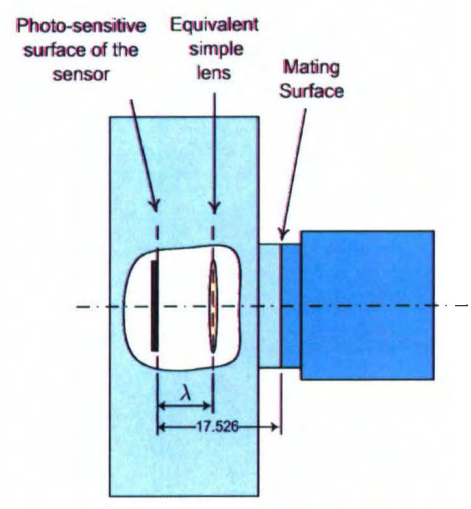


Figure 4.3: Details of camera, lens and CCD sensor location

4.2 Tests and simulations

The proposed method has been tested using the parameters of the 5-DOF eye-in-hand CRS robot. Since image processing is not of interest, a simple black rectangle on a white background is used as a target in tests. Four corners of this target are extracted and tracked in the image space as visual features. Various researchers has thoroughly studied the application of visual servoing control schemes to real objects and environments [94 96]. The proposed method can be directly applied to real objects if matched points in the initial and desired images are available and can then be tracked. Due to the nature of the path planning scheme, visibility constraint needs to be taken into account as the fundamental constraint to generate appropriate paths for visual servoing. This constraint is the basic constraint that provides necessary modifications to PRM in order to generate visual paths. Joint limits constraint is also considered during preprocessing phase of PRM construction and therefore visibility and joint limits constraints are imposed on the basic queried path. Visual occlusion and obstacle avoidance are other constraints that make PRM capable to generate

Table 4.1: Preprocessing time for roadmap construction

N	Preprocessing (min)
5000	34.3
7000	62.6
10000	108.1
50000	613.9

more flexible visual paths. The path planner is designed to generate trajectories for target of unknown CAD model.

Test and simulations performed to validate the proposed path planning method are presented in the following subsections.

Part A : The results regarding probabilistic roadmap construction are presented.

Part B : Visibility constraint and joint limits constraint are taken into account in this series of simulations. A general off-line path planning is performed to confirm the extended robustness provided with the method with respect to modeling error and noises.

Part C : In these set of off-line experiments, visual occlusion constraint and visibility constraint are studied in more detail in path planning and visual servoing.

4.2.1 Part A

Probabilistic Roadmaps are constructed in MATLAB using mex-files, EML (Embedded Matlab) features to run extensive computations faster. Table 4.1 summarizes the computational details of probabilistic roadmap construction algorithm implemented in C-Mex.

Note that joint limits constraint is considered during preprocessing phase of PRM construction. The sampling is performed on the allowable kinematic range of mo-

Table 4.2: PRM preprocessing and query parameters

Parameter	Value
N	50000
D_Q	10
Number of neighbors k (prepro.)	30
Number of neighbors k (query)	20
Joint weights w_i	$[1, 0.8, 0.8, 0.35, 0.2]$

tion of each joint. Therefore the queried paths will be generated with imposed joints limit constraint. There is a trade-off between postprocessing and query durations with respect to obstacle avoidance detection. If the obstacle avoidance is postponed to be performed during query phase, the preprocessing will be much faster but the query time will increase. Since visibility and occlusion avoidance constraint are computationally expensive tasks to perform at every configuration, the obstacle collision detection is performed in the PRM preprocessing phase. Table 4.2 list all the parameters employed to construct the PRM.

Previous research has suggested that the relative importance of the rotation component in equation (3.17) decreases as the planning queries become harder [90]; thus w_t and w_r are chosen 0.7 and 0.3.

4.2.2 Part B

In this section, it is shown that planning a trajectory for visual servoing is of general interest. Using a designed time-varying reference $\mathbf{s}^*(t)$ rather than a constant reference $\mathbf{s}^*$ improves the performance of a classical visual servo. The robustness of the classical visual servo is tested against coarse calibration of the camera. Path planning is also performed to demonstrate the improved performance of the visual servo. Three different scenarios are simulated:

1. Classical visual servo is carried out with correct calibration of intrinsic parameters. A constant reference $\mathbf{s}^*$ is used.

2. A 50% error is introduced in the intrinsic parameters and the same constant reference $\mathbf{s}^*$ is used to perform the test once more.

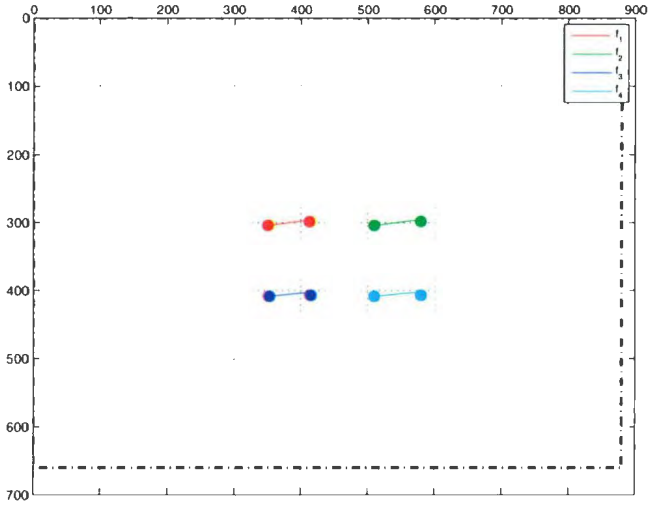
3. An image trajectory is designed for the tracking controller to track; thus time-varying $\mathbf{s}^*(t)$ is used. A 50% error is introduced in the intrinsic parameters.

The results of the above scenarios are depicted in Figures 4.4, 4.5 and 4.6, respectively. The feature trajectories are straight as expected with correct calibration parameters (Figure 4.4). However with modeling errors the trajectories are not straight lines in the image plane (Figure 4.5). The motion of the camera is not predictable and the features may exit the field of view. As depicted in Figure 4.6, introducing time-varying reference $\mathbf{s}^*(t)$ improves the performance of the visual servo and the expected trajectories are obtained even when an important calibration error exists. It is important to note the differences of the feature errors in image space in Figures 4.4, 4.5 and 4.6.

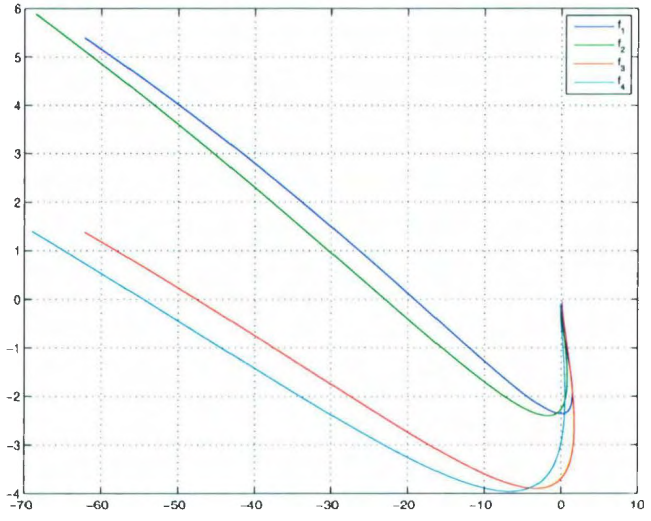
As it can be seen, the error signals in Figure 4.6, decreases smoothly towards zero. Using the path planning for visual servoing thus makes the feature trajectories in image space predictable as straight lines and provides better performance than merely controller-induced motion and significantly improves the robustness of the visual servo with respect to modeling errors.

4.2.3 Part C

One of the benefits of the proposed method is that it does not require the model of the target. In all of the simulations, the path planning is performed for a target



4.4.1: Target trajectory $\mathbf{s}(t)$ in image space

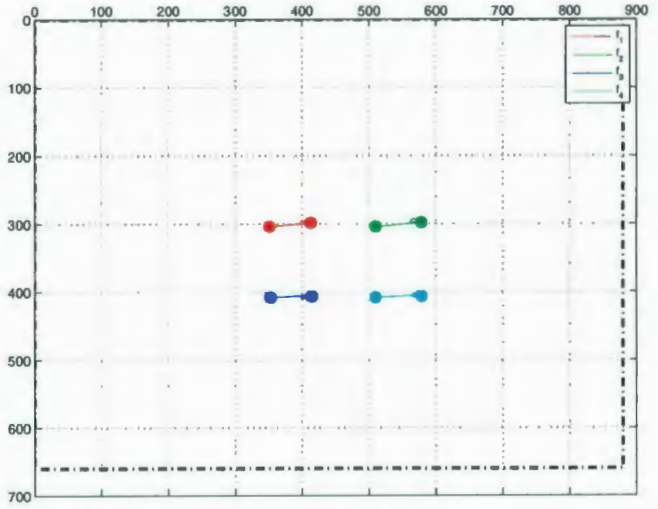


4.4.2: Error trajectory $(\mathbf{s}(t) \quad \mathbf{s}^*)$ in pixels in image space

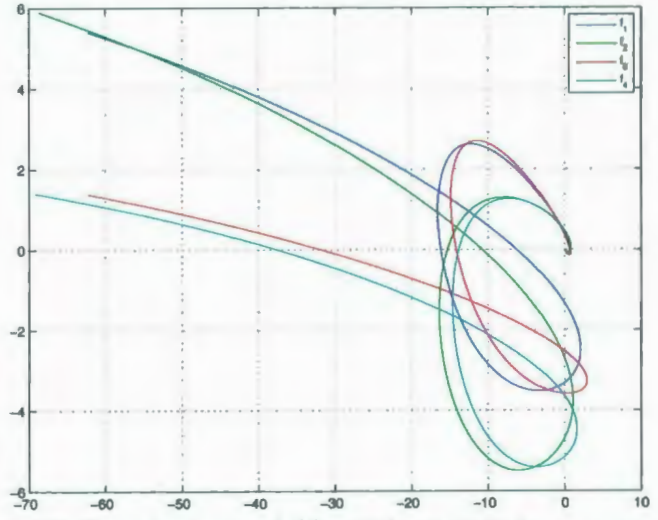
Figure 4.4: Visual Servoing using $\widehat{\mathbf{I}}^+(\mathbf{s}^*, \widehat{Z^*}, \widehat{\mathbf{C}})$ with correct intrinsic parameters

with an unknown model. In these sets of off-line experiments, visibility and joint limits constraints are considered in the path planning. The joint limits constraint is considered during preprocessing phase of PRM construction due to the sampling method. Several set of simulations are presented to validate the performance of the proposed trajectory generator.

In these tests, queries are made to the constructed PRM; the returned paths are



4.5.1: Target trajectory $\mathbf{s}(t)$ in image space



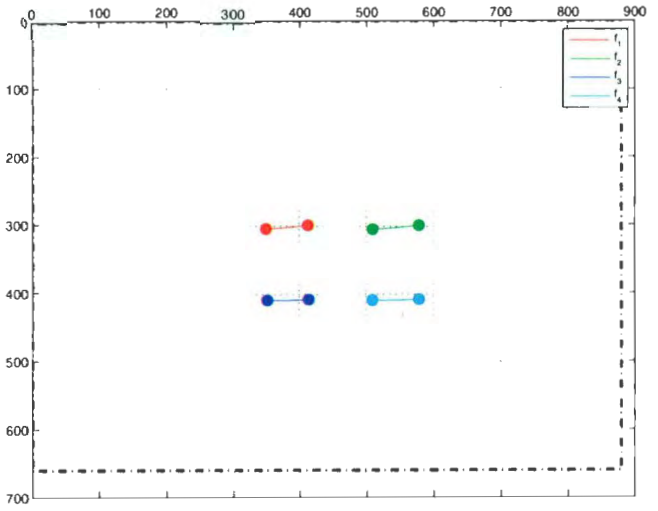
4.5.2: Error trajectory $(\mathbf{s}(t)-\mathbf{s}^*)$ in pixels in image space

Figure 4.5: Visual Servoing using $\widehat{\mathbf{I}}^{+}(\mathbf{s}^*, \widehat{Z^*}, \widehat{\mathbf{C}})$ with 50% error in calibration

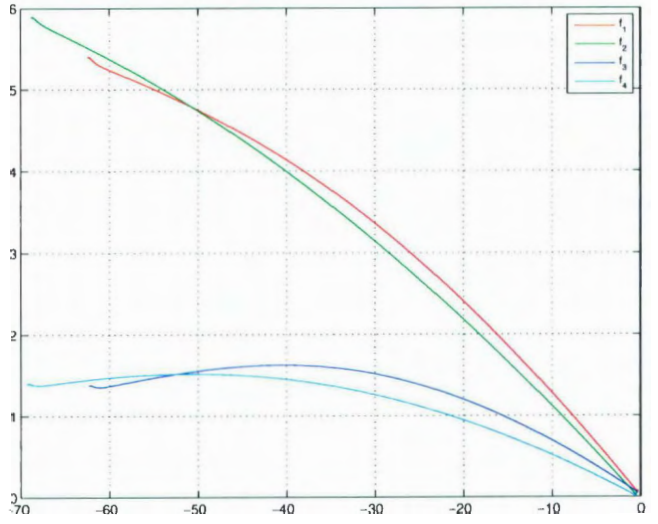
post-processed and used to generate a cubic spline representation that is employed as the reference for the visual servo tracking controller.

Simulation of visibility constraint

The visibility constraint developed in Chapter 3 is the primary constraint that should be used to generate visual paths for a visual tracker to follow. The performance of



4.6.1: Target trajectory $\mathbf{s}(t)$ in image space

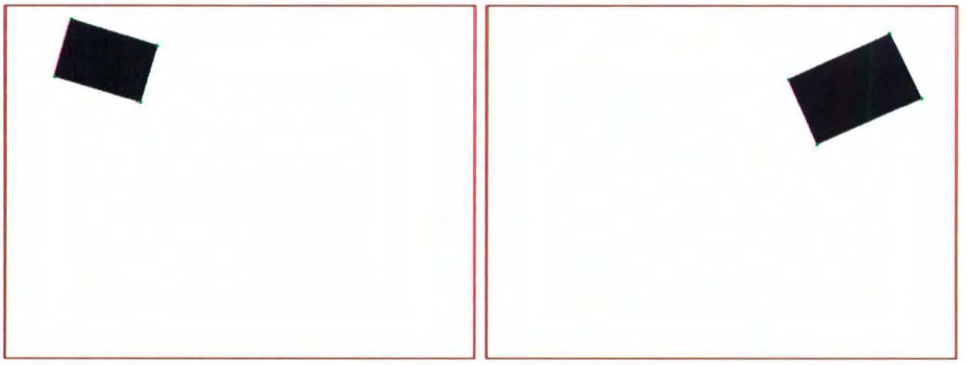


4.6.2: Error trajectory $(\mathbf{s}(t) - \mathbf{s}^*)$ in pixels in image space

Figure 4.6: Visual Servoing using $\widehat{\mathbf{I}}^+(\mathbf{s}^*(t), \Psi(t), \widehat{\mathbf{C}}, \widehat{d^*})$ with 50% error in calibration

the trajectory planning system is examined by performing benchmarking simulations. The following simulation is not convergent with classical IBVS. The system accomplishes the task with good results. The initial and desired images used to design the path are depicted in Figure 4.7. The fiducial features are the four corners of the rectangular target (shown with small green circles).

The postprocessed camera trajectory in workspace for this scenario is depicted

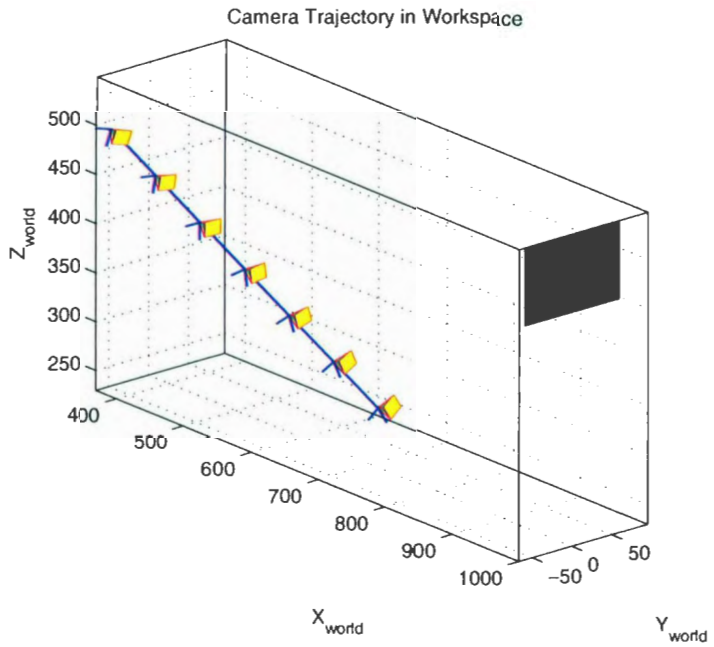


4.7.1: Initial image $\mathbf{s}^i$ 4.7.2: Desired image $\mathbf{s}^*$

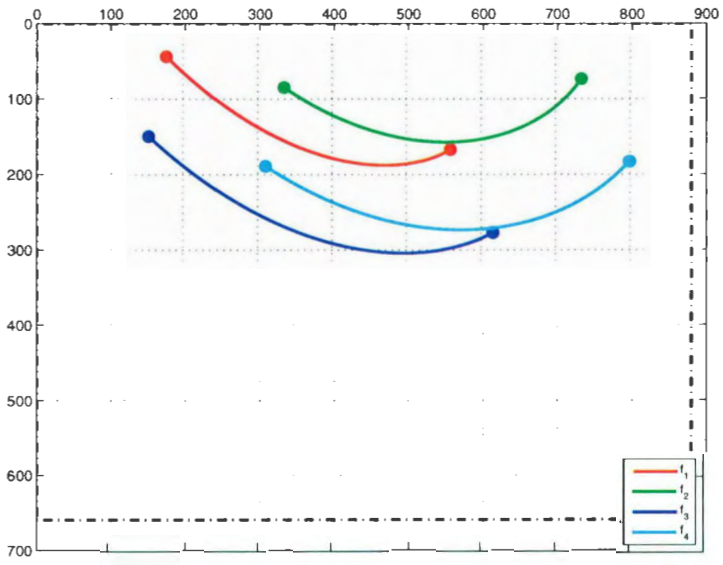
Figure 4.7: Original initial and desired images (with binary threshold)

in Figure 4.8. The grey rectangle shown in Figure 4.8.1 represents the unknown planar target. The four corners of the rectangle are the features to be tracked. ζ is chosen as 0.1 thus resulting $\delta = 400$ for this example. Figure 4.9.1 shows the tracked feature trajectories by the IBVS controller (see Chapter 3, Section 3.8). As it can be seen in Figure 4.9.2, the tracking error does not exceed 10 pixels. The exponential decrease of the error terms $(\mathbf{s}(t) - \mathbf{s}^*(t))$ is shown in Figure 4.9.3 which is also depicted alternatively in Figure 4.9.4. Since the path is generated using the joint limits constraint, all the joint positions are within limits during visual servo. To facilitate the illustration of the joint positions on one plot, joint positions are normalized into $[-1; 1]$ where -1 and 1 represent the joint limits (see Figure 4.9.5).

Figure 4.9.6 depicts the induced camera velocities or in other words computed control law. Note that the integration of angular velocity about x-axis, w_x, provides extra information and is not used in inverse kinematics of the 5-DOF robot arm; however, the corresponding computed control law decreases to zero. The corresponding joint position displacement between the desired and the initial camera frames is large (i.e. $(-7.5, 0, 0, -7.2, -14.32) \rightarrow (7.5, -90, -10, 16.8, 25.06)$). It is important to note that, as mentioned previously, classical IBVS fails in this case.

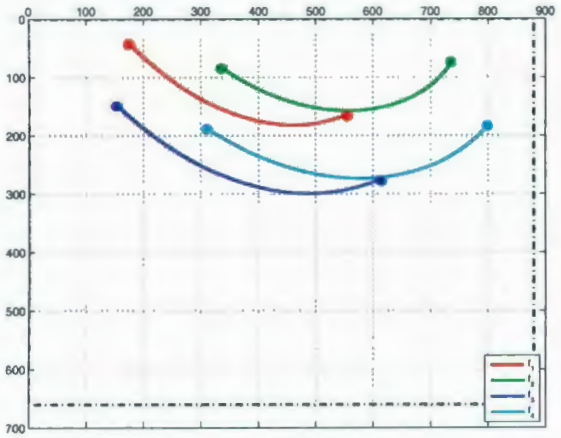


4.8.1: Postprocessed camera trajectory in workspace

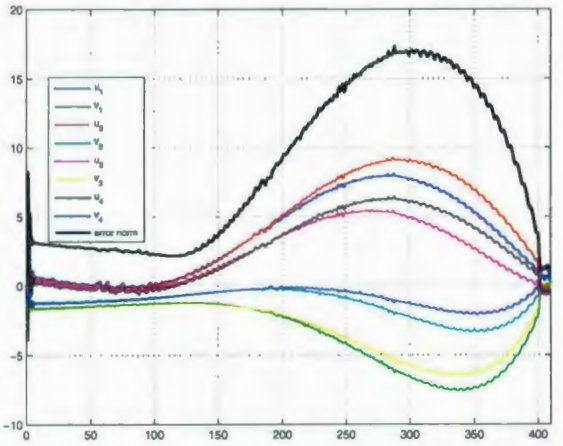


4.8.2: Planned image plane trajectories $\mathbf{s}^*(t)$

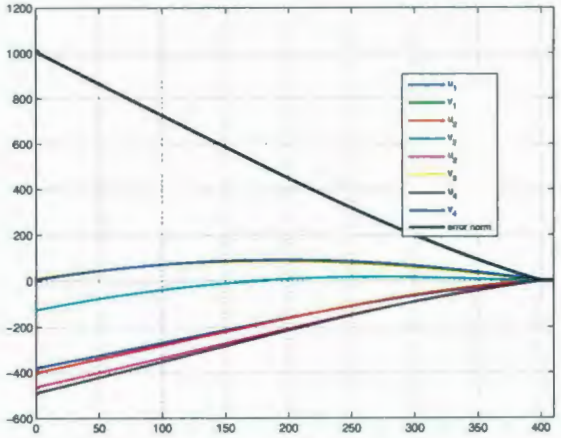
Figure 4.8: Planned camera and feature trajectories



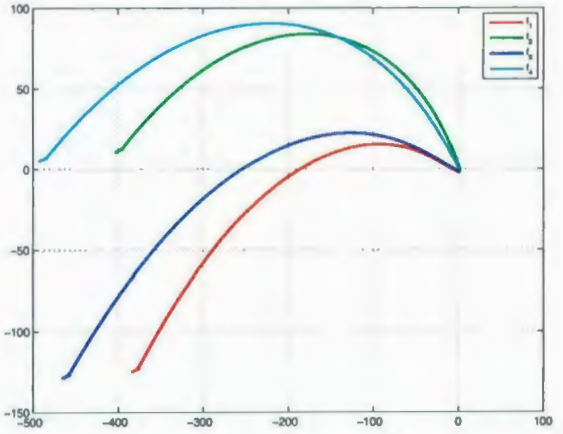
4.9.1: Tracked image plane trajectories $\mathbf{s}(t)$



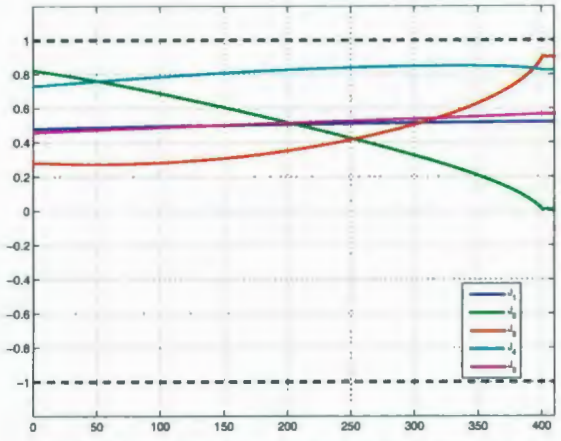
4.9.2: Tracking error $(\mathbf{s}(t) - \mathbf{s}^*(t))$



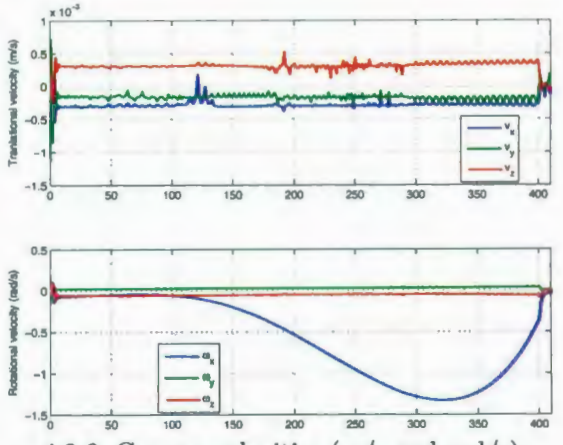
4.9.3: Error $(\mathbf{s}(t) - \mathbf{s}^*)$



4.9.4: Error trajectory $(\mathbf{s}(t) - \mathbf{s}^*)$ in image space (pixels)



4.9.5: Normalized trajectories in joint space



4.9.6: Camera velocities (m/s and rad/s)

Figure 4.9: Visual servoing using the designed path in Figure 4.8

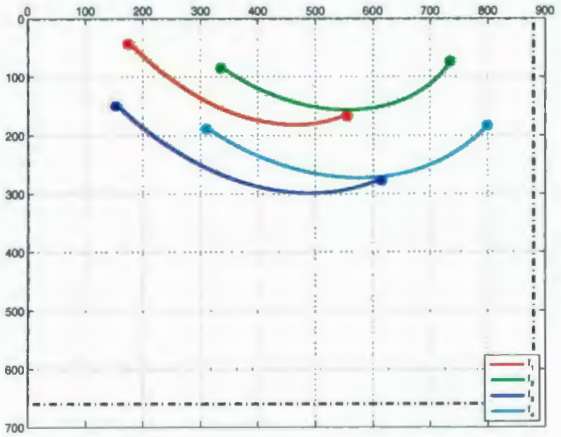
In the experiment whose results are reported in Figure 4.9, the intrinsic parameters given by the camera manufacturer are used. The same test is performed to examine the robustness of the system to noise and calibration errors. Figure 4.10 shows the performance of the system with 20% noise in the calibration parameters which is considered a coarse calibration. Figure 4.11 shows the performance of the system with 45% noise in the calibration parameters i.e. a bad calibration.

It is important to note the high performance of the tracker in Figure 4.11. It is obvious that the visual servo task is performed successfully while the joint positions are within their kinematic ranges, as expected. One of the other benefits of path planning and tracking the trajectory is that the computed control signal is kept at a level which will not cause severe excitation of the joint motors, even though some filtering may be carried out.

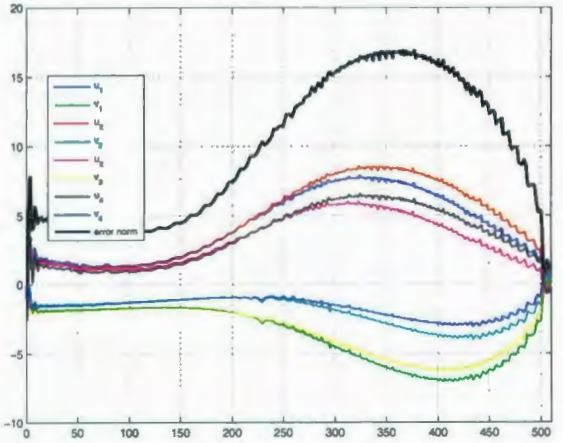
Simulation of visual occlusion avoidance

In this section, the visual occlusion avoidance algorithm is activated and an obstacle with known model is added to the workspace. In this case the path planner will attempt to circumvent the obstacle. Classical IBVS and PBVS cannot achieve visual servo tasks when there is an occlusion in the induced feature trajectories. The initial and desired images used to design the path are depicted in Figure 4.12.

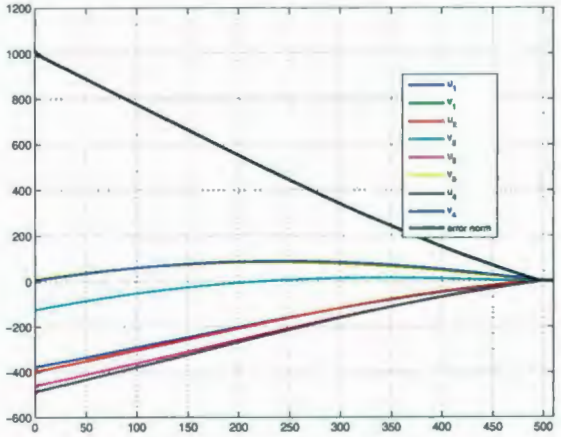
Figure 4.13 depicts the camera trajectory directly from the PRM planner. The obstacle is illustrated as a blue block in Figure 4.13.1. To further smooth the path, A* search is performed once more on the returned path. Furthermore the path postprocessing algorithm explained in [57] is used to improve the path. Figure 4.14 shows the postprocessed version of the path shown in Figure 4.13. The trajectories are designed such that the target remains in the FOV while visual servo control solution takes the initial image to the desired one. It also ensures that joint positions are



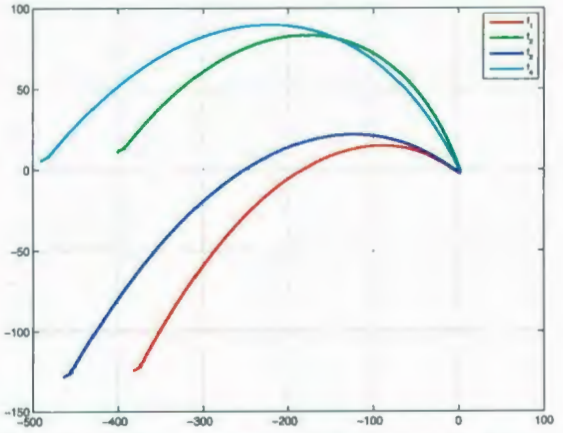
4.10.1: Tracked image plane trajectories $\mathbf{s}(t)$



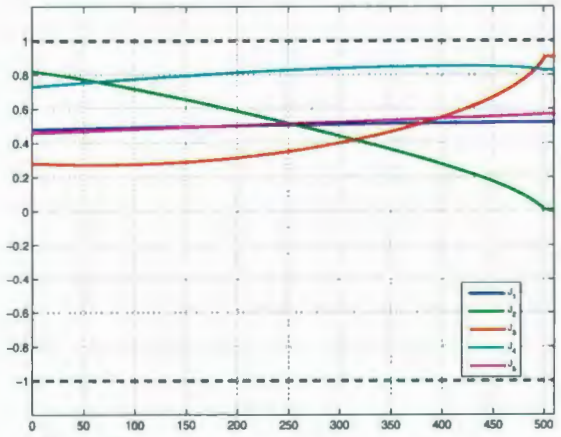
4.10.2: Tracking error $(\mathbf{s}(t) - \mathbf{s}^*(t))$



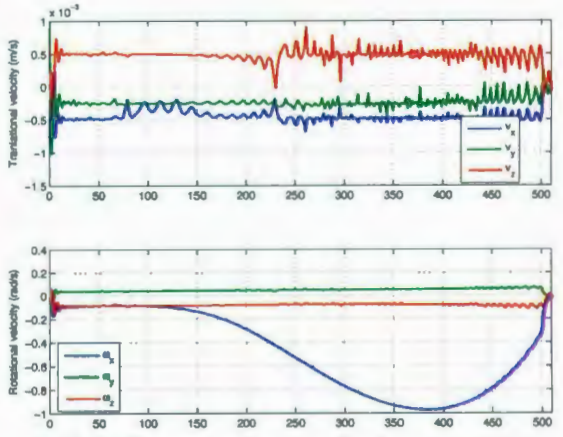
4.10.3: Error $(\mathbf{s}(t) - \mathbf{s}^*)$



4.10.4: Error trajectory $(\mathbf{s}(t) - \mathbf{s}^*)$ in image space (pixels)

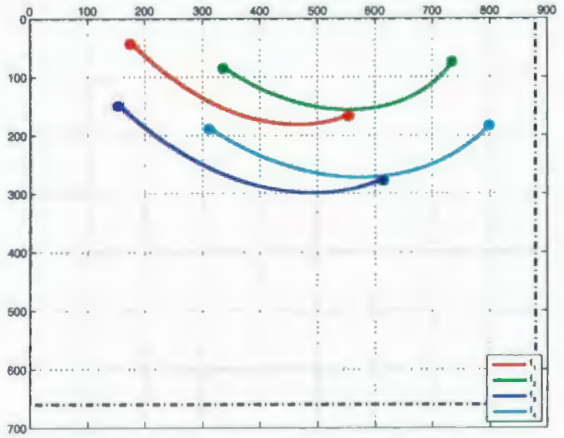


4.10.5: Normalized trajectories in joint space

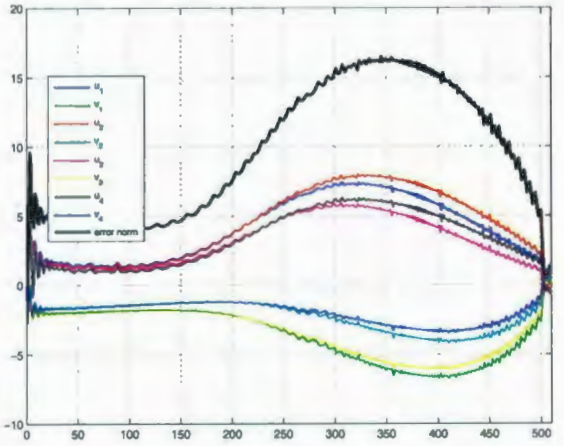


4.10.6: Camera velocities (m/s and rad/s)

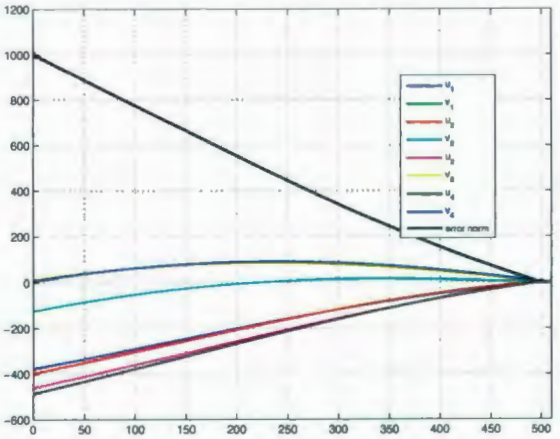
Figure 4.10: Visual servoing using the designed path in Figure 4.8 with 20% noise



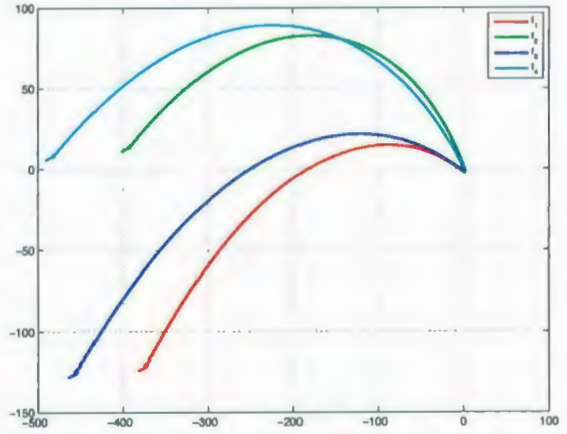
4.11.1: Tracked image plane trajectories $\mathbf{s}(t)$



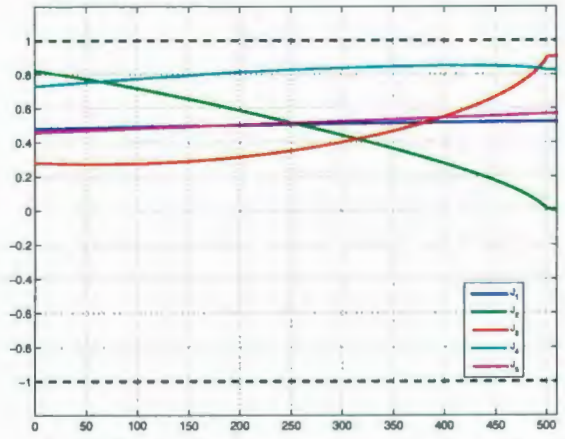
4.11.2: Tracking error $(\mathbf{s}(t) - \mathbf{s}^*(t))$



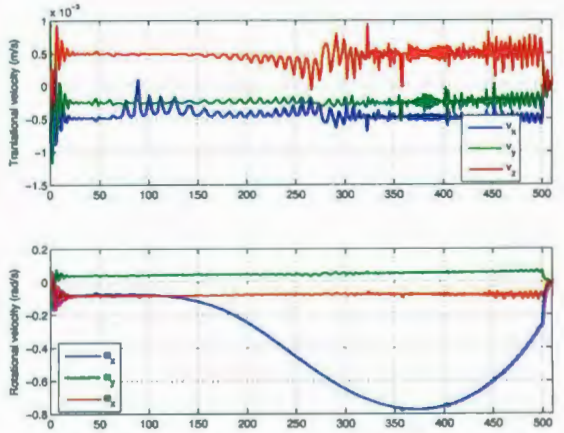
4.11.3: Error $(\mathbf{s}(t) - \mathbf{s}^*)$



4.11.4: Error trajectory $(\mathbf{s}(t) - \mathbf{s}^*)$ in image space (pixels)

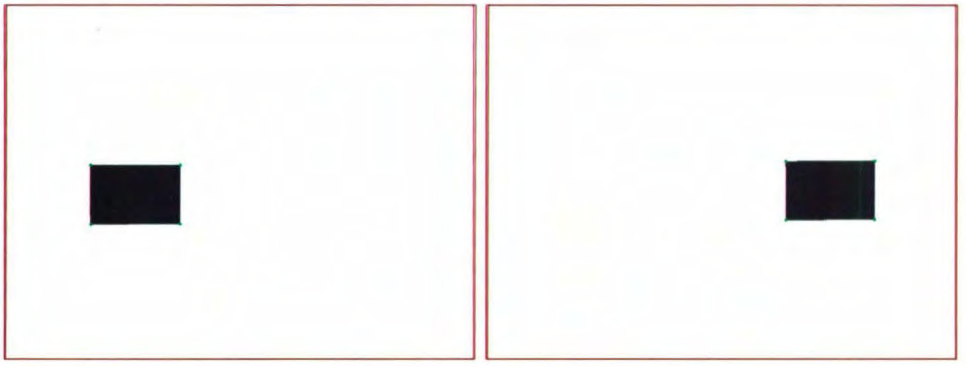


4.11.5: Normalized trajectories in joint space



4.11.6: Camera velocities (m/s and rad/s)

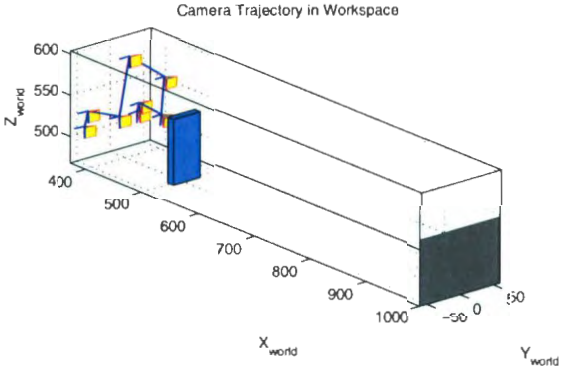
Figure 4.11: Visual servoing using the designed path in Figure 4.8 with 45% noise



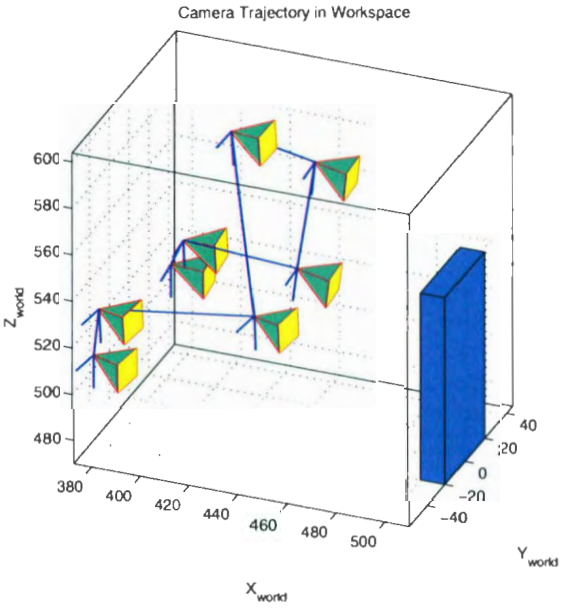
4.12.1: Initial image $\mathbf{s}^i$ 4.12.2: Desired image $\mathbf{s}^*$

Figure 4.12: Original initial and desired images (with binary threshold)

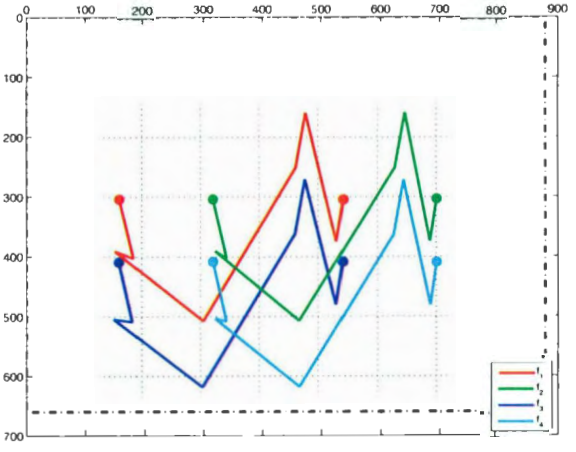
always within limits and that visual occlusion is performed properly. The visual servo controller has accomplished performing the task. Figure 4.15 shows the convergent results of visual servo of the postprocessed path shown in Figure 4.14. As it is shown in Figure 4.15.2, the tracking error is not more than 4 pixels. The visual servo converges with an error of less than 1 pixel. The break point in image trajectory causes a *corner* in joint trajectory (e.g. see Figure 4.15.5). The same path in Figure 4.8 is thus further smoothed in the spline interpolation phase such that the break point is removed (see Figure 4.16.1 and 4.16.5). Although smoothing the path will provide a better velocity and acceleration profiles for joints, the generated path should be checked for occlusion, specifically in smoothing regions. If there is any occlusion, the path will be rejected. The simulation results of the visual servo using the smoothed path are depicted in Figure 4.16. It is important to note the difference in the computed control law between the smoothed path (see Figure 4.16.6) and the postprocessed path (see Figure 4.15.6). To show that the system is not sensitive to modeling errors, a 40% camera calibration is introduced into the system. It is obvious that the system is robust to errors and perturbations (see Figure 4.17).



4.13.1: Camera trajectory in workspace

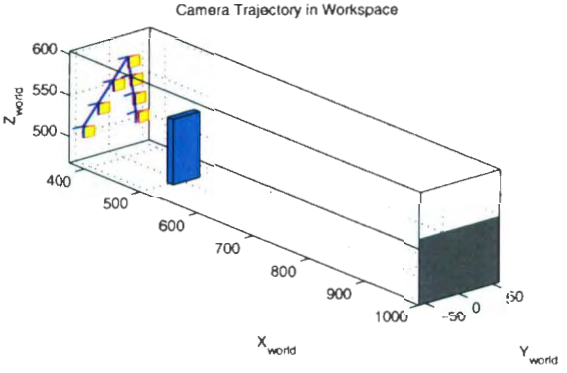


4.13.2: Magnified camera trajectory

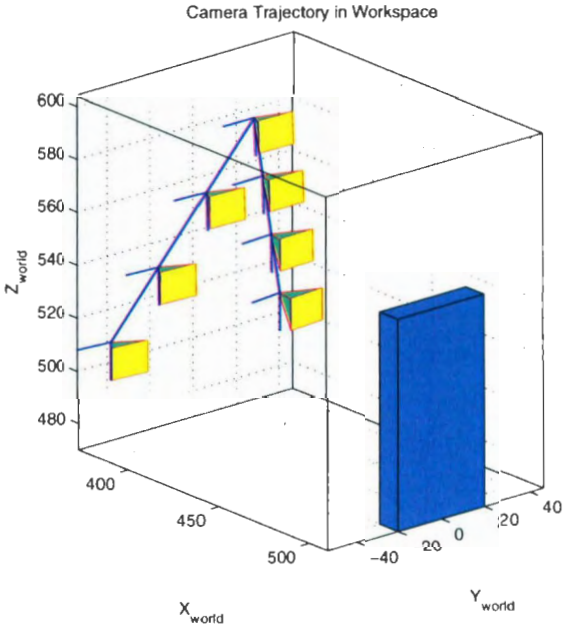


4.13.3: Planned image plane trajectories $\mathbf{s}^*(t)$

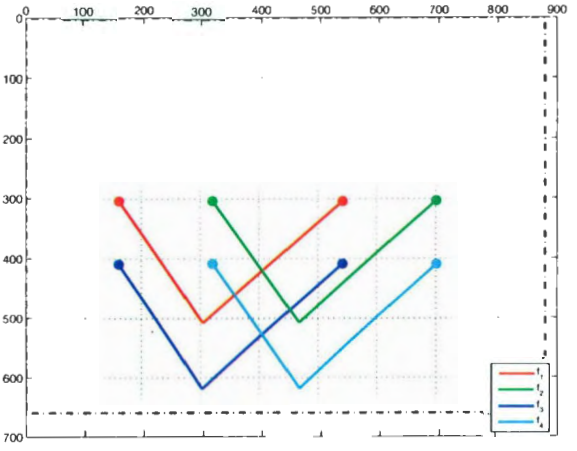
Figure 4.13: Planned camera and feature trajectories without postprocessing



4.14.1: Camera trajectory in workspace



4.14.2: Magnified camera trajectory

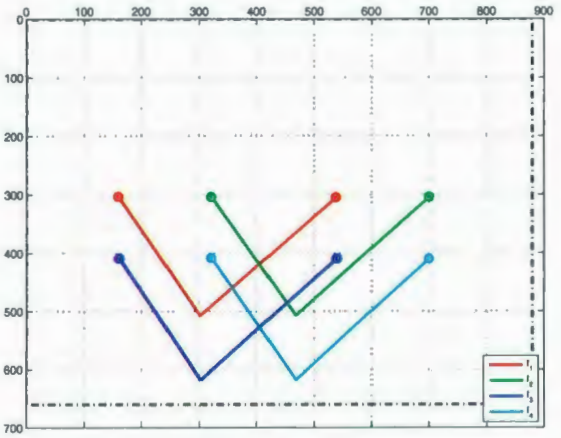


4.14.3: Planned image plane trajectories $\mathbf{s}^*(t)$

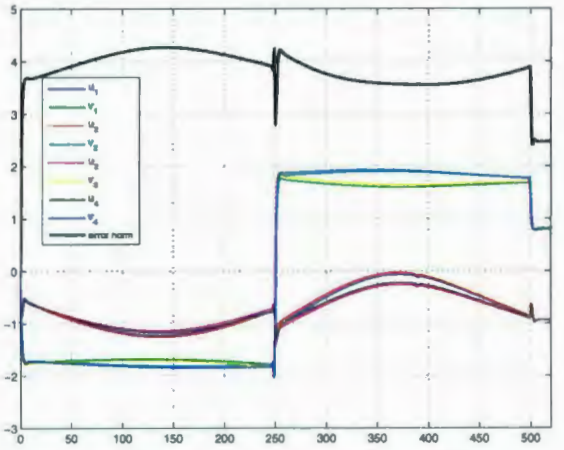
Figure 4.14: Postprocessed camera and feature trajectories in Figure 4.13

4.3 Limitations and discussion

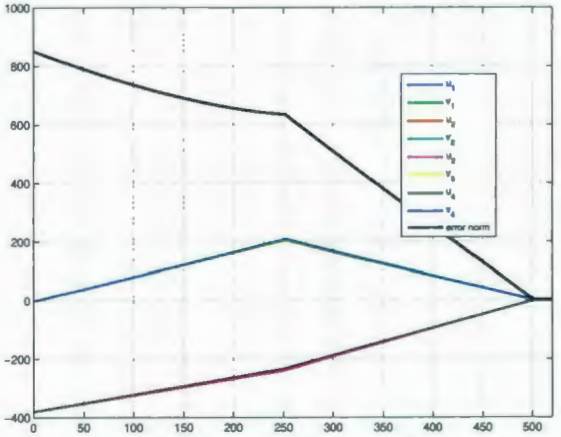
A set of parameters influence the convergence of the proposed system. This section discusses the effect of these parameters on the convergence and performance of the proposed system.



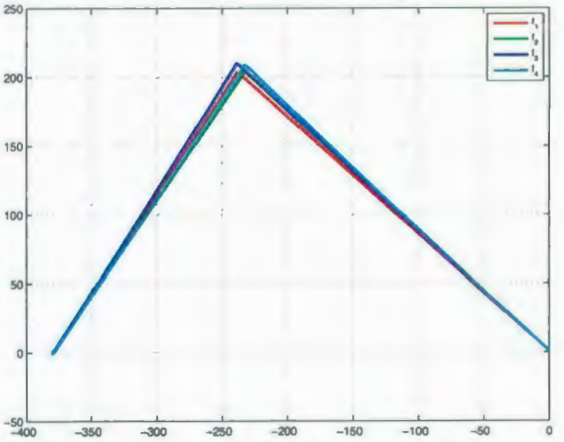
4.15.1: Tracked image plane trajectories $\mathbf{s}(t)$



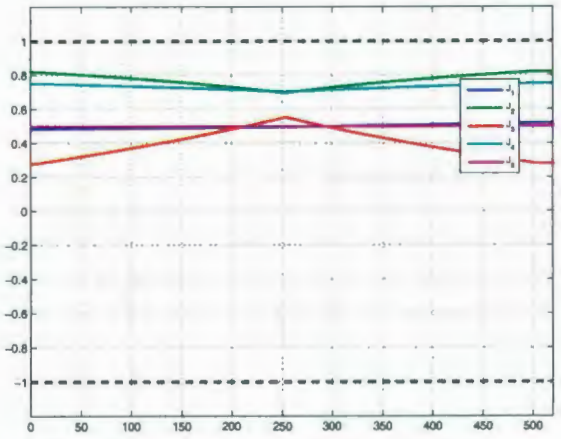
4.15.2: Tracking error $(\mathbf{s}(t) - \mathbf{s}^*(t))$



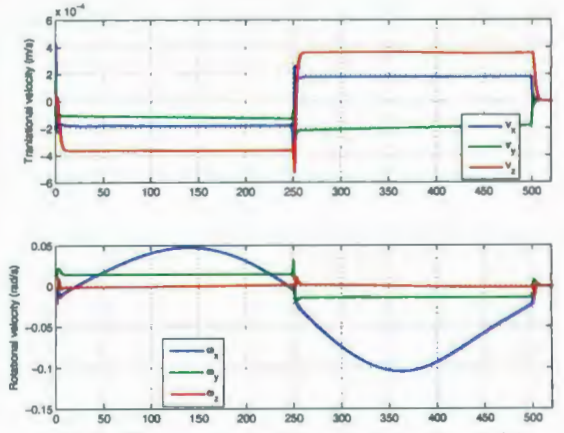
4.15.3: Error $(\mathbf{s}(t) - \mathbf{s}^*)$



4.15.4: Error trajectory $(\mathbf{s}(t) - \mathbf{s}^*)$ in image space (pixels)

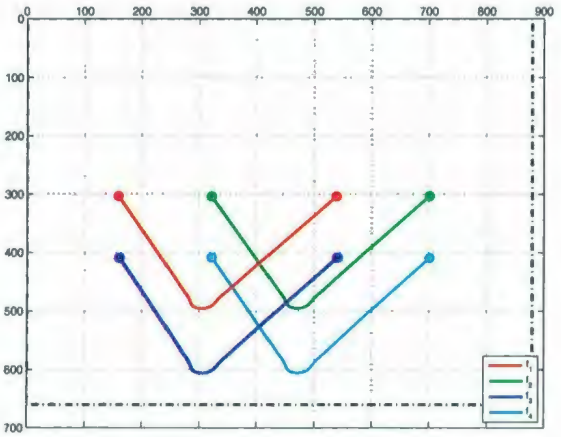


4.15.5: Normalized trajectories in joint space

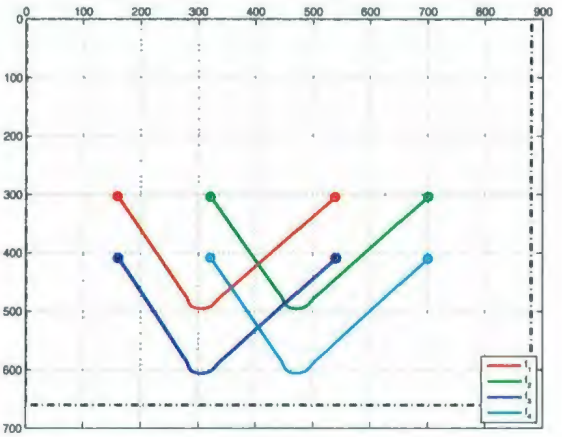


4.15.6: Camera velocities (m/s and rad/s)

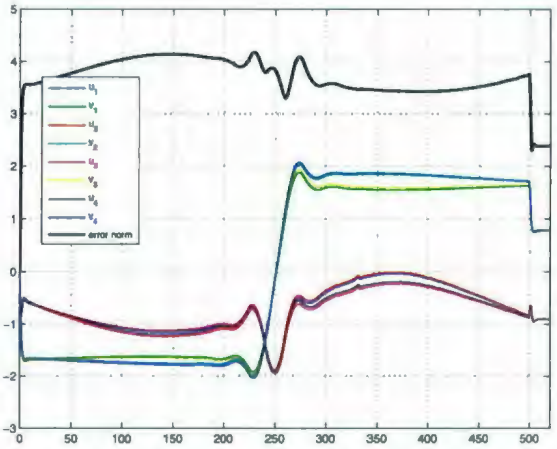
Figure 4.15: Visual servoing using the designed path in Figure 4.14



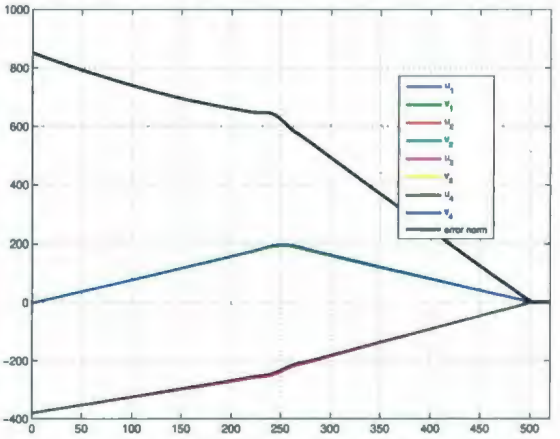
4.16.1: Spline-smoothed image plane trajectories $\mathbf{s}^*(t)$



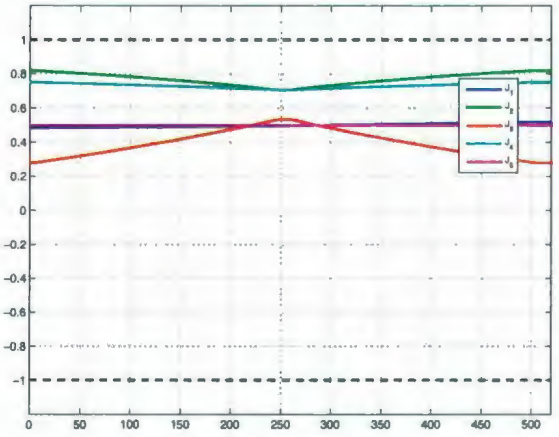
4.16.2: Tracked image plane trajectories $\mathbf{s}(t)$



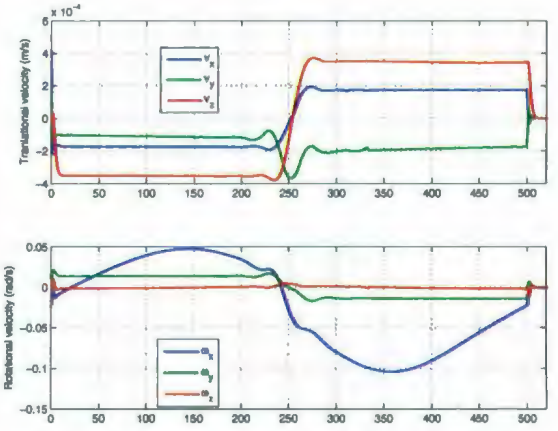
4.16.3: Tracking error $(\mathbf{s}(t) - \mathbf{s}^*(t))$



4.16.4: Error $(\mathbf{s}(t) - \mathbf{s}^*)$

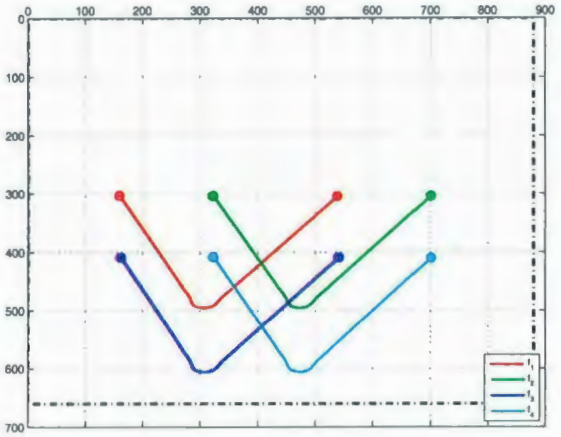


4.16.5: Normalized trajectories in joint space

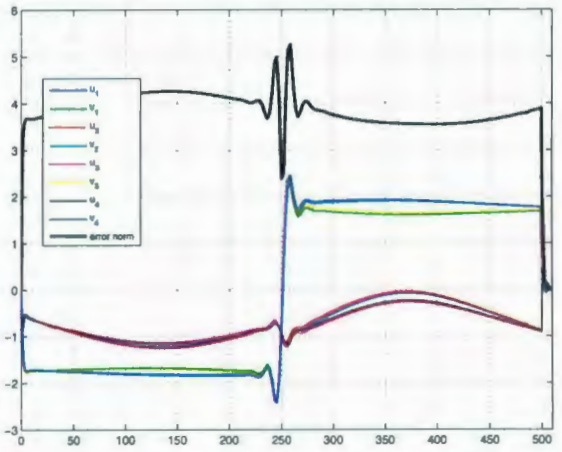


4.16.6: Camera velocities (m/s and rad/s)

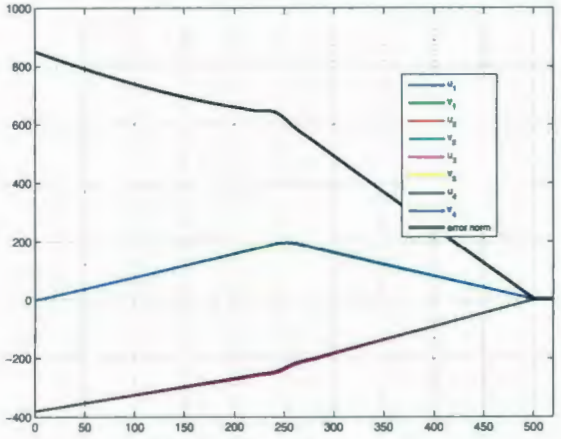
Figure 4.16: Visual servoing using smoothed feature trajectories vs. Figure 4.15



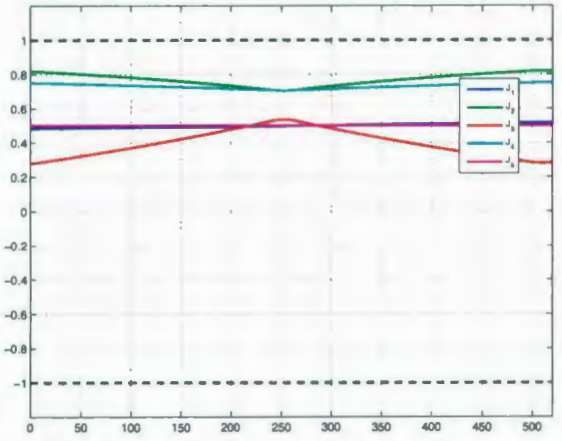
4.17.1: Tracked image plane trajectories $\mathbf{s}(t)$



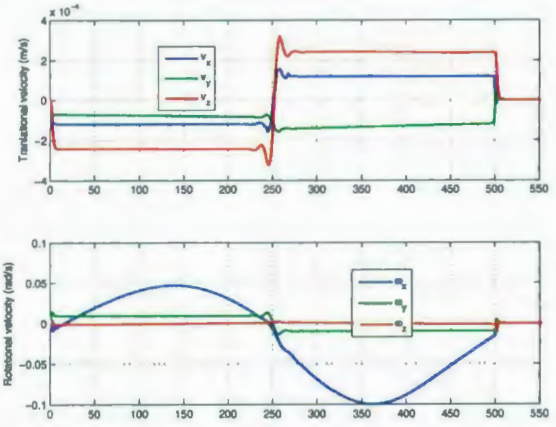
4.17.2: Tracking error $(\mathbf{s}(t) - \mathbf{s}^*(t))$



4.17.3: Error $(\mathbf{s}(t) - \mathbf{s}^*)$



4.17.4: Normalized trajectories in joint space



4.17.5: Camera velocities (m/s and rad/s)

Figure 4.17: Visual servoing using feature trajectories in Figure 4.16.1 with 40% noise

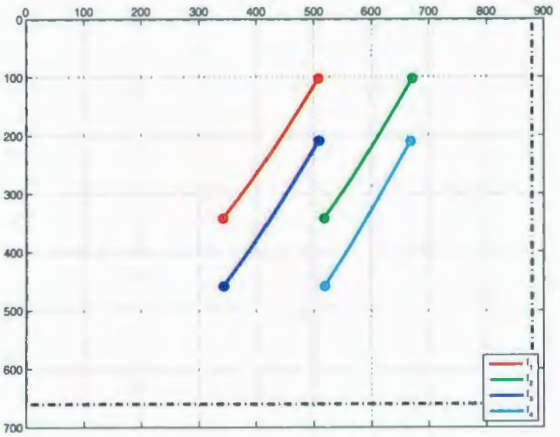
PRM method is known to be probabilistically complete [57]. Several general parameters control the construction and query of the probabilistic roadmap which are application specific. For instance, as the number of milestones N in PRM increases, the probability of finding a path increases and a more optimal path is expected, if any.

Although it is shown that the method is robust to camera calibration errors, estimation of the depth of the target to the origin of the camera frame at the desired pose, d^*, plays an important role in the convergence of the system. Coarse approximations of d^* will cause the biased approximation of $\mathbf{t}^*$ using homography decomposition. It is noteworthy that the approximation of $\mathbf{R}^*$ is exact; therefore the designed path will take the initial image to the desired image with respect to the rotational component. Then a control solution such as the classical visual servo has to compensate for the erroneous translational part which requires visual servo of three degrees of freedom instead of six. A switching controller may be used to take control after the designed path has been followed.

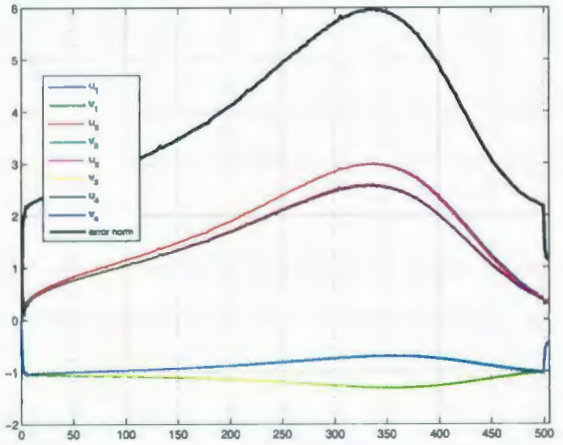
To evaluate the system performance, an error of 20% is introduced in d^*. In this case, the estimated $\mathbf{t}^*$ is biased ($\hat{\mathbf{t}}^* = [477.6210,\ 0,\ 460.3918]$ instead of $\hat{\mathbf{t}}^* = [502.7589,\ 0,\ 484.6230]$). The results of this simulation are shown in Figures 4.18 and 4.19. The tracking control solution will take the initial image to the estimated pose whose results are shown in Figure 4.18; then a pure IBVS controller will regulate the remaining translational error ($\mathbf{t}^* - \hat{\mathbf{t}}^*$) in image space (see details in Figure 4.19). In other words, the tracking control solution regulates the rotational component of the workspace error. Since the magnitude of the computed control law of the controllers is not compatible, the response of the controllers are depicted separately. As shown in Figure 4.19.5, the angular velocity of the camera frame is zero with pure IBVS controller due to the complete regulation of the rotational error by the tracking

controller.

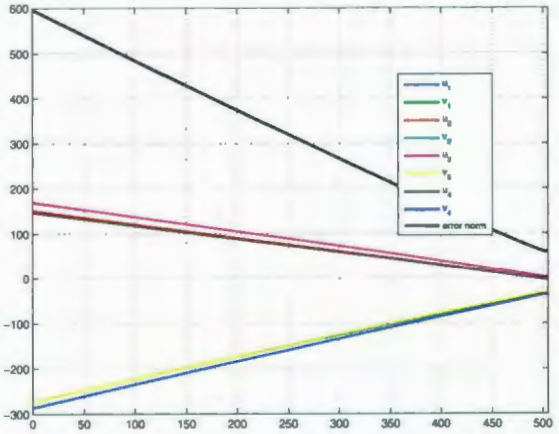
It is obvious that the error in image space is not regulated completely due to the coarse estimation of d^* (see Figures 4.18.3, 4.18.4, 4.19.2 and 4.19.3) although the trajectory tracker has accomplished the tracking with less than 1 pixel final error.



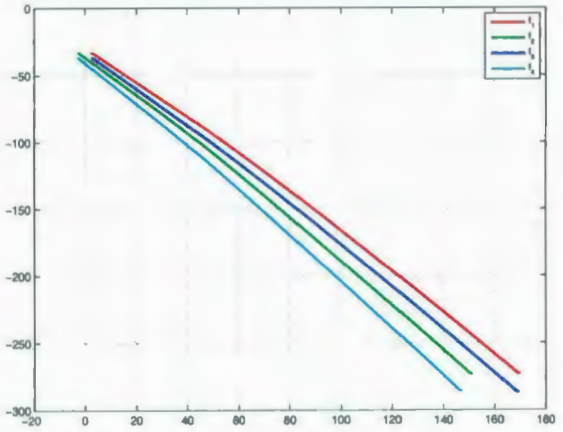
4.18.1: Tracked image plane trajectories $\mathbf{s}(t)$



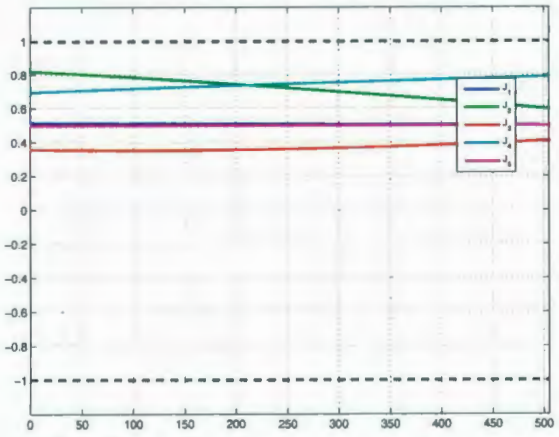
4.18.2: Tracking error $(\mathbf{s}(t) - \mathbf{s}^*(t))$



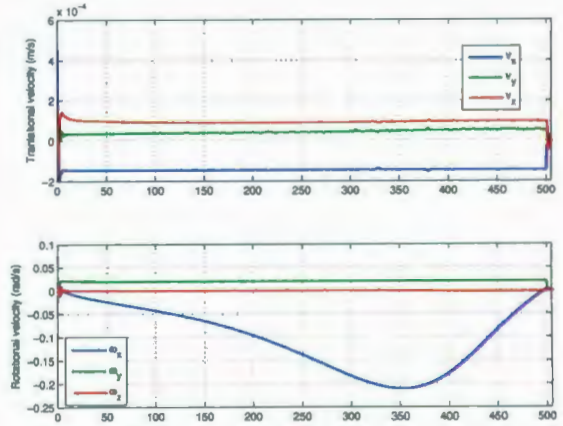
4.18.3: Error $(\mathbf{s}(t) - \mathbf{s}^*)$



4.18.4: Error trajectory $(\mathbf{s}(t) - \mathbf{s}^*)$ in image space (pixels)

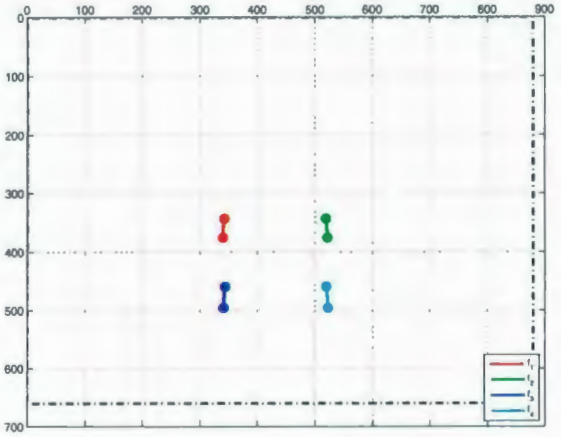


4.18.5: Normalized trajectories in joint space

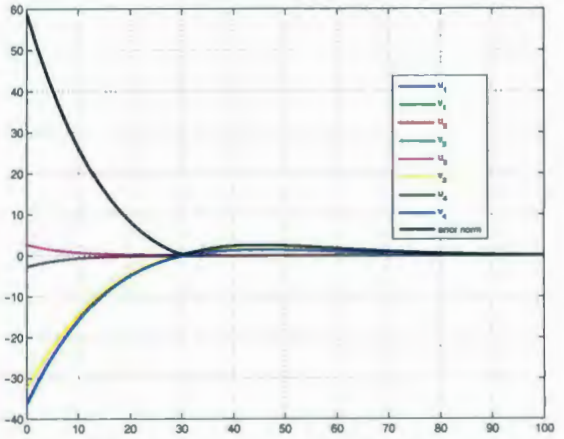


4.18.6: Camera velocities (m/s and rad/s)

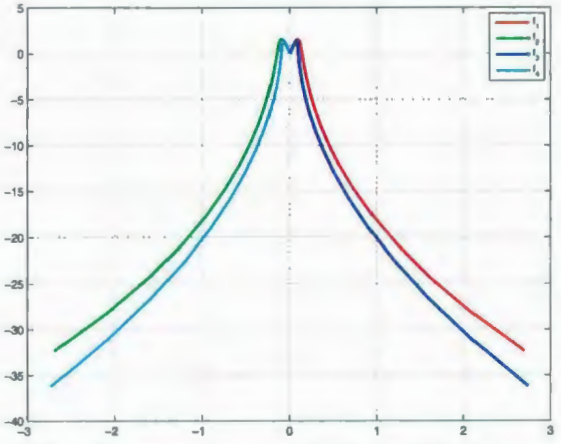
Figure 4.18: Tracking of the designed path using erroneous d^*



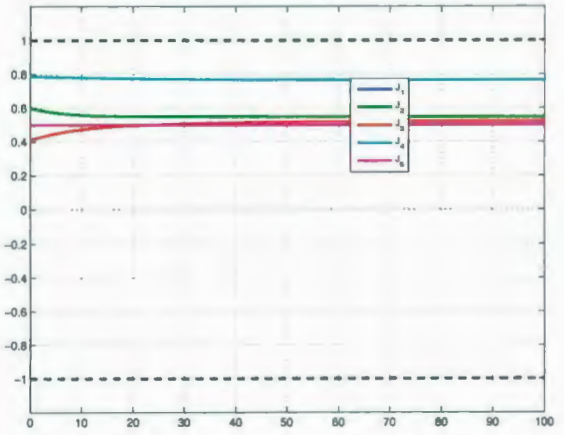
4.19.1: Target trajectories in image plane $\mathbf{s}(t)$



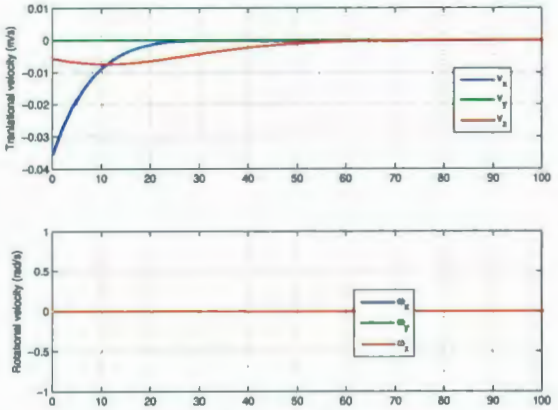
4.19.2: Error trajectories $(\mathbf{s}(t) - \mathbf{s}^*)$



4.19.3: Error trajectories $(\mathbf{s}(t) - \mathbf{s}^*(t))$ (pixels)



4.19.4: Normalized trajectories in joint space



4.19.5: Camera velocities (m/s and rad/s)

Figure 4.19: 3 degree-of-freedom IBVS for regulation of translational error

Chapter 5

Conclusion

The main goal of this thesis is to investigate a path planning scheme for visual servoing. The designed path takes the initial pose to the desired pose such that the induced camera and image feature trajectories are collision-free and such that the trajectories are not occluded with any obstacles and the features are retained in FOV. The visibility constraint and visual occlusion avoidance constraint are developed which are integrated with a probabilistic roadmap to perform the visual path planning. The visibility constraint ensures that the target features remain in FOV and the occlusion avoidance constraint guarantees that the generated path is occlusion-free. Joints limit constraint and obstacle avoidance are implemented in the preprocessing phase of PRM path planning. Probabilistic roadmap uses a workspace metric in the query of the paths and a C-space based metric for the preprocessing.

Once the relation between the known initial and desired poses are computed using the projective homography between initial and desired images, a path is designed that connects the known initial and computed desired poses in a PRM framework. It is noteworthy that the proposed method does not require a model of the target. Since the generated path is a discrete set of intermediate poses in the workspace and

points in the image plane, cubic spline interpolation is used to find a continuous and differentiable trajectory. A visual servo control solution then tracks the generated path so that the error remains small and thus the local stability of the controller holds.

A series of simulations and off-line path planning is performed to validate the method. It is demonstrated that path planning makes visual servoing more robust by performing obstacle avoidance and occlusion detection. It is observed that the method has minimal sensitivity to the calibration errors, although there is an unpleasant dependence on the estimation of the depth of the target at the desired pose.

5.1 Future Work

During this research the following areas were identified to have possible future research potential:

Visual Path Planning in a Dynamic Environment The dynamic planning is path planning in dynamic environment. In the research of this thesis, the target and obstacle are static. In case of dynamic planning, the implications of the moving obstacles and target in the designed constraints and the visual servo controller are necessary to be studied.

Optimal Path Planning The current research results provide a path planning scheme in the workspace; thus the generated paths are optimal in the workspace. One possible extension is to integrate joint space and image space in probabilistic roadmap with some performance index to generate paths so that workspace and image trajectories are optimal. In this case, the interaction of the planning components of the integrated planner requires an investigation.

Eye-to-hand Visual Path Planning This research investigates the eye-in-hand

path planning. Study of a path planning scheme for eye-to-hand visual requires a slight modification to the PRM structure and constraints. Similarly, in this case every milestone corresponds to a point in the feature space. The visual servo controller will be the same but a negative sign will appear in the control law.

Appendix A

K-nearest neighbor query

Kd-tree is an *orthogonal range query* technique used to perform an efficient search in terms of space and time in a d-dimensional data structure to find k nearest neighbors. A d-dimensional kd-tree takes as input a set of N points in d dimensions and constructs a binary tree that decomposes space into cells along the d dimensions such that no cell contains too many points.

In the 5-dimensional case, each point is characterized with five values. In order to construct the kd-tree, at the root the set V of N points is split with a vertical hyperplane $\mathcal{L}$ perpendicular to the first dimension x_1 into two subsets of roughly equal size. The splitting hyperplane is stored at the root. V_{left}, the subset of points to the left or on the splitting hyperplane, is stored in the left subtree, and V_{right}, the subset to the right of it, is stored in the right subtree. At the left child of the root, V_{left} is split into two subsets with another hyperplane which is this time perpendicular to the second dimension x_2; the points to the left or on it are stored in the left subtree of the left child, and the points to the right are stored in the right subtree. The left child itself stores the splitting hyperplane. Similarly, at the right child of the root, the set V_{right} is split with the hyperplane perpendicular to x_2 into two subsets, which

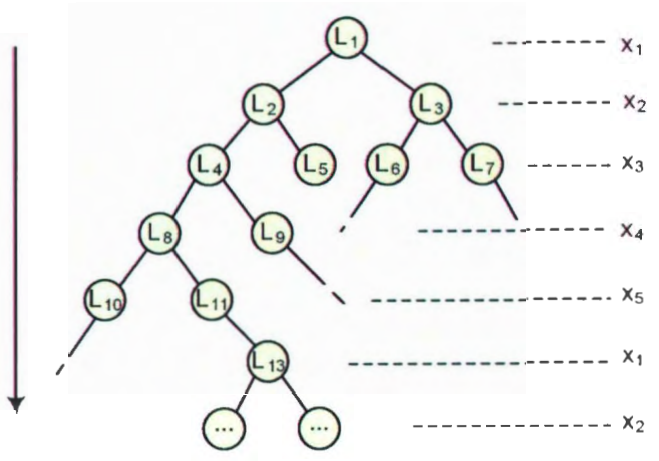


Figure A.1: An example of a binary tree for a kd-tree

are stored in the left and right subtree of the right child. At the grandchildren of the root, the points are split with the hyperplane perpendicular to x_3 into two subsets at each of the four children of the root. The splitting is performed continuously with hyperplanes of x_4 then x_5. This process cycles using hyperplanes of x_1 to x_5, until a point remains on the branches. Figure A.1 illustrates how the binary tree looks like. A tree like this is called a kd-tree.

The kd-tree is constructed with the recursive procedure outlined below. This procedure has three parameters: a set of points V, the current depth *depth* and the number of dimensions d. The first parameter is the set for which the kd-tree has to be built; initially this is the set V. The second parameter is depth of recursion or, in other words, the depth of the root of the subtree that the recursive call constructs. The depth parameter is zero at the first call. The third parameter is the number of dimension of the configuration space. The procedure returns the root of the kd-tree. Nodes in a d-dimensional kd-tree correspond to regions of the d-dimensional configuration space.

A kd-tree for a set of N points uses $\mathcal{O}(N)$ storage and can be constructed in

Algorithm 6 BUILDKDTREE($V, depth, d$)

Require: A set of points V, the current depth *depth* and the number of dimensions d.

Ensure: The root of a kd-tree storing V.

```
1: if PV contains only one point then
2:     return a leaf storing this point
3: else
4:     for i = 1 to d do
5:         Split P into two subsets with a hyperplane ℒ through the median x_i-
           coordinate of the points in V.
6:         V_1 ← set of points to the negative of ℒ or on ℒ.
7:         V_2 ← set of points to the positive of ℒ.
8:         ν_left ← BUILDKDTREE(V_1, depth + 1)
9:         ν_right ← BUILDKDTREE(V_2, depth + 1)
10:        Create a node ν storing ℒ, make ν_left the left child of ν, and make ν_right
           the right child of ν.
11:        return ν
12:    end for
13: end if
```

$\mathcal{O}(N \log N)$ time, considering that d is constant. Nodes in a d-dimensional kd-tree correspond to regions. The query algorithm visits those nodes whose regions are properly intersected by the query range, and traverses subtrees (to report the points stored in the leaves) that are rooted at nodes whose region is fully contained in the query range. It can be shown that the query time is bounded by $\mathcal{O}(N^{1-1/d}+k)$ where k is the number of the reported neighbors [82]. The query algorithm is explained in the following, which takes the root of the kd-tree and a range D and $lc(\nu)$ and $rc(\nu)$ denote the left and right child of a node ν, respectively.

Finding the nearest neighbor NN_q to a given target point q not in the tree, is based on the ability to discard large portions of the tree by performing a simple test. The tree is searched in a depth-first fashion and at each stage it makes an approximation to the nearest distance. When the algorithm decides that there cannot possibly be a closer point it terminates, giving the nearest neighbor.

Algorithm 7 SEARCHKDTREE(ν, D)

Require: The root of (a subtree of) a kd-tree, and a range D.
Ensure: All points at leaves below ν that lie in the range.

```
1: if ν is a leaf then
2:     return the point stored at ν if it lies in D
3: else
4:     if region spanned by lc(ν) lies fully in D then
5:         return all the leafs stored in lc(ν)
6:     else if region spanned by lc(ν) intersects D then
7:         SEARCHKDTREE(lc(ν), D)
8:     end if
9: else
10:    if region spanned by rc(ν) lies fully in D then
11:        return all the leafs stored in rc(ν)
12:    else if region spanned by lc(ν) intersects D then
13:        SEARCHKDTREE(rc(ν), D)
14:    end if
15: end if
```

Bibliography

[1] S. A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Trans. Robotics and Automation*, 12(5):651 670, Oct 1996.

[2] F. Chaumette and S. Hutchinson. Visual servo control, part II: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109 118, mar 2007.

[3] Bernard Espiau. Effect of camera calibration errors on visual servoing in robotics. In *The 3rd International Symposium on Experimental Robotics III*, pages 182 192, London, UK, 1994. Springer-Verlag.

[4] Ed K. Hashimoto. Visual servoing: Real time control of robot manipulators based on visual sensory feedback. *of World Scientific Series in Robotics and Automated Systems*, Singapore: World Scientific, 1993,.

[5] Lee Weiss, Arthur C Sanderson, and C. P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal on Robotics and Automation*, RA-3, October 1987.

[6] C. C. W. Hulls W. J. Wilson and G. S. Bell. Relative end-effector control using cartesian position-based visual servoing. *IEEE Trans. Robot. Automat.*, vol. 12,:pp. 684 696, Oct. 1996.

[7] Daniel F. Dementhon and Larry S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, 1995.

[8] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. In *Selected Papers from the Workshop on Geometric Reasoning for Perception and Action*, pages 106–136, London, UK, 1993. Springer-Verlag.

[9] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In *The Confluence of Vision and Control. ser. LNCIS, D. Kriegman, G. Hager, and A. Morse, Eds.*, pages 66 78. New York: Springer Verlag, 1998.

[10] P. I. Corke and S. A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Trans. on Robotics and Automation*, 17(4):507 515, Aug 2001.

[11] Youcef Mezouar and François Chaumette. Path planning for robust image-based control. *IEEE Trans. on Robotics and Automation*, 18:534–549, 2002.

[12] Jian Chen, Darren M. Dawson, Warren E. Dixon, and Vilas K. Chitrakaran. Navigation function-based visual servo control. *Automatica*, 43(7):1165 1177, 2007.

[13] J. Craig. *Introduction to Robotics: Mechanics and Control.* Addison-Wesley Publishing Company, E.U.A., segunda edition, 1989.

[14] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint.* MIT Press, Cambridge, MA, USA, 1993.

[15] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3:485–508, 1988.

[16] Ezio Malis and François Chaumette. $2\frac{1}{2}$D visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *Int. J. Comput. Vision*, 37(1):79 97, 2000.

[17] Boubakeur Boufama and Roger Mohr. Epipole and fundamental matrix estimation using the virtual parallax property. In *Proceedings of the 5th International Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 1030 1036, June 1995.

[18] Richard I. Hartley. In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580 593, 1997.

[19] E. Marchand, F. Chaumette, and A. Rizzo. Using the task function approach to avoid robot joint limits and kinematic singularities in visual servoing. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'96*, volume 3, pages 1083 -1090, Osaka, Japan, November 1996.

[20] Bradley J. Nelson and Pradeep K. Khosla. Increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. *Journal of Robotics Research*, 14:418 423, 1995.

[21] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. on Robotics and Automation*, 8(3):313 326, June 1992.

[22] Khosla P. P. Papanikolopoulos, N. P. and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control. *IEEE Trans. on Robotics and Automation*, 9(1):14 35, 1993.

[23] De Mathelin M. Abba G. Gangloff, J.A. 6 DOF high speed dynamic visual servoing using GPC controllers. In *Proceedings. IEEE International Conference on Robotics and Automation*, pages 2008 2013, May 1998.

[24] Tae Won Kim Il Hong Suh. Fuzzy membership function based neural networks with applicationsto the visual servoing of robot manipulators. *IEEE Transactions on Fuzzy Systems*, Volume: 2, Issue: 3:203 220, Aug 1994.

[25] Fuentes O. Nelson R. Jägersand, M. Experimental evaluation of uncalibrated visual servoing forprecision manipulation. In *Proceedings. IEEE International Conference on Robotics and Automation*, pages 20–25, Apr. 1997.

[26] E. Marchand, A.I. Comport, and F. Chaumette. Improvements in robust 2D visual servoing. In *IEEE Int. Conf. on Robotics and Automation, ICRA'04*, volume 1, pages 745 750, New Orleans, Louisiana, April 2004.

[27] F. Chaumette and S. Hutchinson. Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82 90, December 2006.

[28] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *ICRA*, pages 1843 1848. IEEE, 2004.

[29] P. I. Corke. *Visual Control of Robots: High-Performance visual servoing.* Research Studies Press (John Wiley), 1996.

[30] Ezio Malis, François Chaumette, and Sylvie Boudet. 2D $\frac{1}{2}$ visual servoing. *IEEE Trans. Robotics and Automation*, Vol. 15, No. 2, April 1999.

[31] W.E.; Behal A. Jian Chen; Dawson, D.M.; Dixon. Adaptive homography-based visual servo tracking for a fixed camera configuration with a camera-in-hand extension. *IEEE Transactions on Control Systems Technology*, Volume 13, Issue 5,:Page(s): 814 – 825, Sept. 2005.

[32] Jian Chen and D. M. Dawson. UAV tracking with a monocular camera. In *Decision and Control, 2006 45th IEEE Conference on*, pages 3873–3878, 2006.

[33] Shanghai Daqing Yi Jian Wu Ping Jiang Tongji Univ. Iterative learning control for visual servoing with unknown homography matrix. In *IEEE International Conference on Control and Automation, 2007. ICCA 2007*, pages 2791 2796, May 30 2007.

[34] W.E.; Dawson D.M.; McIntire M. Chen, J.; Dixon. Homography-based visual servo tracking control of a wheeled mobile robot. In *Proceedings. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003)*, pages 1814–1819, Vol.2, 17-31 Oct. 2003.

[35] S.A. Corke, P.I.; Hutchinson. A new hybrid image-based visual servo control scheme. In *Proceedings of the 39th IEEE Conference on Decision and Control, 2000*, pages 2521–2526 , Volume 3, 2000.

[36] Ville Kyrki. New shortest-path approaches to visual servoing. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 349–355, 2004.

[37] S.A. Gans, N.R.; Hutchinson. An asymptotically stable switched system visual controller for eye in hand robots. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, pages 735 742 Vol.1, Oct. 2003.

[38] F. Malis, E.; Chaumette. Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods. *IEEE Transactions on Robotics and Automation*, Volume 18, Issue 2:176 - 186, April 2002.

[39] Guillaume Morel, Thomas Liebezeit, Jérôme Szewczyk, Sylvie Boudet, and Jacques Pot. Explicit incorporation of 2D constraints in vision based control of robot manipulators. In *The Sixth International Symposium on Experimental Robotics VI*, pages 99 108, London, UK, 2000. Springer-Verlag.

[40] François Chaumette and Éric March. A redundancy-based iterative scheme for avoiding joint limits: Application to visual servoing. *IEEE Trans. on Robotics and Automation*, 17:1720–1725, 2001.

[41] D. Littau R. Singh, R. M. Voyle and N. P. Papanikolopoulos. Alignment of an eye-in-hand system to real objects using virtual images. In *Proc. Workshop Robust Vision. Vision-based Control of Motion, IEEE Int. Conf. Robotics Automation*, May 1998.

[42] Andreas Ruf and Radu Horaud. Visual trajectories from uncalibrated stereo. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 83 91, 1997.

[43] Youcef Mezouar and Francois Chaumette. Design and tracking of deserable trajectories in the image space by integrating mecahnical and visibility constraints. In *in International Conference on Robotics and Automation, Seoul, Korea, 2001, IEEE*, pages 731 6. IEEE, 2001.

[44] Youcef Mezouar, François Chaumette, and Irisa Inria Rennes. Path planning in image space for robust visual servoing. In *In IEEE Int. Conf. on Robotics and Automation*, pages 2759 2764, 2000.

[45] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[46] Daniel E. Koditschek and Elon Rimon. Robot navigation functions on manifolds with boundary. *Adv. Appl. Math.*, 11(4):412 442, 1990.

[47] E. Rimon and D. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation,*, 8(5):501 - 518, Oct. 1992.

[48] Noah J. Cowan and Daniel E. Koditschek. Planar image based visual servoing as a navigation problem. In *International Conference on Robotics and Automation*, pages 611 617. IEEE, 1999.

[49] Noah J. Cowan, Gabriel A. D, Lopes Daniel, and E. Koditschek. Rigid body visual servoing using navigation functions. In *Conference on Decision and Control*, pages 3920 3926. IEEE, 2000.

[50] Noah J. Cowan, Joel D. Weingarten, and Daniel E. Koditschek. Empirical validation of a new visual servoing strategy. In *Conf. Control Applications, Mexico City, Mexico*, Sept. 2001.

[51] Elon Rimon. *Exact robot navigation using artificial potential functions.* PhD thesis, New Haven, CT, USA, 1990. Adviser-Daniel E. Koditschek.

[52] H. Zhang and J. Ostrowski. Visual motion planning for mobile robots. *IEEE Trans. Robot. Automat.*, Vol. 18:199 208, Apr. 2002.

[53] Noah J. Cowan, Joel D. Weingarten, Student Member, Daniel E. Koditschek, and Senior Member. Visual servoing via navigation functions. *IEEE Trans. on Robotics and Automation*, 18:533, 2002.

[54] Noah Cowan, Omid Shakernia, René Vidal, and Shankar Sastry. Vision-based follow-the-leader. In *In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[55] Guilherme A. S. Pereira, Aveek K. Das, Vijay Kumar, and Mario F. M. Campos. Formation control with configuration space constraints. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, pages 2755 2760 , Vol.3, Oct. 2003.

[56] K. Hosoda, K. Sakamoto, and M. Asada. Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3D reconstruction. In *IROS '95: Proceedings of the International Conference on Intelligent Robots and Systems-Volume 1*, page 29, Washington, DC, USA, 1995. IEEE Computer Society.

[57] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents)*. The MIT Press, June 2005.

[58] Lydia E. Kavraki Petr Vestka, Jean claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566 580, 1996.

[59] Lydia Kavraki and Jean claude Latombe. Randomized preprocessing of configuration space for path planning: Articulated robots. In *IROS '94. Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Sep 1994.

[60] J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[61] S. Caselli and M. Reggiani. ERPP: An experience-based randomized path planner. In *Proceedings IEEE International Conference on Robotics & Automation*, 2000.

[62] James J. Kuner, Jr. Steven, and M. Lavalle. RRT-connect: An efficient approach to single-query path planning. In *In Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995 1001, 2000.

[63] B. Baginski. *Motion planning for manipulators with many degrees of freedomThe BB Method.* PhD thesis, Technische Universität München, 1998.

[64] D. Hsu, R. Kindel, J-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*. A.K. Peters, Wellesley, MA, 2001.

[65] J. Yakey, S. M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(6):951 958, dec 2001.

[66] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 233 246. A.K. Peters, Wellesley, MA, 2001.

[67] Timothy Bretl, Jean-Claude Latombe, and Stephen Rock. Toward autonomous free-climbing robots. In Paolo Dario and Raja Chatila, editors, *ISRR*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 6 15. Springer, 2003.

[68] A. Casal. *Reconfiguration Planning for Modular Self-Reconfigurable Robots*. PhD thesis, Stanford University, Stanford, CA, 2002.

[69] Z. Rus D. Fitch, R. Butler. Reconfiguration planning for heterogeneous self-reconfiguring robots. In *Proceedings. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003).*, pages 2460 2467, Oct. 2003.

[70] Florent Lamiraux and Lydia E. Kavraki. Planning paths for elastic objects under manipulation constraints. *International Journal of Robotics Research*, 20:188 208, 2001.

[71] X. Ji and J. Xiao. Planning motion compliant to complex contact states. *International Journal of Robotics Research*, 20(6):446 465, 2001.

[72] Kavraki L. E. Danner, T. Randomized planning for short inspection paths. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 971 976 vol.2, April 24-28 2000.

[73] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *International Journal of Robotics Research*, 5(3):72 89, 1986.

[74] G. Sánchez and J.-C. Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *International Journal of Robotics Research*, 21(1):5 26, 2002.

[75] Peter J. Leven. *A framework for real-time path planning in changing environments.* PhD thesis, Champaign, IL, USA, 2001. Adviser-Seth Hutchinson.

[76] T. Jaillet, L.; Simeon. A PRM-based motion planner for dynamically changing environments. In *Proceedings. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004).*, pages 1606 1611, Sept. 2004.

[77] Lydia E. Kavraki. Random networks in configuration space for fast path planning. Technical report, Stanford, CA, USA, 1995.

[78] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. *Computer Graphics*, 24(4):327 335, aug 1990.

[79] S. Quinlan. Efficient distance computation between non-convex objects. In *Proc. IEEE Int. Conf. on Robotics and Automation, San Diego, CA*, pages 117 123, 1994.

[80] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171 180, New York, NY, USA, 1996. ACM.

[81] E. Malis. *Contributions à la modélisation et à la commande en asservissement visuel.* PhD thesis, Univ. Rennes I, IRISA, France, Nov. 1998.

[82] Mark Overmars Otfried Schwarzkopf Mark de Berg, Marc van Kreveld. *Computational Geometry: Algorithms and Applications.* Springer; 3rd edition, March 2008.

[83] D.P. Dobkin Barber, C. B. and H.T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, pages 469 483, Vol. 22, No. 4, Dec. 1996.

[84] Gildardo Sánchez-Ante and Jean-Claude Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In Ray Jarvis and Alexander Zelinsky, editors, *ISRR*, volume 6 of *Springer Tracts in Advanced Robotics*, pages 403 417. Springer, 2001.

[85] R. Geraerts and M. Overmars. Sampling techniques for probabilistic roadmap planners. In *Proceedings International Conference on Intelligent Autonomous Systems*, 2004.

[86] R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, dec 2002.

[87] Nancy M. Amato, O. B Bayazit, Lucia K. Dale, Christopher Jones, and Daniel Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. Technical report, College Station, TX, USA, 1998.

[88] Roger Penrose. *The Road to Reality: A Complete Guide to the Laws of the Universe*. Vintage, January 2007.

[89] Kumar V. Zefran, M. and C.B. Croke. On the generation of smooth three-dimensional rigid body motions. *IEEE Transactions on Robotics and Automation,*, pages 576 589, Aug 1998.

[90] James J. Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In *In IEEE International Conference on Robotics and Automation*, pages 3993 3998, 2004.

[91] L. Sciavicco, Bruno Siciliano, and B. Sciavicco. *Modelling and Control of Robot Manipulators*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2000.

[92] Begnaud Francis Hildebrand. *Introduction to numerical analysis: 2nd edition*. Dover Publications, Inc., New York, NY, USA, 1987.

[93] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, 2001.

[94] Gregory D. Hager and Kentaro Toyama. X vision: A portable substrate for real-time vision applications. *Computer Vision and Image Understanding*, 69:23 37, 1998.

[95] E. Marchand and F. Chaumette. Features tracking for visual servoing purpose. In D. Kragic and H. Christensen, editors, *Advances in Robot Vision - From Domestic Environments to Medical Applications*, pages 10 20, Sendai, Japan, September 2004.

[96] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87 119, October 1995.