University of Alberta

Uncalibrated Vision-Based Control and Motion Planning of Robotic Arms in Unstructured Environments

by

Azad Shademan

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

To my father, Arsalan-
for teaching me to think critically,
and
to my mother, Shahnaz-
for teaching me to love unconditionally.

Abstract

Many robotic systems are required to operate in unstructured environments. This imposes significant challenges on algorithm design. Particularly, motion control and planning algorithms should be robust to noise and outliers, because uncertainties are inevitable. In addition, independence from scene model and calibration parameters is preferred; otherwise, the tedious model extraction and calibration procedures need to be redone with every change in the environment. The basic problem that this thesis addresses is how to robustly control the motion of a vision-based manipulator and plan occlusion-free paths in unstructured environments.

Vision-based motion control without using calibration or a geometric model is studied in Uncalibrated Visual Servoing (UVS). In this thesis, we adopt a framework based on UVS and contribute to two distinct areas: robust visual servoing and robust randomized path planning. We develop a statistically robust algorithm for UVS, which detects outliers and finds robust estimates of the uncalibrated visual-motor Jacobian, a central matrix in the visual servoing control law. We integrate the robust Jacobian estimation into a real-time feedback control loop and present case studies. To avoid the visual and joint-limit constraints, we propose a robust sampling-based path planning algorithm. The proposed planner fits well within the UVS framework and facilitates occlusion-free paths, despite not knowing the obstacle model.

Finally, our third and last contribution is a novel UVS approach based on extracting the geometry of three images in the form of the trifocal tensor. We experimentally validate this approach and show that the proposed UVS controller handles some of the most challenging degenerate configurations of image-based visual servoing.

Acknowledgements

The PhD program is a journey to treasure island. The treasure is the maturity and experience required to identify research opportunities and open problems. Many individuals have shown me where to find gems along my journey; gems that collectively form my treasure. I am eternally grateful to all.

First and foremost, I would like to extend my sincere gratitude to my PhD advisor, Prof. Martin Jägersand, who spotted my pre-application forms and encouraged that I proceed with the application. He gave me the opportunity to freely work on a set of intriguing open problems in robotics and visual servoing. He has made it possible for me, and other researchers in his lab, to work with some of the most advanced robotic and vision platforms that exist in the world. His guidance has made my research skills mature and my understanding of computer and robot vision more profound.

I am indebted to Amir massoud Farahmand, my friend and colleague, for always prompting me to pay attention to the core research problem and to think more critically. I am thankful for his contributions to this research through discussions and paper co-authorship. I have enjoyed each and every one of our coffee chats in the Student Union Building.

I would like to thank my supervising committee members, Prof. Hong Zhang and Prof. Mahdi Tavakoli, and other members of the PhD Candidacy Exam Committee for their valuable feedback on my PhD proposal and research. I thank Prof. Tavakoli and Prof. Szepesvári for their trust in me and their invitations to present lectures in their robotics courses. They have provided me with invaluable teaching experience.

I would like to thank Simon Leonard and Neil Birkbeck, my friends and alumni lab mates, who helped me get started when I first arrived to the lab. They both did a phenomenal job and shared their experience without hesitation and continued to do so over the years. Special thanks are due to Simon for developing openman, the open source C++ object-oriented library he developed for the control of the WAM Arm, and to Neil for his trajectory generation plug-in. I also thank my friend and alumnus lab mate, Chris Parker, for making the camera mount, as well as for the many stimulating conversations we had on Canadian history, culture, travel, and food. Many thanks are due to the other members and alumni of the robotics and computer vision lab, Alejandro, Camilo, Romeo, Karteek, Adam, David, Robert, Kiana, and Dana, for their help and for making these years very memorable. I would also like to thank all the hard working staff of the Computing Science Department, for their punctuality and, in particular, our past graduate program director, Edith Drummond, for her additional care.

My utmost gratitude goes to my wife Mina, my parents Shahnaz and Arsalan, and my sister Baharak, for their love, care, patience, and moral support throughout my journey. I am the luckiest person on earth to have such a wonderful family. This dissertation is dedicated to my parents as a token of my appreciation for the sacrifices they have made in raising me and for their unconditional love.

Finally, I would also like to acknowledge the support of Alberta Innovates- Future Technologies (formerly known as iCORE and Alberta Ingenuity Fund) and Alberta Advanced Education & Technology and thank them for their scholarships that has made this research possible. I also thank the University of Alberta and the Department of Computing Science for the Provost Doctoral Entrance Award and multiple teaching assistantships, and my advisor, for research assistantship support.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

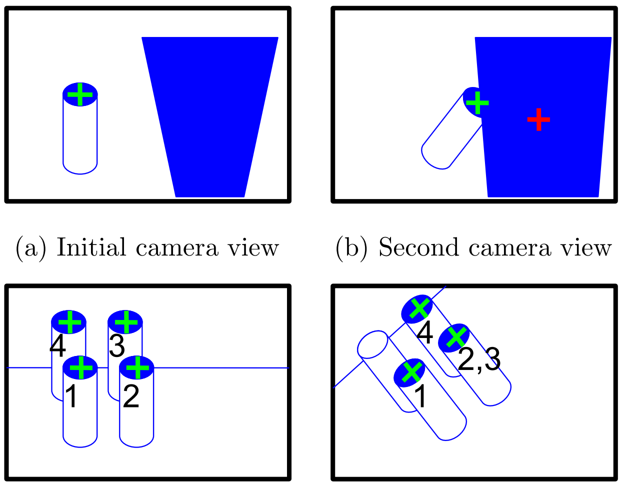
Autonomous and semi-autonomous robots have a broad range of applications such as planetary space exploration, surgical robotics, rehabilitation, and household applications. A common attribute of such applications is that the robot needs to operate in unstructured environments rather than structured industrial workcells or controlled laboratory settings. Motion control and trajectory planning for robots in unstructured environments face significant challenges due to uncertainties in environment modeling, sensing modalities, and robot actuation. This thesis attempts to solve a subset of these challenges.

Robots working in unstructured environments do not have full scene models *a priori*. They either need to reconstruct the scene for motion control and planning or use model-free algorithms. It is possible to reconstruct the scene model from sensory information if sensors are calibrated, but accurate calibration of the sensor is required for model-based control and planning. In addition, the resource-intensive reconstruction procedure needs to run continuously to capture any changes to the model. This scheme also assumes knowledge of the robot kinematic and dynamic parameters obtained from robot calibration. Robot calibration should not be assumed, because the robot should have the ability to interact with the environment and manipulate unknown objects that change the kinematics and dynamics of the robot during operation. *Therefore, it is desirable to have model-free algorithms and control architectures that are free of both sensor calibration and robot calibration.*

In unstructured settings, sensory input is often corrupted by measurement noise and outliers. Uncertainties in the motor readings could exist as well. Under the unstructured-environment assumption, robots need algorithms that are *robust* to outliers and other uncertainties in the input-output sensory-motor space.

Visual sensing at frame rate is a key element in vision-based applications. Nonetheless, visual sensing errors are inevitable in unstructured environments for several reasons. The main reasons are attributed to

- Using non-descriptive visual features that are easy to compute and meet the real-time processing requirements, and
- Fast robot motions, under which visual tracking fails. The latter is more typical when the camera is rigidly attached to a moving robot arm (the eye-



(a) Initial camera view (b) Second camera view

(c) Initial camera view (d) Second camera view

Figure 1.1: Illustration of visual sensing outliers. (a) A target feature on a cylinder is marked. An obstacle with similar colour is also seen in the initial camera view. (b) Occlusions due to similarities between the target object and the obstacle object occur, which leads to erroneous estimation of the image features. (c) Four similar target features are seen in the initial camera view.(d) Fast camera motion results in erroneous image feature correspondences in the next view. Vision-based control with such outliers is subject to failure.

in-hand configuration).

Two typical examples of visual sensing errors for a moving camera are depicted in Figure 1.1. The visual feature of interest here is the center coordinate of the cylinder's top surface. In Figure 1.1 (top), the feature is occluded by an object with similar colour properties. A colour-based visual tracking algorithm, such as the Continuously Adaptive Mean Shift (CAMSHIFT) visual tracker [1], returns a large visual sensing error in this case. In Figure 1.1 (bottom), there is an error in feature correspondences between the left and right images due to large inter-frame motion: features numbered 3 and 4 are associated with 2 and 3, respectively. For an eye-in-hand camera configuration, large inter-frame motions are typical, because a fast arm motion translates to a very fast camera motion. This type of error is called an *outlier* to the underlying model, because the feature does not fit to the model it belongs to. A main focus of this thesis is to design image-based control algorithms that are robust to outliers and suited for unstructured settings.

Path planning is another focus of this work. Some level of *planning* is usually required before taking actions reactively. This thesis contributes to planning for vision-based control in unstructured environments. To avoid physical or sensory constraints, the robot should have the ability to think ahead, plan trajectories, follow the planned trajectories, and replan, if necessary. Many path planning algorithms use a user-specified scene model to plan motion trajectories and replan when an obstacle, which is not in the model, is detected. Some other planning algorithms extract models from the sensory data, given some assumptions about the environment, and then plan according to the extracted model. Both approaches

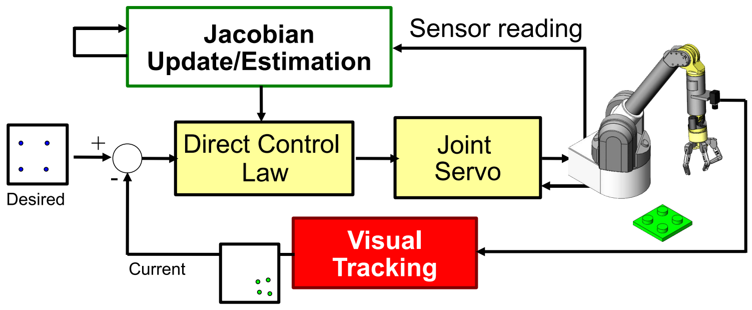


Figure 1.2: A general Uncalibrated Visual Servoing (UVS) block diagram is shown for an eye-in-hand robot arm. The direct control law, compares the current image to a desired image and closes the feedback on the corresponding image error. The direct control law relates this image error to joint commands using a Jacobian matrix, which relates the joint velocities to image velocities. This Jacobian should either be estimated or updated from sensory-motor readings. Details of UVS control are presented in Section 2.4

could be considered model-based planning; however, when models cannot be reliably inferred from data, these approaches are not suitable. A path planning algorithm that directly uses the input-output sensory-motor data, instead of an extracted or user-specified model, would be more useful in many practical scenarios. Clearly, the planning algorithm should also be *robust* to the outliers, because it is based on the raw input-output sensory-motor data, which might be corrupted by noise.

1.2 Overview

1.2.1 Visual servoing in unstructured environments

There is a strong demand to use vision-based robots in everyday environments, because vision adds versatility to a robot. Real-time motion control of robots from visual feedback, *visual servoing*, is distinct from regular robot control in that it uses the (projective) camera coordinates instead of a fixed Euclidean robot base frame. Visual servoing is a well-studied framework for real-time vision-based motion control of robots [2, 3, 4]. Many elementary robotic tasks, such as manipulation, benefit from visual servoing [5]. A formal discussion of the visual servoing problem, along with a comprehensive review of the literature, the available approaches, their strengths and limitations will be presented in Chapter 2. Here, we briefly present where this thesis stands within the broad visual servoing literature.

Visual servoing in everyday environments should be ideally independent from geometric structures, models, and calibration parameters. This class of visual servoing systems are called *uncalibrated* in the literature [6, 7, 8, 9]. The Uncalibrated Visual Servoing (UVS) approach fits the model-free and uncalibrated assumptions pretty well; therefore, this work is based on the UVS framework. The diagram in Figure 1.2 illustrates a simple UVS system for an eye-in-hand robot arm. A visual tracking module processes high-dimensional image data, obtained from the camera, to generate low-dimensional image features. The image features are used to calculate an *image error* that is zero at the desired configuration. The goal is to move the arm to its desired configuration by regulating the image error to zero. The direct control law is an image-based approach that uses a *Jacobian matrix* and the image error to generate a control signal in terms of joint velocities. This Jacobian plays a central role in the UVS approach and will be explained in further detail in Chapter 2, Section 2.4. There are several methods to estimate this Jacobian as explained in Sections 2.4.1, 2.4.2, 2.4.3.

There are many open problems in visual servoing. In this work, we develop methods to address some open problems in UVS system design.

One open problem concerns the handling of noise and outliers in the raw sensory-motor data. A contribution of this thesis is to develop a robust estimation algorithm as mentioned briefly in Section 1.5 and presented in detail in Chapter 4.

Another open problem concerns the choice of the visual features used in UVS. As we mention briefly in Section 1.5, the geometry of multiple views can be used to derive visual features from uncalibrated images. In Chapter 6, we present the details of our contribution. We develop an uncalibrated approach that uses the elements of the *trifocal tensor* as visual features. The trifocal tensor encapsulates the geometry of three views: the initial view, the desired view, and the midway view that evolves from the initial to the desired views.

1.2.2 Planning to avoid constraints

Carrying out complex real-world tasks requires not only the ability to move, but also the ability to plan trajectories to avoid obstacles and physical/visual constraints. Traditionally, planning is done in a Euclidean framework based on the geometric models of the environment. While the traditional method works in known environments, geometric models are not available for most everyday unstructured settings. For example, in vision-based indoor service robotics or outdoor field robotics, the planning algorithm has to rely on visual sensing.

Kazemi *et al.* [10] provide an extensive survey on path planning approaches to visual servoing. Path planning has been shown to address some of the standard problems in visual servoing such as the convergence of visual servo for distant goals [11, 12], or convergence in the presence of visual, physical, and joint constraints [10]. Path planning methods also increase the basin of locality for the image-based control and provide a solution to avoid the field-of-view constraint [10, 12]. While several papers discuss path planning methods for visual servoing, only a handful consider an uncalibrated robot/camera system without a geometric target model [13, 14, 15]. The problem with the simple uncalibrated approaches, such as the early work of Hosoda *et al.* [13], is that they cannot be generalized to other image features, hand/eye configurations, or more realistic unstructured environments. The other methods rely on partial scene reconstruction or homography interpolation [14, 15], which are ill-posed problems if the scene model is not known. What is lacking in the literature is a planning algorithm for uncalibrated visual servoing to avoid constraints without using or reconstructing a model from the visual-motor data corrupted by outliers. For a detailed literature review, the reader is referred to Chapter 3.

Robust path planning for uncalibrated visual servoing without geometric models is an open problem that we address as briefly mentioned in Section 1.5. We develop an algorithm to avoid visual occlusions by mapping obstacles seen in the image to the robot motor coordinates. This maps the obstacles in the motor joint configuration space of the arm, avoiding the need for a mundane calibration of the robot/camera or extraction of the geometric target/obstacle model. The details are presented in Chapter 5.

1.3 Assumptions and Problem Statement

The concept of *robustness* has been used in different senses in the robotics literature. Some authors have used robustness in the general sense of repeatability in complex situations. In that sense, a robust algorithm is one that achieves the same result in different, but similar, experiments. Others have used robustness in a specific technical sense. In the visual servoing literature, robustness has been studied in different contexts. Kragic [16] summarizes robustness issues as perceptual robustness with respect to visual sensing, robustness of trajectory planning with respect to physical constraints, and robustness with respect to system design. She focuses on perceptual robustness and develops an integrated vision-based grasping and manipulation system. Multiple other authors have also worked on perceptual robustness and robust visual tracking [17, 18, 19, 20, 21, 22, 23]. There is a completely different aspect of robustness, which is derived from control-theoretic stability analysis using adaptive robust control [24, 25, 26, 27, 28, 29]. Statistical robustness has also been used for specific visual servoing control laws [30, 31].

While these methods are very interesting and tackle important problems, they do not study robustness in sensor-based planning and control for model-free and uncalibrated vision-based robots. Our most important assumptions are

- Sensors are not calibrated and sensor information is not used to construct a metric model of the world;
- The robot kinematics could change after interacting with the scene; therefore, we do not use kinematic parameters;
- A user specifies a target object (to be manipulated) and an obstacle object (to be avoided) in the sensor space through a user interface.

Under these assumptions, the problem of uncalibrated visual servoing and planning to avoid physical and/or visual constraints becomes very challenging.

The basic problem that this thesis addresses is statistical robustness to visual-motor outliers in control and planning within the context of data-driven model-free visual servoing in unstructured environments. Specifically, we address robust uncalibrated visual servoing and robust sampling-based planning for uncalibrated visual servoing to avoid occlusion, field-of-view, and joint limit constraints. We also develop a novel uncalibrated visual servoing scheme based on the projective geometry of three views, where we estimate the trifocal tensor and then use its elements as visual features to control the motion of a robot arm.

1.4 Thesis Outline

The nature of this research is interdisciplinary and spans many areas in computing science and engineering including computational vision, vision-based control theory, robust statistics, and path planning. In the remainder of this section, we present a brief overview of the thesis framework and outline the open problems in visual servoing, path planning, and task specification. We mention our proposed and anticipated contributions in each area. This organization shall highlight the most important points of the thesis immediately. The rest of this thesis is organized in two parts.

Part I includes the comprehensive background in two distinct subareas in sensor-based robotics: vision-based motion control of robots (visual servoing) in Chapter 2 and path planning for visual servoing in Chapter 3. Each subarea is reviewed in a designated chapter to formulate the basic problem and provide a comprehensive related work, which leads to open problems that we address in the subarea.

In Part II, we present our theoretical chapters. In Chapter 4, we present the robust visual-motor Jacobian estimation algorithm and the robust uncalibrated visual servoing system that utilized the robust Jacobian. In Chapter 5, we present the sampling-based planning algorithm to avoid occlusion, field-of-view constraint, and joint-limit constraint for uncalibrated visual servoing. In Chapter 6, we present the uncalibrated visual servoing algorithm that uses 3D computer vision and the geometry of three-views (the trifocal tensor).

In Part III, we present the experiments and the evaluation of our presentation in Part II. In Chapter 7, we present experiments related to both Chapters 4 and 5, since some experiments are relevant to both chapters. In Chpater 8, we present experiments related to Chapter 6. The diagram in Figure 1.3 shows the relation between chapters in this thesis. This diagram can be used a guide on how to follow the presentation. Next, a summary of the contributions is presented.

1.5 Contributions

The contributions of this thesis are three fold: (1) An algorithm to detect outliers and estimate the visual-motor Jacobian for a statistically-robust UVS system. (2) A new sampling-based planning algorithm for the statistically-robust UVS system. (3) A new set of image measurements derived from the multiple-view geometry to be used in the feedback loop of an UVS system.

Robust Uncalibrated Visual Servoing

We present a statistically robust Jacobian estimation algorithm in Chapter 4. This algorithm detects the visual-motor outliers, which do not belong to the underlying visual-motor model, and de-weighs their contribution to the visual-motor Jacobian estimation. The details of what constitutes a visual-motor outlier and the relevance of the visual-motor Jacobian in the UVS control law can be found in Chapter 2. The proposed algorithm does not require the 3D geometric model of the target or robot/camrea calibration and is suitable to be used with an UVS system. We have validated our algorithm through experiments (see Chapter 7).

The contributions are twofold:

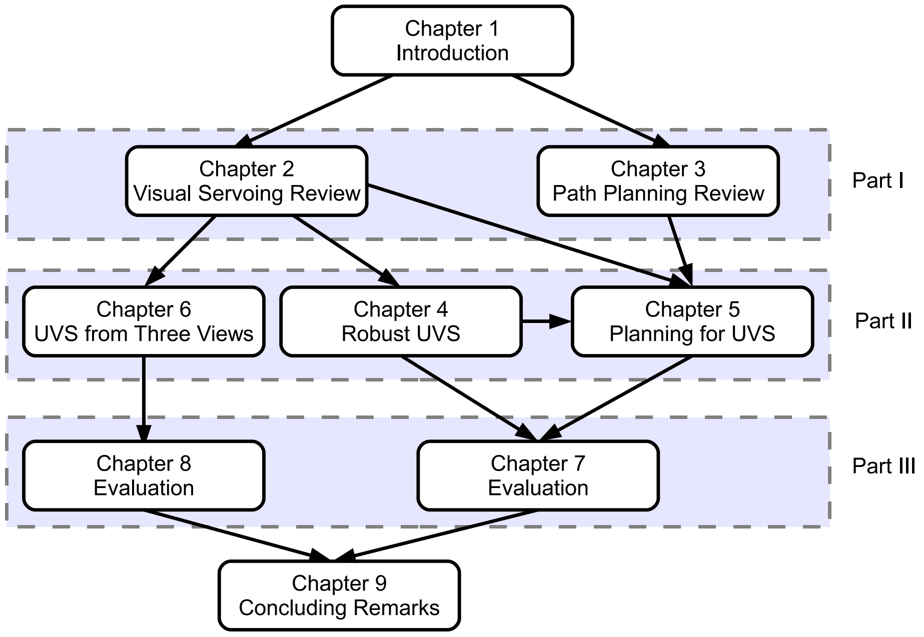


Figure 1.3: Dissertation outline.

1. We develop an algorithm based on robust M-estimators to numerically estimate the visual-motor Jacobian from raw visual-motor data. In our proposed method, the outliers that are due to different visual tracking errors are statistically rejected.

2. We present a control framework that uses the robust Jacobian. The robust Jacobian estimation algorithm provides information about the can be also used to label outliers and determine if a feature point is an outlier. We present a method to recover an outlier query, *i.e.,* estimate the correct value of the outlier features to use this estimate in the closed-loop control. The procedure is completely based on sensed values, not *a priori* models.

The block diagram of our proposed system is shown in Figure 1.4.

Sampling-Based Planning for Uncalibrated Visual Servoing

The Rapidly-exploring Random Tree (RRT) planner [32] is a sampling-based planning algorithm that provides a global solution, if one exists. RRT-based planners have been used in eye-in-hand 6 DOF image-based visual servoing in calibrated settings recently by Kazemi *et al.* [33]. Although the approach of Kazemi *et al.* is appealing, it assumes known geometric model of the object and known camera intrinsic parameters. Therefore, their approach does not work under the assumptions of our problem (Section 1.3), where known geometric model of scene objects are not available *a priori* or extracted.

Nonetheless, since the geometric models of the object or the environment are not used, we develop a special variant of the RRT planner, the UvsBiRRT algorithm,

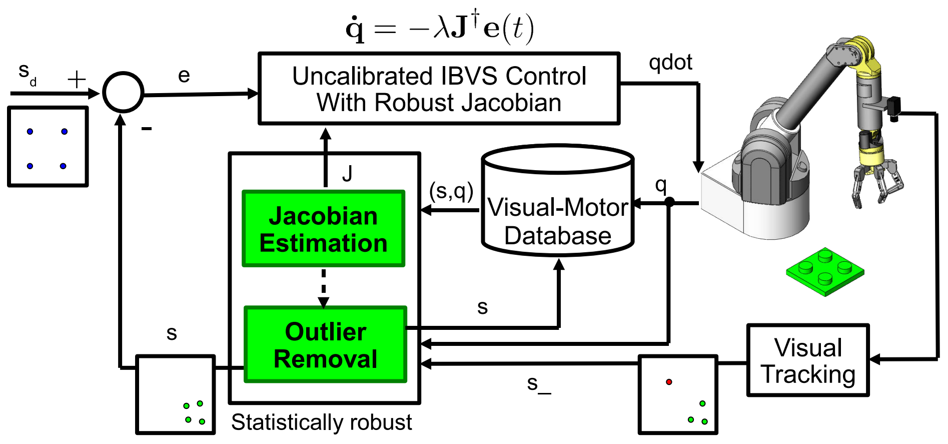


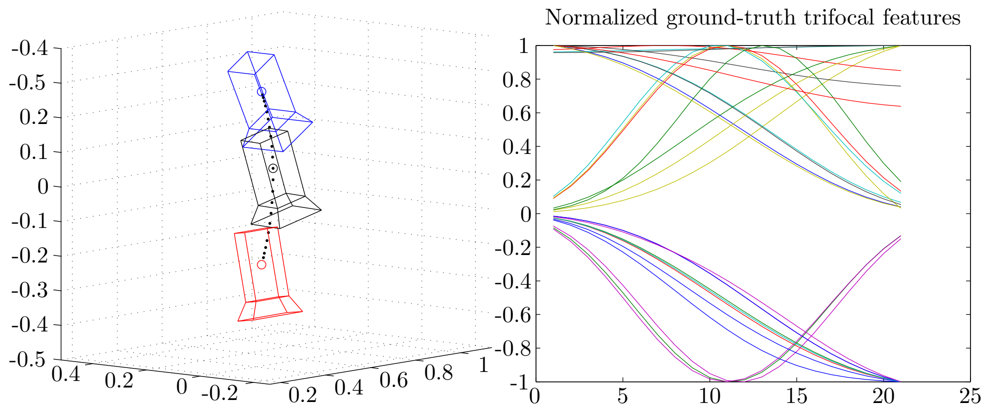
Figure 1.4: Robust uncalibrated planning and visual servoing block diagram. A contribution of this thesis is the robust Jacobian estimation and control for an uncalibrated visual servoing system (see Chapter 4). An eye-in-hand manipulator is considered.

which is suitable for unstructured settings. The UvsBiRRT algorithm, which is based on the bidirectional RRT planner [34] is presented in Chapter 5. The basic RRT algorithm and the bidirectional RRT are reviewed in Chapter 3.

The UvsBiRrt algorithm is an efficient robust sampling-based path planning algorithm designed for a UVS system. We consider an eye-in-hand configuration with a planar target and a planar obstacle *without* their geometric models. The proposed algorithm is a modified version of the RRT planner with a special data structure. The planner works in the visual-motor space and provides a path to avoid visual occlusions of the target by the obstacle that might occur during servoing, in addition to avoiding joint and field-of-view (FOV) constraints.

A neighbourhood of a visual-motor sample that violates either the joint-limit constraint or the FOV constraint belongs to the occupied space and cannot be used for planning. The occupied space is then updated by the neighbourhoods of the visual-motor samples, where the target object is visually occluded by the obstacle in the image. Since visual occlusion is determined in the image space, there is no need for an explicit geometric model of the target or the obstacle. Once the occupied and free spaces are estimated, the RRT data structure can be extended by adding new nodes after checking for occlusions, FOV, or joint limit violations.

As emphasized previously, outliers in the visual-motor space are unavoidable. They need to be handled properly such that their effect on the control and planning algorithms is mitigated. As we show in Chapter 5, the same robust estimation algorithm presented in Chapter 4 can be used in the planning phase to eliminate outliers from the visual-motor database. In addition, the robust Jacobian estimate along the solution path is required to follow a planned path using the control law. When a new random point is sampled, the algorithm estimates the robust visual-motor Jacobian at the sample and labels it as inlier or outlier simultaneously. The outlier can be replaced by a nearby inlier (according to some distance), where the tree gets extended to. If a nearby inlier does not exist, the randomly sampled point



(a) Cartesian space (b) Tensor space

Figure 1.5: (a) A sample camera trajectory from the start (top, blue) to the goal (bottom, red) pose. An arbitrary intermediate configuration on the trajectory is also shown (middle, black). (b) The evolution of the elements of the trifocal tensor along this trajectory is depicted. The elements are normalized with respect to their maximum absolute value to fit in the same graph.

is discarded. As such, robustness to outliers is ensured by tree construction. The proposed tree data structure includes the joint vector, the visual feature vector, as well as the corresponding visual-motor Jacobian estimate. This incorporates robustness into the tree extension algorithm of the proposed RRT-based planner.

The algorithms are explained in Chapter 5 and evaluated in Chapter 7 together with the robust Jacobian estimation and control algorithms, since there was some overlap between them.

Uncalibrated Visual Servoing from the Projective Geometry of Three Views

We propose a new class of visual servoing, Projective-Geometric Visual Servoing (PGVS), where the error signal is based on a projective-geometric measure. In Chapter 6, we propose to use the projective geometry of three views for uncalibrated visual servoing.

Almost all UVS systems to-date have used the image coordinates of scene points as servoing features [6, 7, 8, 9]. In Chapter 6, we develop a formulation based on the trifocal tensor as a new type of features for error generation and closing the feedback loop in the UVS control architecture. The trifocal tensor encapsulates the projective geometry of three images. In a visual servoing system, these three images are the desired image, the initial image, and the midway image along the control trajectory. The elements of the trifocal tensor evolve smoothly as the camera moves from the initial configuration to the desired configuration as shown in Figure 1.5 We define an error based on the elements of the trifocal tensor, where the aim of the control law is to regulate this error to zero. Once the error reaches zero, the robot reaches the desired 6-DOF Cartesian configuration.

The proposed 6-DOF uncalibrated trifocal visual servoing is a special case for the PGVS. The evaluations and experimental results are presented in Chapter 8.

Part I

Background and Literature Review

Chapter 2

Visual Servoing Methods

In this chapter, we review the visual servoing literature and formulate the uncalibrated visual servoing problem, which is the underlying framework of this thesis. We discuss available Jacobian estimation algorithms, and motivate why in an unstructured environment these methods cannot be used. We also review the literature on robust visual servoing and place the contributions of this thesis in perspective.

2.1 Overview of Visual Feedback in Motion Control of Robots

The goal of many robotic applications is to place the robot at a desired configuration to manipulate an object in an environment. Computer vision adds versatile sensing to a robotic application. The early approaches to vision-based robotics include monitoring and inspection applications, where visual feedback is not used in a closed-loop control scheme.

To place the end-effector of the manipulator at a desired position with respect to the object, the rigid-body transformation between the object and the robot base and between the robot base to the end-effector must be known. Figure 2.1 shows a manipulator with a camera and an object and the corresponding coordinate frames. Let the robot base frame be denoted by $\{B\}$, the frame at the end-effector by $\{E\}$, the object frame by $\{O\}$, the transformation from $\{B\}$ to $\{O\}$ by W_O^B, the transformation from $\{B\}$ to $\{E\}$ by W_E^B, and the transformation from $\{O\}$ to $\{E\}$ by W_E^O. Given W_O^B (*e.g.,* object on a fixture with calibrated distance from the base) and the desired W_O^B, one can calculate W_E^B and then by solving the inverse kinematics problem, find the robot configuration. Despite the limiting assumption to calculate W_O^B and perfect robot calibration a-priori, many industrial applications such as factory automation and visual part inspection still use this open-loop framework. It is clear that this approach is limited to very structured environments and does not apply to unstructured settings.

To add flexibility to vision-based robots, visual feedback can be used. Figure 2.2 shows the addition of a camera frame $\{C\}$ to the previous vision-based manipulator shown in Figure 2.1. The other transformations of interest are the object-to-camera transformation W_C^O and the end-effector-to-camera transformation W_C^E. Addition of a camera sensor enables bypassing of the robot base frame to calculate the relative object to robot transformation. In particular, there is no need to place the object

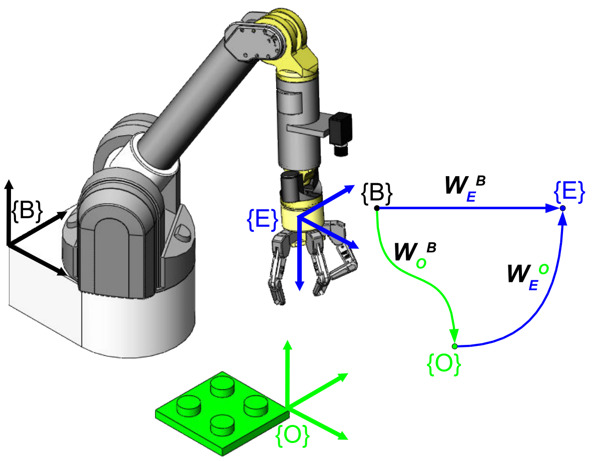


Figure 2.1: Open-loop robot control without using feedback from the camera. There are three frames: robot base, object, and end-effector. The forward kinematics transformation is denoted by W_E^B, the base to object transformation by W_O^B, and object to end-effector by W_E^O. Robot configuration can be updated by solving the inverse kinematics from known W_O^B (object on know fixture in structured settings) and W_E^O (user-defined).

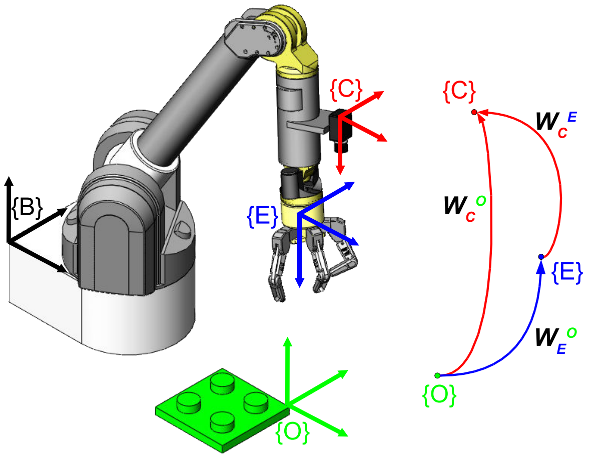


Figure 2.2: Closed-loop robot control using the relative object-to-camera pose. With a feedback signal from the camera, the robot base frame and fixed object fixture can be bypassed. The other three frames are the object, the end-effector, and the camera. Transformation W_C^E takes the end-effector frame to the camera frame and is found by calibration. Transformation W_C^O denotes the relative object-to-camera pose. As we shall see in Section 2.2, the position-based architecture is formulated around the transformations in this figure.

on a known fixture.

One strategy is to compute transformation W_E^O from a calibrated transformation W_C^E and estimate the relative camera-to-object relative pose W_C^O from pose estimation algorithms. Once the transformation W_E^O is computed, the robot can be moved towards the desired pose without further pose estimation. This is called the static *"look and move"* control architecture [35].

In general, visual feedback is provided by one or more cameras that are either rigidly attached to the robot (eye-in-hand configuration as in Figures 2.1 and 2.2) or static in the environment looking at the robot motions (eye-to-hand configuration, not shown in figures). The initial and desired states define an error, which is to be minimized and regulated to zero at the desired state.

Computer vision algorithms are used for the tracking of visual features on the object. The visual features can be used to compute the relative object-to-camera pose or to compute an error in the image space. The typical visual tracking features are either geometric primitives or appearance-based. Examples of geometric features are dots, lines, contours, and their higher order moments [36]. An example of an appearance-based feature is the Sum of Squared Distances (SSD) tracker [37].

Visual servoing is a framework where real-time visual feedback is used to control a robot to a desired configuration [2, 3, 4]. Visual servoing is also studied as vision-based motion control and robotic hand-eye coordination with a feedback.

Depending on the type of the error in the control law, one can classify the visual servoing system to three main classes: position-based visual servoing (PBVS) or 3D visual servoing [38], image-based visual servoing (IBVS) or 2D visual servoing [39], and hybrid visual servoing (HVS), which is either implemented as a $2\frac{1}{2}$D visual servoing [24, 40], or a hybrid switching controller (HSC) [41, 42]. Stability analysis, control uncertainty, and performance studies of these approaches are available in the literature [11, 40, 43, 44, 45].

In PBVS the relative pose of the end-effector (with respect to the object) is found from the feature points in the image. The control law is defined in the Cartesian space and requires the geometric model of the object and perfect camera/robot calibration. Therefore, the classic PBVS is not applicable to unstructured settings. The control law in direct IBVS is defined without using the object model and directly in the image space; therefore, it could be used in unstructured environments with some considerations. In $2\frac{1}{2}$D HVS, the control law has two components. One component is found directly from the initial and desired images, but the other component is found from the scaled Euclidean transformation. In HSS HVS, a higher-level discrete system switches between IBVS and PBVS subsystems to ensure that the overall system is stable [42], or the system alternates between PBVS and IBVS to avoid singularities, image-space, Cartesian-space, or joint-space constraints with the help of a path planner [41]. Because the model of the object is partially used in the HVS approach, it also does not apply to unstructured settings.

In the following sections, we will first review PBVS briefly, and then study the IBVS and uncalibrated visual servoing approaches more elaborately as they are essential to this thesis.

2.2 Position-Based Visual Servoing (PBVS)

In the case that the error is expressed in terms of a 3D rigid-body transformation, the relative transformation from the initial to the desired state must be inferred and updated from images. This approach is studied in PBVS [38, 3]. Figure 2.3 shows

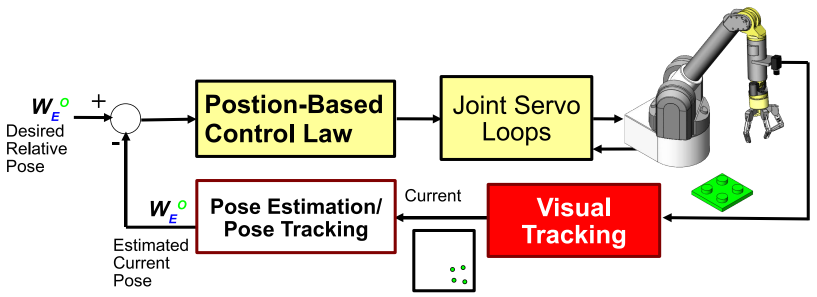


Figure 2.3: Position-Based Visual Servoing (PBVS) diagram. A pose estimation or pose tracking algorithm is called on the output of the visual tracking module to estimate the relative object-to-camera pose. The control law regulates the pose error to zero using Cartesian-space control.

a block diagram for a typical PBVS system. The camera acts as a 3D sensor in PBVS and the previous illustration in Figure 2.2 is also PBVS. The PBVS control law ensures global asymptotic stability if the pose estimation or tracking is perfect and there are no uncertainties in the robot model [3]. The pose estimation or pose tracking algorithm must be realized in real-time to calculate the error at every control iteration. Usually, pose tracking algorithms by the Extended Kalman Filter (EKF) [38] or variations of EKF [46, 47] are preferred for speed requirements. A recent efficient pose estimation algorithm from n points, with a computation time of $O(n)$ can also be used [48].

The theoretical stability proofs for PBVS under ideal conditions are attractive, but in practice the pose estimation/tracking algorithms are sensitive to image measurement noise, Kalman filter tuning [47], and outliers [48]. There is another problem with PBVS. Since there is no control on the visually-tracked image features, the image features might leave the field-of-view (FOV) during servoing, which results in the failure of the pose estimation/tracking and ultimately failure of the overall system. This occurs because the control law sets the camera to follow the shortest trajectory in the camera Cartesian space and the image-space trajectories might leave the FOV. In summary, *the PBVS approach is not suitable for unstructured environments, because the pose estimation requires full knowledge of the geometric object model, the camera intrinsic calibration, and the robot kinematic calibration.*

2.3 Image-Based Visual Servoing (IBVS)

In the IBVS approach, the error is defined between the current and desired images and directly in the image space [39]. This is done without the explicit calculation of the relative frames as opposed to PBVS. Figure 2.4 shows a typical IBVS block diagram. A typical IBVS task is demonstrated in Figures 2.5 and 2.6 using MATLAB® simulations.

The IBVS approach is more robust to modeling and measurement noise than PBVS and converges even with a coarsely calibrated camera [49]. It is also more accurate because the control law is defined in the image space [3]. However, the IBVS approach only guarantees local asymptotic stability with a local control law

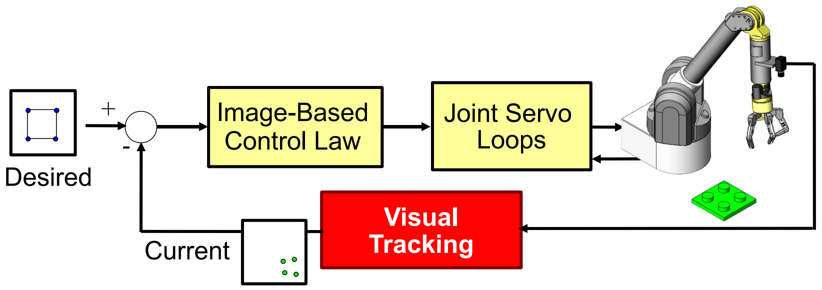


Figure 2.4: The Image-Based Visual Servoing (IBVS) diagram. A visual tracking module process the image feed to track desired image features in real-time. The image error is found directly from the desired image and the current image. The image-based control law in (2.5) regulates the image error to zero by generating appropriate camera velocity.

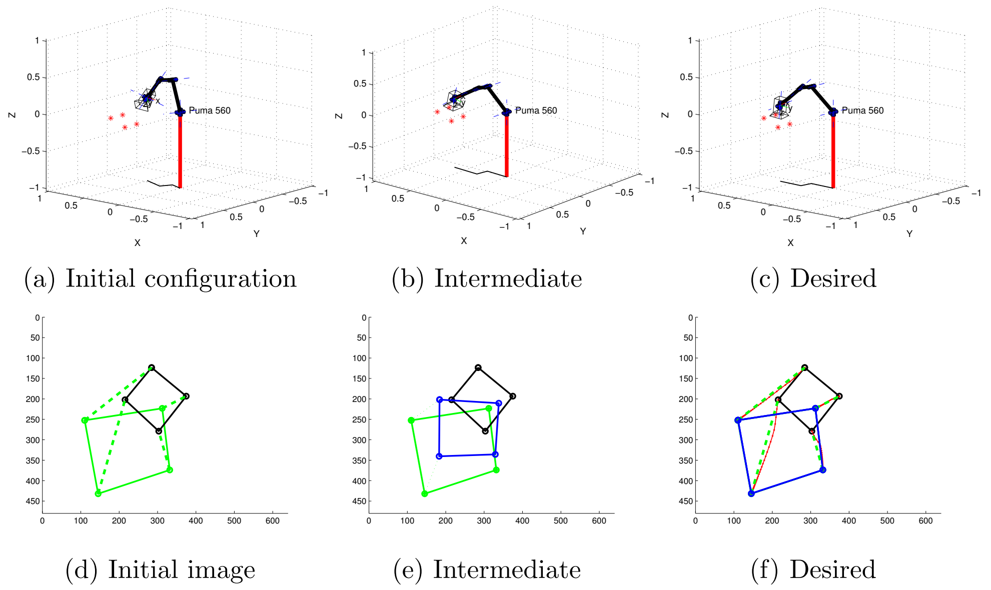
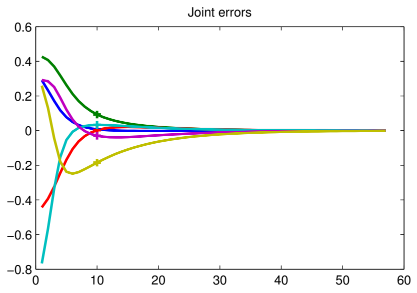
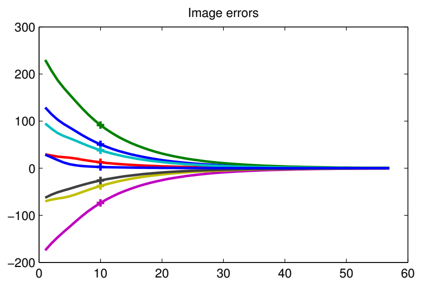


Figure 2.5: IBVS demo with MATLAB simulations. A planar object with four interest points is considered. The robot arm is equipped with an in-hand camera. (a) Initial robot configuration. (d) Initial image (small square, black). The desired image (large square, green) is overlayed with the dotted line in the image space corresponding to the shortest path in the image. (b), (e) An arbitrary intermediate state. The intermediate image is shown in blue and the initial/desired images are overlayed. (c) The desired (goal) robot configuration. This is not known *a priori*. (f) The final image (blue) and the image-space trajectory, which is close to the dotted shortest path, but not identical due to the kinematic constraints of the arm. The goal image is provided as an input (teach-by-showing concept). Figure best seen in colour.

theoretically [3]. When the initial and desired images correspond to large motions, the resulting camera trajectory is not the shortest. In addition, there are local



(a) Joint error [rad]



(b) Image error [pix]

Figure 2.6: Joint and Image errors for the IBVS demo in Figure 2.5. (a) Joint errors converge to zero but not exponentially. (b) Image errors show an exponential decay as expected by controller design. The intermediate state as in Figures 2.5b and 2.5e is marked by crosses on the graphs.

minima and singularity problems [11, 3]. To better understand these problems, more detail about the control law is provided here.

Let $\mathbf{s}(t)$ be the vector of visual measurements at time t and $\mathbf{s}^*$ be the desired vector at the desired configuration. The time variation of $\mathbf{s}(t)$, *i.e.,* the velocity of the visual features, is related to the camera velocity screw $\mathbf{v} = [\mathbf{V} \;\; \boldsymbol{\Omega}]^\top$ [39, 3]:

$$\dot{\mathbf{s}} = \mathbf{L}(\mathbf{s}(t))\,\mathbf{v} \tag{2.1}$$

where $\mathbf{L}(\mathbf{s}(t))$ is a Jacobian matrix usually referred to as the *interaction matrix* in the visual servoing community, vector $\mathbf{V}$ is the translational velocity and $\boldsymbol{\Omega}$ is the rotational velocity. This matrix relates the rate of changes of the image feature velocities $\dot{\mathbf{s}}$ to the rate of change of the pose parameters which is the camera velocity screw[50]. The analytic form of the interaction matrix is known for particular types of visual features [39][36]. For example, let us consider 3D point $[X\ Y\ Z]^\top$ expressed in the camera frame and its projection onto the image plane $[x\ y]^\top$. Suppose we take the image coordinates of the 3D point as a visual features (dot feature). Then, the interaction matrix has the following form [39]:

$$\mathbf{L} = \begin{bmatrix} -f\frac{1}{Z} & 0 & f\frac{x}{Z} & f\frac{xy}{Z} & -\frac{f^2+x^2}{f} & y \\ 0 & -f\frac{1}{Z} & \frac{y}{Z} & \frac{f^2+y^2}{f} & -\frac{xy}{Z} & -x \end{bmatrix}, \tag{2.2}$$

where f is the camera focal length obtained from intrinsic camera calibration. The analytic interaction matrix depends on the camera calibration and also more importantly *depends on the depth of the 3D point from the camera.* In fact, the interaction matrices of other visual features [36] also depend on a 3D parameter that needs to be estimated, therefore, an approximation $\widehat{\mathbf{L}}$ is always used in practice. For the purpose of operation in unstructured environments, this dependence on a 3D parameter is a limiting factor. We will show later how this problem can be remedied by adopting an uncalibrated IBVS approach, where there is no need to know or estimate this 3D parameter.

The Lyapunov stability theory provides the tool to design an analyze the IBVS control law. Following the task function approach [52], the visual servoing error $\mathbf{e}(t)$ is defined as

$$\mathbf{e}(t) = \mathbf{s}(t) - \mathbf{s}^*, \tag{2.3}$$

where the goal of IBVS is to regulate $\mathbf{e}(t)$ to zero. Let the candidate Lyapunov function be $U(t) = \frac{1}{2}\mathbf{e}^\top\mathbf{e}$ and its derivative $\dot{U}(t) = \mathbf{e}^\top\dot{\mathbf{e}}$. An exponentially-decaying error $\dot{\mathbf{e}} = -\lambda\mathbf{e}$, results in $\dot{U}(t) = -\lambda\mathbf{e}^\top\mathbf{e} < 0$. If we could design a controller with such an error, then exponential stability would be ensured [51]. The time derivative of (2.3) provides

$$\dot{\mathbf{e}} = \mathbf{L}\,\mathbf{v} = -\lambda\mathbf{e}. \tag{2.4}$$

The IBVS control law is obtained as

$$\mathbf{v} = -\lambda\,\widehat{\mathbf{L}}^\dagger\,\mathbf{e}, \tag{2.5}$$

where $\widehat{\mathbf{L}}$ is an approximation to the interaction matrix $\mathbf{L}$, and $\widehat{\mathbf{L}}^\dagger = \widehat{\mathbf{L}}^\top(\widehat{\mathbf{L}}\,\widehat{\mathbf{L}}^\top)^{-1}$ is the Moore-Penrose pseudoinverse of $\widehat{\mathbf{L}}$.

To further analyze the stability of the resulting control law, we check to see if the Lyapunov conditions holds:

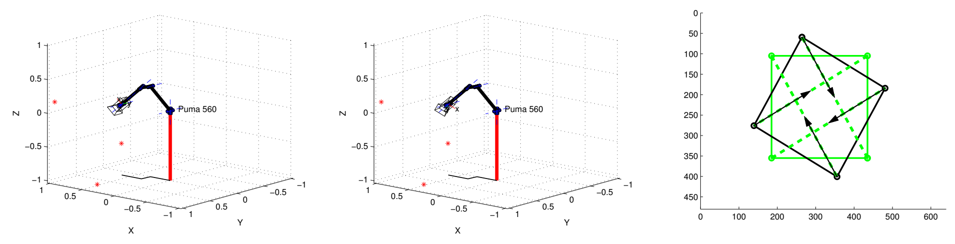
$$\begin{aligned}\dot{U} &= \mathbf{e}^\top\dot{\mathbf{e}} && (2.6)\\ &= \mathbf{e}^\top\mathbf{L}\,\mathbf{v} = -\lambda\,\mathbf{e}^\top\mathbf{L}\,\widehat{\mathbf{L}}^\dagger\,\mathbf{e}. && (2.7)\end{aligned}$$

Therefore, for the IBVS to be asymptotically stable, $\mathbf{L}\,\widehat{\mathbf{L}}^\dagger$ should be full-rank and positive-definite, however, $\mathbf{L}\,\widehat{\mathbf{L}}^\dagger$ is only positive semi-definite and for some non-zero values of $\mathbf{e}$, $\mathbf{e}^\top\mathbf{L}\,\widehat{\mathbf{L}}^\dagger\,\mathbf{e} = 0$. Hence, asymptotic stability by Lyapunov theory cannot be demonstrated [51]. It was only shown that the system is stable.

Visual features should be selected such that matrices $\mathbf{L}$ and $\widehat{\mathbf{L}}^\dagger$ are full rank. This is a limiting condition for the interaction matrix in (2.2) as explained by the following simple example. To control the 6 degrees-of-freedom (DOF) of a robot, the rank of the interaction matrix must be at least 6 and at least 3 dot features should be used to meet the stability criteria. However, with 3 dot features there are 4 distinct global minima for camera positions where $\mathbf{e} = 0$ [3]. There is also the local minima problem for more than 3 dot features, where two distinct camera configurations give similar image features [3]. The local minima problem is not exclusive to dot features. In fact, a great deal of research is concentrated on finding better visual features for the perspective camera model [36, 53] and spherical projection camera model[54, 55]. In addition to local minima, the interaction matrix or the kinematic Jacobian might become singular during operation. This results in an unreliable control signal. Nelson and Khosla have studied the singularity of the interaction matrix as configurations at which motion is not visually resolvable [56]. Such methods usually avoid the kinematic singularity at the low-level control library. In Section 2.4, we will show how both singularities can be avoided at once.

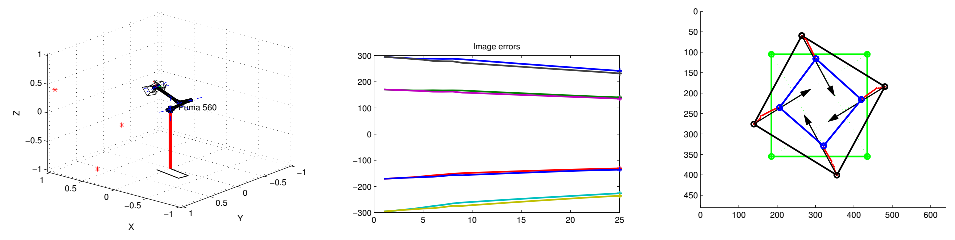
The IBVS has some known degenerate configurations for the control law in (2.5) [11]. One of these degenerate configurations correspond to rotations around view-axis. In Figure 2.7, the initial and desired robot configurations and the corresponding images are depicted. The IBVS control in (2.5) tries a generate robot motions that result straight image trajectories. This results in the camera to retreat from the desired configuration. This unwanted motion generates desired image trajectories with a decaying image error as shown in Figure 2.8.

In Chapter 6, we show the preliminary results of one of our algorithms that solves the camera retreat problem while not depending on 3D parameters, camera, or robot calibration.



(a) Initial (b) Desired (c) Image trajectory

Figure 2.7: IBVS rotation around view-axis setup. (a) Initial configuration. (b) The desired configuration looks similar to the initial because the camera is rotated around the view axis (optical axis rotation). (c) The initial image in black is a pure rotation with respect to the desired image in green. The arrows show the direction of the shortest path in the image space.



(a) Camera *retreats* (b) Image error decays (c) Linear trajectory

Figure 2.8: The IBVS camera retreat problem for pure rotation around the view axis (Figure 2.7) is illustrated. Regulation of the image error to zero results in a wrong robot motion, where the camera retreats from the object.

We have presented the main problems with the classic IBVS approach to motivate new developments proposed in this work. Detailed discussions on the stability and convergence of the IBVS is beyond the scope of this document and we refer the reader to [3, 11, 44, 51].

2.4 Uncalibrated Visual Servoing (UVS)

Uncalibrated visual servoing (UVS) studies vision-based motion control of robots without using the camera intrinsic parameters, the calibration of the robot-to-camera transformation, or the geometric object/scene models [4, 7]. This is a demanding problem with increasing applications in unstructured environments, where no prior information is assumed [7, 8, 9].

The control law in the UVS should be defined without the need to reconstruct the depth or other 3D parameters. One way to define the uncalibrated control law is an approach similar to IBVS. Let $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^M$ be the mapping from the configuration $\mathbf{q} \in \mathbb{R}^N$ of a robot with N joints, to the visual feature vector $\mathbf{s} \in \mathbb{R}^M$ with M visual features. For example, for a 6 DOF robot with 4 point features (8 coordinates in total), $N = 6$ and $M = 8$. The visual-motor function of such vision-based robotic

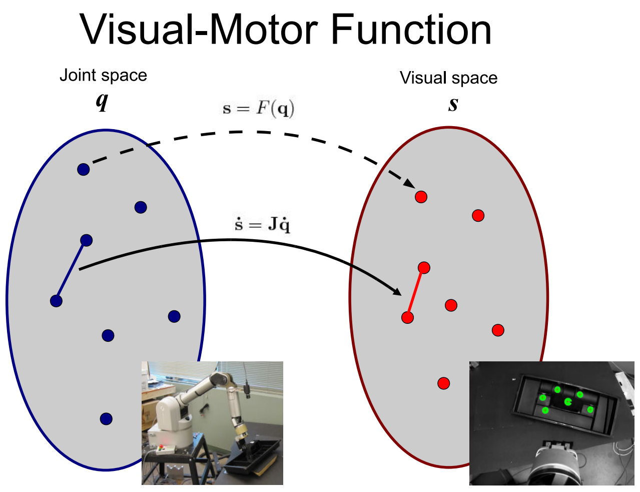

Visual-Motor Function
Joint space
q
Visual space
s
$\mathbf{s} = F(\mathbf{q})$
$\dot{\mathbf{s}} = \mathbf{J}\dot{\mathbf{q}}$

Figure 2.9: Visual-motor space diagram. Each joint reading (motor) is uniquely mapped to a sensor reading (visual). Variations in the motor space is mapped to variations in the visual space through a Jacobian matrix known as the visual-motor Jacobian. The reader is referred to (2.8)-(2.11) to see how the visual-motor Jacobian is defined and how it appears in the control law.

system can be written as

$$\mathbf{s} = \mathbf{F}(\mathbf{q}). \tag{2.8}$$

This formulation is general and covers both eye-in-hand and eye-to-hand systems.

The time derivative of the visual-motor function in (2.8) leads to

$$\frac{\partial \mathbf{s}}{\partial t} = \frac{\partial \mathbf{F}(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t} = \mathbf{J_u}(\mathbf{q})\,\dot{\mathbf{q}}, \tag{2.9}$$

in which $\mathbf{J_u} \in \mathbb{R}^{M \times N}$ is called the *visual-motor Jacobian*. The discrete-time approximation of (2.9), when $\mathbf{J_u}(\mathbf{q})$ is replaced by $\widehat{\mathbf{J}}_\mathbf{u}(\mathbf{q})$ is

$$\Delta\mathbf{s} \simeq \widehat{\mathbf{J}}_\mathbf{u}(\mathbf{q})\,\Delta\mathbf{q}. \tag{2.10}$$

Figure 2.9 summarizes the above notations in a diagram.

Similar to the IBVS control law in (2.5), the estimated visual-motor Jacobian, $\widehat{\mathbf{J}}_\mathbf{u}$, appears in the uncalibrated control law:

$$\dot{\mathbf{q}} = -\lambda\,\widehat{\mathbf{J}}_\mathbf{u}^{\dagger}\,(\mathbf{s} - \mathbf{s}^*), \tag{2.11}$$

where $\widehat{\mathbf{J}}_\mathbf{u}^{\dagger}$ is the Moore-Penrose pseudoinverse of $\widehat{\mathbf{J}}_\mathbf{u}$.

It is worthwhile to see how the uncalibrated Jacobian is related to the interaction matrix in (2.2). The kinematic Jacobian of a manipulator, $\mathbf{J}(\mathbf{q})$, relates $\mathbf{v}$, the velocity screw of its end-effector, to $\dot{\mathbf{q}}$, the joint velocities [51]:

$$\mathbf{v} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}. \tag{2.12}$$

Recall from (2.1) that the rate of change of the visual features is related to the velocity screw of the camera frame by the interaction matrix: $\dot{\mathbf{s}} = \mathbf{L}\,\mathbf{v}$. Assuming that the camera frame is aligned exactly on the end-effector frame, (2.1) and (2.12) results in

$$\dot{\mathbf{s}} = \mathbf{L}\,\mathbf{v} = \mathbf{L}\,\mathbf{J}\,\dot{\mathbf{q}}, \tag{2.13}$$

which is the same equation as (2.9) with $\mathbf{J_u} = \mathbf{L}\,\mathbf{J}$. When either the kinematic Jacobian or the interaction matrix becomes near-singular, the uncalibrated Jacobian becomes poorly conditioned. In this case, the numerical computations in the control scheme become unstable. Figure 2.10 shows a generic block diagram for a typical UVS system. In the following subsections, we provide a brief literature review on the Jacobian estimation methods (direct estimation or update) for the UVS.

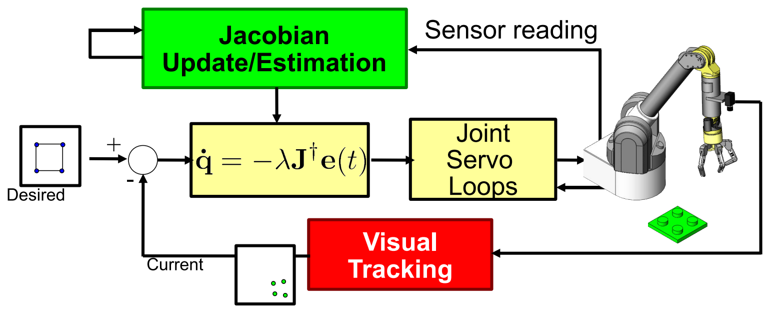

Jacobian Update/Estimation
Sensor reading
+
-
Desired
$\dot{\mathbf{q}} = -\lambda \mathbf{J}^{\dagger}\mathbf{e}(t)$
Joint Servo Loops
Current
Visual Tracking

Figure 2.10: Uncalibrated Visual Servoing (UVS) block diagram. The diagram is similar to the IBVS block diagram (Figure 2.4) with a couple of differences. The control law uses the visual-motor Jacobian here not the interaction matrix. As a result joint velocities are computed with a joint-space controller. In addition, the visual-motor Jacobian needs to be estimated, *e.g.,* by one of the methods explained in sections 2.4.1-2.4.3.

2.4.1 Jacobian estimation by orthogonal motions

Since the analytic form of the Jacobian is not available, we can estimate the Jacobian by choosing small orthogonal exploratory motions [57].

Consider a small displacement angle δ of the joints, for which the difference of the image features can be measured and is larger than the visual-tracking noise. The Jacobian can be found from N visual feature displacements $\Delta\mathbf{s}^{(1)}, \cdots, \Delta\mathbf{s}^{(N)}$ from N orthogonal motions:

$$\begin{aligned} \Delta\mathbf{s}^{(1)} &\simeq \widehat{\mathbf{J}}\begin{bmatrix}\delta & 0 & \dots & 0\end{bmatrix}^{\top} \\ \vdots \quad & \quad \vdots \\ \Delta\mathbf{s}^{(N)} &\simeq \widehat{\mathbf{J}}\begin{bmatrix}0 & \dots & 0 & \delta\end{bmatrix}^{\top}, \end{aligned} \tag{2.14}$$

$$\mathbf{J_R} \simeq \frac{1}{\delta}\begin{bmatrix}\Delta\mathbf{s}^{(1)} & \Delta\mathbf{s}^{(2)} & \dots & \Delta\mathbf{s}^{(N)}\end{bmatrix} \tag{2.15}$$

In simulations, we can choose an arbitrarily small value for δ and measure $\Delta\mathbf{s}^{(i)}, i = 1, \cdots, N$ with high accuracy. The orthogonal motions is not practical for online estimation of the Jacobian in real robotic systems, because it requires extra undesired moves. However, it is a meaningful reference to evaluate the accuracy of the Jacobian estimation at a desired configuration. We usually use it to evaluate the accuracy of different Jacobian estimation algorithms. The estimation error can be measured by the Frobenius norm of the online estimate, $\widehat{\mathbf{J}}_{\mathbf{u}}$, to this reference Jacobian, $\mathbf{J}_{\mathbf{R}}$:

$$\nu = \|\mathbf{J}_{\mathbf{R}} - \widehat{\mathbf{J}}_{\mathbf{u}}\|_F = \sqrt{\sum \operatorname{diag}\left((\mathbf{J}_{\mathbf{R}} - \widehat{\mathbf{J}}_{\mathbf{u}})^\top (\mathbf{J}_{\mathbf{R}} - \widehat{\mathbf{J}}_{\mathbf{u}})\right)} \tag{2.16}$$

where $||\cdot||_F$ is the Frobenius norm.

2.4.2 Jacobian estimation by the Broyden update rule

A Broyden rank-one secant update has been proposed by Jägersand *et al.* [7] and Hosoda and Asada [6] to estimate the visual-motor Jacobian.

$$\mathbf{J_u}^{(k+1)} = \mathbf{J_u}^{(k)} + \alpha \frac{(\Delta\mathbf{s} - \mathbf{J_u}^{(k)}\Delta\mathbf{q})\Delta\mathbf{q}^\top}{\Delta\mathbf{q}^\top\Delta\mathbf{q}}, \tag{2.17}$$

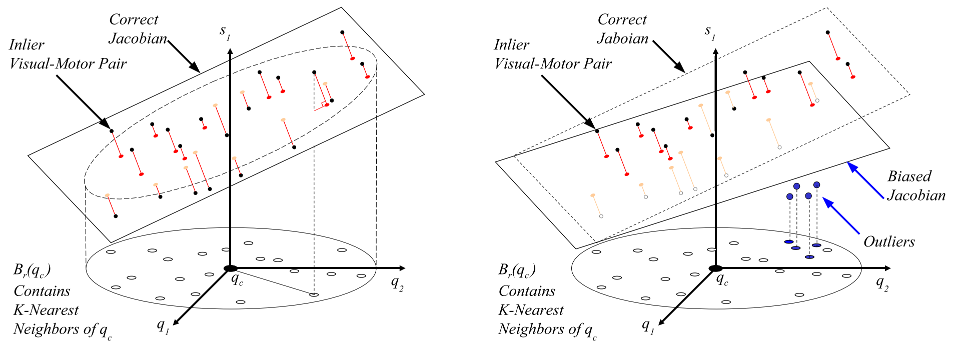
where $\Delta\mathbf{s}$ and $\Delta\mathbf{q}$ are the visual measurement and joint differences between the current reading at iteration $k+1$ to the previous reading at iteration k. Parameter $\alpha > 0$ is a forgetting factor which is used to lessen the weight of previous data during the estimation process. This update considers a static object. Piepmeier *et al.* introduced a new term to consider a dynamic object [8]. One of the problems with methods based the Broyden update is that they require a good initial guess. This method is not a direct estimation method, but an update (or tracking) one. This is important for some critical tasks, where measurements are available but a Jacobian estimate is not available. Another problem with this scheme is the lack of a framework to include old data. If a robot is following the same trajectory over and over, the Broyden Jacobian update would not improve.

2.4.3 Jacobian estimation by the least-squares based methods

Farahmand *et al.* [9] propose the local least-squares (LLS) estimation to utilize the memory of visual-motor data. They estimate the visual-motor Jacobian in simulated 3 DOF eye-to-hand experiments. This method is general and estimates the Jacobian of any point in the workspace directly from raw visual-motor data in a close neighborhood of the point under consideration. The LLS method [9] is similar to the work of Lapresté *et al.* [58], where the least squares problem is solved directly for the pseudoinverse Jacobian. However, random perturbations around the desired pose are used as offline training in [58]. Instead of random perturbations, the LLS method considers a memory of the previous measurements and find the forward Jacobian directly from these measurements.

For a memory with P visual-motor data pairs and a new visual-motor query point $\mathbf{d}_c = (s_c, q_c)$, the uncalibrated Jacobian estimation problem is posed as the following optimization problem [9]:

$$\widehat{\mathbf{J}}_{\mathbf{u}}(q)\Big|_{q=q_c} = \arg\min_{\mathbf{J_u}} \sum_{k:\, q_k \in B_r(q_c)} (\Delta s_k - \mathbf{J_u}\Delta q_k)^2, \tag{2.18}$$



(a) No outliers (b) With outliers

Figure 2.11: Visual-motor Jacobian estimation using least-squares methods. (a) Estimation using the Local Least-Squares (LLS) method [9] according to (2.18). (b) The effect of outliers on the Jacobian estimates by the LLS method. The estimation of the Jacobian hyperplane is biased towards the outliers.

where $B_r(q_c) = \{q_p : \|q_c - q_p\| < r\,, p = 1, \cdots, P\}$ is an open ball with radius r in the joint space, $\Delta s_k = s_c - s_k$, and $\Delta q_k = q_c - q_k$. This method fits the best hyperplane to the visual-motor data around q_c. An illustration of this method is given in Figure 2.11a, where a hyperplane is fitted to 2×1-dimensional data (2 DOF for joints and a single image feature).

The problem with the least-squares-based methods is that they are sensitive to outliers. This is illustrated in Figure 2.11b, where the Jacobian is biased towards the outlier visual-motor pairs. The visual-motor outliers are inevitable in unstructured environments for several reasons. The main reasons attribute to the non-descriptive visual features used in tracking and the real-time processing requirements of visual servoing systems. Some concrete examples of outliers is presented in Figure 1.1. The visual feature used in the first row is the centroid and the area of the cylinder top. When the feature is occluded by an object with similar properties, the measured visual feature has large errors. The visual features used in the second row are the centroid of the cylinder tops. Because of the fast motion of an eye-in-hand robot, features 3 and 4 slip on the other two features and generate a visual-motor outlier for the given visual-motor configuration.

In the next section, we review the literature on robust visual servoing. We have also developed a statistically robust method to deal with outliers that will be presented later in Chapter 4.

2.5 Robust Visual Servoing (RVS)

Robustness in visual servoing has been studied from different perspectives: robust visual tracking, statistically robust IBVS, and control-theoretic stability analysis using adaptive robust control.

2.5.1 Robust visual tracking for visual servoing

Many authors discuss robust visual tracking for a visual servoing system. Toyama and Hager [17] present the Incremental Focus of Attention (IFA) hierarchical architecture for robust visual tracking. The architecture consists of different layers of search and tracking and a transition policy between the layers. A high-precision tracker is the top layer of the hierarchy, which transitions to a lower-precision tracker when there are visual disturbances. Multiple tracking algorithms and search heuristics are used in the IFA architecture to achieve a robust system. Kragic and Christensen [18, 19] propose robust visual tracking using a voting scheme and visual cue integration to achieve robustness in unstructured settings. The RANdom SAmple Consensus (RANSAC) algorithm [59] has been used in conjunction to other robust methods. Preisig and Kragic [20] use robust M-estimation and RANSAC for 3D tracking. Comport *et al.* [21] compare statistically robust real-time visual tracking algorithms. Tran and Marchand [22] propose a fast and efficient feature descriptor for tracking and use RANSAC to reject matching outliers between two views. Robust M-estimation has been proposed for articulated object tracking with applications to virtual reality by Comport *et al.* [23]. In these methods, the robustness is considered for the visual tracking module. They do not utilize the important information embedded in the joint encodings and try to reject the outliers using only visual information. Next, we will mention some papers that consider robustness within the control law, not just the visual tracking module.

2.5.2 Statistically robust visual servoing

Comport *et al.* [30] use M-estimation to find visual tracking outliers during visual servoing. Robustness is achieved by de-weighting the rows of the parametric interaction matrix associated with outliers. Dionnet and Marchand [31] use a similar control law for stereo 3D tracking and visual servoing. This set of papers [30, 31] discuss statistical robustness within the IBVS control law.

2.5.3 Control-theoretic robust and adaptive visual servoing

The other approaches to robust visual servoing are either model-based [24, 25, 26], or parametric model-free [27, 28, 29]. These authors study robustness from a control-theoretic point of view. This is not directly related to our work, but it is mentioned for completeness. Robustness to calibration parameters may be achieved in model-based visual servoing [24, 26, 25] (see Section 2.6 for our definitions of model-based, model-free and parametric in visual servoing). Adaptive control has also been used with parametric model-free IBVS [27, 28, 29]. These works estimate the linearized camera/robot calibration parameters using a depth-independent Jacobian without estimating the depth directly. Liu *et al.* [27] propose a depth-independent Jacobian for an eye-to-hand system, to estimate the linearized camera parameters online during visual servoing of point features. They assume known 3D coordinates of the features with respect to the robot. Wang *et al.* [28] propose a depth-independent Jacobian to estimate the linearized camera parameters online. They use this depth-independent Jacobian for eye-in-hand visual servoing of point and line features without knowledge of 3D coordinates. Wang *et al.* [28] extend the work in [27] to eye-in-hand visual servoing of point and line features without knowledge of 3D coordinates. Hu *et al.* [29] propose a homography-based robust adaptive controller to control translation and orientation of an eye-in-hand system in the presence of uncertainty in the intrinsic camera parameters as well as uncertainty in the depth information. These methods do not study robustness to visual-motor outliers.

2.6 Model-Based versus Model-Free Visual Servoing

Classical approaches to visual servoing use some knowledge about the model and/or the system parameters [3]. For example, PBVS (see Section 2.2) uses a calibrated camera and known geometric model of the 3D features to reconstruct the relative pose of the camera with respect to the desired object. In this sense, PBVS is *model-based* and not suitable for unstructured settings. The IBVS (see Section 2.3) does not require the geometric model of the object, but the analytic form of the image Jacobian is often used in the control loop. This image Jacobian contains the intrinsic camera parameters and often a 3D parameter expressed in the camera frame (*e.g.,* for point features, this 3D parameter is the depth). In this sense, IBVS depends on the parametric model of the image Jacobian which must be analytically derived for each feature type beforehand. Therefore, we consider classical IBVS as a model-based approach.[1] An issue with the classical IBVS approach is that the 3D parameter cannot be directly measured and must be estimated from images. In other words, despite knowing the parametric model of the Jacobian and a calibrated camera, the exact value of the Jacobian cannot be determined directly and must be estimated [3]. In addition, implementation of the traditional PBVS and IBVS requires the knowledge of the extrinsic calibration of the camera with respect to the end-effector.

Advanced visual servoing approaches have been proposed to address some of the above issues [4]. One of these advanced methods is the so-called UVS, which does not need the scene model or camera/robot calibration (see Section 2.4). The uncalibrated Jacobian can be estimated from the previous measurements of the visual-motor data. This sets the system free from any models or parameters. In this sense, UVS is *model-free* and *parameter-free* (see, *e.g.,* [6, 7, 9]) [2]. Such uncalibrated methods are particulary important for unconventional robots, such as a tendon-driven robot neck [60], where the analytic form of the Jacobian is tedious to derive. Throughout this thesis, we use the *uncalibrated Jacobian* in this parameter-free and model-free sense.

2.7 Summary

In this chapter, we reviewed the classic approaches to visual servoing and described the mathematical background on UVS. An interesting application of UVS is its use in uncalibrated settings and unstructured environments.

[1]Some authors refer to IBVS as model-free in the sense that it does not depend on the geometric model of the object (*e.g.,* [26]). This should not be confused with our usage of model-based that refers to a parametric Jacobian model.

[2]There are also uncalibrated IBVS methods which are model-free, but not parameter-free [27, 28, 29]. These methods study the problem from an adaptive control-theoretic point of view and estimate the linearized camera/robot calibration parameters using a depth-independent Jacobian without estimating the depth directly. See Section 2.5.3.

The classic approaches, namely the PBVS and IBVS, do not apply to uncalibrated settings. The PBVS requires the geometric model of the object and therefore is not ideal for the problem. The IBVS is more promising, however, some 3D quantities appear in the analytic form of the interaction matrix. These 3D quantities (depth, for example) need to be estimated using a calibrated camera.

Alternatively, one may use only raw input-output data (sensory-motor information) for visual servoing. This is studied in UVS, which is the approach this thesis is centred around. The UVS approach has plenty of open problems. One of the problems concern noise and outliers in the raw sensory-motor data. Visual tracking errors often play a key role in the failure of a visual servo, therefore, robustness to tracking errors should also be considered. This has motivated us to contribute by developing robust algorithms for UVS. Further details can be found in Chapter 4.

Conventional image-based methods [39] provide local asymptotic stability [11], therefore are only safe to use when the goal state is close to the initial state. There does not seem to be a general way to determine the basin of locality given a control law. Recent work has studied and analyzed different image-based control laws [61], which highlights the inherent stability problems in purely image-based control laws. For distant goal states, the tracked visual features might leave the camera field-of-view (FOV) resulting in the overall failure of the visual servoing system. In addition, waypoints in the image space should be specified. These open problems can be tackled by path planning. In Chapter 3, we give an overview of the path planning problem and mention how it improves the convergence for distant goals and solve the FOV problem with visual servoing control. In Chapter 5, we contribute by developing a new path planning algorithm for a UVS system with outliers present in the visual-motor information.

The image-based control law has known degenerate cases, for which the eye-in-hand arm fails to reach the goal state. An example of a degenerate case, is a 180° camera rotation around the view axis with a target parallel to the image plane. The image-based control law makes the camera retreat from the target, instead of rotation to reach the goal state. The choice of the visual features is important in such cases. The visual features generally affect the performance of the visual servoing system and choosing the right features for a task is another open problem in visual servoing. Image-moments have been proposed for planar [36] and non-planar objects [53] to have better performance results. More recently, Hadj-Abdelkader *et al.* [62] have proposed features from a spherical projection model for a decoupled hybrid visual servoing. These methods are either purely 2D or hybrid. In Chapter 6, we develop a new set of features that are derived from the projective geometry of three states: the initial state, the goal state, and the intermediate state. The visual features are empirically validated for a UVS system.

Chapter 3

Planning Methods in Visual Servoing

In Chapter 2, we introduced two main classes of visual servoing: position-based (PBVS) and image-based (IBVS). These approaches work well for vision-based robotic tasks if enough information about the structure of the scene is available. However, the PBVS methods are intrinsically not suitable for unstructured environments, because they depend on the geometric model of the object. On the other hand, the IBVS approaches have stability and convergence issues with their local controller, as discussed in Section 2.3. In addition, the classic approaches do not handle occlusions, field-of-view constraints, or kinematic constraints.

A path planning approach can address both the convergence problem in IBVS and handling of different constraints. In this Chapter, we review the integration of path planning and visual servoing towards improved system performance and convergence. Specifically, we study how this approach could be generalized to uncalibrated visual servoing for unstructured settings, which is our core research problem.

3.1 Overview of the Path Planning Problem in Robotics

Motion planning is one of the most well-studied yet challenging core problems in robotics. There has been numerous theoretical and practical progress over the past decades. Some of the main theoretical results are on the computational complexity of the problem [63] and why it is difficult to have a general algorithm that works for all situations. The main practical results discuss fast path planners given specific assumptions about the environment or the robot [64][65][66]. Planning is now the subject studied in many classic texts and textbooks [63][67] [68][69][51].

In path planning, a *configuration* is the location of all points on the robot. A useful representation of the configuration is the vector of joint variables $\mathbf{q}$. The *configuration space* of a robot is the set of all configurations [51]. If there are some obstacles in the workspace, the robot cannot access its entire configuration space. It can only access the *obstacle-free configuration space* (*free space* for short), where no occlusions between the robot and obstacles occur. We call the rest of the configuration space the *occupied configuration space* (*occupied space* for short). A path planning algorithm finds a path from an initial configuration to a target configuration, given obstacles in the environment and other constraints such as joint and velocity limits. This is usually done by partitioning the configuration space into the free space and occupied space and then searching the free space to find a *path* from the initial configuration to the desired configuration.

The most basic form of the path planning problem is *the generalized piano mover's problem*, where the goal is to find any collision-free paths for a free-flying rigid object consisting of connected polyhedra (a piano). This problem includes the specific case of finding collision-free paths with robots with specific kinematics, such as manipulator arms or mobile robots.

The generalized piano mover's problem was proved to be PSPACE-hard[1] by Reif in 1979 [71]. This suggests that any algorithm to generate a path may require exponential time in the worst case [63]. The early algorithms to solve the mover's problem were doubly exponential in the robot's degrees of freedom [72]. This was the state-of-the-art in the 1970's and most of the 1980's until in 1988 Canny showed that finding an exact path for the robot is only singly exponential in the dimension of the configuration space (Canny's Roadmap Algorithm) [63]. Canny's algorithm uses semi-algebraic representation of shapes to find critical points in order to find a roadmap. If we denote by p the number of constraint polynomials representing obstacles, d the maximum degree and a the coefficient of these constraint polynomials, for a robot with an n-dimensional configurations space, the running time of Canny's algorithm is [63][2]:

$$T = p^n \left(\log p (2d)^{O(n)} (\log a)^3 \, + \, (2dn)^{O(n^2)} (\log a)^2) \right) . \tag{3.1}$$

This bound can be further weakened to the product of two terms: a geometric term $O(p^n \log p)$, which grows with the number of obstacles and the robot's degrees-of-freedom, and an algebraic term
$(2dn)^{O(n^2)}(\log a)^3$, which grows with the complexity of the constraint polynomial and the robot's degrees-of-freedom, but not the number of obstacles. Both terms grow singly exponential not doubly exponential. Canny also showed that the generalized mover's problem was PSPACE-complete[3]. This was a breakthrough from the theoretical point of view because it established complexity bounds for this approach in path planning, however, the implementation is not practical in robotics [68].

Efficient algorithms to provide an exact path for the general case are not possible, because of the complexity of the problem. Optimal planning is NP-hard, even for a point robot without dynamics moving in a 3D polyhedral environment [67]. There are always heuristics and assumptions on the environment, configuration space, and the robot in order to be able to solve the problem in polynomial time. In addition, it may not be feasible to explicitly find the free space representation for the purpose of path planning. This has lead to the development of two main methods for path planning, which we review in the following sections: potential fields and sampling-based methods. A third class of methods for path planning is based on

[1]PSPACE is the class of decision problems that are decidable in polynomial space on a deterministic Turing machine [70]. If a problem B is PSPACE and every A in PSPACE is polynomial-time reducible to B, then B is PSPACE-complete. PSPACE is known to be an improper subset of EXPTIME: PSPACE $\subseteq$ EXPTIME. EXPTIME the class of all decision problems which can be solved by a deterministic Turing machine in time $O(2^{f(n)})$, where $f(n)$ is a polynomial function of n.

[2]The running time is simplified as $p^n \, \log(p) \, d^{O(n^4)}$ by Choset *et al.* [68].

[3]A special instance of the generalized mover's problem, the *Sokoban*, was independently proved to be PSPACE-complate by Culberson [73].

cell decomposition. We omit the discussion because of their poor performance for real-time replanning in dynamic environments.

A path planning algorithm that provides only the solution for a specific instance of the problem (a specific query of initial and desired configurations), if such a solution exists, is called a *single-query* algorithm. Some algorithms can be used to solve different instances of the same planning problem. Such an algorithm is *multiple-query.*

3.1.1 Potential fields methods

A scalar *potential field* is a differentiable single-valued function $U : \mathbb{R}^n \to \mathbb{R}$. Consider a robot as a point particle in the configuration space, which moves according to the torques generated from the potential field $U(\mathbf{q})$. The value of U can be viewed as energy and hence its gradient results in torques $t\tau$ [51]:

$$t(\mathbf{q}) = -\nabla U(\mathbf{q}). \tag{3.2}$$

For path planning methods based on potential fields, the potential field U is constructed such that the resulting torques move the robot towards the desired configuration $\mathbf{q}_d$, where $U(\mathbf{q}_d)$ is globally minimized. Therefore, the path planning problem is an optimization problem and the solution is the path followed by the gradient descent algorithm. The optimization algorithm stops when $\nabla U(\mathbf{q}^*) = 0$, where $\mathbf{q}^*$ is a *critical point* [4] of U.

The potential field can be chosen as the sum of an attractive field $U_\oplus(\mathbf{q})$ that moves the robot greedily towards $\mathbf{q}_d$ and a repulsive field $U_\ominus(\mathbf{q})$ that repels the robot from obstacles:

$$U(\mathbf{q}) = U_\oplus(\mathbf{q}) + U_\ominus(\mathbf{q}). \tag{3.3}$$

An example for the attractive field [68, 51] is $U_\oplus(\mathbf{q}) = \frac{1}{2}\,\zeta\, d^2(\mathbf{q}, \mathbf{q}_d)$, where $d(.,.)$ is a distance such as the Euclidean distance and ζ is a parameter that controls the attractive-ness of the desired configuration. The repulsive potential is usually defined for the obstacles in the robot workspace coordinates. For example, consider the safe distance to navigate around the obstacle to be ρ_0 and the closest point on the robot to have a distance $\rho(\mathbf{q})$ to the obstacle. If $\rho(\mathbf{q}) > \rho_0$, there is no need for a repulsive field. For $\rho(\mathbf{q}) \leq \rho_0$, the repulsive field [68, 51] can be defined as $U_\ominus(\mathbf{q}) = \frac{1}{2}\,\eta\,(\frac{1}{\rho(\mathbf{q})} - \frac{1}{\rho_0})^2$, where η determines the gain of repulsive torques.

The main problem with this kind of approach is that the gradient descent may converge to a local minimum, not a global minimum. This happens when the attractive and repulsive torques cancel out each other and the robot gets stuck in a local minimum. This is either a result of the placement of the obstacles and the desired configuration, or poorly tuned parameters (ζ, η, etc.).

To leverage the problem with local minima, Barraquand *et al.* propose a randomized potential-fields approach that generates random walks to escape the local minima [74]. This algorithm is an example of a larger class of algorithms, where the potential field or the roadmap is augmented with a search-based planner [68, 51], such as the general Randomized Path Planner (RPP) [75].

[4]A critical point of U is either a minimum, a maximum, or a saddle point. One can look at the second derivative (Hessian) of U to learn more about the critical point. For a non-singular Hessian, a positive-definite Hessian corresponds to a local minimum and a negative-definite Hessian corresponds to a local maximum[68].

Another solution comes from the idea behind the navigation functions [76, 77], where the potential field is defined with only one minimum [68]. The disadvantage of the navigation function approach is its requirement to have a completely known and stationary environment [77].

The planning algorithms based on both the potential fields and their variant, the navigation functions, are single-query algorithms. Later in Section 3.4, we review two important contributions in IBVS that use the potential fields [12] and the navigation functions [78].

3.1.2 Sampling-based methods

A *roadmap* in the configuration space is the set of collision-free paths in the free space that includes all possible initial and desired configurations. A complete roadmap represents the connectivity of the free space [69]. The theoretical results by Canny [63], which were discussed in Section 3.1, were also based on the roadmap concept. There are different types of representations for a roadmap (visibility map, generalized Voroni diagram, etc.) with a corresponding graph representation (visibility graph, generalized Voroni graph, etc.) [68]. Early path planning algorithms based on the roadmaps idea have two components: (1) Constructing the corresponding graph representation of the roadmap, and (2) searching the graph for paths from the initial configuration vertex to the desired configuration vertex. A trivial algorithm is to use an unweighted graph representation and use basic search algorithms such as Breadth-First or Depth-First Search (BFS, DFS). One can also assign weights to edges based on distance in the configuration space. In this case, a heuristic search method like $A*$ or its variants can be used to find the minimum-length path [69]. Once the graph is constructed, the search algorithm can find paths for any queries, therefore these methods are multiple-query. A review of the roadmap representations and deterministic algorithms is beyond the scope of this document and we refer the reader to [68] and [69] for details. For our purpose, it suffices to mention that finding the complete roadmap is inefficient and must be repeated with any changes in the environment.

The roadmap can also be constructed probabilistically. Kavraki *et al.* propose the Probabilistic RoadMap planner (PRM), where the roadmap is constructed by sampling the free space and connecting the resulting configurations by a fast local planner [64]. If a roadmap cannot be found, the algorithms returns failed. Once the roadmap is constructed, a graph search algorithm can find paths for multiple queries of initial and desired configurations. Roadmap planners are offline planners that find the roadmap given the environment, the robot, and some constraints. The more realistic the constraints are, the more challenging it is to find a roadmap. Motion planning problems for robots with 5 or more DOFs might require very large amounts of memory to store the thousands of configurations to be connected for a solution with the standard PRM planner. In addition, many authors report that the standard PRM planner is infeasible for some hard problems (see *e.g.,* [34]). An example of a hard problem is the kinodynamic planning problem, where both the configuration and velocity of the robot is planned. Although the standard PRM planner is not suitable for kinodynamic planning, it could be used to solve it after some modifications. Tsianos and Kavraki compute the sampling-based motion trees for problems in static environments incrementally through consecutive replanning steps [79]. These motion trees are then evaluated using a navigation function to choose the overall path. One of the important attributes of this replanning approach is its bounded memory. This approach suits environments with dynamic obstacles, however, finding the right navigation function could be difficult.

Randomized sampling techniques could be utilized to help with some of the shortcomings of other methods. We mentioned the RPP algorithm at the end of Section 3.1.1, which is used to help a potential fields-based path planner stuck in local minima. Sampling-based planners have many attractive characteristics [68]. For example, they can work without an explicit free space and they are not required to have complete knowledge of the obstacle boundaries in the configuration space. Most sampling-based planners are *probabilistically complete* in the sense that if a solution path exists, the probability of finding the solution goes to 1 as the number of samples goes to infinity. Another nice property is their modular algorithms, where several modules such as collision/obstacle avoidance can be integrated without too much effort. For all these reasons, randomized path planning algorithms have become popular to solve single-query path planning for challenging high-dimensional problems within the past decade.

One of the major achievements in sampling-based path planning is based on a randomized data structure, the rapidly-exploring random tree (RRT) [32]. A useful property of the RRT is that it expands in the unexplored regions of the state space. The RRT is designed to handle a multitude of path planning problems: holonomic, nonholonomic, and kinodynamic. It can also be used in planning problems with high-dimensional state spaces, which is an improvement compared to PRM for many practical applications. Kuffner and LaValle extend the basic RRT algorithm to develop RRT-connect, which can be used in single-query path planning problems with no differential constraints. Since there are no differential constraints, the problem can be expressed in the configuration space [66]. RRT-connect works by growing an RRT from the initial configuration and another RRT from the desired configuration. The trees use a greedy heuristic to advance towards each other. When the two trees join each other, a solution to the path planning problem is found. The state-space variant of the RRT-connect is called the bidirectional RRT or BiRRT algorithm [34]. LaValle and Kuffner use BiRRT to solve the challenging problem of kinodynamic planning, where dynamic constraints are also taken into account [34]. We will spend some time to review the basic BiRRT algorithm, because we refer to RRT-based visual planners in multiple occasions in the remainder of this section. Also, we will later propose to extend the concept of RRT to develop an algorithm, which suites vision-based motion control in unstructured environments.

Our presentation of the BiRRT algorithm is based on [34]. The basic BiRRT algorithm (see Algorithm 2) starts by initializing tree T_a at the initial state x_{init} and tree T_b at the initial state x_{goal}. Then a random state generator, RandomState(), samples the unexplored state space and generates x_{rand}. Now, this newly generated state should be added to the RRT. This is done by extending the tree T_a towards x_{rand} by another function ExtendRRT. The algorithm for ExtendRRT is presented in Algorithm 1. First, x_{near}, the nearest vertex or edge of the current tree to x_{rand} is determined by NearestNeighbor(). A naive algorithm would just connect x_{near} to the new state to expand the tree. However, one may consider a local controller, LocalController(), which extends the x_{near} state towards x_{rand}. This would make the integration of differential constraints or other constraints into the algorithm possible. In Figure 3.1, the local controller is shown by an arrow. The resulting new state, x_{new} is resulted from applying the control signal, u_{new}, from the local controller for a short period of time, Δt. If the resulting new state, x_{new}, avoids collisions with obstacles, it may be added to the tree. The control signal is applied until x_{new} is reasonably close to x_{rand} (the algorithm returns REACHED) or a maximum time of t_{max} is reached. If x_{rand} is not reached, but a new state is generated by advancing x_{near}, the algorithm returns ADVANCED. If the new state cannot be added to the tree, the algorithm returns TRAPPED. The function of ExtendRRT is also shown in Figure 3.1. When the two trees of T_a and T_b have a common vertex, a path from x_{init} to x_{goal} is returned by Path() (see Algorithm 2).

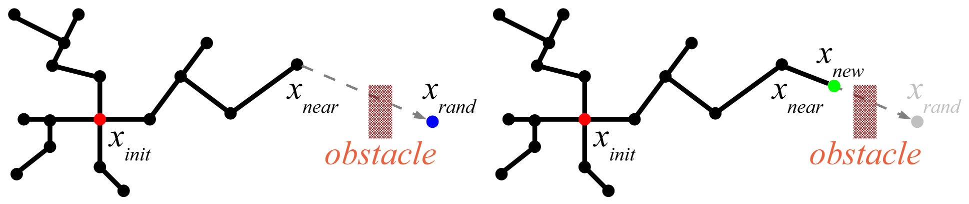
Algorithm 1 ExtendRrt(T, $\mathbf{x}_{rand}$)

```
1:  $\mathbf{x}_{near} \leftarrow$ NearestNeighbor($\mathbf{x}_{rand}$, $T$)
2:  $i \leftarrow 1$
3:  $\mathbf{x}_i \leftarrow \mathbf{x}_{near}$
4:  while $i < MAX$ do
5:      $\mathbf{x}_{i+1} \leftarrow$ LocalController ($\mathbf{x}_i$, $\mathbf{x}_{rand}$)
6:      if Collision($\mathbf{x}_{i+1}$) then
7:          $\mathbf{x}_{new} \leftarrow \mathbf{x}_i$;
8:          flag $\leftarrow$ TRAPPED
9:      else if $||\mathbf{x}_{i+1} - \mathbf{x}_{rand}|| < \epsilon$ then
10:         $\mathbf{x}_{new} \leftarrow \mathbf{x}_{i+1}$;
11:         flag $\leftarrow$ REACHED
12:     else
13:         flag $\leftarrow$ ADVANCED
14:     end if
15:     if flag is REACHED or TRAPPED then
16:         $T.add(\mathbf{x}_{new})$
17:         return flag
18:     else
19:         $\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}$; $i \leftarrow i + 1$
20:     end if
21: end while
```

We have implemented a 2D version of the BiRRT algorithm to show how the algorithm works in practice. Figure 3.2 shows successful path planning with 2, 3, and 4 arbitrary walls as obstacles. As seen in this figure the final path could further be shortened by a simple algorithm, which finds a shorter path from the initial path. In these examples a uniform sampling has been used and the path is found before the maximum iteration of $K = 500$. It is easy to see that as the environment becomes more complex, a uniform sampling scheme might not work. In Figure 3.3, we consider 5 walls as obstacles with a narrow passage for the middle wall. This is an extremely challenging case for any path planning algorithms. Both trees (from the initial state and the goal state) are extended rather uniformly in the free space, however, they have not joined each other through the narrow passage in the middle. Figure 3.3 (Left) shows a snapshot of the RRTs at step 269 (almost half way through) and Figure 3.3 (Right) shows failure at the maximum step ($K = 500$). To avoid

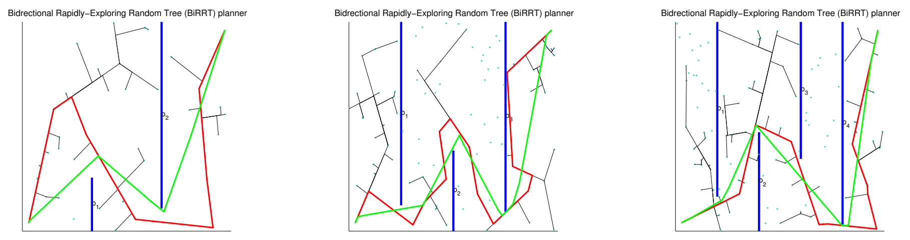
Algorithm 2 BiRrt($\mathbf{x}_{init}, \mathbf{x}_{goal}$)

```
1:  $T_a.init(\mathbf{x}_{init})$
2:  $T_b.init(\mathbf{x}_{goal})$
3:  for $k = 1$ to $K$ do
4:      $\mathbf{x}_{rand} \leftarrow$ RandomState()
5:      if ExtendRrt($T_a$, $\mathbf{x}_{rand}$) is not TRAPPED then
6:          if ExtendRrt($T_b$, $\mathbf{x}_{new}$) is REACHED then
7:              return Path($T_a$, $T_b$)
8:          end if
9:      end if
10:     Swap($T_a$, $T_b$)
11: end for
12: return FAILURE
```



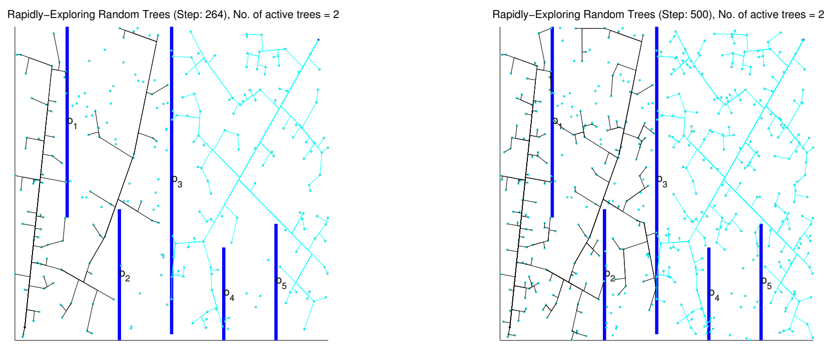
(a) $\mathbf{x}_{near}$ starts extension. (b) Extension stops at $\mathbf{x}_{new}$.

Figure 3.1: A schematic diagram for the classic ExtendRRT algorithm. (a) The RRT is first built from the initial node, x_{init}. The random state x_{rand} is sampled next and the nearest node from the tree is found as x_{near}. The tree should not be naively extended from x_{near} to x_{rand} by adding an edge, as the edge does not necessarily correspond to the free space. Instead, the LocalController is called to extend x_{near} toward x_{new} checking for collisions at every control iteration. (b) After the extension cannot be further advanced, or has reached x_{rand}, the algorithm returns the resulting final state as x_{new} and adds it to the tree.



(a) BiRrt with 2 walls (b) BiRrt with 3 walls (c) BiRrt with 4 walls

Figure 3.2: 2D BiRRT planner examples. The start node is at the bottom left corner (initial node for the first tree) and the goal node is at the top right corner (initial node for the second tree). The two trees connect once the have a common node (red path). A simple algorithm shortens the path at the end (green path). (a) Two obstacle walls, (b) three obstacle walls, and (c) Four obstacle walls. Figure best seen in colour.


Rapidly–Exploring Random Trees (Step: 264), No. of active trees = 2
Rapidly–Exploring Random Trees (Step: 500), No. of active trees = 2
o_1
o_2
o_3
o_4
o_5

(a) Explored space after 264 steps. (b) Explored space after 500 steps.

Figure 3.3: BiRrt planner may fail with narrow passages. The start node is at the bottom left corner (initial node for the first tree) and the goal node is at the top right corner (initial node for the second tree). There are five obstacle walls in this example with a very narrow passage in the middle wall. (a) The tree progression after 264 steps. (b) The tree progression after 500 steps. At this point, the algorithm returns FAILURE, as the maximum number of iterations is set as $K = 500$ in this example. Figure best seen in colour.

such results, biased sampling should be used. The bias is usually chosen to sample more from the problem areas, but this requires full knowledge of the environment and obstacles.

3.2 Path planning for visual servoing

Some of the shortcomings of the visual servoing approaches were outlined in Chapter 2. For example, the visual features may leave the field-of-view (FOV), which leads to the failure of visual servo. If the initial and desired states are not reasonably close to each other, IBVS may fail because the controller ensures only local asymptotic stability. Another failure is due to the possibility of either the interaction matrix or the kinematic Jacobian become singular during servoing. Generally, one can consider two main reasons for the failure of the visual servo: visual constraints (FOV, interaction matrix singularity, visual occlusions, etc.) and configuration constraints (kinematic/dynamic singularity, obstacles, etc.).

Path planning approaches may be used to overcome the shortcomings of the IBVS approach. A recent article by Kazemi *et al.* [10] surveys planning algorithms for visual servoing. They classify the literature as (1) image-space path-planning (epipolar geometry, projective homography, projective invariance), (2) optimization-based path-planning, (3) potential field-based path-planning, and (4) global path-planning. There is some overlap between these classes, for example, some of the potential field-based approaches [12] are also image-space planning methods. Another example is an optimization-based planning [80], which is also a global path-planning algorithm. Hence, we present a different review of the literature on path-planning algorithms for visual servoing.

We study the methods that perform the planning in the Cartesian space of camera pose as position-based path planning (Section 3.3). Although the planned trajectory might not be necessarily followed by a PBVS controller, in principle it could be as the pose of the camera along the trajectory is known. Some of these methods try to optimize a cost function, while others are efficient sampling-based methods implemented in the configuration space or the Cartesian camera space. A typical requirement of these methods is the geometric knowledge of the scene and obstacles.

On the other hand, some methods provide image-space trajectories that are followed by an IBVS controller. We study these papers under image-based path planning in Section 3.4. Some methods use both the image-space trajectories and Cartesian camera trajectories, in which case we have looked at the controller to classify them as either image-based or position-based. There are three main classes of image-based path planners: projective-geometric, potential fields and navigation functions, and sampling-based. This classification allows to study sampling-based methods for position-based and image-based approach, separately.

3.3 Position-Based Path Planning

To ensure the FOV constraint for PBVS, usually the geometric model of the object and the obstacle is required, in addition to the camera calibration parameters. Therefore, this approach is not suitable for operation in unstructured environments. Nonetheless, our review of the path planning approaches in visual servoing would be incomplete without inclusion of position-based methods because some of these methods require a minimal knowledge of the parameters and may be applied in some semi-structured environments. We distinguish two major approaches, both of which satisfy the visibility constraint: methods that use the PBVS control law alongside an optimal camera path and methods that use sampling-based planning techniques in the configuration space.

3.3.1 Planning for the optimal camera path

Kyrki *et al.* proposes a new PBVS approach that ensures visibility, while servoing along a straight line [81]. While not a classical path-planning approach, the generated trajectory has the shortest translation. They separately control the translation and orientation of the camera. For the translation they use the classical PBVS controller (as discussed in Section 2.2). For the orientation, they define a virtual 3D point at the center of the object and use 2 DOFs to control its projection lies on the view axis. They use the final DOF to control rotation around the view axis. This approach requires knowledge of 6D pose, which is found from a known geometric object model. They also apply the same technique in conjunction with controllers derived by Euclidean homography decomposition (see Section 3.4.1). This relaxes the dependency on the object model, however a calibrated camera is still required and the depth of the feature points to the camera must be known or estimated. This technique cannot be easily extended to include scene obstacles.

Chesi and Vicino present a method that generates circular-like image trajectories with short translational motions to ensure visibility constraint and global asymptotic stability over large displacements [82]. The CAD model of the object is not used, but a calibrated camera is used. The uncertainty bounds for extrinsic and intrinsic camera parameters is also provided. Given the camera calibration and point correspondences between the initial and desired images, they estimate the Essential matrix between the two views, from which it is possible to estimate the rotation and (normalized) translation. This work uses an angle-and-axis representation for the rotation with the following control strategy: the optical axis is rotated between the initial and the desired camera frames and at the same time the camera frame is rotated along the optical axis to compensate for any discrepancies in rotation angles. For the translation control, instead of moving along the straight path of the frame centers, the camera is moved with an angle such that the camera views the object along the path. This results in a circular-like curve. Although it is not a path planning paper per se, it lays out the foundations for a path-planning paper by Chesi and Hung [83]. In addition, Chesi and Vicino provide a comprehensive comparison with other VS methods (IBVS, 2.5D VS [24], PBVS in [25] PBVS in [84]).

The work by Chesi and Hung [83] develops a global path planning for visual servoing that considers joint, workspace, and FOV constraints while providing a framework to optimize the camera trajectory in some sense (shortest length, curvature, etc.). They use images and the object model to reconstruct the 6D pose (rotation and translation). If the CAD model is not available, a robust object reconstruction method can be used. The 6D Cartesian camera trajectory is parameterized by two polynomials: one for rotation and one for translation. This method finds the set of all polynomials from the initial camera frame to the desired frame, considering the constraints. The optimization with respect to a proper cost function is then carried out to reach the optimal path. Once the optimal path is found, it is tracked by an IBVS tracker which also takes the FOV and joint constraints into account similar to Mezouar and Chaumette [12]. The advantage is that an optimal path in the Cartesian camera space is found, however, the success depends on limited calibration errors and numerical stability in the object reconstruction phase. It also depends on the knowledge of obstacle pose to consider the workspace constraints. Chesi has extended their previous method to parameterize the camera path with Homogeneous Forms and Linear Matrix Inequality (LMI) optimization [85]. This parametrization is more general than the previous case and allows for using the more powerful LMI optimization tools. The constraints for the camera path are derived in terms of the positivity of a homogeneous term. A very nice property of this method is that it turns the constrained path planning problem into a convex optimization problem (with a unique solution that can be found efficiently). Although the methods by Chesi *et al.* use a local IBVS control law to *track* the projection of the planned trajectory, the trajectory itself is planned in the Cartesian camera space and could be tracked by any visual servo including a PBVS controller. This is why we have reviewed these methods in a Path Planning for PBVS section.

3.3.2 Sampling-based planning in the configuration space

For eye-to-hand robots that may occlude their targets, or an eye-in-hand robots with dynamic objects in the scene, an efficient occlusion checker is needed. Dynamic occlusion checkers have been extended from dynamic collision checkers that determine all collision-free configurations along a path. Schwarzer *et al.* developed an adaptive dynamic collision checker (DCC) algorithm [86]. Collisions are efficiently detected from samples in the configuration space with local adjustments to the sampling resolution. The DCC algorithm in [86] is particularly suitable to be used with the PRM planner for manipulators with many moving parts. This is due to the requirement of the PRM planner to determine efficiently whether the given path segments collide. Leonard *et al.*have extended the DCC to include dynamic visibility checking (DVC) [87]. An occlusion-free path is generated by a motion, which does not result in the occlusion of the object in the image (by another object, but not the robot) and does not make the object image to leave the FOV. They use the DVC algorithm in conjunction with the PRM planner on an eye-in-hand robot. The edges in the PRM graph represent the occlusion-free motions. Using the DVC with the planner results in a position-based control that satisfies the FOV constraint. In a similar paper, Baumann *et al.* exploit the CAD model of the object and obstacle for visibility checking and propose the vision-based PRM (VBPRM) planner for an eye-in-hand system [88]. These algorithms require a calibrated camera and usually the CAD model of the object and obstacle. They also do not discuss self-occlusion. However, self-occlusion might happen to an eye-to-hand system (fixed camera looking at the robot). Leonard *et al.*extend their previous visibility checking algorithm of [87] to propose a dynamic occlusion checker (DOC) algorithm with the PRM planner for eye-to-hand systems [89]. Occlusions are modeled as collisions between an object and the frustum of a pixel. The DOC algorithm determines which pixels are occluded during motion and addresses visual self-occlusions of the object by the robot arm.

3.4 Image-Based Path Planning

IBVS is a local feedback control scheme. Therefore, there is no convergence guarantee when the initial and target states are largely distant. Another eminent problem in IBVS is the possibility that the tracked features may leave the FOV, because there is no FOV constraints embedded in the IBVS control law. These types of problem may be tackled by path planning in the sensor space. There are three main approaches: projective-geometric methods (including the projective reconstruction of the camera trajectory and homography interpolation), potential fields and navigation functions, and sampling-based planning in the image-space. Such approaches usually have some assumptions on the uncertainty of the camera calibration and depth estimation. Another approach which is more suitable, but less treated in the literature, is completely uncalibrated image-based planning.

3.4.1 Projective-geometric methods

Concepts from projective geometry can be used to design trajectories in the image space. Some of these methods are based on the epipolar geometry, which describes the geometry of two views. The two views could either be obtained from a stereo rig (both eye-in-hand and eye-to-hand configurations), or from the initial and desired images for the eye-in-hand configuration. Another projective-geometric concept, which is used here is the projective transformation that maps geometric objects (points, lines, etc.) to corresponding geometric objects in the projective space. For example, a *homography* is a matrix that maps a set of geometric objects of projective

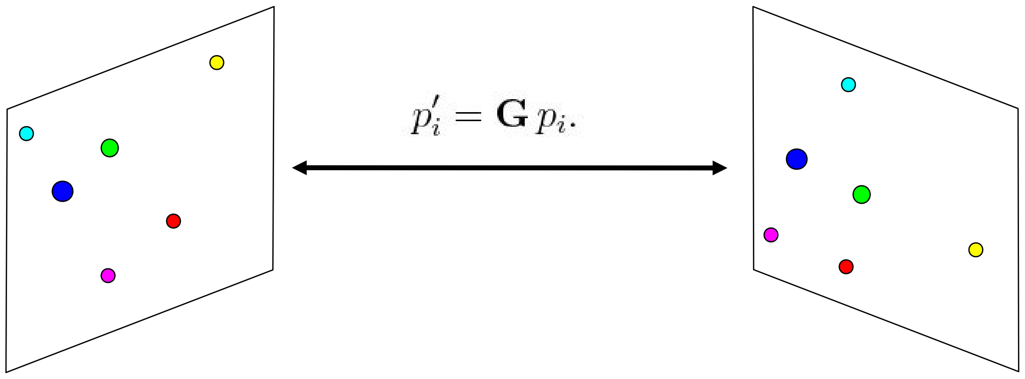


Figure 3.4: Projective homography maps homogeneous coordinates from one image to another image. The Euclidean homography in (3.5) is the calibrated version of the projective homography as shown in (3.4). It can be decomposed into a rotational term and a translational term as in (3.6) for planning.

space $\mathbb{P}^2$ in one view to another view. A *collineation* or a *projective displacement* maps geometric objects of projective space $\mathbb{P}^3$ to each other.

The major idea in image-space path planning is to decompose and interpolate the homography or the projective displacement such that the corresponding camera motion can be realized by a local IBVS controller, while the visibility constraints are satisfied. The following brief presentation, illustrates the decomposition of homography matrix into rotational and translational terms.

Let $p_i = [u_i,\, v_i,\, 1]^\top$ and $p'_i = [u'_i,\, v'_i,\, 1]^\top$, with $i = 1, \cdots, M$, denote the homogeneous coordinates of the initial and desired images of M feature points, respectively, as seen in Figure 3.4. Homography matrix $\mathbf{G}$ relates these coordinates according to

$$p'_i = \mathbf{G}\, p_i. \tag{3.4}$$

If the intrinsic camera calibration matrix $\mathbf{K}$ is available, the Euclidean homography $\mathbf{H}$ could be found up to a scalar factor k:

$$k\,\mathbf{H} = \mathbf{K}^{-1}\mathbf{G}\mathbf{K}. \tag{3.5}$$

The Euclidean homography can be decomposed into a rotation matrix $\mathbf{R}$ and a translation $\mathbf{t}$ between the initial and desired frames:

$$\mathbf{H} = \mathbf{R} + \frac{\mathbf{t}}{d}\mathbf{n}^\top, \tag{3.6}$$

where d is the distance between the plane of coplanar feature points and the origin of the initial camera frame.

Malis *et al.* [24] have used this homography decomposition to decouple the control signal of the translational velocities from the rotational velocities. This scheme is usually called the hybrid visual servoing or 2.5D visual servoing. Malis and Chaumette [26] have shown that the 2.5D visual servoing has improved performance compared to both the IBVS and PBVS. Deguchi [90] has also used a similar decoupling of the translational and rotational velocities similar to 2.5D visual servoing with the epipolar constraint to generate a camera trajectory. In the context of planning in the image space for IBVS, the homography decomposition can be used to interpolate the rotational and translational terms separately as will be discussed next.

Planning by the interpolation of the projective homography

Borgstadt and Ferrier [91] develop a path interpolation algorithm for a monocular eye-to-hand robot with a known 3D model for the object rigidly attached to the end effector. Their method uses a canonical projection of the object onto the image plane, which must be known *a priori*, to interpolate the 2D homography in (3.4) between the images of the object at the initial and desired configuration. Their proposed framework is limited not only because the 3D object model must be known, but also in the sense that robot physical constraints, or FOV constraints cannot be integrated.

Allotta and Fioravanti [15] propose a 3D camera path planning approach for a monocular eye-in-hand robot with 4 coplanar feature points. Their method does not depend on the 3D object model, but requires the camera intrinsic parameters for Euclidean reconstruction. They interpolate the Euclidean homography in (3.5) to generate a scaled 3D camera trajectory. The smoothness of the 3D camera trajectory is guaranteed by choosing a circular helicoidal trajectory[5] Then, they modify the trajectory to ensure FOV and follow the image-space trajectory by a standard IBVS control law.

Mezouar and Chaumette [92] propose a method to generate image-space trajectories, which are derived to ensure an optimal Cartesian camera trajectory. Their approach is based on an implicit decomposition of the Euclidean homography in (3.5) without exact knowledge of the calibration matrix or object model. They extend their optimal trajectory generation to include image-space and joint-space constraints using the potential fields. This method generates a smooth and optimal collineation path between the initial and desired images. It is uncalibrated in the sense that exact camera calibration or object model are not required. This is an improvement to their earlier potential field-based path planning method [12], where the optimality of the Cartesian camera trajectory is not guaranteed (potential fields-based path planning was discussed in Section 3.1.1).

Schramm and Morel [93] address the FOV issues by ensuring visibility, while planning the image trajectories for an uncalibrated eye-in-hand manipulator. The projective displacement is decomposed into a term corresponding to translation and another term corresponding to rotation. The parametric forms of each term is recovered by assuming an auxiliary image obtained from a pure translation from the desired image. This assumption is somewhat limiting because such an image might not be available. The object depth is recovered, up to a scale, from the auxiliary image and the desired image using 4 or more non-coplanar points. This depth is then used to identify the parameters of the projective displacement from

[5]A *helix* is a smooth 3D curve that has a constant angle between the tangent and the helix axis at any given point on the curve. A circular helix is defined by $(a\cos(t), a\sin(t), b\,t)$, where a defines the helix radius and $2\pi\,b$ defines its pitch. A coil spring is an example of a circular helix. See `http://en.wikipedia.org/wiki/Helix`.

6 or more points. The resulting projective displacement can be interpolated by interpolating the translation component linearly and the rotation component using eigen-decomposition. The planned image trajectory is then found from the projections of the points using the interpolated projective displacements. To include the FOV constraint in the same framework, the authors translate the camera away from the object and analytically find the corresponding image trajectories that ensures visibility for the whole target. A nice property of this framework is that it can be extended such that a desired image point follows a straight line, while all other points satisfy the FOV constraint. While this is an interesting approach, the planned trajectory is not guaranteed to generate feasible camera motions. This approach is based on the assumption that the projective displacement can be recovered in the presence of measurement errors. In our view, this depends on the data and the type of the measurement errors, therefore, this might be a limitation if there are systematic outliers in the visual data (see Figure 1.1 for examples of such outliers). Also, consideration of the physical constraints (such as straight camera trajectory) is not straightforward as shown by the authors in a later paper [94].

Image-space planning using stereo eye-to-hand cameras

Park and Chung [95] use uncalibrated stereo eye-to-hand and plan image trajectories that move the robot on a straight line towards the goal for a large pose alignment and grasping task. FOV constraint is implicitly satisfied, because the initial and desired images are both within the field of view and the image path is a straight line. The epipolar geometry of the stereo rig is used to reconstruct 3D projective coordinates up to a projective transformation. The plane at infinity is used to interpolate the projective transformation, which ensures pure translation of projective points on the gripper. No 3D information about the gripper or the object is assumed. However, self-occlusion is not addressed, which is not a practical assumption for this kind of task. This framework does not automatically incorporate the configuration constraints, such as kinematic singularity. The authors extend their previous work to include kinematic constraints into account [96], but this requires an additional orientation generating operator, calculation of its Jacobian, and control of the orientation in the null space of the Jacobian to avoid kinematic singularities. While the initial idea by Park and Chung [95] is elegant, the extension seems to be unnecessarily overcomplicated and yet it does not address self-occlusion.

Hosoda *et al.* [13] propose another image-space path planning approach for stereo eye-to-hand robots. The obstacle-free path is generated from images using the epipolar constraint, which can be found from point correspondences between the camera images. The camera are assumed to be rigidly attached to the stereo rig with calibrated parameters. This scheme is based on the following heuristic for an occlusion-free camera path: if the image path in at least one of the cameras is obstacle-free in the image space, then the camera path is occlusion-free. Based on this simple heuristic, they plan around the obstacle in one of the images along the epipolar lines and use a straight image path in the other image. Although 3D reconstruction is not required and image-space path following is performed with an uncalibrated visual servo similar to [6], this scheme would only work for simple scenarios where the heuristic results in feasible camera trajectories (free of joint constraints, velocity limits, etc.).

3.4.2 Potential fields and navigation functions

Two major approaches use potential fields [12] and navigation functions [78] for path-planning in image space with constraints (FOV, joint limit avoidance, etc.). The potential fields-based methods are prone to local minima, while the navigation functions have a unique global minimum. However, derivation of the navigation function for a new problem is cumbersome and in some cases not easy to accomplish.

Mezouar and Chaumette [12] provide an interesting path-planning solution based on the potential fields for eye-in-hand IBVS. Their main contribution is to provide a feasible trajectory in the image space that can be followed by the local IBVS controller, even under poor camera calibration accuracy. They consider two types of constraints: FOV and joint-limit avoidance. However, one of the problems that they do not address is the local minima problem of the potential fields-based path planning methods. This problem can be solved by designing a proper navigation function [76], which has a unique minimum at the desired configuration, as shown by Cowan *et al.* [78]. The navigation function-based method requires some knowledge of the scene to design a subset of the occlusion-free configurations. This subset should be conservative to result in safe motions with respect to visual occlusions. In addition, this method requires approximate kinematic calibration and intrinsic camera calibration, although some robustness to uncertainty can be established through the closed-loop feedback. While the theoretical foundations are sound, only examples of eye-to-hand position control, and 6 DOF eye-in-hand servoing have been empirically validated for FOV constraint and visual self-occlusion. The derivation of the such navigation functions to include other obstacles in the scene is not straightforward. Other authors have also reported that the navigation function-based planning cannot be easily extended to more complex environments [10]. While the potential fields-based and the navigation functions-based methods address many concerns of IBVS, they do not seem suitable for unstructured and uncalibrated settings.

3.4.3 Sampling-based planning for IBVS with 3D object model

Sampling-based planning has been recently used with the IBVS scheme. Chan *et al.* [97] use a calibrated camera and the 3D object model to estimate object visibility at different camera poses. If the initial and desired configurations are distant and a feasible trajectory cannot be executed by the local IBVS control, the PRM is used to find intermediate configurations (waypoints) satisfying the FOV constraint. The sampling is performed by a virtual IBVS controller (this requires a model-based approach with knowledge of the camera calibration, object, etc.). Since the resulting path is a concatenation of several smooth paths, where the robot comes to a stop at each waypoint, a further interpolation in the robot joint space is required.

Another sampling-based planning algorithm that is used with IBVS is the RRT planner. RRT-like planners usually assume some knowledge of where the robot is with respect to a model of the environment. Although these assumptions are fairly relaxed, adopting the RRT-like planners to sensor-based planning (not configuration-space planning) is not straightforward. Assume that you have a serial link manipulator consisting of a series of revolute joints with a camera rigidly attached to the last link. The camera is the only sensor to be used in this setting (eye-in-hand configuration). At each control step the robot only has a partial view of the whole scene. Moreover, there are known singularity an local minima problems for the eye-in-hand robot control [11]. In essence, the image-based controllers that are used for visual servoing of such a system are only locally asymptotically stable. This means that the goal state must lie inside the local basin of convergence. There are no easy ways to determine the boundaries of such a basin. However, one can decompose a seemingly global task into a series of local tasks each of which are guaranteed to converge.

Kazemi *et al.* [33] considers the problem of planning for eye-in-hand IBVS with several constraints, including FOV, visual occlusion by workspace obstacles or self-occlusion, and joint constraints. They build the RRT by iteratively exploring the camera space and the image space. The RRT is first sampled in the Cartesian camera space. Given the 3D object model, the project the feature points onto the image at this random state. If the image-space constraints are all satisfied, the RRT is extended by a local IBVS controller. Similar to Chan *et al.* [97], the final camera trajectory should be further checked for feasibility. Alternatively, randomized kinodynamic planning [34] could be used by including the velocities in the state variable [98].

It is not clear how much modeling noise can these sampling-based methods handle with an IBVS controller. There is also no justification for using the IBVS scheme over the PBVS, if the 3D object model and camera calibration is available. Most PBVS methods perform better with a perfect knowledge of geometry and calibration. Also, the optimal configuration-space planning method discussed in Section 3.3 provide optimality with some sense, whereas RRT-based methods provide a feasible solution, not an optimal one. The bottom line is that these approaches assume to have the geometric model of object and the camera intrinsic parameters *a priori*. This makes them useless for unstructured environments.

3.5 Summary

In this chapter, we presented the mathematical background on path planning and reviewed the applications of path planning in visual servoing.

There are several papers on path planning for visual servoing, but only a handful of them consider an uncalibrated system without any additional information on the object model, depth from object, obstacle model, and robot/camera calibration. The early work of Hosoda *et al.* [13], is limited in the sense that it cannot be generalized to other image features, hand/eye configurations, or more realistic unstructured environments. Other recent methods [14, 15] rely on partial scene reconstruction or homography interpolation, which are ill-posed problems in the case of unknown scene model. The uncalibrated approach is still an open problem in path planning.

It is desirable to design planning algorithms that depend only on the raw sensory-motor data. Ideally, the planning algorithms should also be statistically robust and reject outliers due to visual tracking error, disturbances, etc.

The sampling-based path planning approach is a viable option for image-based visual servoing systems. In particular, RRT-based planners have desirable properties including a global solution (if one exists) and efficient implementation. RRT-based planners have been successfully used in eye-in-hand 6 DOF image-based visual servoing by Kazemi *et al.* [33]. Although the approach of Kazemi *et al.* is appealing because of the nice properties of RRT, it assumes a known geometric model of the object and known camera intrinsic parameters. Therefore, their approach is not completely suitable for unstructured environments.

Avoiding visual occlusions by unwanted objects (obstacles) is a very challenging task, given the assumptions of our problem. A simple method to avoid visual occlusions is to determine visual occlusions in the image space and translate the corresponding visual-motor data as occupied space.

In Chapter 5, we present a new sampling-based path planner based on the RRT planner to be used with a UVS system. We show how algorithms from the robust Jacobian estimation and control chapter (Chapter 4) can be borrowed to equip the planning algorithm with statistical robustness against outliers.

Part II

Contributions

Chapter 4

Robust Uncalibrated Visual Servoing

In this chapter[1], we introduce the robust uncalibrated visual servoing (RUVS) system. The proposed RUVS system incorporates statistical robustness against outliers into the general framework of UVS. The contributions of this chapter are twofold:

1. We develop an algorithm based on robust M-estimators to numerically estimate the visual-motor Jacobian from raw visual-motor data. In our proposed method, the outliers that are due to different visual tracking errors are statistically rejected.

2. We present a control framework that uses the robust Jacobian. The robust Jacobian estimation algorithm provides information about the visual-motor data, which can be also used to label outliers and determine if a feature point is an outlier. We present a method to replace an outlier query with a reconstructed inlier, *i.e.,* estimate the correct value of the outlier features, to use in the closed-loop control. The procedure is completely based on sensed values, not *a priori* models.

The uncalibrated control and the concept of visual-motor space is reviewed in Section 4.1. In Section 4.2, we show how visual-motor outliers affect the Jacobian estimates through a simple example consisting of a two-link arm. The methods introduced in this chapter and the next chapter use a visual-motor database, which is formalized in Section 4.3. The statistics theory behind the estimation of the visual-motor Jacobian in a general case and the JacobianEstIRLS algorithm are explained next in Section 4.4. Finally, the outlier removal method and the overall RUVS system are presented in Sections 4.5 and 4.6. The algorithms and methods introduced here are evaluated in Part III, Chapter 7.

4.1 Uncalibrated Control in the Visual-Motor Space

An intuitive space for vision-based control and planning is the space that maps visual measurements to robot configuration.

[1]The results of this chapter have been partially published in the proceedings of and presented at the IEEE International Conference on Robotics and Automation (ICRA) in May 2010 [99].

The joint-space representation of the configuration space for a robot with N degrees of freedom (DOF) is usually a subset $\mathcal{C}$ of the product $\mathcal{Q}_1 \times \mathcal{Q}_2 \times \cdots \times \mathcal{Q}_N \subseteq \mathbb{R}^N$ with $\mathcal{Q}_i = [0,\, 2\pi]$, $1 \leq i \leq N$. Similarly, the sensor (visual) space of a camera measuring M visual features corresponding to scene objects is represented by a subset $\mathcal{S}$ of $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_M \subseteq \mathbb{R}^M$. The *visual-motor space* $\mathcal{V}$ is then defined as the product of the configuration space and the visual space:

$$\mathcal{V} = \mathcal{S} \times \mathcal{C}. \tag{4.1}$$

In other words, the elements of this visual-motor space are tuples $(\mathbf{s},\, \mathbf{q}) \in \mathcal{V}$, where visual measurement vector $\mathbf{s} \in \mathcal{S}$ is an M-vector and joint vector $\mathbf{q} \in \mathcal{C}$ is an N-vector.

Visual measurements may be the coordinates of the corners of scene objects, their area in the image plane, or other higher-order moments [36]. If visual features are chosen correctly, for each configuration, there exists a unique visual measurement. Four is the minimum number of point coordinates for a unique camera configuration. With 3 points, there exists exactly 4 camera poses from which the same images are seen [11].

In Figure 4.1, the camera is placed on the elbow of a WAM arm, therefore there are controllable DOFs ($N = 4$). The visual features are the horizontal and vertical coordinates of five target points; therefore, $M = 5 \times 2 = 10$. Figures 4.1a and 4.1c show the robot and the image in the initial state. The goal configuration and image are shown in Figures 4.1b and 4.1d.

With the state of the art implementations of computer vision libraries [101, 102], visual features can be computed and tracked at video frame rate; however, fast arm motions with an in-hand camera often result in the failure of the visual tracker. When a visual feature in the image does not represent its corresponding object, it becomes an outlier observation. The control and planning algorithms should be able to handle outliers; otherwise, they will treat outliers in the same fashion as inlier observations. This generally results in failure of the system.

To map a configuration to its corresponding visual measurement, we define the visual-motor map as

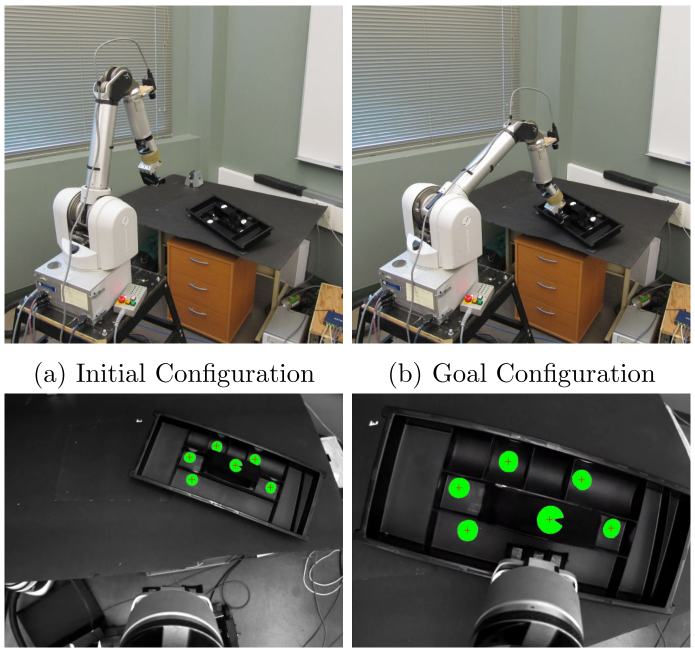
$$\mathbf{F} : \mathcal{C} \to \mathcal{S}. \tag{4.2}$$

The visual-motor map is also known as the perceptual kinematic map in the literature [100]. It defines an N-dimensional manifold on the $(N + M)$-dimensional visual-motor space. While the geometric structures of the visual-motor map and the kinematic perceptual map are similar, the kinematic perceptual map uses scene structure. The mapping from the motor readings to the visual measurements is direct; therefore, it is prudent to design uncalibrated planning and control algorithms that depend only on the raw visual-motor data and do not depend on scene reconstruction, the target depth, or other 3D measurements from the scene. This differentiates our work from the work of Sharma and Sutanto [100].

The time derivative of the visual-motor map $\mathbf{s} = \mathbf{F}(\mathbf{q})$ leads to $\dot{\mathbf{s}} = \mathbf{J_u}(\mathbf{q})\, \dot{\mathbf{q}}$, where $\mathbf{J_u}(\mathbf{q}) = \partial \mathbf{F}(\mathbf{q}) / \partial \mathbf{q}$ is the uncalibrated visual-motor Jacobian.

The UVS control law is defined without the need to reconstruct the depth or other 3D parameters:

$$\dot{\mathbf{q}} = -\lambda\, \widehat{\mathbf{J}}_{\mathbf{u}}^{\dagger}\, \mathbf{E}(\mathbf{s},\, \mathbf{s}_{goal}), \tag{4.3}$$


(a) Initial Configuration (b) Goal Configuration

(c) Initial Image (d) Goal Image

Figure 4.1: Setup for UVS in the visual-motor space. The goal and initial states are shown. A visual tracking algorithm [101] tracks white blobs (filled with green) and returns the coordinates of their centres marked by cross marks. The UVS control law in (4.3) uses the image error from the coordinates in the goal image and the coordinates in the current image to generate proper joint commands.

where $\widehat{\mathbf{J}}_{\mathbf{u}}$ is an estimate of the visual-motor Jacobian and $\widehat{\mathbf{J}}_{\mathbf{u}}^{\dagger}$ is its Moore-Penrose pseudoinverse, $\mathbf{s}_{goal}$ is the vector of visual feature measurements at the goal state, and $\mathbf{E}()$ is a vector-valued function that computes the error between the current and goal visual measurements. For feature point coordinates, $\mathbf{E}(\mathbf{s}, \mathbf{s}_{goal}) = \mathbf{s} - \mathbf{s}_{goal}$.

Some Jacobian estimation methods were reviewed in Section 2.4. The new robust Jacobian estimation algorithm will be presented in Section 4.4.

4.2 Motivational Example: The Two-Link Robot Arm

The effect of outliers on Jacobian estimation is best explained through a simple example. Consider a two-link planar robot arm with a 2D workspace and a sensor that measures the position of the end-effector as shown in Figure 4.2. This sensor models an orthographic projection camera overlooking the arm and the workspace. The projection lines are parallel and perpendicular to the image plane. The robot has two controllable joint angles represented by a tuple $\mathbf{q} = (q_1, q_2) \in \mathcal{C}$, where the motor space $\mathcal{C}$ can be represented by

$$\mathcal{C} = \mathcal{Q}_1 \times \mathcal{Q}_2 \subseteq \mathbb{R}^2. \tag{4.4}$$

The measurements for the end-effector coordinates may be represented by $\mathbf{s} = (s_1, s_2) \in \mathcal{S}$, where the sensor space $\mathcal{S}$ is represented by

$$\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \subseteq \mathbb{R}^2. \tag{4.5}$$

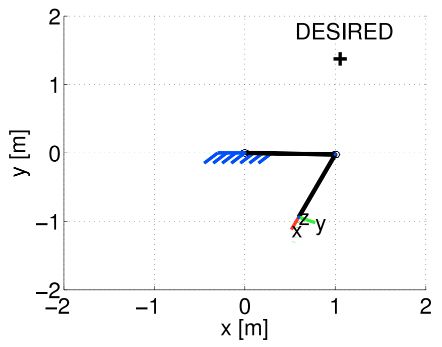


Figure 4.2: Two-link planar robot arm. An overhead camera overlooks the robot end-effector and the workspace in an eye-to-hand configuration. For simplicity, and without loss of generality, the measurements have been chosen to be the same as metric coordinates of the end-effector. The goal is to control the end-effector to reach the desired position in the workspace denoted by a cross.

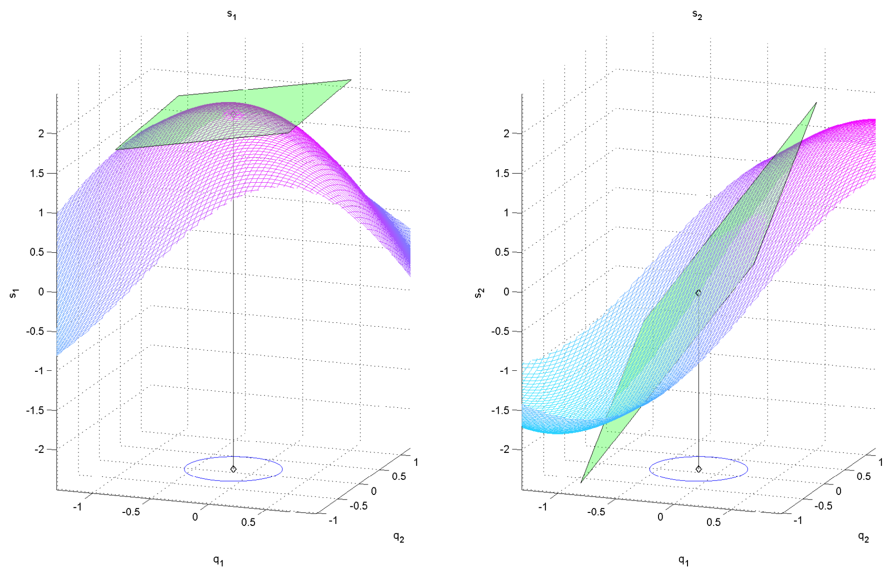


Figure 4.3: Sensory-motor surface for the two-link robot arm example in Figure 4.2. An arbitrary point on the surface is shown with a the Jacobian hyperplane tangent to the surface. The Jacobian hyperplane is represented by a 2×2 matrix in this case.

In the simple case of the planar two-link arm, the sensory-motor space $\mathcal{V} = \mathcal{S} \times \mathcal{C}$ is a hypersurface, which can be visualized by the two surfaces shown in Figure 4.3. The sensory-motor Jacobian matrix $\mathbf{J}_u$ in $\dot{\mathbf{s}} = \mathbf{J}_u \dot{\mathbf{q}}$ is a 2×2 matrix here. The Jacobian defines a hyperplane tangent to the sensory-motor space. The goal of a numerical Jacobian estimation method is to find the Jacobian matrix given samples from the sensory-motor surface. The sensory-motor Jacobian matrix is used to control the arm to a desired configuration using the control law in (4.3) using sensor measurements only. A snapshot of control iterations for the example under study is shown in Figure 4.4.

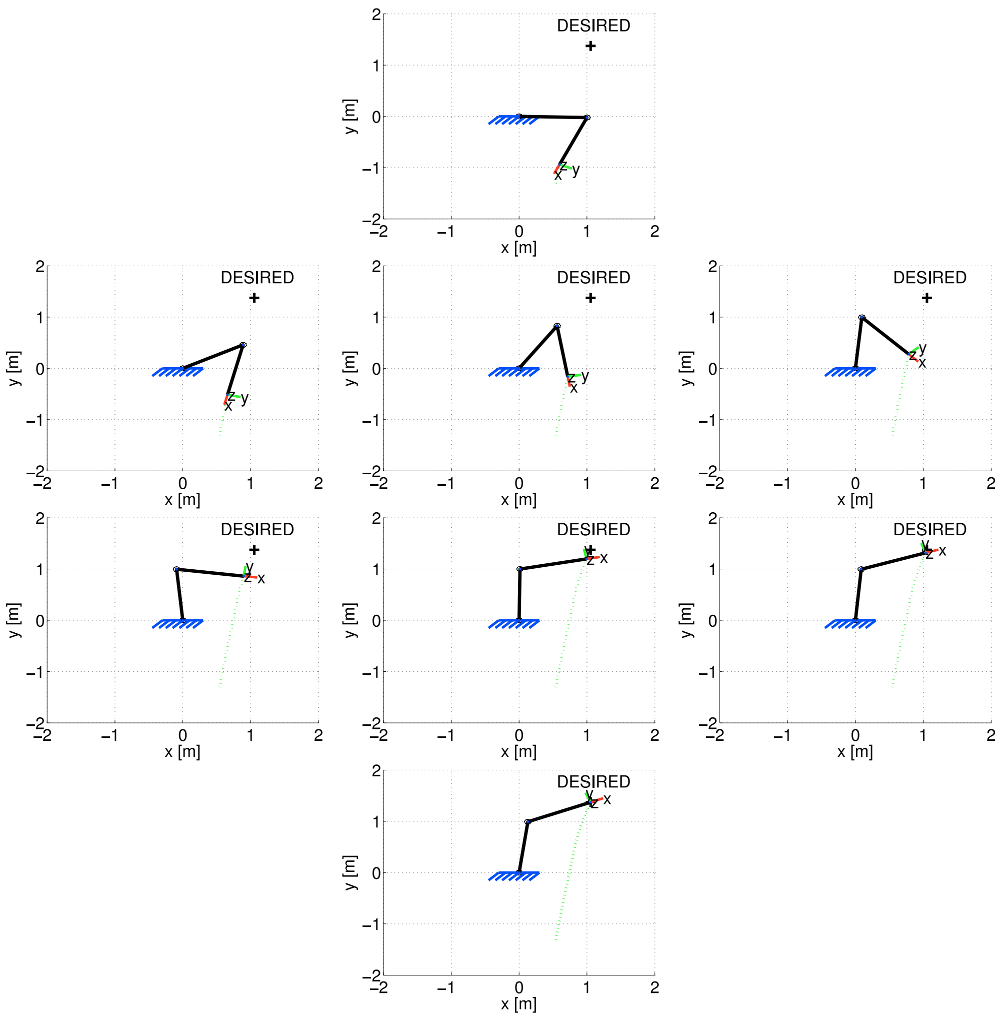


Figure 4.4: Visual servoing of the two-link arm using the control law in (4.3). For simplicity, the visual measurements have been chosen to be the same as the end-effector coordinates. The robot reaches the goal position, while the image trajectory (the same as end-effector position in this example) is nearly straight.

It was established in Section 2.4.3 that the least squares-based methods [9] can be used to estimate the Jacobian matrix from samples. Nonetheless, least squares-based methods are not robust to outliers and a naive least-squares estimation leads to faulty Jacobian estimates. This is illustrated as largely erroneous estimates of the Jacobian hyperplanes in Figure 4.5, when outliers are present. The Jacobian hyperplane estimated by the least-squares method is shown in red and has largely differs from the true Jacobian shown in green. A robust Jacobian hyperplane, estimated later by the algorithm presented in Section 4.4.4 is shown in blue. It is very close to the true Jacobian shown in green. In the following sections, we will explain the theory of robust Jacobian estimation and the related algorithms.

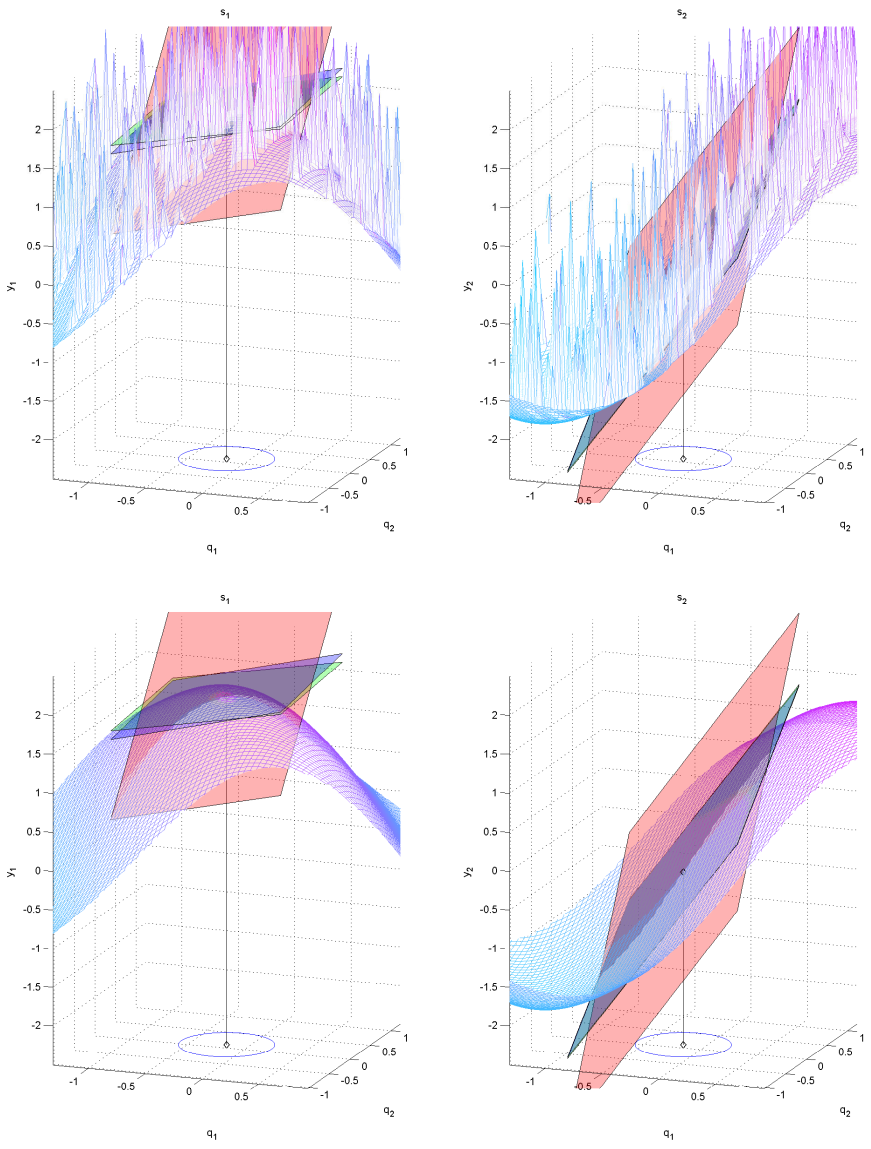


Figure 4.5: Least-squares Jacobian estimation fails when outliers are present in the data samples. The red hyperplane shows the result of least-squares based Jacobian estimation as explained in Section 2.4.3. The true Jacobian is shown in green. The robust Jacobian estimate, the contribution of this chapter, is shown in blue. The bottom figure shows the same Jacobian hyperplanes overlayed atop the no-outliers hypersurface for better visualization result. The top figure is cluttered. This figure is best seen in colour.

4.3 Visual-Motor Database

Almost all robotics systems in new environments require an offline learning phase, where they can process the sensory-motor information and learn the scene from it.

The algorithms and methods that we present next are based on analyzing the statistics of the visual-motor space. That is to say, there is a need to have a statistically significant number of visual-motor samples for the proposed methods. By exploring and sampling the space randomly, a database of visual-motor tuples $(\mathbf{s}_p, \mathbf{q}_p)$ is stored in database D, where in its simplest form

$$D = \{(\mathbf{s}_p,\, \mathbf{q}_p)\}_{p=1}^{P}, \tag{4.6}$$

where P is the length of the database. Other fields that depend on joint vector $\mathbf{q}_p$ may also be added to the data structure. For example, the visual-motor Jacobian matrix at each point can be stored alongside the visual-motor tuple to be retrieved later. For each scene object, a separate database can be created. It is assumed that scene objects and the robot base frame are static (please see the chapter summary in Section 4.7 for a discussion of this assumption).

The database initialization process can be performed either semi-autonomously or manually. For both methods, the user first initializes the visual tracker algorithm for the objects of interest through a Graphical User Interface (GUI). In manual initialization, the WAM arm is put in gravity compensation, where the user can freely move the arm around and sample the visual-motor space. In semi-autonomous initialization, the WAM arm follows previously-recorded trajectories to record the visual-motor samples after the visual tracking initialization. A new sample is always checked for repeatability before being appended to the database. This helps avoid overfitting in the numerical optimization methods explained in the next section. A typical database after initialization has around 3,000 records, which are obtained in only 5 minutes at a sampling frequency of 10 Hz.

After the initialization phase, new samples are incrementally added to the database to create a dense database overtime. A dense database carries better statistics.

4.4 Robust Jacobian Estimation

In the presence of visual tracking discrepancies, the corresponding visual-motor pair $(\mathbf{s}, \mathbf{q})$ are outliers and do not belong to the visual-motor space of the object of interest. The outliers should have no significant influence on the Jacobian estimates for a successful UVS system. Here, we apply some widely-accepted machinery from the theory of robust statistics to estimate the uncalibrated visual-motor Jacobian.

4.4.1 Problem formulation

Let $D = \{(\mathbf{s}_p, \mathbf{q}_p)\}_{p=1}^{P}$ be a database of P visual-motor tuples, which are potentially contaminated by less than 50% outliers. Given a visual-motor query point $\mathbf{d}_c = (\mathbf{s}_c, \mathbf{q}_c) \notin D$, the problem of estimating the Jacobian in (4.3) at $\mathbf{d}_c$ is formulated as an optimization problem, in which the sum of a robust norm of the residuals are minimized:

$$\widehat{\mathbf{J}}_{\mathbf{u}}(q)\Big|_{q=\mathbf{q}_c} = \arg\min_{\mathbf{J}_{\mathbf{u}}} \sum_{k:\,\mathbf{q}_k \in B_r(\mathbf{q}_c)} \rho(\Delta\mathbf{s}_k - \mathbf{J}_{\mathbf{u}}\Delta\mathbf{q}_k), \tag{4.7}$$

where $B_r(\mathbf{q}_c) = \{\mathbf{q}_p : \|\mathbf{q}_c - \mathbf{q}_p\| < r\,, p = 1, \cdots, P\}$ is an open ball with radius r in the joint space, $\Delta\mathbf{s}_k = \mathbf{s}_c - \mathbf{s}_k$, and $\Delta\mathbf{q}_k = \mathbf{q}_c - \mathbf{q}_k$, and $\rho(\cdot)$ is an M-estimator. This formulation is quite general and can also describe the least-squares based methods [9], where the least-squares (LS) estimator $\rho(x)$ is written as $\rho(x) = x^2$. Then, (2.18) repeated here has a similar form to (4.7):

$$\widehat{\mathbf{J}}_{\mathbf{u}}(\mathbf{q})\Big|_{q=\mathbf{q}_c} = \arg\min_{\mathbf{J}_{\mathbf{u}}} \sum_{k:\,\mathbf{q}_k \in B_r(\mathbf{q}_c)} (\Delta\mathbf{s}_k - \mathbf{J}_{\mathbf{u}}\Delta\mathbf{q}_k)^2. \tag{4.8}$$

Both methods fit the best hyperplane to the visual-motor data around $\mathbf{q}_c$. As illustrated in Figure 2.11 for a simple system with 2 DOFs and single visual feature, a plane is fitted to 2×1-dimensional data. As mentioned earlier, the LS estimator is not robust to outliers. In a nutshell, the main difference between (4.8) and (4.7) is the choice of the estimator. The sensitivity of the least-squares Jacobian estimate to outliers is also depicted in Figure 2.11, where the Jacobian is biased towards the outlier visual-motor pairs.

4.4.2 The choice of the estimator

The breakdown point of an estimator

In the robust statistics literature, an estimator is usually characterized by how well it handles outliers. The breakdown point (BDP) of an estimator is a measure of its resistance to outliers. It refers to the smallest proportion of incorrect samples that the estimator can tolerate before they arbitrarily affect the model fitting [103]. The maximum BDP possible is 50%, because if the number of outliers is larger than the number of inliers, the estimator captures the statistics of the outliers. It is important to note that the maximum BDP of 50% concerns estimators in the form of functions, but not algorithms. For example, the RANSAC algorithm [59] can often handle a larger percentile of outliers, but it uses priors on the data.

The LS estimator is very sensitive to outliers and can be biased with only one of the samples. Therefore, the BDP of the LS estimator is $1/N$, where N is the sample size [104]. For large sample sizes, the LS estimator has a BDP of 0 since with increasing the sample size, $1/N$ tends to zero.

Ideally, one would like to use a robust estimator with a BDP of 50%. There is also another way to characterize estimators based on their influence and weight functions, as explained next.

The influence and weight functions

The approach based on the influence function [103] characterizes an estimator $\rho(x)$ based on its *influence function* $\psi(u)$

$$\psi(u) = \frac{d}{du}\rho(u), \tag{4.9}$$

and its weight function $w(u)$

$$w(u) = \frac{1}{u}\psi(u). \tag{4.10}$$

The data concentrated at the tail of the distribution should not influence the estimation result. In other words, the weight function assigns smaller weights to outlier data.

The LS estimator has a constant weight function and outliers receive the same constant weight as data. Another choice for the estimator is the L_1-norm $\rho(x) = |x|$, which is more robust than LS estimator. However, both have the least possible breakdown point [104]. In addition, M-estimators with a redescending influence function, outperform bounded estimators that are not redescending (for example, L_1-norm) [105, 103].

Several redescending M-estimators have been used in computer vision and robotics literature. Tukey's Biweight (BW) function is popular among the image-based visual tracking and servoing [30, 23, 31, 22, 106] because of high Gaussian efficiency. The influence function of the Biweight function can be written as follows:

$$\psi_{BW}(u) = \begin{cases} u(1 - \frac{u^2}{c^2})^2, & |u| < c \\ 0, & |u| \geq c \end{cases} \tag{4.11}$$

where parameter c is a constant of the BW estimator which is set to 4.6851 to achieve 95% Gaussian efficiency [105].

A scaled version of the influence function can be derived with a simple variable transformation in the form of

$$u(x) = \frac{x}{\sigma},$$

Parameter σ is in fact selected as robust measure of scale as explained later in Section 4.4.3. With this transformation, one can define the influence function as a multi-variate function that depends on scale σ as well:

$$\begin{aligned} \psi(x, \sigma) &= \frac{d}{dx}\rho(u(x)) && (4.12) \\ &= \frac{d}{du}\rho(u)\frac{du(x)}{dx} && (4.13) \\ &= \frac{1}{\sigma}\frac{d}{du}\rho(u) = \frac{1}{\sigma}\psi(\frac{x}{\sigma}). && (4.14) \end{aligned}$$

With this transformation, Tukey's Biweight can be rewritten as

$$\psi_{BW}(xm\sigma) = \begin{cases} \frac{x}{\sigma^2}(1 - \frac{x^2}{c^2\sigma^2})^2, & |x| < c\sigma \\ 0, & |x| \geq c\sigma \end{cases} \tag{4.15}$$

The influence of the outliers starts to decrease when the residual error is larger than the inflection point of the estimator. The inflection point for the BW estimator is calculated as $\frac{c}{\sqrt{5}}\sigma \simeq 2.0952\sigma$, *i.e.,* larger points have less influence:

$$|x|_{BW} > \frac{c}{\sqrt{5}}\sigma \simeq 2.0952\sigma.$$

The Geman-McClure (GM) estimator has been successfully used in the computer vision community for pattern matching and optical flow estimation [107].

The Geman-McClure estimator has a smooth influence function, which means that outliers are always placed in a zone of doubt, but not fully discraded [103]. On the other hand, Tukey's Biweight estimator does assign zero weight to outliers and does not consider a zone of doubt. The GM estimator has a simple form of

$$\rho_{GM}(u) = \frac{u^2}{u^2+1}, \tag{4.16}$$

which after the $u(x) = \frac{x}{\sigma}$ transformation becomes

$$\rho_{GM}(x,\sigma) = \frac{x^2}{x^2+\sigma^2}. \tag{4.17}$$

Its influence function is derived by taking the derivative of the above:

$$\psi_{GM}(x,\sigma) = \frac{2x\sigma^2}{(x^2+\sigma^2)^2}. \tag{4.18}$$

For the GM estimator, the inflection point is calculated as $\sigma/\sqrt{3} \simeq 0.577\sigma$, which means that residuals larger than this inflection point start losing their influence:

$$|x|_{GM} > \sigma/\sqrt{3} \simeq 0.577\sigma.$$

Figure 4.6 illustrates some of the discussed estimators and their corresponding influence and weight functions. The BW estimator assigns relatively large weights to residual errors in $[2\sigma, 3\sigma]$. For our visual-motor data set, the outliers tend to be not too far from the inliers and stricter outlier-rejection criteria is desired. The GM estimator has a tighter inflection point than the BW estimator. It does a better job of rejecting the visual-motor outliers than the BW estimator as validated by experiments presented in Chapter 7. Hence, we use the GM robust M-estimator as the robust M-estimator in (4.7).

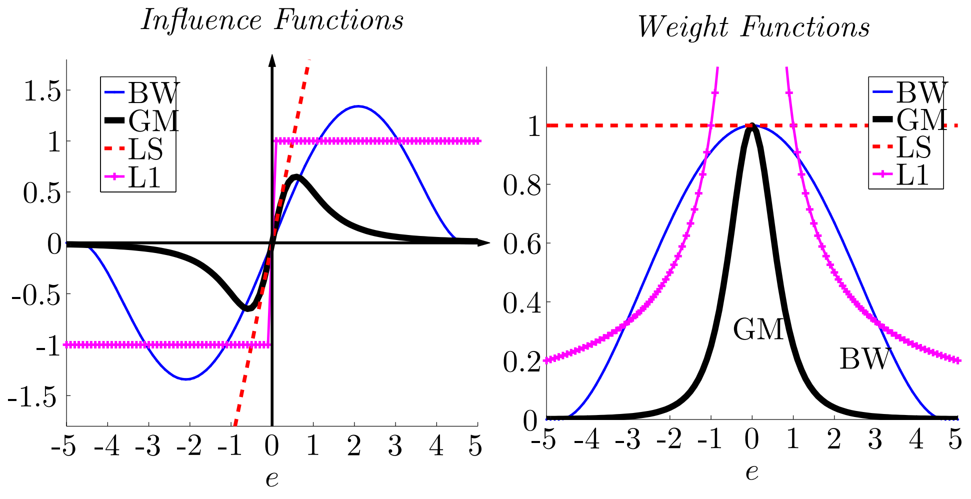


Figure 4.6: The influence and weight functions of estimators L_1-norm (L1), Least-squares (LS), Tukey's Biweight (BW), and Geman-McClure (GM) estimators. (Left) The influence functions, (Right) The normalized weight functions, for $\sigma = 1$.

4.4.3 Measure of scale

Scale σ is a parameter that quantifies how the probability distribution is spread. For example, variance is a measure of scale for the normal distribution. Robust measures of scale are needed to estimate the scale parameter in the M-estimator.

The Median Absolute Deviation (MAD) is one of the most common estimators of scale, which has the highest possible BDP of 50% and a bounded influence function [103]:

$$\sigma_{MAD} = B \operatorname*{med}_i \left\{ \left| x_i - \operatorname*{med}_j \{x_j\} \right| \right\}, \tag{4.19}$$

where B is a constant chosen to make MAD consistent with the normal distribution using the cumulative normal distribution function $\Phi(\cdot)$: $B = 1/\Phi^{-1}(3/4) = 1.4826$.

Although its Gaussian efficiency is rather low at 37%, the MAD scale is computationally very efficient [103]. Thus, we select it as a computationally-efficient measure of scale to estimate the variance of the inlier samples. Samples $\{x_k\}_{k=1}^{k=K}$ should be scalar because the median is defined over scalar values. The visual-motor space is multi-dimensional and not scalar. We use the norm of the image space differences for nearest neighbours to calculate σ_{MAD}, *i.e.,* $x_k = ||\Delta \mathbf{s}_k||$, where $(\mathbf{s}_k, \mathbf{q}_k)$ are the K-nearest neighbours of the query $(\mathbf{s}_c, \mathbf{q}_c)$ in the joint space.

4.4.4 The JacobianEstIRLS Algorithm

Solving the robust M-estimation problem in (4.7) can be challenging. A common practice is to solve (4.7) by iteratively recomputing the weights of a weighted least squares problem [105]. This method is known as the Iteratively Reweighted Least Squares (IRLS) algorithm. The IRLS algorithm is widely used as an efficient implementation of robust M-estimation in many practical nonlinear optimization domains. Examples include robust visual tracking and visual servoing [30, 23, 31, 22, 106, 107]. The robust Jacobian estimation algorithm works as follows.

A.1 **Initialize visual-motor database**: Start by an offline database initialization for the visual-motor data pairs by collecting visual-motor pairs. This will generate visual-motor database $\{(\mathbf{s}_p, \mathbf{q}_p)\}_{p=1}^{P}$. Later on, new readings are incrementally added to the database.

A.2 **Find nearest neighbours**: For the visual-motor query $(\mathbf{s}_c, \mathbf{q}_c)$, find the neighbouring visual-motor pairs in $\{(\mathbf{s}_k, \mathbf{q}_k)\}_{k=1}^{K}$, which contains the K-nearest neighbors of $\mathbf{q}_c$.

A.3 **Estimate initial scale**: Use MAD as in (4.19) to find the initial measure of scale σ.

A.4 **Find initial weights**: Initialize weight matrix $\mathbf{W}_0$ according to the norm and scale found in (4.19). We assign binary weights [30, 104] to the K-nearest neighbors of query point $(\mathbf{s}_c, \mathbf{q}_c)$ according to:

$$w_k = \begin{cases} 1 & : |x_k| \leq 2.5\sigma \\ 0 & : \text{otherwise} \end{cases}, \tag{4.20}$$

where $\mathbf{W}_0 = \operatorname{diag}[w(e_1) \cdots w(e_K)]$.

A.5 **Estimate the Jacobian:** Given a query point $(\mathbf{s}_c, \mathbf{q}_c)$, its K-nearest neighbors from the memory $\{(\mathbf{s}_k, \mathbf{q}_k)\}_{k=1}^{K}$, scale σ, robust estimator $\rho(x, \sigma)$, and the initial weight matrix $\mathbf{W}_0$, call JACOBIANESTIRLS to estimate the Jacobian. The pseudocode is presented in Algorithm 3.

A.6 **Update control signal**: The Jacobian estimated in the previous step is used in (2.11) to generate the control signal.

A.7 **Update database**: The new visual-motor pair is added to the database for later use. $P = P + 1$.

A.8 **Return**: Goto step A.2.

Algorithm 3 JACOBIANESTIRLS($(\mathbf{s}_c, \mathbf{q}_c)$, $\{(\mathbf{s}_k, \mathbf{q}_k)\}_{k=1}^{K}$, σ, $\rho(e; \sigma)$, $\mathbf{W}_0$)

```
1:  $\mathbf{W} \leftarrow \mathbf{W}_0$
2:  $t \leftarrow 1$
3:  $\widehat{\mathbf{J}}_{\mathbf{u}}(0) \leftarrow \mathbf{0}$
4:  for $k = 1$ to $K$ do
5:      $(\Delta s_k, \Delta \mathbf{q}_k) \leftarrow (\mathbf{s}_c, \mathbf{q}_c) - (\mathbf{s}_k, \mathbf{q}_k)$
6:  end for
7:  $\Delta \mathbf{S}_{[K \times M]} \leftarrow [\Delta \mathbf{s}_1 \cdots \Delta \mathbf{s}_K]^\top$
8:  $\Delta \mathbf{Q}_{[K \times N]} \leftarrow [\Delta \mathbf{q}_1 \cdots \Delta \mathbf{q}_K]^\top$
9:  while $||\widehat{\mathbf{J}}_{\mathbf{u}}(t) - \widehat{\mathbf{J}}_{\mathbf{u}}(t-1)|| > \epsilon$ do
10:     $[U, \Sigma, V] \leftarrow \text{SVD}\ (\mathbf{W} \Delta \mathbf{Q})$
11:     $\widehat{\mathbf{J}}_{\mathbf{u}}(t) \leftarrow [(U \, \Sigma \, V^\top)^\top \, \mathbf{W} \, \Delta \mathbf{S}]^\top$
12:     $[e_1 \cdots e_K]^\top \leftarrow \|\mathbf{W} \Delta \mathbf{Q} \, \widehat{\mathbf{J}}_{\mathbf{u}}(t) - \mathbf{W} \Delta \mathbf{S}\|$
13:     for $k = 1$ to $K$ do
14:         $w(e_k) \leftarrow \frac{1}{u} \frac{\partial}{\partial u} \rho(u; \sigma)\big|_{u=e_k}$
15:     end for
16:     $\mathbf{W} \leftarrow \text{diag}\,[w(e_1) \cdots w(e_K)]$
17:     $t \leftarrow t + 1$
18: end while
19: return $\widehat{\mathbf{J}}_{\mathbf{u}}(\mathbf{t})$, $\mathbf{W}$
```

4.5 Outlier Replacement

The goals of the statistically-robust Jacobian estimation algorithm are twofold: (i) To estimate the robust Jacobian $\widehat{\mathbf{J}}_{\mathbf{u}}$, and (ii) to label local samples as inliers or outliers according to the output weights $\mathbf{W}$. The labeled inliers are used to determine if the query vector contains outliers and recover the error.

In the Jacobian estimation problem formulation, it was mentioned that the query point $(\mathbf{s}_c, \mathbf{q}_c)$ is not already in the database D. After the nearest neighbours are found in D, one may append the query point to the neighbours set and run the JACOBIANESTIRLS algorithm on the $K+1$-set. Examining the element of $\mathbf{W}$, the inliers and outliers within the $K+1$-set are determined. The last diagonal element of $\mathbf{W}$ corresponds to the query point. If the query point is an inlier, it can be used to calculate the image error and close the feedback loop. Otherwise, it is an outlier and cannot be used in control. It needs to be discarded and replaced by an inlier estimate.

If the visual feature vector $\mathbf{s}_c$ contains independent visual measurements, one can call JacobianEstIRLS for each individual elements of $\mathbf{s}_c = [s_{c,1} \cdots s_{c,M}]^\top$ (M times) and label any of the individual elements as either inlier or outlier. The outlier elements are then replaced by a robust measure of location from the inlier neighbours, say, replaced by their median. This is possible because the median is well-defined for scalars. For $i = 1, \cdots, M$:

$$\widehat{s}_{c,i} = \underset{k \in \text{inliers}}{\text{med}} \{s_{k,i}\}. \tag{4.21}$$

In feedback control $\widehat{\mathbf{s}}_c = [\widehat{s}_{c,1} \cdots \widehat{s}_{c,M}]^\top$ will be used.

If the elements of the visual feature vector $\mathbf{s}$ are not independent from each, then inlier/outlier labeling and outlier replacement should be treated with care. For image point features, for example, the horizontal and vertical coordinates belong to the same object and should be treated together. If an image point is an outlier, both coordinates must be labeled accordingly. We use a variant of the JacobianEstIRLS algorithm to estimate the visual-motor Jacobian of point features. For m points, the query vector $\mathbf{s}_c$ is of size $M = 2 \times m$, where $\mathbf{s}_c^{(i)} \in \mathbb{R}^2$ for $i = 1, \cdots, m$, represent the horizontal and vertical coordinates of the point feature. The point-wise Jacobian $\widehat{\mathbf{J}}^{(i)}$ is a $2 \times N$ matrix that relates joint velocity to the image velocity of a point: $\dot{\mathbf{s}}_c^{(i)} = \widehat{\mathbf{J}}^{(i)}(\mathbf{s}_c^{(i)}, \mathbf{q}_c)\, \dot{\mathbf{q}}_c$. Each point-wise Jacobian is estimated in a similar fashion to (4.7):

$$\widehat{\mathbf{J}}^{(i)}(\mathbf{q}_c) = \underset{\mathbf{J}^{(i)}}{\operatorname{argmin}} \sum_{k=1}^{K} \rho(||\Delta\mathbf{s}_k - \mathbf{J}^{(i)}\Delta\mathbf{q}_k||;\, \sigma), \tag{4.22}$$

where K is the number of nearest neighbors, $\Delta\mathbf{s}_k = \mathbf{s}_c^{(i)} - \mathbf{s}_k^{(i)}$, $\Delta\mathbf{q}_k = \mathbf{q}_c - \mathbf{q}_k$, and $\rho(\cdot)$ is a robust estimator as explained previously. Note that the L_2-norm of the residue is used as the robust norm is defined on scalar values. This is an approximation that should be taken into account when evaluating the method in Chapter 7.

The point-wise Jacobian matrices are augmented to obtain a full $M \times N$ Jacobian:

$$\widehat{\mathbf{J}}_{\mathbf{u}} = [\widehat{\mathbf{J}}^{(1)}\, ;\, \widehat{\mathbf{J}}^{(2)}\, ;\, \cdots\, ;\, \widehat{\mathbf{J}}^{(m)}]. \tag{4.23}$$

The point-wise query $(\mathbf{s}_c^{(i)}, \mathbf{q}_c)$ and all the K neighbors are labeled as inlier/outlier. Similar to (4.21), an outlier query is recovered from the median of the norm of the coordinates of inliers within the K nearest neighbors.

4.6 The RUVS System

The goal of the robust Jacobian estimation algorithm is to provide the robust Jacobian. In addition, if the visual measurements become erroneous during a control loop iteration, the control law might fail despite having a robust Jacobian estimate. The goal of the outlier removal algorithm is to deal with outliers that occur during operation. The block diagram of the RUVS system is shown in Figure 4.7. The

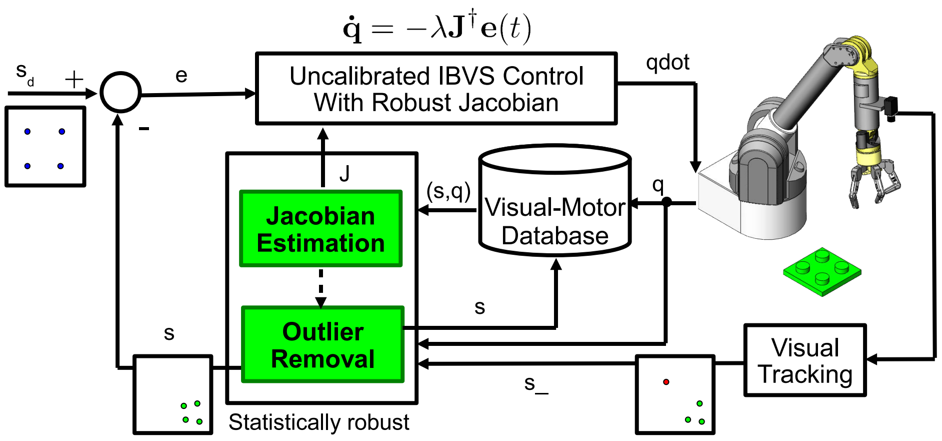


Figure 4.7: Robust Uncalibrated Visual Servoing (RUVS) block diagram. The RUVS system first finds a statistically-robust visual-motor Jacobian using a database, then removes visual-motor outliers to calculate a valid image error to use in the uncalibrated image-based control law. An eye-in-hand manipulator is considered.

UVS control law is equipped with a robust Jacobian estimation algorithm, which is robust against statistical outliers and an outlier removal method to close the feedback loop during control. The system does not require any prior knowledge of either the camera/robot calibration or the geometric model.

4.7 Summary

The UVS framework is very desirable for robot applications in unstructured environment, because the camera intrinsic parameters, the calibration of the robot-to-camera transformation, or the geometric object/scene models are not required. Robustness to noise and outliers is an important feature that needs to be incorporated into the UVS framework. Visual occlusion of image features, visual tracking mismatches, and large visual tracking errors are among the most common reasons for outliers to exist in vision-based robotics.

In this chapter, a framework based on robust M-estimation was presented for UVS control. We developed a robust visual-motor Jacobian estimation algorithm, JacobianEstIRLS, which exploits the visual-motor database to build robustness against visual-motor outliers (without explicitly fitting the visual-motor samples to a global model). The proposed Jacobian estimation algorithm is less sensitive to outliers than the least squares-based algorithms [9]. Unlike methods like the Broyden rank-one update [6, 7], which cannot exploit the visual-motor memory, our proposed method could use visual-motor memory to increase the quality of the Jacobian estimate gradually.

Further, we presented an outlier replacement method to reconstruct the inlier and replace the query visual-motor sample with its robust estimate. These algorithms were integrated into the RUVS system (Figure 4.7).

Simulations and experiments with a WAM arm with eye-in-hand configuration will be presented in Part III, Chapter 7 to validate the proposed algorithms.

Comparison to relevant robust visual servoing approaches

There are a few other robust visual tracking [18, 19] and visual servoing [30] papers in the literature, but none have considered robustness of the visual-motor Jacobian estimates.

The approach of Comport *et al.* [30], which proposes a statistically robust IBVS control law, is the most relevant to our contribution in this chapter. They use a parametric Jacobian, where robustness is achieved by down-weighting the rows of the parametric Jacobian associated with *corrupted* features. Thus, our proposed method differs from the work of Comport *et al.* in that we directly exploit the statistics of the sampled visual-motor space to estimate the robust Jacobian. In addition, we use an uncalibrated approach where robot-to-camera calibration parameters are not required, but they use an IBVS control law, in which the robot-to-camera transformation is already calibrated and depth of features in the camera frame is estimated using intrinsic camera calibration. On the other hand, we presented an outlier replacement method to reconstruct the inliers, which is helpful if there are not too many redundant features. Comport *et al.* detect which features are outliers, remove them, and close the control loop on inlier features. This suggests that there are an adequate number of redundant features available and a minimal set are always inliers.

Discussion on static object/robot assumption

The assumption of static objects and a static robot base may sound unpractical at first. We would like to point out that many robot arms are part of a larger mobile manipulator system, where the mobile base is equipped with a Simultaneous Localization And Mapping (SLAM) algorithm for localization in indoor environments. For example, a mobile manipulator which is meant to manipulate objects in a kitchen environment, has the ability to position itself to many static objects in the environment such as faucets, cabinets, fridge, etc. Once a visual-motor database for a static object is created at a specific location, it can be used or improved upon revisiting the same location. In addition, for dynamic objects moving slowly, the Jacobian matrix estimated by the proposed robust Jacobian estimation algorithm could be updated by a Broyden update rule (see Section 2.4.2). This is possible, because the variations of the Jacobian hypersurface would be small in this case. As such, the methods presented in this chapter are applicable to a wide range of applications.

Chapter 5

Sampling-Based Planning for Uncalibrated Visual Servoing

In this chapter[1], we contribute by introducing an efficient sampling-based planning framework to avoid visual and joint constraints in a RUVS system.

Path planning addresses some classic problems in visual servoing. An extensive recent survey can be found in Kazemi *et al.* [10]. Path planning can improve the convergence of the visual servo for distant goals by avoiding visual, physical, or joint constraints. However, most methods usually consider calibrated robot/cameras and known scene/target models. These strong calibration and modeling assumptions are not attractive for robots operating in unstructured environments.

A successful planner should be efficient, while building *robustly* on the raw sensory-motor data. Sampling-based planning is an efficient approach, which has been recently used in the context of visual servoing [33, 88, 97, 98]. Sampling-based methods are central to the contribution of this chapter. For a review of sampling-based methods, please see Chapter 3.

Only a handful of papers have proposed path planning for uncalibrated robot/camera systems with an unmodeled target object [13, 14, 15]. Early approaches [13] are limited and cannot be generalized to more realistic unstructured environments. Other uncalibrated methods are based on partial scene reconstruction or homography interpolation [14, 15], which are ill-posed problems for an unknown scene model.

In unstructured environments, planning is a quite challenging problem, because models are not known *a priori* and sensor measurements contain errors and outliers (Figure 1.1). While there has been progress in incorporating planning algorithms for visual servoing, there is generally a lack of robust frameworks in the literature to address the challenges in uncalibrated domains.

In Section 5.1, the problem setup and assumptions are described. The target and obstacle visual-motor databases are defined in Section 5.2. The databases are later used to determine which parts of the visual-motor space could be used in planning. Three types of constraints are modeled in this work: Joint limits (JLIM), field-of-view limits (FOV), and visual occlusion (VO) of target object by an obstacle

[1]The results of this chapter have been accepted for publication in the proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in October 2012 [108].

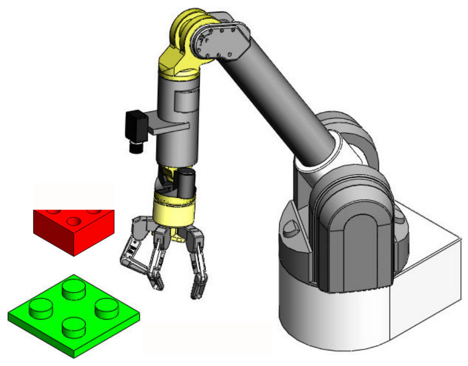
object. The modeling of these constraints are explained in Section 5.3. The proposed UvsBiRrt algorithm is presented in Section 5.4. The proposed framework is built on the success and efficiency of the sampling-based planners, while incorporating robustness to outliers in both planning and control.

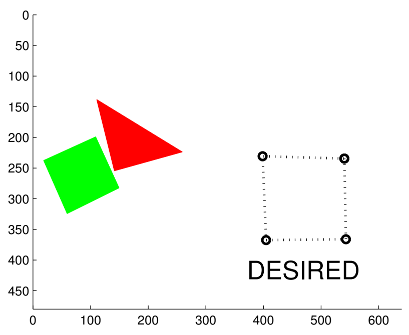
Planning experiments are presented in Part III, Chapter 7.

5.1 Planning to Avoid Constraints: Problem Setup and Assumptions

We consider a manipulator with an eye-in-hand configuration, a visual tracking algorithm to track interest points on a target object and on an obstacle object *without* using their geometric model, and an uncalibrated visual servoing system as described in Chapter 4. Our formalization of the control and planning framework is based on UVS and applies to both eye-in-hand and eye-to-hand camera configurations. To simplify the presentation and without loss of generality, we consider an eye-in-hand configuration as shown in Figure 5.1. The task is to manipulate a target object, which is shown as a square in Figure 5.1a, using only visual information. The obstacle is a shown as triangle. The images of the target and obstacle objects at the initial state are shown in Figure 5.1b. The desired (or goal) image of the target object, which is used to compute an image-based error in the UVS control law (4.3), is also shown in the initial image by dotted lines.

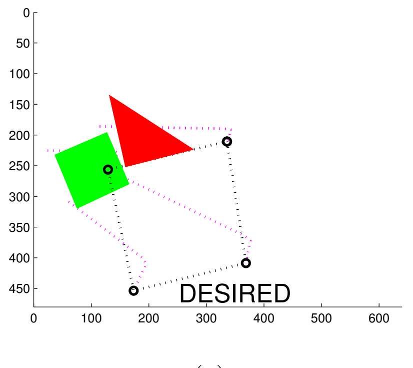
The system could fail, if planning to avoid constraints is not implemented for the UVS control. For example, the target image might leave the FOV resulting in the loss of the visual trackers or the robot might be driven to parts of the workspace where task-specific JLIM are violated. Another problem is the visual occlusion violation (VOV) as illustrated in Figure 5.2. The initial images of the target and obstacle images are shown in Figure 5.2a. An image trajectory, which has been generated using the UVS control law without considering the obstacle, is also shown by dotted lines. In Figure 5.2b, it is observed the VOV occurs during visual servoing


(a) Workspace, Initial Configuration

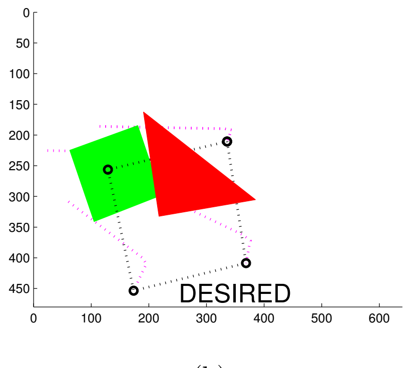


(b) Initial Image

Figure 5.1: Eye-in-hand setup with target and obstacle objects. (a) A Barrett WAM with camera mounted on the elbow. The target object is a *green square*. The obstacle is a *red triangle*. (b) Images of the target and obstacle objects. The desired target image is overlayed (dotted line).



(a)



(b)

Figure 5.2: Visual occlusion violation occurs during visual servoing without planning. (a) The initial image of the rectangle target (green) and the triangle obstacle (red), the image-based control trajectory without planning (dotted line) and the goal or DESIRED target image (dotted line) are shown. (b) Without considering the VOV and planning to avoid it, the rectangle target (green) is visually occluded by the triangle obstacle (red) during visual servoing. As visual occlusion results in failure of visual tracking, the overall visual servoing system fails.

without planning. With the algorithms presented in this chapter, we will show how a planner helps avoid the constraints. In addition, we show that the proposed algorithm are well-suited for an RUVS system (Chapter 4) and can generate paths that are robust to outliers.

Both the UVS control and the constraint violation check utilizes the sampled visual-motor space. The sampled visual-motor space is stored in one or more databases, which are explained next.

5.2 The Target and Obstacle Visual-Motor Databases

A target database D_T with P_t visual-motor tuples is initialized by randomly exploring the visual-motor space offline:

$$D_T = \{(\mathbf{s}_t,\, \mathbf{q}_t)\}_{t=1}^{P_t}. \tag{5.1}$$

The samples could be outliers, but it is assumed that more than 50% of the samples are inliers. An obstacle database

$$D_O = \{(\mathbf{s}_o,\, \mathbf{q}_o)\}_{o=1}^{P_o} \tag{5.2}$$

is initialized similarly. The obstacle database does not have to be created at the same time as the target database. Obstacles may be added to the scene at any time, and obstacle-specific databases are created for each new obstacle. The databases are later used for two purposes: (i) Target database D_T for robust visual-motor Jacobian estimation, and (ii) target and obstacles databases D_T and D_O for sampling-based planning and to model visual constraints (and other constraints). During robot operation, the databases are updated by new samples. A dense set of samples improves the success rate of planning, but it is not necessary for Jacobian estimation.

In practice, samples are acquired at 10 Hz or higher for a moving arm. New samples are tested against repetition before updating the database. Within 5 minutes, the database contains around $3,000$ samples, an adequate number to start robust estimation and planning for a static scene.

5.3 Modeling of the Constraints

Our problem requires modeling of both visual constraints and physical/joint constraints. The standard configuration space does not allow modeling of the visual constraints; therefore, we plan in the visual-motor space $\mathcal{V}$ defined in Section 4.1.

Both types of constraints (physical/joint and visual), are mapped to the visual-motor space and marked as $\mathcal{V}_{occupied}$, the *occupied visual-motor space.* Sample points that do not violate any constraints are in $\mathcal{V}_{free}$, the *free visual-motor space.* From now on, we refer to them simply as *occupied* or *free* space. The planning is performed in the free space.

5.3.1 Joint and FOV Limits

Labeling of the joint limit (JLIM) and FOV limit in the occupied space are straightforward. In the visual-motor database, sample $(\mathbf{s}_k, \mathbf{q}_k)$ is labeled as occupied, when either violates their respective constraint. The gathered points in the visual-motor database are usually in the free space as visual-motor readings are within the FOV and joint limits. In practice, a safety margin $\zeta_{fov} = 10$ pixels is used for the FOV as most trackers perform erratically close to the image border. A visual-motor point with any of the coordinates within ζ_{fov} pixels of the image border is not used in planning (considered in the occupied space). Similarly, extra safety limits on the joints are implemented to add safety.

5.3.2 Visual Occlusion

Modeling of visual occlusions needs elaboration. We are concerned with two sets of visual information: the target features $\mathbf{s}_T$ or the obstacle features $\mathbf{s}_O$. The image-based task specification in (4.3) is based on $\mathbf{s}_T$.

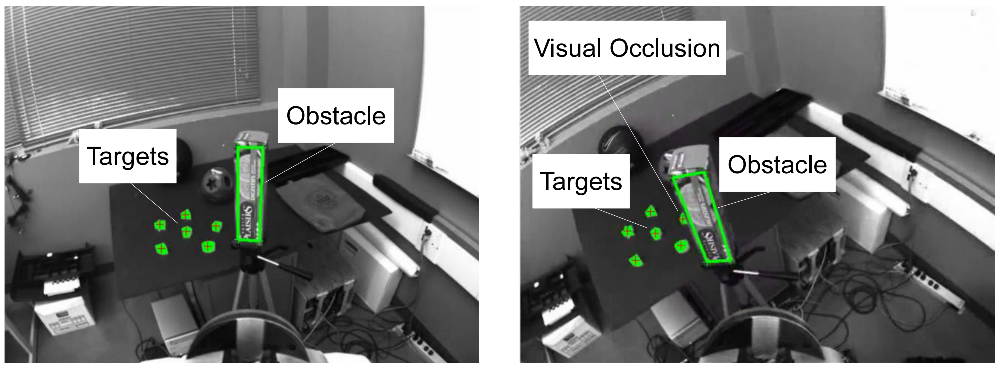
There is a visual-motor database for the target object $\{(\mathbf{s}_t, \mathbf{q})\} \in D_T$ and a visual-motor databases for an obstacle object $\{(\mathbf{s}_o, \mathbf{q})\} \in D_O$. Assuming that the obstacle is a simple planar object and closer to the camera than the target object, visual occlusion occurs when the polygon defined by $\mathbf{s}_o$ masks any of the target visual features $\mathbf{s}_t$ in the image plane. We can find Visual Occlusion Violations (VOV) from geometric properties of points, lines, and polygons. For instance, using the homogenous coordinates in the image plane one can formulate the line going through two points on the obstacle as

$$\mathbf{l}_{obs} = \mathbf{s}_{o,1} \times \mathbf{s}_{o,2}, \tag{5.3}$$

and the point-to-line distance for each corner point of the target as

$$d_i = \mathbf{s}_{t,i} \cdot \mathbf{l}_{obs}. \tag{5.4}$$

If $d_{\min} = \operatorname{argmin}_i d_i$ is smaller than a margin ζ_{vov}, then VOV has either occurred or about to occur for the corresponding point, therefore, $(\mathbf{s}_t, \mathbf{q}) \notin \mathcal{V}_{free}$. Since visual



(a) No VOV (b) VOV

Figure 5.3: Visual occlusion violation (VOV) in practice. (a) Target and obstacle features seen from an in-hand camera. (b) A target features is visually occluded by the obstacle. $d_{min} < \zeta_{vov}$.

occlusion is determined entirely in the image space, there is no need for an explicit 3D model of either the target or the obstacle objects.

Figure 5.3 shows an example of visual occlusion violation. Coplanar points are used as our target object. The obstacle is a box and modeled in the image by tracking one of its faces. It is important to note that while visual occlusion has not technically occurred in this example, during control and planning we flag this as a VOV, since $d_{min} < \zeta_{vov}$. In practice, we use a safety margin of $\zeta_{vov} = 10$ pixels.

The proposed sampling-based planning algorithm, which is based on the Bidirectional RRT (BiRRT) algorithm [34] is explained next. A modified data structure, which incorporates the sensory input from the cameras, is used. In addition, robust methods from Chapter 4 are utilized in planning and path following.

5.4 The Proposed UvsBiRrt Algorithm

The RRT [32, 34] is a tree data structure that efficiently covers the state space by sampling the space and extending the existing tree to the newly sampled state. Because it is efficient and probabilistically complete, it has been used in robotics extensively.

We use visual-motor tuples $\mathbf{x} = (\mathbf{s}, \mathbf{q})$ as states and build a tree in the visual-motor space directly. To extend the RRT, a random state generator, RandomState(), first samples the unexplored free space $\mathcal{V}_{free}$ and picks a random state $\mathbf{x}_{rand} \in \mathcal{V}_{free} \subset D_T$. Next step is to find $\mathbf{x}_{near}$, the nearest vertex or edge of the tree to $\mathbf{x}_{rand}$, by NearestNeighbor(). The random state should not be naively added to the tree, as the extension of $\mathbf{x}_{near}$ to $\mathbf{x}_{rand}$ might not be feasible in the free space. We consider the UVS($\mathbf{s}$,$\mathbf{s}_{goal}$, Duration) in (4.3), to advance $\mathbf{x}_{near}$ towards $\mathbf{x}_{rand}$. This makes the integration of differential constraints possible and allows to check for violation of other constraints. The robust Jacobian estimate, as explained in Chapter 4, is used to ensure robustness against outliers during planning. State $\mathbf{x}_{near}$ is ADVANCED until control is TRAPPED due to constraint violations or $\mathbf{x}_{rand}$ is REACHED. In case of TRAPPED, $\mathbf{x}_{new}$ is the last state, which does not violate the constraints. In case of REACHED, $\mathbf{x}_{new}$ is the same as the random

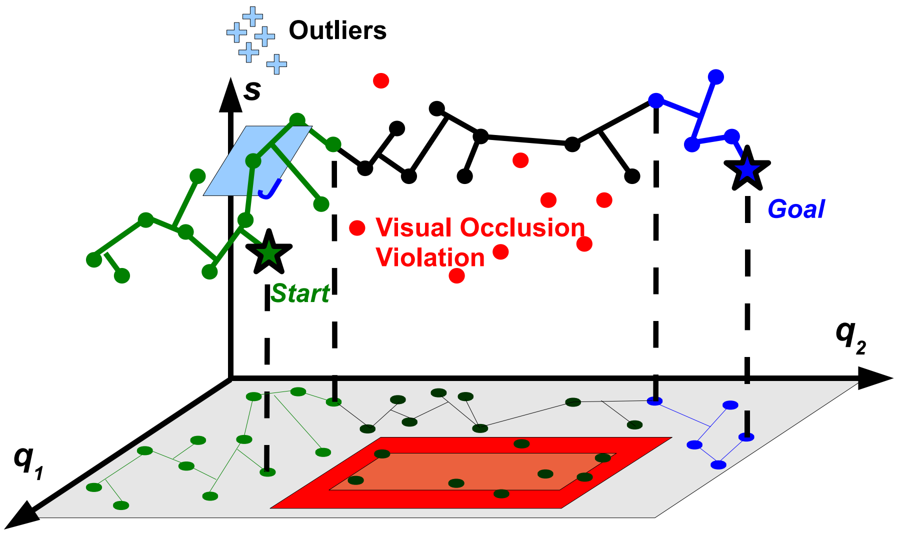


Figure 5.4: Illustration of the UvsBiRrt Algorithm. The Visual Occlusion Violation (VOV) is projected on to the joint space with a safety margin. At each state a robust Jacobian is estimated and kept for path following.

sample. The final state, $\mathbf{x}_{new}$, does not violate VOV, FOV, or JLIM constraints in either case. The pseudocode for the tree extension algorithm, ExtendUvsRrt is presented in Algorithm 4.

Our planning strategy uses a bidirectional version of the RRT planner with the above tree extension algorithm as introduced in UvsBiRrt algorithm (Algorithm 5). The algorithm starts by initializing tree T_a at the initial state $\mathbf{x}_{init}$ and tree T_b at the goal state x_{goal}. When a new random state is sampled, we first estimate its local robust visual-motor Jacobian from the visual-motor database according to (4.7) and (4.23). When the random state contains outlier features, we remove the outliers robustly to obtain $\mathbf{x}_{robust}$ as explained in Section 4.5. As such, statistical robustness is built into the planning algorithm. Note that to follow a planned path using the UVS controller, estimates of the visual-motor Jacobian along the path will be needed; therefore, this step does not increase the computational time significantly.

Once the robust random state $\mathbf{x}_{robust}$ is generated, the two trees are extended towards each other until they have a common vertex. At this point, a path from $\mathbf{x}_{init}$ to $\mathbf{x}_{goal}$ is returned as the solution by Path(). If a path is not found within a fixed number of iterations, the algorithm returns a FAILURE. The diagram in Figure 5.4 illustrates the proposed algorithm schematically.

A vanilla unidirectional RRT planner can also be used to reach the goal state, but UvsBiRrt is more efficient. The image-based task is specified by known goal visual features $\mathbf{s}_{goal}$, but the full state which contains the goal configuration $\mathbf{q}_{goal}$ is not known. To remedy this problem, we estimate the goal state using a virtual visual servoing scheme as explained next.

Algorithm 4 ExtendUvsRrt(T, $\mathbf{x}_{rand}$)

```
1:  $\mathbf{x}_{near} \leftarrow$ NearestNeighbor($\mathbf{x}_{rand}$, $T$)
2:  $i \leftarrow 1$; $\mathbf{x}_i \leftarrow \mathbf{x}_{near}$
3:  while $i < MAX$ do
4:      $\mathbf{x}_{i+1} = (\mathbf{s}_{i+1}, \mathbf{q}_{i+1}) \leftarrow$ UVS ($\mathbf{s}_i$, $\mathbf{s}_{rand}$, $\Delta t$)
5:      if VOV($\mathbf{x}_{i+1}$) $\vee$ FOV($\mathbf{s}_{i+1}$) $\vee$ JLim($\mathbf{q}_{i+1}$) then
6:          $\mathbf{x}_{new} \leftarrow \mathbf{x}_i$; flag $\leftarrow$ TRAPPED
7:      else if $||\mathbf{x}_{i+1} - \mathbf{x}_{rand}|| < \epsilon$ then
8:          $\mathbf{x}_{new} \leftarrow \mathbf{x}_{i+1}$; flag $\leftarrow$ REACHED
9:      else
10:          flag $\leftarrow$ ADVANCED
11:     end if
12:     if flag is REACHED or TRAPPED then
13:         $T.add(\mathbf{x}_{new})$
14:         return flag
15:     else
16:         $\mathbf{x}_i \leftarrow \mathbf{x}_{i+1}$; $i \leftarrow i + 1$
17:     end if
18: end while
```

Algorithm 5 UvsBiRrt($\mathbf{x}_{init}$,$\mathbf{x}_{goal}$)

```
1:  $T_a.init(\mathbf{x}_{init})$; $T_b.init(\mathbf{x}_{goal})$
2:  for $k = 1$ to $K$ do
3:      $\mathbf{x}_{rand} \leftarrow$ RandomState()
4:      $[\mathbf{x}_{robust}, \widehat{\mathbf{J}}_{\mathbf{u}}] \leftarrow$ JacobianEstIRLS($\mathbf{x}_{rand}, D_T$)
5:      if ExtendUvsRrt($T_a, \mathbf{x}_{robust}$) is not TRAPPED then
6:          if ExtendUvsRrt($T_b, \mathbf{x}_{new}$) is REACHED then
7:              return Path($T_a, T_b$)
8:          end if
9:      end if
10:     Swap($T_a, T_b$)
11: end for
12: return FAILURE
```

5.5 Estimation of the Goal State

The goal visual measurement $\mathbf{s}_{goal}$ is known from task specification and is assumed to be occlusion free. To obtain an estimate of the full goal state, the target visual-motor database D_T (see Section 4.3) is used. Simple processing of D_T, such as averaging or linear interpolation of the visual measurements, to obtain $\mathbf{x}_{goal}$ from $\mathbf{s}_{goal}$ is not successful due to two problems: (i) The neighbors of $\mathbf{s}_{goal}$ are not the same as the neighbors of $\mathbf{q}_{goal}$, because of high dimensionality; and (ii) the existence of outliers in the visual-motor database makes it difficult to estimate it using simple methods.

In order to find a reliable estimate, we use an offline *virtual* visual servoing scheme on D_T to simulate the propagation of $\mathbf{q}_{start}$ towards $\mathbf{q}_{goal}$. Similar to the normal visual servoing, we compute the statistically-robust Jacobian according to

(4.7) and (4.23) and use the control law in (4.3). When the next state is not available in D_T, the nearest neighbor joint is chosen and the corresponding visual feature is chosen as the current measurement. A similar outlier replacement scheme as explained in section 4.5 is used to mitigate the effect of outliers. At this stage, the visual occlusion constraint is not considered as the goal is to estimate the goal joint position, which by task definition, does not violate any constraints. The planning scheme explained earlier will take care of occlusion avoidance and other constraints later.

This approach estimates $\widehat{\mathbf{q}}_{goal} \neq \mathbf{q}_{goal}$. In practice it works well and returns an error of only a few degrees. This means that after planning, a short servoing to regulate the error to zero might be necessary to close the gap between $\widehat{\mathbf{q}}_{goal}$ and $\mathbf{q}_{goal}$.

5.6 Summary

Path planning can be used to avoid physical and visual constraints in visual servoing under calibrated assumptions [10]. A study of the literature (Chapter 3) shows room for improvement in two categories: (i) Model/calibration dependence. Most methods rely on having an accurate model of the obstacle to model visual occlusions or physical collisions; therefore, they are of limited use in unstructured environments. (ii) Robustness against outliers. Current methods do not consider outliers in the visual measurements caused by visual tracking malfunction. Unfortunately, in real unstructured scenarios, visual tracking failures are quite common and planner should be robust against them.

In this chapter, we presented UvsBiRrt, a robust and efficient sampling-based planning framework based on RRT, to avoid constraints during UVS. The proposed framework is relevant to situations, where exact models of the environment are either non-existent or tedious to acquire. Other methods in the literature require either a calibrated camera or scene model [33, 85, 98].

The UvsBiRrt algorithm builds a tree in the visual-motor space. A visual-motor database was used in planning to avoid joint limit (JLIM) and field-of-view (FOV) limit as well as to avoid visual occlusions (VO). Some constraints, *e.g.,* the FOV, can be checked in the visual space. Some other constraints, *e.g.,* the JLIM, can be checked in the motor (joint) space. Visual occlusions can be checked in the visual space, as only the images of the obstacle and target in one view are needed to determine that. However, to check the VOV when the obstacle and target databases are not created at the same time, the obstacle visual-motor space and the target visual-motor space need to be *aligned* in the joint space first. Working in the visual-motor space has the advantage that the modeling of all types of constraints are unified.

The proposed planning algorithm is extended from the Rapidly-Exploring Random Tree (RRT) planner [34] and works in the visual-motor space to avoid visual occlusions of the target by the obstacle during servoing. A unique property of the planner is that it works directly with the raw data and is able to reject outliers online, after a brief offline training session. The raw visual-motor data has been used for both control (Chapter 4) and planning (Chapter 5) purposes.

There are limitations to what can be inferred from raw sensory data. In particular, self-collision and collision with physical obstacles cannot be avoided, because geometric models are not available. Another caveat with such an uncalibrated approach is that the unvisited sections of the visual-motor space are not included in the planning phase.

Experimental results and evaluations are presented in Part III, Chapter 7.

Chapter 6

Uncalibrated Visual Servoing from Three Views

In the previous chapter, we showed how a statistically robust Jacobian estimation technique could be utilized to achieve robustness in unstructured settings. In this chapter[1], we present another method for uncalibrated visual servoing, which is based on using pure projective-geometric measurements obtained directly from image feature correspondences.

The three-view projective geometry of the initial, current, and desired views is exploited to select features for uncalibrated visual servoing of a 6-DOF manipulator. This geometry is encapsulated by the trifocal tensor, which is independent of the scene and depends only on the projective relations between the cameras [110]. We begin by distinguishing a new approach to visual servoing: The projective-geometric visual servoing (PGVS). In this approach, the geometric properties of the sensor (camera) is exploited and projective-geometric measures are controlled directly towards completing a task. One of these projective-measures is obtained from the geometry of three views, namely the trifocal tensor. We will briefly describe how the trifocal tensor is numerically computed and then formulate our 6-DOF uncalibrated approach to PGVS based on the trifocal tensor.

6.1 Projective-Geometric Visual Servoing

Two-view geometry has been extensively studied in visual servoing. Epipolar geometry can be used to estimate depth, which appears in the interaction matrix of point features [111] for an improved stability [26]. Chesi *et al.* exploit the symmetry of epipolar geometry without point correspondences to control a holonomic mobile robot from a partially calibrated camera [112]. Mariottini *et al.* use epipolar geometry for visual servoing of non-holonomic mobile robots [113]. Becerra and Sagues develop a sliding mode control law using epipolar geometry for non-holonomic mobile robots [114]. Homography-based methods have also been studied in visual servoing. Benhimane and Malis develop a homography-based approach without reconstructing any 3D parameters [115]. Lopez-Nicolas *et al.* design a homography-based controller

[1]The results of this chapter have been partially published in the proceedings of and presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) in October 2010 [109].

which considers the non-holonomic constraints [116]. These methods use the two-view geometry between the observed and desired views and ignore their relation with the *initial view*. In addition, epipolar geometry is not well-conditioned if the features are coplanar or the baseline is short, while homography-based approaches require dominant planes [117].

Projective geometry of three views can also be used in vision-based motion control. The trifocal tensor relates three views in a similar manner that the fundamental matrix relates two views. The application of trifocal tensor to visual servoing has been neglected in the visual servoing literature until very recently [117, 118]. Becerra and Sagues use a simplified trifocal tensor as measurement and estimate and track the pose of a non-holonomic mobile robot with Extended Kalman Filter (EKF) [117]. Lopez-Nicolas *et al.* [118] also use the constrained camera motion on a mobile robot and linearize the input-output space for control. This approach provides an analytic interaction matrix, which relates the variations of 9 elements of the trifocal tensor to robot velocities. To the best of our knowledge, the trifocal tensor has not been used to control a 6-DOF robot prior to our work [109]. This is likely due to the difficulty in linearizing the input-output space in the case of the generalized 6-DOF camera motions.

The main contribution of this chapter is to propose a 6-DOF uncalibrated visual servoing approach which uses the elements of the trifocal tensors as *visual features*. The advantage of uncalibrated methods over analytical methods is clear when it is not easy to derive the interaction matrix for a particular set of features. The visual features, here, are a subset of the 27 elements of the $3 \times 3 \times 3$ trifocal tensor. We estimate the Jacobian matrix that relates joint velocities to the rate of change of these tensor-based visual features. This work does not fall into the conventional 2D, 3D, and hybrid classifications of visual servoing based on control law. There seems to be a missing visual servoing class, *projective visual servoing*, where the control loop is closed over projective measures (for example, controlling the epipoles [113], trifocal tensor features [118], etc.). In essence, we control one such projective measure, which is found directly from images across three views, without explicitly recovering the camera pose or directly closing the loop in the image space.

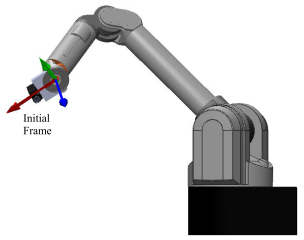
6.2 The Geometry of Three Views

Figure 6.1 illustrates the start, intermediate, and goal camera frames at three different poses. A 3D point $\mathbf{X} \in \mathbb{R}^3$ in Euclidean space projects onto the image plane by a 3×4 projection matrix

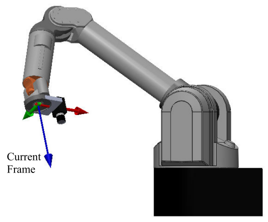
$$\mathbf{P} = \mathbf{K}[\mathbf{R} \quad -\mathbf{R}\tilde{\mathbf{C}}], \tag{6.1}$$

where $\mathbf{K}$ is the camera intrinsic matrix, $\mathbf{R}$ is the rotation of the camera frame with respect to the world frame, and $\tilde{\mathbf{C}}$ is the coordinate of the camera frame expressed in the world frame. The homogeneous coordinates of image point $\mathbf{x}$ can be found from

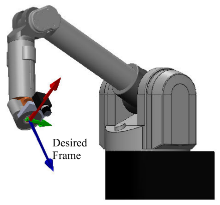
$$\mathbf{x} = \mathbf{P}\mathbf{X}. \tag{6.2}$$



(a) Start



(b) Intermediate



(c) Goal

Figure 6.1: The trifocal tensor is independent of the scene and depends on the camera projection matrices of three views. (a) The start robot configuration and camera frame, which is considered static, (b) The intermediate robot configuration and camera frame, which is variable and moves from the start to the goal, and (c) The goal robot configurations and camera frames (also considered static).

Let the projection matrices for the start, intermediate, and goal camera frames be $\mathbf{P}_s$, $\mathbf{P}_i$, and $\mathbf{P}_g$, respectively. A 3D point $\mathbf{X}$ projects to

$$\begin{bmatrix} \mathbf{x}_s \\ \mathbf{x}_i \\ \mathbf{x}_g \end{bmatrix} = \begin{bmatrix} \mathbf{P}_s \\ \mathbf{P}_i \\ \mathbf{P}_g \end{bmatrix} \mathbf{X}, \tag{6.3}$$

where $\mathbf{x}_s$ is the image of the 3D point in the start (initial) view, $\mathbf{x}_i$ is the image of the same 3D point in the intermediate (current) view, and $\mathbf{x}_g$ is the image of that point in the goal (desired) view. The three-view point correspondence is denoted by

$$\mathbf{x}_s \leftrightarrow \mathbf{x}_i \leftrightarrow \mathbf{x}_g. \tag{6.4}$$

In our case, the start camera matrix $\mathbf{P}_s$ and the goal camera matrix $\mathbf{P}_g$ are constant. It is only the intermediate camera matrix $\mathbf{P}_i$ and the image points in the corresponding view that change as the robot configuration changes.

6.3 The Trifocal Tensor

The trifocal tensor encapsulates the geometry of three views in a similar manner that the fundamental matrix encapsulates the geometry of two views. They are independent of the scene and depend only on the camera projection matrices. Given one view and the fundamental matrix, image points in the first view can be transferred to the second view. Similarly, given two views and point (or line) correspondences across two-views, and the trifocal tensor, image points (or lines) can be transferred to the third view.

The trifocal tensor is more general than combining the existing epipolar geometries between views (1,2), (1,3), and (2,3). For example, epipolar transfer fails for points lying on the trifocal plane, the plane defined by the three camera centres, but the trifocal tensor can be used for point transfer in this case [110]. The trifocal tensor and its computation is described in detail in several chapters of the "Multiple View Geometry in Computer Vision" textbook by Hartley and Zisserman [110]. A brief presentation, which motivates the proposed application of the trifocal tensor in 6-DOF uncalibrated visual servoing follows.

Consider canonical representations for three camera projection matrices $\mathbf{P}_s = [\mathbf{I}|\mathbf{0}]$, and $\mathbf{P}_i = [\mathbf{A}|\mathbf{a}_4]$, and $\mathbf{P}_g = [\mathbf{B}|\mathbf{b}_4]$, where $\mathbf{I}$ is the 3×3 identity matrix, $\mathbf{O}$ is the null 3×1 vector, $\mathbf{A}$ and $\mathbf{B}$ are rank-two 3×3 matrices, and vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ for $i = 1, \cdots, 4$ are the i-th columns of camera matrices $\mathbf{P}_i$ and $\mathbf{P}_g$, respectively. Let

$$\mathbf{T}_i = \mathbf{a}_i \mathbf{b}_4^\top - \mathbf{a}_4 \mathbf{b}_i^\top, \tag{6.5}$$

for $i = 1, 2, 3$. The set of three matrices $\{\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3\}$ constitute the trifocal tensor in matrix notation:

$$\mathbf{T}_1 = \begin{bmatrix} \tau_1 & \tau_4 & \tau_7 \\ \tau_2 & \tau_5 & \tau_8 \\ \tau_3 & \tau_6 & \tau_9 \end{bmatrix} \tag{6.6}$$

$$\mathbf{T}_2 = \begin{bmatrix} \tau_{10} & \tau_{13} & \tau_{16} \\ \tau_{11} & \tau_{14} & \tau_{17} \\ \tau_{12} & \tau_{15} & \tau_{18} \end{bmatrix} \tag{6.7}$$

$$\mathbf{T}_3 = \begin{bmatrix} \tau_{19} & \tau_{22} & \tau_{25} \\ \tau_{20} & \tau_{23} & \tau_{26} \\ \tau_{21} & \tau_{24} & \tau_{27} \end{bmatrix}, \tag{6.8}$$

where τ_k for $k = 1, \cdots, 27$ are the elements of the trifocal tensor. The trifocal tensor is, in fact, a $3 \times 3 \times 3$ cube of cells with 27 elements. The equivalent camera projections are specified, up to a projective transformation, by only 18 parameters. This enforces 8 additional internal algebraic constraints on the elements of the trifocal tensor [110].

The trifocal tensor is completely represented by the camera projection matrices, it is easy to derive the individual elements for the canonical representation to give an idea of their form.

Let the camera intrinsic matrix $\mathbf{K}$ in (6.1) be the identity matrix. The canonical camera projection matrices in this case can be represented by

$$\mathbf{P}_s = \left[\begin{array}{c|c} \mathbf{I} & \mathbf{0} \end{array} \right], \tag{6.9}$$

$$\mathbf{P}_i = \left[\begin{array}{c|c} R(\phi_i, \theta_i, \psi_i) & \begin{matrix} X_i \\ Y_i \\ Z_i \end{matrix} \end{array} \right], \tag{6.10}$$

$$\mathbf{P}_g = \left[\begin{array}{c|c} R(\phi_g, \theta_g, \psi_g) & \begin{matrix} X_g \\ Y_g \\ Z_g \end{matrix} \end{array} \right], \tag{6.11}$$

where the 3×3 matrix R represents rotation with ZYX Euler angles (also called *Roll-Pitch-Yaw angles*), with roll angle ϕ, pitch angle θ, and yaw angle ψ. The last columns of the projection matrices represent the translation. The rotation matrix R is

$$R(\phi, \theta, \psi) = \begin{bmatrix} \mathbf{c}\phi\, \mathbf{c}\theta & \mathbf{c}\phi\, \mathbf{s}\theta\, \mathbf{s}\psi - \mathbf{s}\phi\, \mathbf{c}\psi & \mathbf{c}\phi\, \mathbf{s}\theta\, \mathbf{c}\psi + \mathbf{s}\phi\, \mathbf{s}\psi \\ \mathbf{s}\phi\, \mathbf{c}\theta & \mathbf{s}\phi\, \mathbf{s}\theta\, \mathbf{s}\psi - \mathbf{c}\phi\, \mathbf{c}\psi & \mathbf{s}\phi\, \mathbf{s}\theta\, \mathbf{c}\psi - \mathbf{c}\phi\, \mathbf{s}\psi \\ -\mathbf{s}\theta & \mathbf{c}\theta\, \mathbf{s}\psi & \mathbf{c}\theta\, \mathbf{c}\psi \end{bmatrix}, \tag{6.12}$$

where notations $\mathbf{c}$ and $\mathbf{s}$ are short for cos and sin, respectively.

In this canonical form, the trifocal tensor elements τ_k have the following form:

$$\begin{aligned}
\tau_1 &= -\mathbf{c}\phi_g\,\mathbf{c}\theta_g\,X_i + \mathbf{c}\phi_i\,\mathbf{c}\theta_i\,X_g, && (6.13)\\
\tau_2 &= -\mathbf{c}\phi_g\,\mathbf{c}\theta_g\,Y_i + \mathbf{s}\phi_i\,\mathbf{c}\theta_i\,X_g, && (6.14)\\
\tau_3 &= -\mathbf{c}\phi_g\,\mathbf{c}\theta_g\,Z_i - \mathbf{s}\theta_i\,X_g. && (6.15)
\end{aligned}$$

$$\begin{aligned}
\tau_4 &= -\mathbf{s}\phi_g\,\mathbf{c}\theta_g\,X_i + \mathbf{c}\phi_i\,\mathbf{c}\theta_i\,Y_g, && (6.16)\\
\tau_5 &= -\mathbf{s}\phi_g\,\mathbf{c}\theta_g\,Y_i + \mathbf{s}\phi_i\,\mathbf{c}\theta_i\,Y_g, && (6.17)\\
\tau_6 &= -\mathbf{s}\phi_g\,\mathbf{c}\theta_g\,Z_i - \mathbf{s}\theta_i\,Y_g. && (6.18)
\end{aligned}$$

$$\begin{aligned}
\tau_7 &= \mathbf{s}\theta_g\,X_i + \mathbf{c}\phi_i\,\mathbf{c}\theta_i\,Z_g, && (6.19)\\
\tau_8 &= \mathbf{s}\theta_g\,Y_i + \mathbf{s}\phi_i\,\mathbf{c}\theta_i\,Z_g, && (6.20)\\
\tau_9 &= \mathbf{s}\theta_g\,Z_i - \mathbf{s}\theta_i\,Z_g. && (6.21)
\end{aligned}$$

$$\begin{aligned}
\tau_{10} &= (\mathbf{s}\phi_g\,\mathbf{c}\psi_g - \mathbf{c}\phi_g\,\mathbf{s}\theta_g\,\mathbf{s}\psi_g)\,X_i + (-\mathbf{s}\phi_i\,\mathbf{c}\psi_i + \mathbf{c}\phi_i\,\mathbf{s}\theta_i\,\mathbf{s}\psi_i)\,X_g, && (6.22)\\
\tau_{11} &= (\mathbf{s}\phi_g\,\mathbf{c}\psi_g - \mathbf{c}\phi_g\,\mathbf{s}\theta_g\,\mathbf{s}\psi_g)\,Y_i + (\mathbf{c}\phi_i\,\mathbf{c}\psi_i + \mathbf{s}\phi_i\,\mathbf{s}\theta_i\,\mathbf{s}\psi_i)\,X_g, && (6.23)\\
\tau_{12} &= (\mathbf{s}\phi_g\,\mathbf{c}\psi_g - \mathbf{c}\phi_g\,\mathbf{s}\theta_g\,\mathbf{s}\psi_g)\,Z_i + (\mathbf{c}\theta_i\,\mathbf{s}\psi_i)\,X_g. && (6.24)
\end{aligned}$$

$$\begin{aligned}
\tau_{13} &= -(\mathbf{c}\phi_g\,\mathbf{c}\psi_g + \mathbf{s}\phi_g\,\mathbf{s}\theta_g\,\mathbf{s}\psi_g)\,X_i + (-\mathbf{s}\phi_i\,\mathbf{c}\psi_i + \mathbf{c}\phi_i\,\mathbf{s}\theta_i\,\mathbf{s}\psi_i)\,Y_g, && (6.25)\\
\tau_{14} &= -(\mathbf{c}\phi_g\,\mathbf{c}\psi_g + \mathbf{s}\phi_g\,\mathbf{s}\theta_g\,\mathbf{s}\psi_g)\,Y_i + (\mathbf{c}\phi_i\,\mathbf{c}\psi_i + \mathbf{s}\phi_i\,\mathbf{s}\theta_i\,\mathbf{s}\psi_i)\,Y_g, && (6.26)\\
\tau_{15} &= -(\mathbf{c}\phi_g\,\mathbf{c}\psi_g + \mathbf{s}\phi_g\,\mathbf{s}\theta_g\,\mathbf{s}\psi_g)\,Z_i + (\mathbf{c}\theta_i\,\mathbf{s}\psi_i)\,Y_g. && (6.27)
\end{aligned}$$

$$\begin{aligned}
\tau_{16} &= -(\mathbf{c}\theta_g\,\mathbf{s}\psi_g)\,X_i + (-\mathbf{s}\phi_i\,\mathbf{c}\psi_i + \mathbf{c}\phi_i\,\mathbf{s}\theta_i\,\mathbf{s}\psi_i)\,Z_g, && (6.28)\\
\tau_{17} &= -(\mathbf{c}\theta_g\,\mathbf{s}\psi_g)\,Y_i + (\mathbf{c}\phi_i\,\mathbf{c}\psi_i + \mathbf{s}\phi_i\,\mathbf{s}\theta_i\,\mathbf{s}\psi_i)\,Z_g, && (6.29)\\
\tau_{18} &= -(\mathbf{c}\theta_g\,\mathbf{s}\psi_g)\,Z_i + (\mathbf{c}\theta_i\,\mathbf{s}\psi_i)\,Z_g. && (6.30)
\end{aligned}$$

$$\begin{aligned}
\tau_{19} &= -(\mathbf{s}\phi_g\,\mathbf{s}\psi_g + \mathbf{c}\phi_g\,\mathbf{s}\theta_g\,\mathbf{c}\psi_g)\,X_i + (\mathbf{s}\phi_i\,\mathbf{s}\psi_i + \mathbf{c}\phi_i\,\mathbf{s}\theta_i\,\mathbf{c}\psi_i)\,X_g, && (6.31)\\
\tau_{20} &= -(\mathbf{s}\phi_g\,\mathbf{s}\psi_g + \mathbf{c}\phi_g\,\mathbf{s}\theta_g\,\mathbf{c}\psi_g)\,Y_i + (-\mathbf{c}\phi_i\,\mathbf{s}\psi_i + \mathbf{s}\phi_i\,\mathbf{s}\theta_i\,\mathbf{c}\psi_i)\,X_g, && (6.32)\\
\tau_{21} &= -(\mathbf{s}\phi_g\,\mathbf{s}\psi_g + \mathbf{c}\phi_g\,\mathbf{s}\theta_g\,\mathbf{c}\psi_g)\,Z_i + (\mathbf{c}\theta_i\,\mathbf{c}\psi_i)\,X_g. && (6.33)
\end{aligned}$$

$$\begin{aligned}
\tau_{22} &= (\mathbf{c}\phi_g\,\mathbf{s}\psi_g - \mathbf{s}\phi_g\,\mathbf{s}\theta_g\,\mathbf{c}\psi_g)\,X_i + (\mathbf{s}\phi_i\,\mathbf{s}\psi_i + \mathbf{c}\phi_i\,\mathbf{s}\theta_i\,\mathbf{c}\psi_i)\,Y_g, && (6.34)\\
\tau_{23} &= (\mathbf{c}\phi_g\,\mathbf{s}\psi_g - \mathbf{s}\phi_g\,\mathbf{s}\theta_g\,\mathbf{c}\psi_g)\,Y_i + (-\mathbf{c}\phi_i\,\mathbf{s}\psi_i + \mathbf{s}\phi_i\,\mathbf{s}\theta_i\,\mathbf{c}\psi_i)\,Y_g, && (6.35)\\
\tau_{24} &= (\mathbf{c}\phi_g\,\mathbf{s}\psi_g - \mathbf{s}\phi_g\,\mathbf{s}\theta_g\,\mathbf{c}\psi_g)\,Z_i + (\mathbf{c}\theta_i\,\mathbf{c}\psi_i)\,Y_g. && (6.36)
\end{aligned}$$

$$\begin{aligned}
\tau_{25} &= -(\mathbf{c}\theta_g\,\mathbf{c}\psi_g)\,X_i + (\mathbf{s}\phi_i\,\mathbf{s}\psi_i + \mathbf{c}\phi_i\,\mathbf{s}\theta_i\,\mathbf{c}\psi_i)\,Z_g, && (6.37)\\
\tau_{26} &= -(\mathbf{c}\theta_g\,\mathbf{c}\psi_g)\,Y_i + (-\mathbf{c}\phi_i\,\mathbf{s}\psi_i + \mathbf{s}\phi_i\,\mathbf{s}\theta_i\,\mathbf{c}\psi_i)\,Z_g, && (6.38)\\
\tau_{27} &= -(\mathbf{c}\theta_g\,\mathbf{c}\psi_g)\,Z_i + (\mathbf{c}\theta_i\,\mathbf{c}\psi_i)\,Z_g. && (6.39)
\end{aligned}$$

It is seen that all terms are linear combinations of nonlinear trigonometric terms. While the elements are nonlinear, they are smooth and derivatives can be easily computed. A sample camera trajectory and the elements of the trifocal tensor are shown in Figure 6.2.

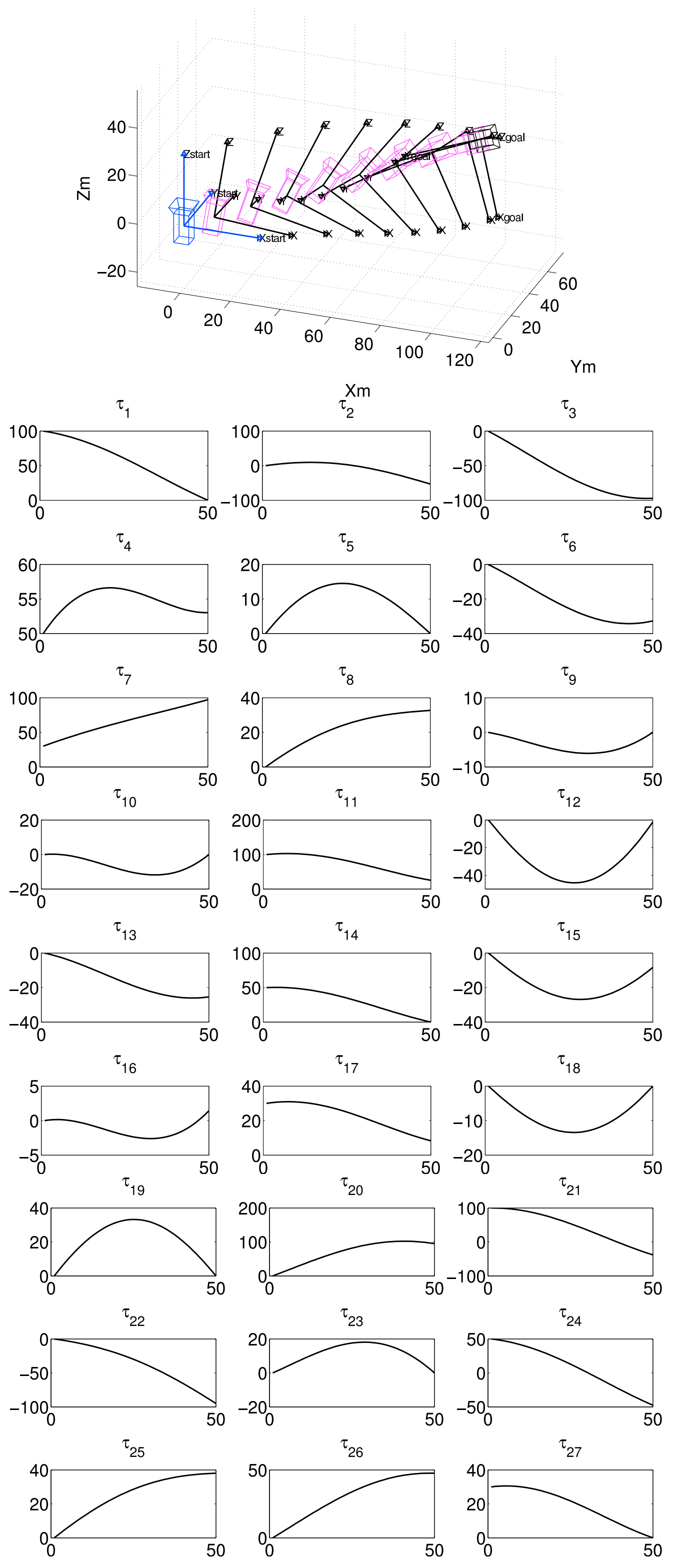


Figure 6.2: A sample camera trajectory and the corresponding variations of the trifocal tensor elements τ_k for $k = 1, \cdots, 27$. The variations of the tensor elements are smooth.

6.4 Estimation of the Trifocal Tensor

The trifocal tensor can be estimated from point or line correspondences across three views [110]. The basic method is a normalized linear algorithm that finds a direct solution from a set of linear equations. If only image point correspondences are used, at least 7 points are required since each point correspondence provides 4 independent equations. The problem with this basic method is that the resulting tensor might violate some of the 8 independent algebraic constraints. An immediate improvement is to use tools from constrained optimization to find a constrained least-squares solution. An iterative algebraic minimization algorithm provides a tensor which satisfies the constraints. Furthermore, this geometrically valid estimate of the tensor can be used as an initial estimate for a maximum likelihood solution to minimize the geometric error. These methods are vulnerable to outliers as they consider all point correspondences as inliers. Robust estimation based on RANSAC [59] can be used to handle outliers in this case.

In practice, it is easier to compute a projective reconstruction from 6 point correspondences across the three views [119], which has been explained in detail as Algorithm 20.1 in Hartley and Zisserman [110]. The corresponding 3D points need to be in general configuration as the resulting algorithm returns

$$\mathbf{X}_1 = \begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \mathbf{X}_2 = \begin{bmatrix}a\\b\\c\\d\end{bmatrix}, \mathbf{X}_3 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \mathbf{X}_4 = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \mathbf{X}_5 = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \mathbf{X}_6 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \tag{6.40}$$

where a, b, c, and d are scalars and $d \neq 0$. The camera projection matrices are found up to a projectivity with that algorithm. There are either one or three solutions. The trifocal tensor can be recovered using (6.5).

At this point, an estimate of the trifocal tensor is available from a minimal set of 6 image point correspondences. To automatically estimate the trifocal tensor, one can use the RANSAC robust estimation [59], label the inliers in the point correspondence set, and compute a geometrically-valid maximum likelihood estimate of the trifocal tensor on the inliers (Algorithm 16.4 in Hartley and Zisserman [110]).

Once the trifocal tensor is estimated, the elements of the tensor can be used in closed-loop feedback control of robot. Becerra and Sagues [114] and Lopez-Nicolas *et al.* [118] have shown that the non-zero elements of the tensor can be used to control a mobile robot with a non-holonomic motion model. The camera is attached to the base of a mobile robot; therefore, camera motion is constrained to moving in a plane in their work. This is a special case and the design and analysis of a controller for the general 6D camera motion by similar methods does not appear to be straightforward.

The approach that we present in this chapter is motivated by the development of the UVS control law as explained in Chapter 2 and the smooth variations of the elements of the trifocal tensor as motivated by the example illustrated in Figure 6.2. The general idea is to estimate the Jacobian matrix that relates the variations of the trifocal tensor to the joint velocities for a camera mounted on a robot arm.

6.5 Problem Formulation

The visual servoing problem is usually set up having known initial and desired features and observing current features (see Fig 6.1). The goal of visual servoing to drive a robot to a desired configuration by regulating a task function to zero. The task function is defined by some features. In our case, these features are found from the projective geometry of three views.

Let $\mathbf{M}_{s,g} : \mathbb{R}^6 \to \mathbb{R}^M$ denote the sensory-motor mapping from configuration $\mathbf{q} \in \mathbb{R}^6$ of a robot with 6 joints (Fig. 6.1), to a vector containing the trifocal tensor elements $\boldsymbol{\tau} \in \mathbb{R}^M$ with M features for start camera matrix $\mathbf{P}_s$, goal camera matrix $\mathbf{P}_g$. The start and goal camera matrices are constant. The intermediate camera matrix $\mathbf{P}_i$ is a function of the robot configuration as it is assumed that the camera is rigidly attached to the robot arm. For visual servoing, measurements with $M \geq 6$ features is needed. The trifocal tensor has 27 elements, which can be found from point correspondences across the three views. A subset of the elements of the trifocal tensor goes into vector $\boldsymbol{\tau} = [\tau_1, \cdots, \tau_M]$. Similar to (2.8), the sensory-motor mapping is written as

$$\boldsymbol{\tau} = \mathbf{M}_{s,g}(\mathbf{q}). \tag{6.41}$$

Fig. 6.2 shows a sample camera trajectory with the evolution of the trifocal tensor elements from the initial to desired pose.

Following a derivation similar to (2.9), the time derivative of the sensory-motor function results in the $M \times 6$ tensor-joint Jacobian, $\mathbf{J}_\tau(\mathbf{q})$:

$$\frac{\partial \boldsymbol{\tau}}{\partial t} = \frac{\partial \mathbf{M}_{s,g}(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t}, \tag{6.42}$$

$$\dot{\boldsymbol{\tau}} = \mathbf{J}_\tau(\mathbf{q})\dot{\mathbf{q}}. \tag{6.43}$$

With an estimate $\widehat{\mathbf{J}}_\tau(\mathbf{q})$ for $\mathbf{J}_\tau(\mathbf{q})$, the discrete-time form of (6.43) becomes

$$\Delta\boldsymbol{\tau} \simeq \widehat{\mathbf{J}}_\tau(\mathbf{q})\Delta\mathbf{q}. \tag{6.44}$$

The tensor-joint Jacobian is an integral part of the visual servoing control law. A weighted Jacobian may be written as follows [61]:

$$\widehat{\mathbf{J}}_\beta = \beta\widehat{\mathbf{J}}_{\mathbf{d}} + (1-\beta)\widehat{\mathbf{J}}_\tau(\mathbf{q}), \tag{6.45}$$

where $\widehat{\mathbf{J}}_{\mathbf{d}} = \widehat{\mathbf{J}}_\tau(\mathbf{q}_{\mathbf{d}})$ is the estimated Jacobian at the desired state. The control law in uncalibrated visual servoing is defined entirely in the feature space. To reach a tensor goal $\boldsymbol{\tau}_d$, the general Jacobian $\widehat{\mathbf{J}}_\beta$ can be used in the following control law:

$$\dot{\mathbf{q}} = -\lambda\widehat{\mathbf{J}}_\beta^\dagger(\boldsymbol{\tau} - \boldsymbol{\tau}_d), \tag{6.46}$$

where $\lambda < 1$ is a positive constant to make joint velocity small, and $\widehat{\mathbf{J}}_\beta^\dagger$ is the Moore-Penrose pseudoinverse of $\widehat{\mathbf{J}}_\beta$.

Similar to the methods described in Chapter 4, the robot keeps a record of the sensory-motor information while it operates in the environment. At each observed state, both the image points and the estimated tensor are recorded. The tensor-joint Jacobian $\mathbf{J}_\tau$ is estimated using the tensor-joint samples from the memory and the JacobianEstIRLS algorithm explained in Section 4.4.4. The only difference is that the vector of the elements of the trifocal tensor, $\boldsymbol{\tau}$, is used here, while previously, we used the coordinates of image features denoted by vector $\mathbf{s}$.

6.6 Summary

We presented a new uncalibrated visual servoing approach based on the trifocal tensor. The trifocal tensor encapsulates the geometry of the start, goal, and intermediate camera views. We use the elements of the trifocal tensor to construct a task function from the desired and current estimation of the trifocal tensor. Unlike conventional image-based methods, where the Jacobian relates the joint velocities to image measurements (coordinates, moments, etc.), the proposed Jacobian directly relates the joint velocities to the rate of change in the elements of the trifocal tensor. The trifocal tensor can be estimated from point correspondences up to a projectivity. Such control laws may be considered as *projective visual servoing*, where the control law uses projective measures instead of 2D, 3D, or hybrid.

This work is closely related to the recent works of Becerra and Sagues [114] and Lopez-Nicolas *et al.* [118] with distinct differences as detailed in the proceeding sections. In summary, the differences are threefold. First, we consider the 6-DOF motion of an eye-in-hand camera, but they consider the planar motion of the camera constrained to the non-holonomic motion model of a mobile robot. Second, they derive an analytic form of the Jacobian and use input-output linearization for control. In contrast, we estimate the Jacobian matrix directly from measurements of the trifocal tensor elements, because an analytic form is cumbersome to derive for the general motion. Finally, we use a control law similar to the typical proportional control law in image-based visual servoing.

Experimental results and evaluations are presented in Part III, Chapter 8. The results show that the proposed method is very promising and handles well “hard” configurations (such as large rotation around the view axis).

Part III

Evaluation and Experimental Results

Chapter 7

Evaluation: Robust Visual Servoing and Planning

7.1 Experimental Setup

An uncalibrated eye-in-hand image-based visual servoing systems is considered as shown in Figure 4.7. We experimentally evaluate the performance of our proposed algorithms and compare it with the LLS method [9] and a reference Jacobian obtained by orthogonal test movements (see Section 2.4.1). The performance is evaluated using four fiducial markers. There are 8 visual features (two coordinates in the image space for each point) and four joints, *i.e.*, $\widehat{\mathbf{J}}_{\mathbf{u}} \in \mathbb{R}^{8\times 4}$.

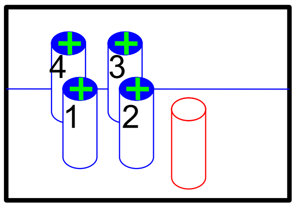
We first perform 100 simulation experiments with a software that was developed in MATLAB. It builds on the Robotics Toolbox [120] and the Epipolar Geometry Toolbox [121]. After validation of simulation results for visual servoing in uncalibrated settings, we designed similar experiments on our experimental setup. The manipulator under study is a 4 degree-of-freedom WAM Arm that runs on an RTAI-Linux box [122]. The vision system consists of a Point Grey Grasshopper camera that captures 640×480 MONO8 images at 60 Hz, and the Visual Servoing Platform (ViSP) [123] for visual tracking.

7.2 Outlier Types

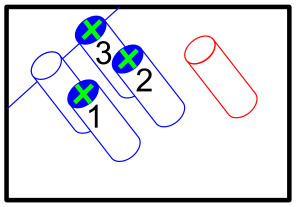
We consider three types of outliers that may occur in a visual servoing scenario because of the mis-tracking or loss of image features. This is depicted in Figure 7.1 and explained next.

7.2.1 Type-1 (lost) outliers

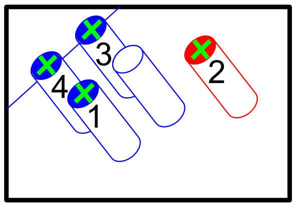
Type-1 outliers represent lost features due to leaving the field of view, getting occluded by obstacles, or other reasons. Lost features are replaced with zero in the visual feature vector. This is done to keep the dimension of the Jacobian matrix numerically consistent in the control law in (4.3). In the analysis, we consider losing only one of point features, which associates with two zeros in the visual feature vector. An illustration of this type of outliers is shown in Figure 7.1b, where feature #4



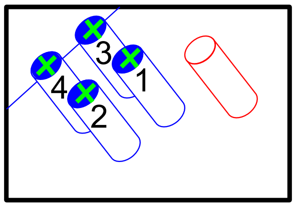
(a) Reference



(b) # 4 is lost



(c) # 2 with offset



(d) # 1 & 2 swapped

Figure 7.1: Different outlier types are considered in this work. (a) The initial image and four features without outliers. (b) *Lost* outlier. Feature no. 4 is lost as explained in section 7.2.1. (c) *Offset* outlier. Feature no. 2 is mis-tracked and as a result the measurement shows an offset as explained in section 7.2.2. (d) *Swap* outlier. Features no. 1 and 2 are swapped. This type of outlier is explained in section 7.2.3.

is lost. The overall norm of the corresponding visual-motor outlier is not arbitrarily large, because other visual features are correct.

7.2.2 Type-2 (offset) outliers

Type-2 outliers are caused by mis-tracking due to a variety of reasons. The most common is due to confusion of the tracking template with the nearby templates having a similar appearance. Because of the real-time servoing constraint, descriptive features cannot generally be used. This increases the mis-tracking problem, especially when the robot moves quickly. Please refer to Figure 7.1c for an illustration. Feature #2 is being mis-tracked with an offset. The offset changes during camera motion, but to model this type of outliers, the corrupted feature is translated by a constant 100 pixels. This type of outlier is very typical and other authors have also considered it [30]. They use a similar model.

7.2.3 Type-3 (swap) outliers

Type-3 outliers are caused by wrong feature association between frames. A major contributor to this type of outlier is the fast motion of the robot. The swap outlier is one of the most challenging types as the vector norm is usually not affected. Human perception sometimes fails to catch it as well, since both features are being tracked. For an illustration of outliers caused by this type of mis-tracking, please see Figure 7.1d, where features #1 and #2 are swapped. In simulation experiments, this type of outlier is modeled by sliding feature no. 2 onto feature no. 1.

7.3 Initialization and Error Measure

The initialization process for the WAM Arm includes the selection of visual features followed by arm motions to store the visual-motor observations into memory. This process takes only a few minutes and is very straightforward. Once a significant number of points are recorded in the memory (approximately 5,000 visual-motor samples to start), reference Jacobian matrices at 50 random points are estimated using the orthogonal motions (Section 2.4.1) for ground-truth comparison.

Jacobian estimation error is measured by the matrix L_2 norm of the difference of the estimated Jacobian to a reference Jacobian, $\mathbf{J_R}$. In simulations, $\mathbf{J_R}$ is found by arbitrarily small orthogonal motions (see Section 2.4.1). This ensures locality and allows for an accurate first-order approximation of the Jacobian hyperplane. In robot experiments, the reference Jacobian is again calculated by orthogonal motions; however, we are limited to the smallest observable visual-motor measurement. In practice, the smallest joint measurement was 0.5 degrees.

7.4 Robust Jacobian Estimation Experiments

7.4.1 Number of neighbours

The first set of experiments concerns the Jacobian estimation error with respect to the number of neighbours. The number of neighbours cannot be very small. When the sampled space contains no outliers, it is expected that estimation error linearly grow with the number of neighbours. The reason is that the least squares-based methods are local and with the addition of more samples, they are forced out of their basin of locality. In addition both the LLS and IRLS Jacobian estimates should be identical, as there are no outliers. This is indeed verified in Figure 7.2a, where both graphs are aligned.

The normalized Jacobian estimation error is observed for the rest of experiments. The outlier percentile is fixed at 30% throughout. The sample size remains constant. A normalized estimation error of 1 means there are no actual errors. A normalized estimation error of 10 means one order of magnitude larger than the ideal.

Figure 7.2b, Figure 7.2c, and Figure 7.2d, show similar results for the different outlier types. The proposed JacobianEstIRLS algorithm (labeled IRLS in the graphs) can easily handle 30% error in all cases for $K > 50$, while the estimation error with the least-squares solution (LLS) is an order of magnitude larger. Figure 7.2e summarizes the performance of the JacobianEstIRLS algorithm with respect to different outliers. As expected, the lost outlier is handled best. The performance against the offset and swap outliers are similar. Figure 7.2f shows a magnified version of the same graph.

7.4.2 Outlier percentile

Choosing a constant number of neighbours, $K = 100$, artificial outliers to the data are introduced to study how Jacobian estimation is affected. The simulation results are summarized in Figure 7.3 for different outlier types. When there are no outliers (Figure 7.3a, both IRLS and LLS algorithms show small and similar errors. With the gradual increase of outliers, the least-squares solution is linearly affected. The

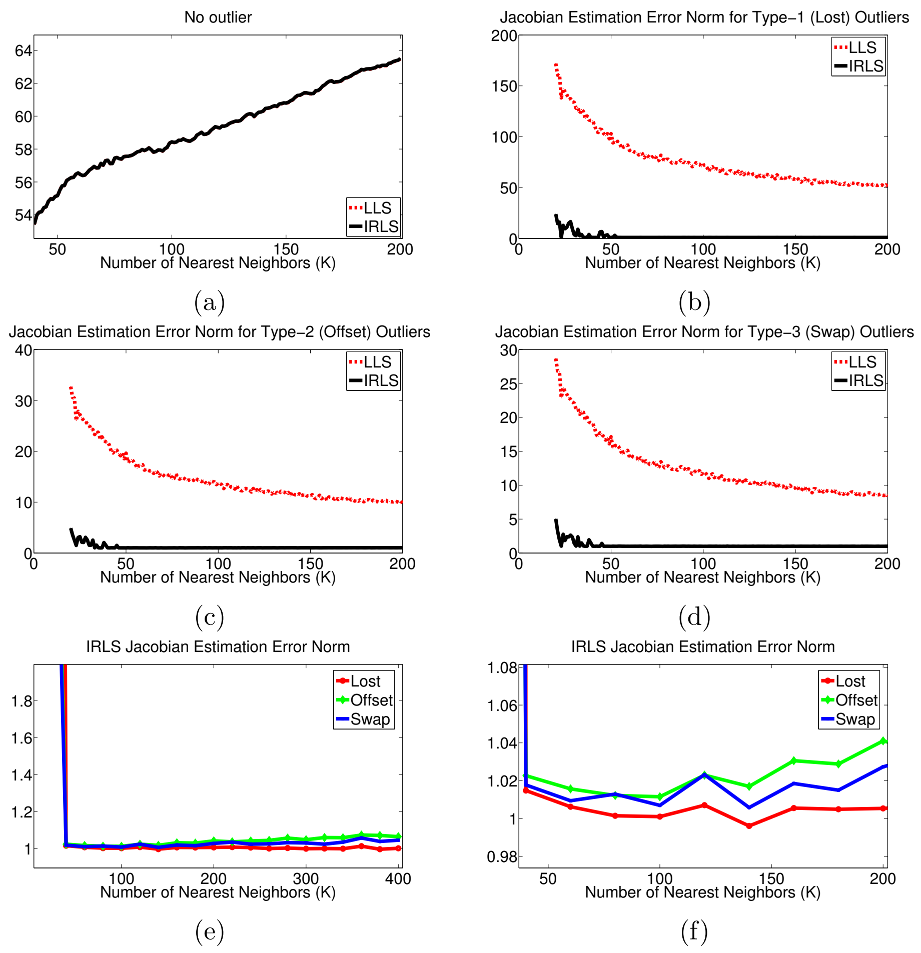


Figure 7.2: Average Jacobian estimation error norm with respect to the number of nearest neighbours over 100 random trials. (a) No outliers. The matrix norm of Jacobian estimate from IRLS and LLS algorithms are almost identical and increase linearly with more neighbours, as expected. (b) Lost outliers. (c) Offset outliers. (d) Swap outliers. For all three outlier types, the Jacobian estimates are normalized to the no-outlier case. In all experiments, outlier percentile is fixed at 30%. (b)-(d) report error norms normalized to the no outlier case. (e) and (f) show the IRLS error norm only for better comparison of different outlier types. For $K > 50$ the normalized Jacobian estimation error norm is comparable to the no-outlier case.

robust solution tolerates all three types of outliers (Figure 7.3b, Figure 7.3c, and Figure 7.3d). The horizontal axis corresponds to the outlier percentile in the neighbourhood. The simulation results are obtained using the Geman-McClure's (GM) M-estimator and averaged over 100 random trials. The robust algorithm (IRLS) tolerates outliers up to 40% in practice. In Section 4.4.2, a comparative study of the GM M-estimator and Tukey's Biweight (BW) was presented. Figure 7.4 summarizes the comparison with robot experiments for lost and offset outliers. The GM estimator outperform the BW estimator and therefore was chosen for all the other experiments in this section.

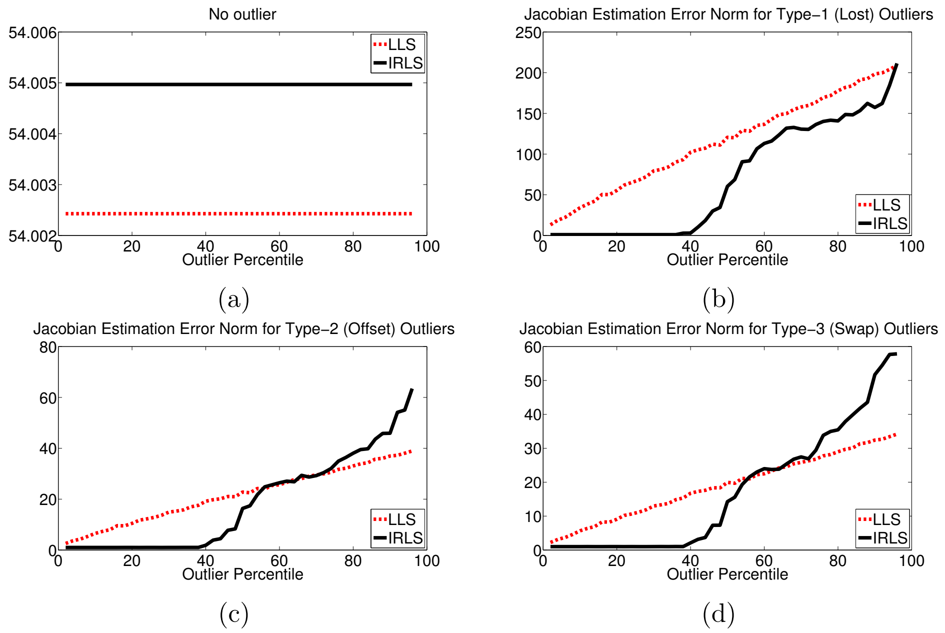


Figure 7.3: Jacobian estimation error norm with respect to outlier percentile with $K = 100$. (a) No outliers. The error norms are constant and almost identical for IRLS and LLS. (b) Lost outliers. (c) Offset outliers. (d) Swap outliers. For all three outlier types, the error norms are normalized to the no-outlier case. For the LLS algorithm, estimation error linearly increases with the outlier percentile (expected) and for the robust algorithm (IRLS) estimation breaks down at around 40% mark.

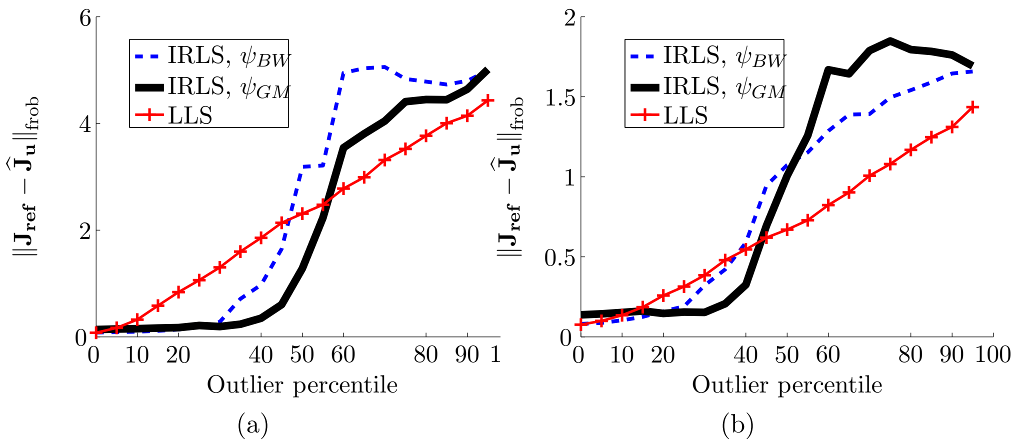


Figure 7.4: Normalized Jacobian estimation results for a WAM Arm. (a) Lost outliers. (b) Offset outliers. Two different M-estimators in this study are Geman-McClure (GM) and Tukey's Biweight (BW). The LLS (non-robust) estimates are also presented for comparison. The GM estimator outperforms others. The IRLS method handles outliers up to 40%. This is consistent with simulation results in Figure 7.3.

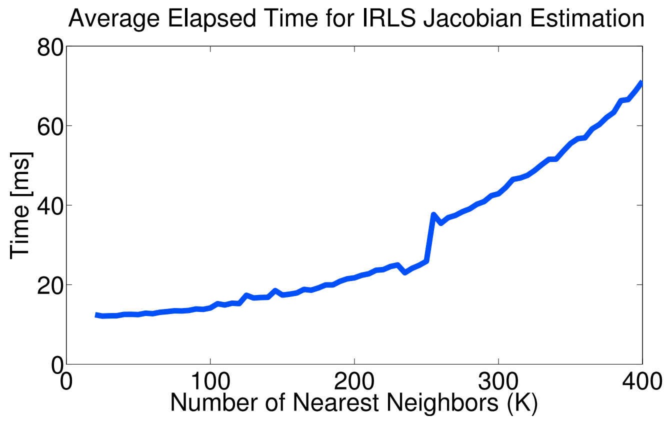


Figure 7.5: Average elapsed time for Jacobian estimation with respect to the number of nearest neighbours. The graph shows the average over 100 trials for 30% outlier percentile. Elapsed time grows exponentially, but for typical number of neighbours around $K = 100$, elapsed time is between 10ms and 25 ms, which means the numerical computations are done at the rate of 40Hz to 100Hz.

7.4.3 Computation time

In this experiment, we evaluate the computation time of the IRLS Jacobian estimation algorithm. Time is averaged over 100 random trials, with a maximum 20 iterations per estimate. The number of neighbours is fixed at $K = 100$ with 30% outliers. Figure 7.5 shows the average elapsed time per number of neighbours. An exponential-like growth can be observed in this graph. Computations for typical number of neighbours around $K = 100$ are ready at the rate of 40-100Hz. This shows that the Jacobian estimation computation is not in the way of real-time operation.

7.5 Outlier Detection and Replacement: MATLAB Experiments

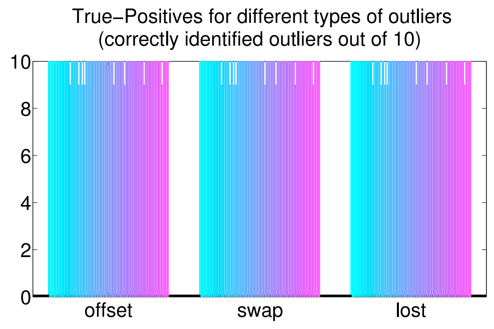
To describe the outlier detection and replacement performance, four sets of experiments with 100 random simulations in each for the eye-in-hand system are considered with 10%-40% offset outliers in the 100-nearest neighbours ($K = 100$). The offset parameter for the offset outlier is set at 100 pixels. Since we have artificially introduced the outliers, the ground-truth labeling of samples are known (they are either inlier or outlier). Bar plots for the true-positives (correctly identified outliers) and box plots[1] of the detection rate (correctly identified outliers by the total number of outliers) are depicted. We also show box plots for the replacement error (in pixels) and the normalized Jacobian estimation error using JacobianEstIRLS with the GM M-estimator are shown in Figures 7.6-7.9.

Figure 7.8 summarizes the MATLAB simulation results for outlier detection and inlier feature reconstruction experiment for 30%. Figures 7.6, 7.7, and 7.9 show the same graphs for 10%, 20%, and 40% outliers, respectively.

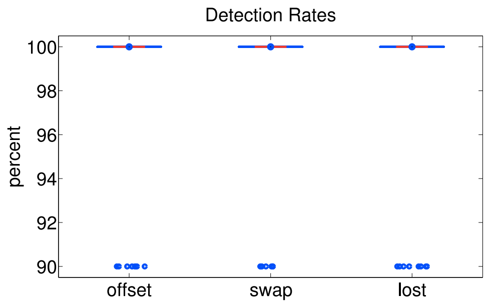
The box plots for these random experiments at 30% outliers are shown in Figure 7.8. Detection Rate for three different outliers are considered as explained in section 7.2. Figure 7.8a shows the bar plot for correctly identified outliers (true-positives). All three types are correctly identified. In Figure 7.9a, the more descriptive case of 40% outliers is shown. The lost outliers are mostly found. The offset outliers are next, and the swap outliers are the most challenging to identify.

Figure 7.8b shows the outlier detection rate for the same represented with box plots. The median detection rate is 100%, which suggests that more than half of the experiments were completely successful. Figure 7.8c shows the box plot for outlier replacement error in pixels. Figure 7.8d shows the box plot for normalized Jacobian estimation error norms. Jacobian estimation error compares well to the no-outlier case in all three cases. A poorly estimated Jacobian would have an error norm which is off by an order of magnitude. These graphs show that the algorithm works well with all three different outlier types and that the swap outlier is more challenging to handle than the other two.

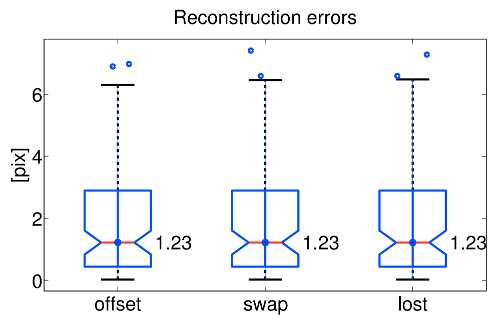
[1] A box plot is a simple and descriptive plot to show the statistical properties of a population without using or knowing its distribution. The following statistics are shown in a graph: upper and lower quartile, the median, and the minimum and maximum. The observations considered to be outliers (within the data- this is different than the visual-motor data) is depicted by dots.



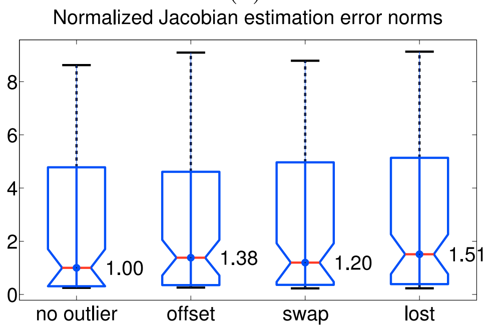
(a)



(b)

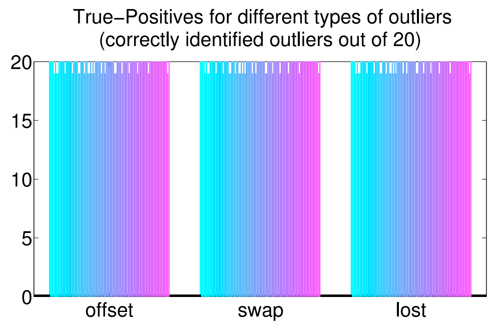


(c)

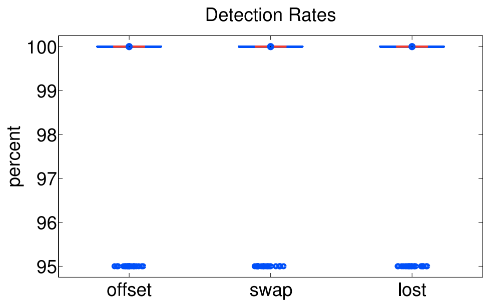


(d)

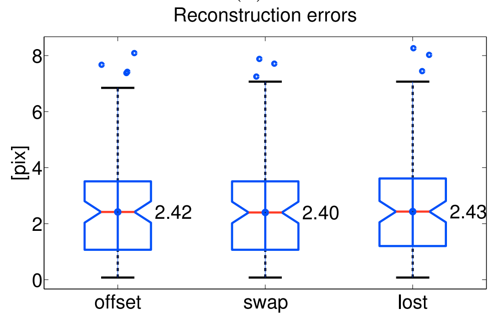
Figure 7.6: Simulation results for outlier detection and replacement with $K = 100$ and 10% outliers. (a) True-positives (correctly identified outliers), (b) Outlier detection rate box plot, (c) Outlier replacement error (in pixels) box plot, (d) Normalized Jacobian estimation error (normalized to median norm with no outliers).



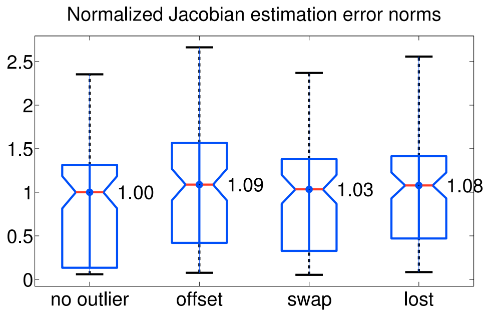
(a)



(b)



(c)



(d)

Figure 7.7: Simulation results for outlier detection and replacement with $K = 100$ and 20% outliers. (a) True-positives (correctly identified outliers), (b) Outlier detection rate box plot, (c) Outlier replacement error (in pixels) box plot, (d) Normalized Jacobian estimation error (normalized to median norm with no outliers).

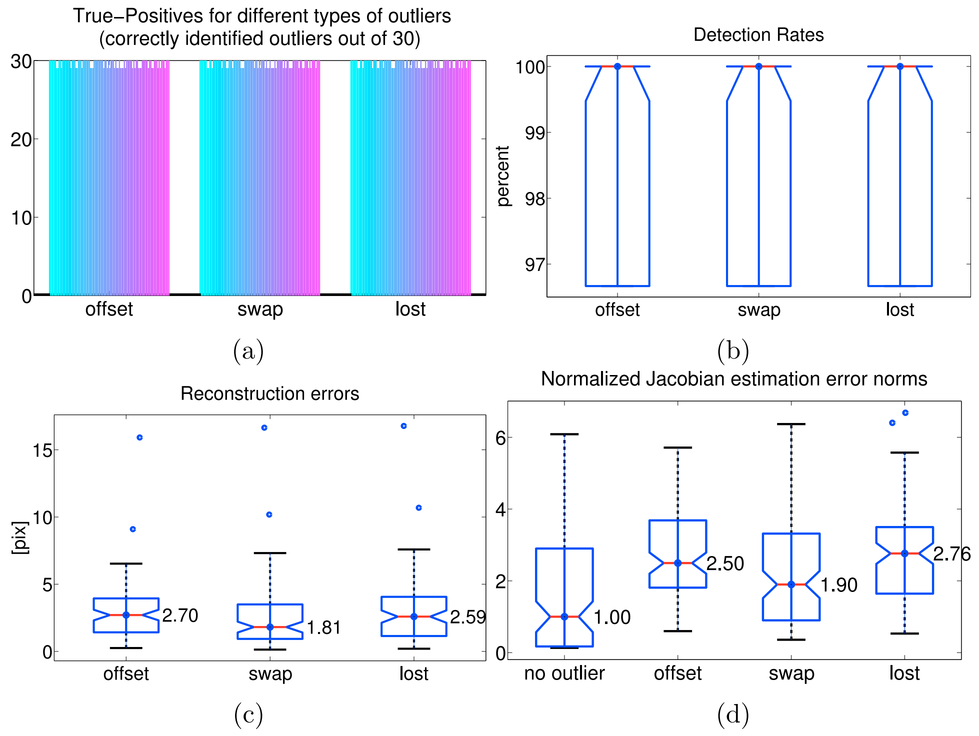


Figure 7.8: Simulation results for outlier detection and replacement with $K = 100$ and 30% outliers. (a) True-positives (correctly identified outliers), (b) Outlier detection rate box plot, (c) Outlier replacement error (in pixels) box plot, (d) Normalized Jacobian estimation error (normalized to median norm with no outliers).

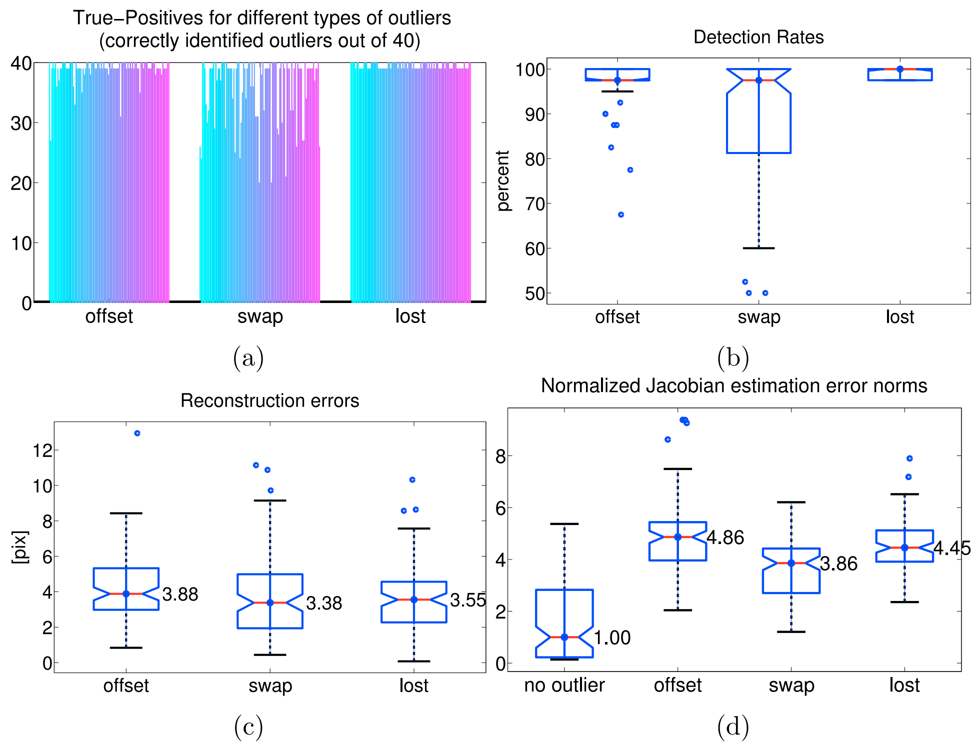


Figure 7.9: Simulation results for outlier detection and replacement with $K = 100$ and 40% outliers. (a) True-positives (correctly identified outliers), (b) Outlier detection rate box plot, (c) Outlier replacement error (in pixels) box plot, (d) Normalized Jacobian estimation error (normalized to median norm with no outliers).

7.6 Outlier Detection and Replacement: WAM Experiments

To evaluate outlier detection and recovery for experiments with the WAM Arm, 50 random configurations are considered. The database contains 5,500 previously-recorded visual-motor samples. The number of nearest neighbours was chosen as $K = 40$ and again the outlier percentile was chosen as 30%, which means approximately one out of three samples are outliers. Controlled experiments help us verify that the simulation assumptions are in line with the experimental test bed. Figure 7.10 summarizes the statistics for outlier detection and replacement for the WAM experiments. The results are indeed similar to the simulations in the previous section. The most challenging outlier type here is again the swap outlier (Figure 7.10a and Figure 7.10b). The inlier reconstruction errors are also very small and below 5 pixels. It is expected that with a denser database, the reconstruction error get smaller.

Successful query reconstruction is essential for the success of the overall closed-loop feedback system. We have demonstrated this reconstruction in controlled experiments. Figures 7.11 and 7.12 show snapshots for the outlier replacement experiments.

We first choose a fixed robot configuration and build a small visual-motor database around that configuration. Moving one of the features, the Jacobian is

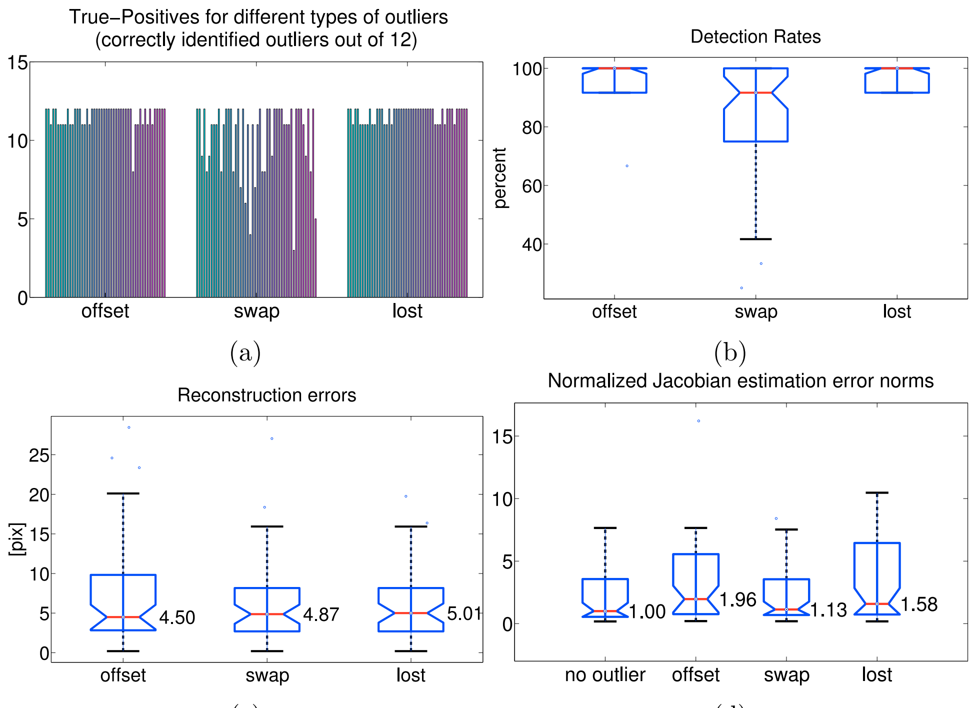
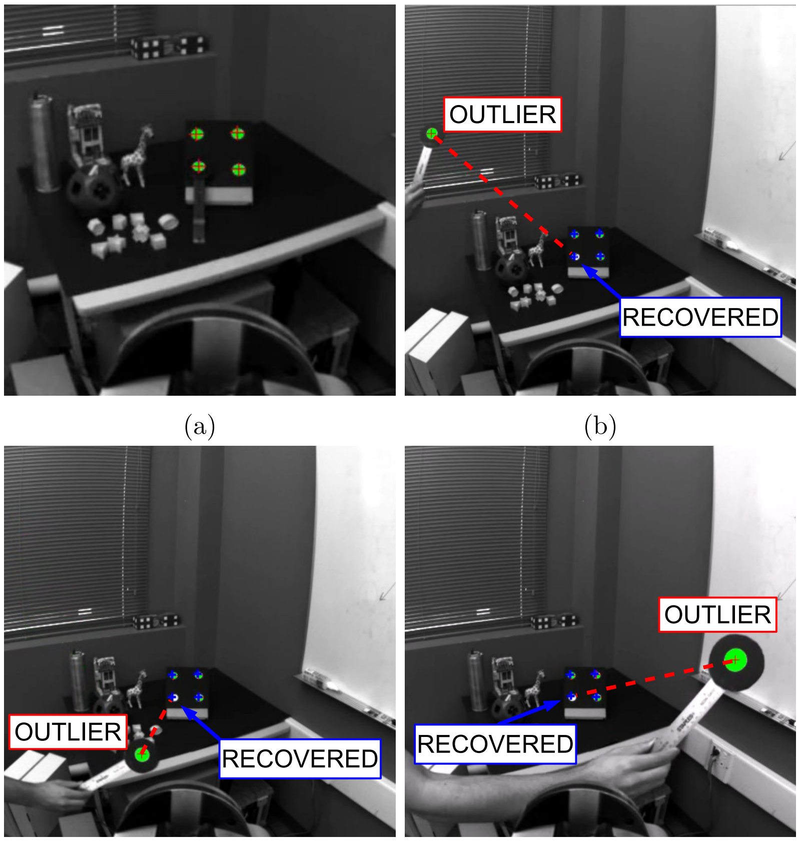


Figure 7.10: WAM experiment. (a) True-positives (correctly identified outliers), (b) Outlier detection rate box plot, (c) Outlier replacement error (in pixels) box plot, (d) Normalized Jacobian estimation error (normalized to median norm with no outliers). In these experiments $K = 40$ with 30% outliers.

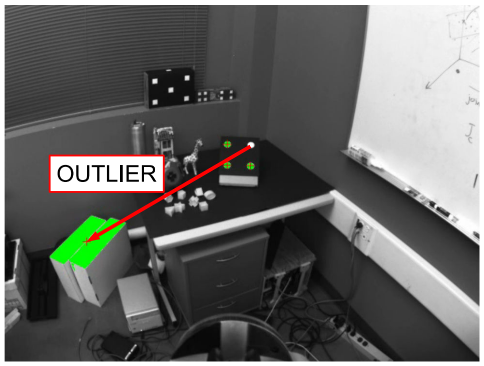


(a) (b) (c) (d)

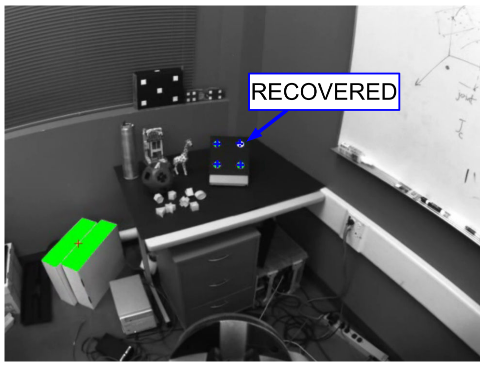
Figure 7.11: Outlier replacement for a static object. Images captured from an eye-in-hand robot arm are shown. The target object consists of four coplanar points. One of the points can be moved around. (a) The visual-motor space around a static point is sampled. (b)-(d) The offset outlier is successfully reconstructed.

constantly estimated and the reconstructed visual features are projected back on to the image plane. Figure 7.11 shows snapshots of the inlier feature reconstruction in this experiment.

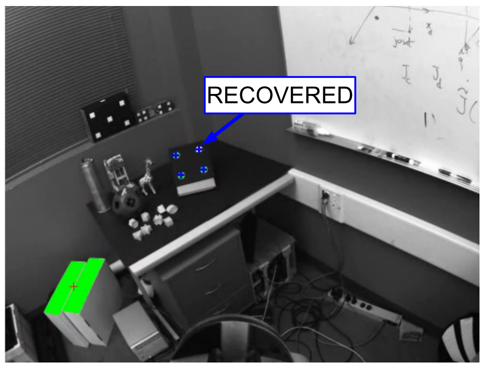
For the second experiment, we move the robot along an arbitrary trajectory and build a visual-motor database around the trajectory. This is done by placing the robot controller in a gravity compensation mode and manually moving the arm to collect data. Once the database is created, we introduce an outlier image feature and replay the trajectory, while estimating the Jacobian and projecting the reconstructed features on the image place. Figure 7.12 shows snapshots of the inlier feature reconstruction in this experiment. The outlier is recovered successfully when there are adequate number of samples present. An observation was that when the robot was at unexplored sections of the visual-motor space, the outlier was recovered to the closest sample with still had an error. Fortunately, one can compare the joint vector against the joint space in the database to determine if the query point is part of the explored space.



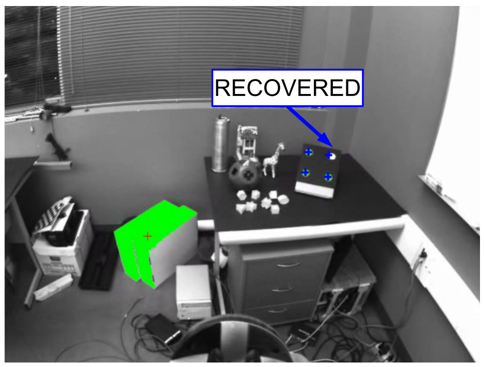
(a)



(b)



(c)



(d)

Figure 7.12: Outlier replacement during robot/camera motion. Images are taken from an eye-in-hand camera rigidly attached to the manipulator. The robot moves along a trajectory and the outlier is recovered from the previously-explored visual-motor space. (a) The target object and the offset outlier are shown. (b)-(d) Snapshots of the experiment.

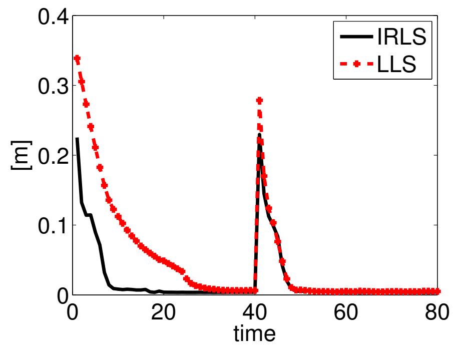
7.7 Eye-in-hand Visual Servoing Experiments

7.7.1 Base experiment: Visual servoing without outliers

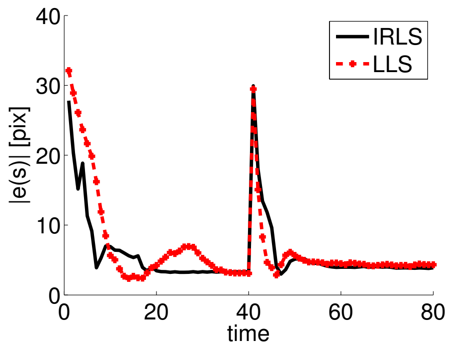
We have used the estimated visual-motor Jacobian using both LLS and IRLS (with GM estimator) in uncalibrated IBVS control law expressed in (2.11). Figure 7.13 (a) depicts the end-effector positioning error and Figure 7.13 (b) shows the norm of image-space error. The goal of this experiment is to show that LLS and IRLS perform similarly with an adequate visual-motor database. constraint. This is due to the highly non-linear visual-motor function. That is, some of the K neighbours will have a relatively large distance to the query point. The robust method downweights such points and reduces their influence on the estimation result. To show this point, we started with a relatively small memory, and let the visual servo drive the arm from an initial position to a desired position. During servoing, new data are incrementally added to the database. The LLS method converges to the desired position but at a slower rate than IRLS. At time $t = 40$, the arm is moved back close to the initial point. With more relevant data available in the database at $t = 40$, both LLS and IRLS converge at the same rate and with similar accuracy.

7.7.2 Experiment: Robust visual servoing with outliers

Finally, we study the effect of outliers on visual servoing performance. We use the same initial starting and desired points as the last section, but add 30% outliers (Type-1) to the data. Figure 7.14 shows a sample visual servoing performance with outliers introduced at time $t = 60$. The IRLS algorithm manages to estimate a meaningful Jacobian to drive the arm towards the goal. However, LLS gives a wrong estimate, which drives the robot in a wrong direction, where the robot gets stuck in a local minimum. Both LLS and IRLS perform similarly without outliers ($t < 60$). This is in agreement with Figure 7.4, where a similar type-1 outlier is used. For the purpose of this experiment, GM estimator is used because of its overall robustness to a larger range of outliers. These results show that LLS is not robust to outliers, however, IRLS tolerated the outliers and could still reach the desired goal.

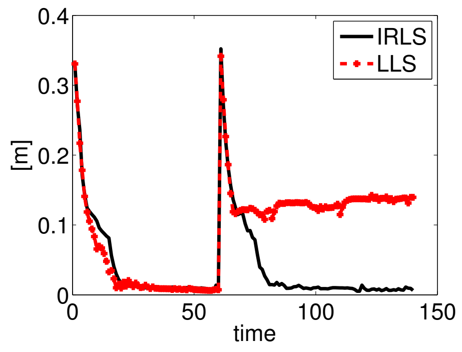


(a)

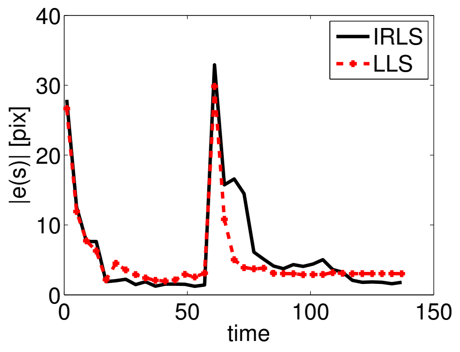


(b)

Figure 7.13: Error measures for LLS and IRLS *without* outliers. (a) End-effector positioning error. (b) Visual space error. IRLS control converges faster with the initial memory $t \leq 40$. For $t > 40$ the memory is richer and performance is similar.



(a)



(b)

Figure 7.14: Error measures for LLS and IRLS with outliers introduced at $t > 60$. (a) End-effector positioning error. (b) Visual space error. The IRLS Jacobian estimates are robust to outliers and drive the arm to the desired point. The LLS estimates are erroneous and drive the robot in the wrong direction. The LLS experiment ends up in a local minimum.

7.8 Sampling-Based Planning Experiments

We consider the uncalibrated eye-in-hand visual servoing set up with a 4-DOF Barrett Whole Arm Manipulator (WAM). The camera is mounted on the elbow of the manipulator. Both simulations and the robotic experiments are carried out in an uncalibrated fashion, that is, the camera/robot calibration or extrinsic/intrinsic camera calibration parameters have not been used by neither the planning algorithm nor the visual servo. The recommendations in Good Experimental Methodology in robotics prepared by EURON [124] is followed throughout this chapter.

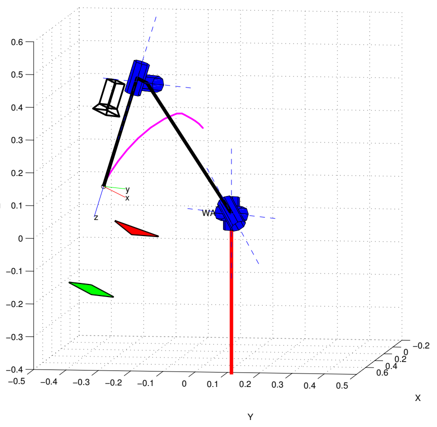
7.8.1 Planning with UVS-BiRRT algorithm: MATLAB

To experimentally evaluate the proposed algorithm, we run MATLAB simulations and empirically validate the performance. Simulations are developed using the Robotics Toolbox [120] and the Epipolar Geometry Toolbox [121].

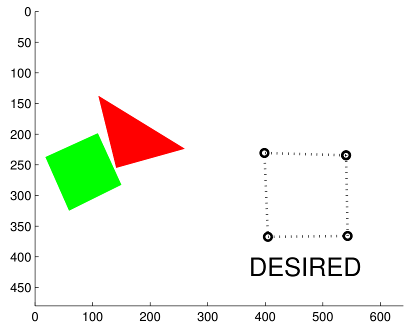
The typical visual servoing simulation setup is shown in Figure 7.15. The task is to align the rectangle target (green) on the dotted area outlined by DESIRED. We consider cases where visual servoing without planning fails due to visual occlusion and planning is required. The planned image trajectory is shown by dotted lines in the image plane.

In Figure 7.16, a visual servoing trajectory is shown without planning to avoid visual occlusions. The obstacle occludes the target during control, which is not desirable. Next, we will show how planning assists in avoiding the visual and physical constraints.

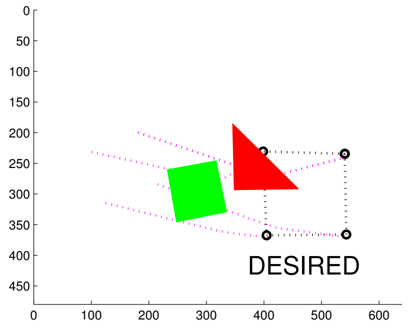
We have run experiments with both 3-DOF and 4-DOF WAM models. For the 3 DOF simulations, joint number 3 of the WAM is locked. This provides a non-redundant positioning arm. In Figure 7.17, the free and occupied space, and the planned path are depicted. For the 4 DOF setup, the visual-motor space has $M + N = 12$ dimensions. The free space is shown by green dots. The convex hulls correspond to the conservative estimate of the VOV. The proposed planner produces a path that goes around the visual occlusion, while avoiding other constraints and handling visual-motor outliers. Figure 7.16 shows an example of the failure of visual servoing without planning. The visual servoing trajectory generates visual occlusions, where visual tracking of points fail. For the same start and goal states, the proposed planning algorithm is successful as shown in Figure 7.18. Note that the goal image is very close to the FOV limits, but the planned image trajectory stays inside a safe margin with $\zeta_{fov} = 10$ pixels. In this experiment the VOV margin is also $\zeta_{vov} = 10$.



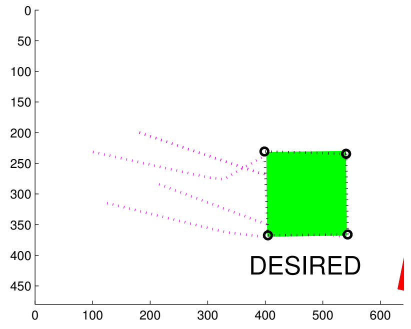
(a)



(b)

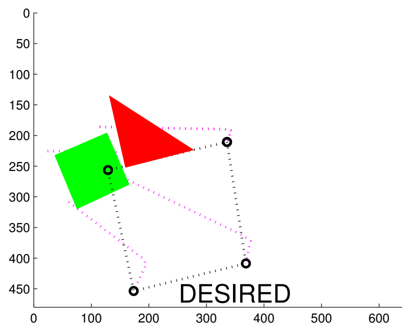


(c)

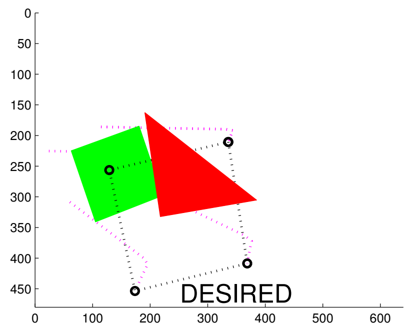


(d)

Figure 7.15: Simulation setup for planning experiments. (a) The desired robot configuration with the Cartesian trajectory. The target is a rectangle below an obstacle in the form of a triangle. (b) Initial and desired target image and initial obstacle image. (c) A snapshot of the target and obstacle images, while following the planned path in dotted lines. (d) The target has reached the desired (goal) state.



(a)



(b)

Figure 7.16: Visual occlusion violation occurs during visual servoing without planning. (a) The initial image of the rectangle target (green) and the triangle obstacle (red), the image-based control trajectory without planning (dotted line) and the goal or DESIRED target image (dotted line) are shown. (b) Without considering the VOV and planning to avoid it, the rectangle target (green) is visually occluded by the triangle obstacle (red) during visual servoing. As visual occlusion results in failure of visual tracking, the overall visual servoing system fails.

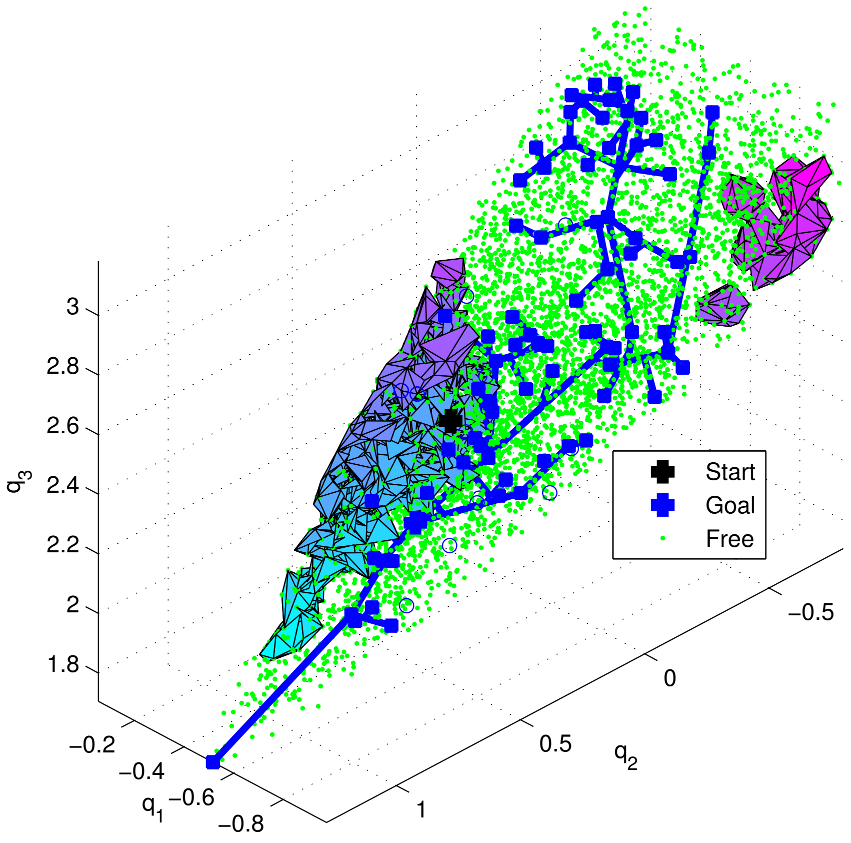


Figure 7.17: Visualization of the free space and the VOV mapped to the configuration space. The visual occlusion constraints are illustrated with convex hulls. The final RRT is shown in blue and the free space with green dots.

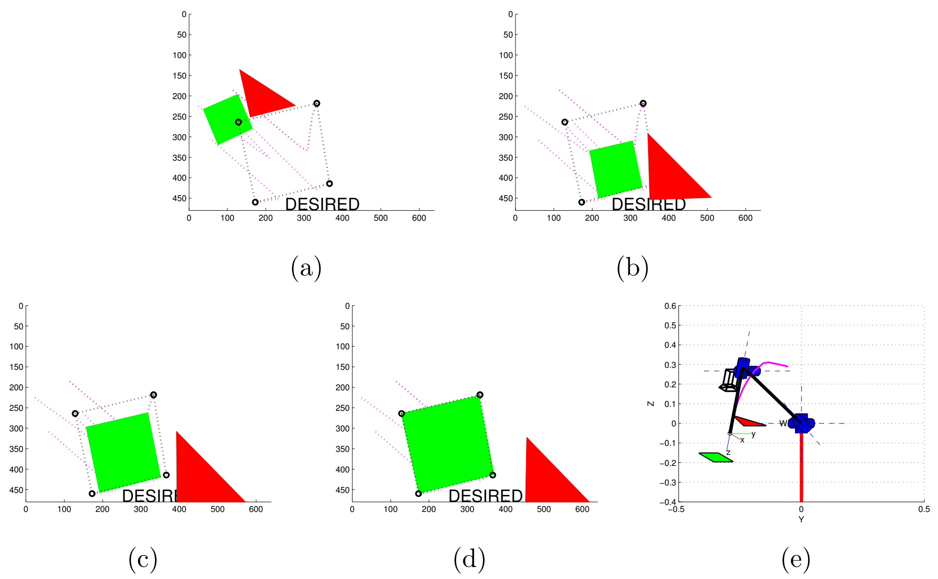


Figure 7.18: Planning with UvsBiRrt results in occlusion-free paths. From left to right, the progression of the initial image to the goal image is shown. The planned path is shown in the image plane. The most left image shows the final configuration and the Cartesian end-effector trajectory.

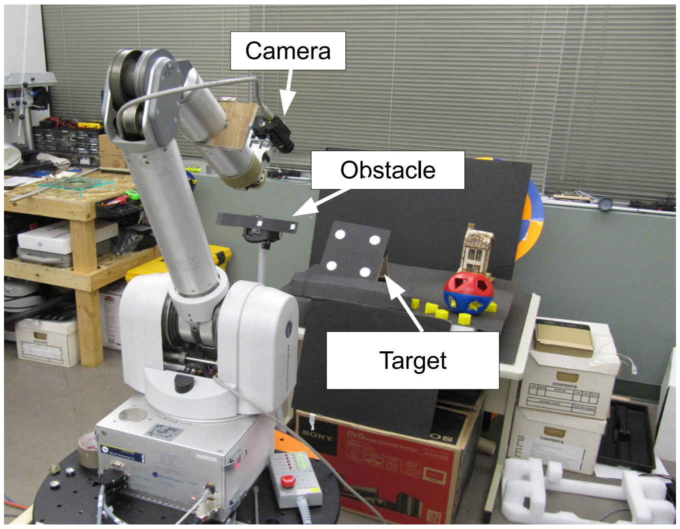


Figure 7.19: The experimental setup. An eye-in-hand WAM arm with 4 DOF is considered. The target is a planar and contains four feature points that can be easily tracked. The obstacle is a line that goes through two feature points on the obstacle target.

7.8.2 Planning with UVS-BiRRT algorithm: WAM

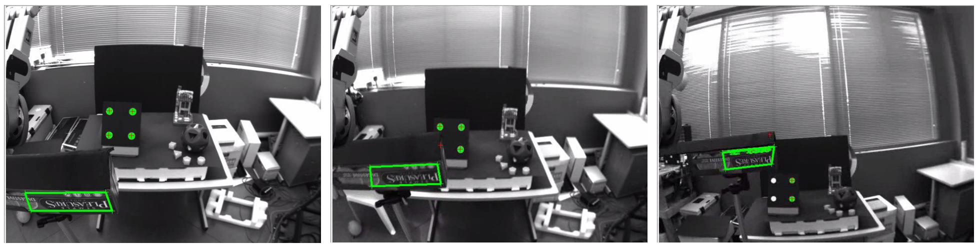
Some experiments with the WAM has been designed and empirically validated. The experimental setup is shown in Figure 7.19 and chosen very similar to the simulation setup in the previous sections. A line obstacle is modeled using two points. The target is a four-point coplanar object as shown in the figure. Models of objects have not been used in our experiments.

We use the Visual Servoing Platform (ViSP) [101] for visual tracking of white dots and integrate it to our custom software that connects to the WAM controller via a socket interface. A MATLAB scripting interface is also developed.

The start state is chosen such that the images of the target and obstacle objects are close, but not occluding. The goal state is chosen such that a direct trajectory results in occlusions as shown in Figure 7.20. When occlusion happens, the affected visual features are lost. Visual servoing under these conditions will fail. The solution is either to use a redundant number of features and replace the affected ones or plan around obstacle occlusions and avoid this situation. The result of the path planning algorithm on a similar setup is presented in Figure 7.21 and Figure 7.22.

In Figure 7.21, images captured by the eye-in-hand WAM from the start to goal are shown. Several midway points are planned and the visual servoing moves the robot to the next state to avoid visual occlusions. Joint limit and field-of-view are also avoided in this experiment. Figure 7.22 corresponds to the positioning and image errors for the same experiment. A video of this experiment is available from `http://webdocs.cs.ualberta.ca/~azad/phd.html`.

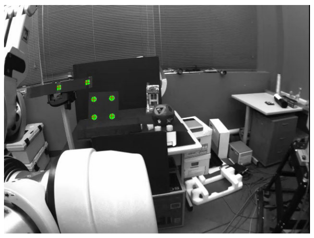
The virtual visual servoing has been validated after a brief data-gathering session. The database contains 2000-3000 samples. The estimated desired state had an error of smaller than 5 degrees in each joint. Exception to this result is when the arm at a singular configuration (for example, when joints 1 and 3 line up and cancel



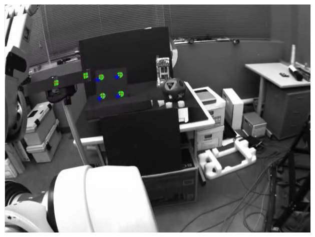
(a) Start (b) Occlusion (c) Lost features

Figure 7.20: Visual occlusion results in the loss of features. The target object is modeled by a planar object with 4 coplanar points. The obstacle object is a box modeled by one of its sides. (a) Start image. (b) Occlusion happens. one of the features are lost. (c) Two feature points are lost.

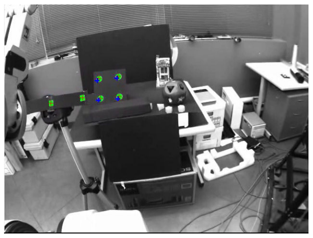
each other), but we do not consider singularity avoidance in this chapter. The small positioning error (Figure 7.22) is a result of inaccuracies in the Jacobian estimation and goal state estimation, but as one can see this error is not large.



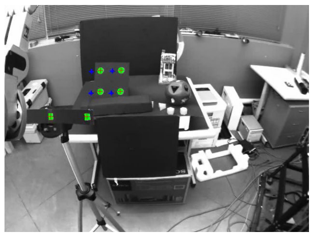
(a)



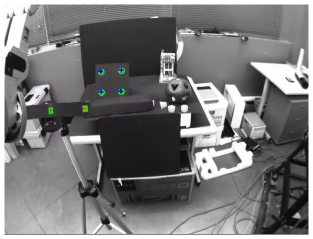
(b)



(c)

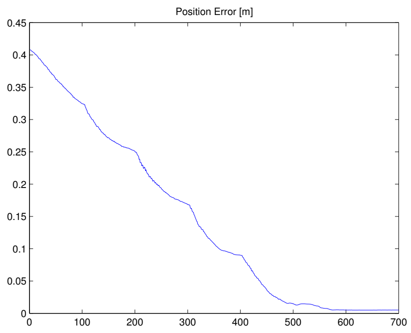


(d)

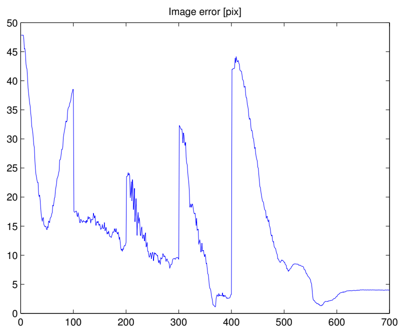


(e)

Figure 7.21: WAM experiments (for a detailed performance please see the attached video). Visual occlusion does not happen in this experiment as a result of planning.



(a) Position error [m]



(b) Image error [pix]

Figure 7.22: Error graphs for UvsBiRrt planning for the WAM setup. (a) The position error of the end-effector. (b) The image error norm for the four-point planar target.

Chapter 8

Evaluation: Three-View Uncalibrated Visual Servoing

In this chapter, we experimentally evaluate the performance of the new features introduced in Chapter 6 with the proposed control law in (6.46) in Section 6.5. Note that this control law uses a new Jacobian and is specified in the space of the trifocal features. To evaluate this new control law, we follow the recommendations in Good Experimental Methodology in robotics prepared by EURON [124]. Specifically, we limit this work to evaluations through controlled simulations. The results for small (local) motions include pure translation, translation along z-axis, and an arbitrary motion including rotations. We also present experimental results for the degenerate case of rotational motion around the z-axis, which is one of the "hard" configurations in visual servoing [11, 3, 61].

We consider the uncalibrated eye-in-hand visual servoing set up with a 6-DOF PUMA 560. The camera is mounted on the end-effector of the manipulator. We emphasize that our proposed method is general to arbitrary camera/robot configurations and the camera does not have to be on the end-effector. No assumptions on camera calibration or camera/robot calibration have been made. Simulations are implemented in MATLAB using the Robotics Toolbox [120] and the Epipolar Geometry Toolbox [121]. We use 6 points in a general configuration to evaluate the performance. Generalization to more point correspondences is straightforward by adopting the RANSAC robust estimation of the trifocal tensor [110]. For a valid trifocal tensor estimation, these 6 points should not be collinear in any of the views. This is somewhat limiting in translational motion experiments. We have chosen the initial and desired states to avoid collinearity.

8.1 Experiment I: Translation along x-axis, y-axis, and z-axis

The first experiment considers a small translation of $[5.4, 5.4, -5.4]cm$ along the three axes. The control parameter is chosen $\beta = 0.5$ for this experiment. Fig. 8.1 (top-left) shows the position of the initial and desired camera. The robot is not shown in this figure. Other values of β result in a similar trajectory in this case, because the initial and desired states are very close. Fig. 8.1 (top-right) shows the evolution of the projections in the image. Note that at the desired image some of

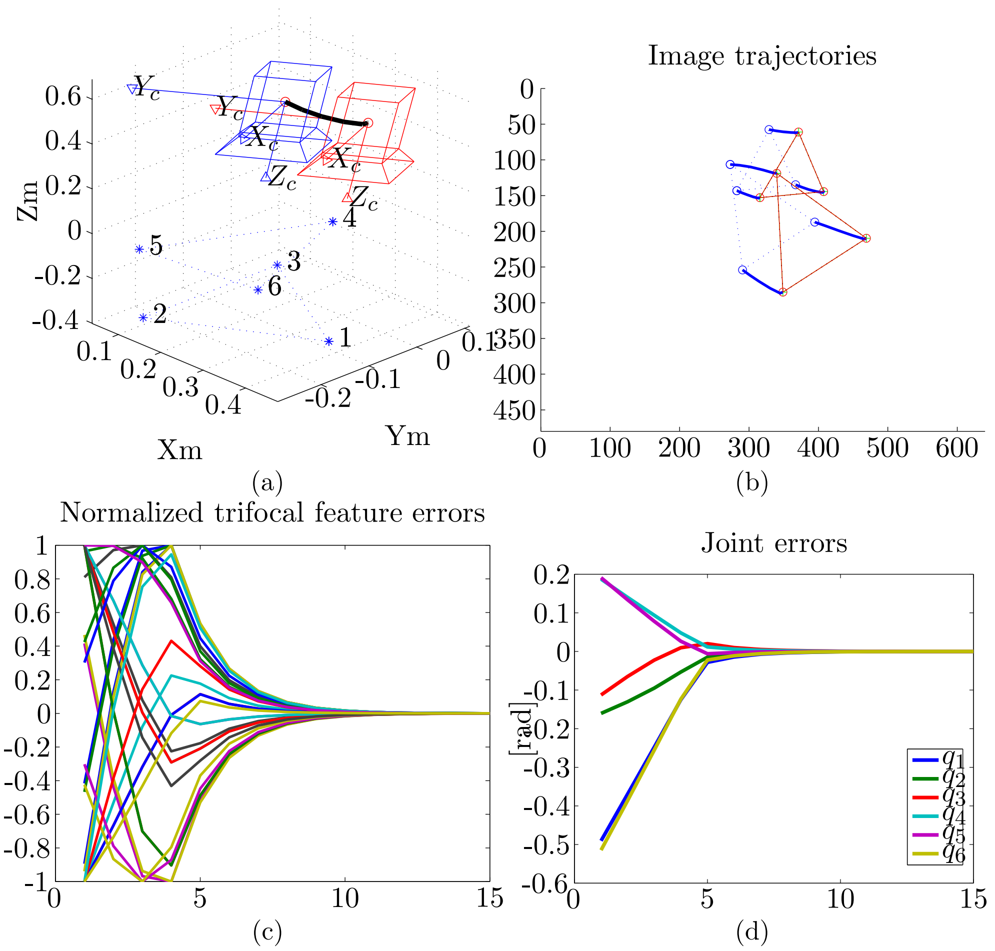


Figure 8.1: Translation Along x-axis, y-axis, and z-axis. (a) Initial/Desired cameras and the camera trajectory. (b) Initial/Desired image coordinates and image trajectory. (c) The normalized trifocal feature errors. (d) Joint values in [rad].

the features are collinear. Fig. 8.1 (bottom-left) shows the evolution of the trifocal features during servoing. Instead of showing all of the features, we normalize the features and show their mean-square-error (MSE). Apart from a discrepancy at the start, the features rapidly converge to zero. The source of this discrepancy is most likely due to the conditioning of the trifocal tensor estimation. Fig. 8.1 (bottom-right) shows the evolution of joint values during servoing. Joints 1, 4, 5, and 6 have large motions and cancel out each other to result in a linear end-effector motion.

8.2 Experiment II: Translation along z-axis

Motion along the z-axis is usually more challenging than xyz motion in the image-based approach. This is due to poor motion resolvability when the camera moves towards the feature points [56]. However, using the trifocal features motion along the z-axis is not different than other motions, as long as the conditions to estimate a valid trifocal tensor are met.

Fig. 8.2 (top-left) shows the camera trajectory for a 17.9cm translation along the z-axis. The initial and desired cameras are parallel to the plane of three of the

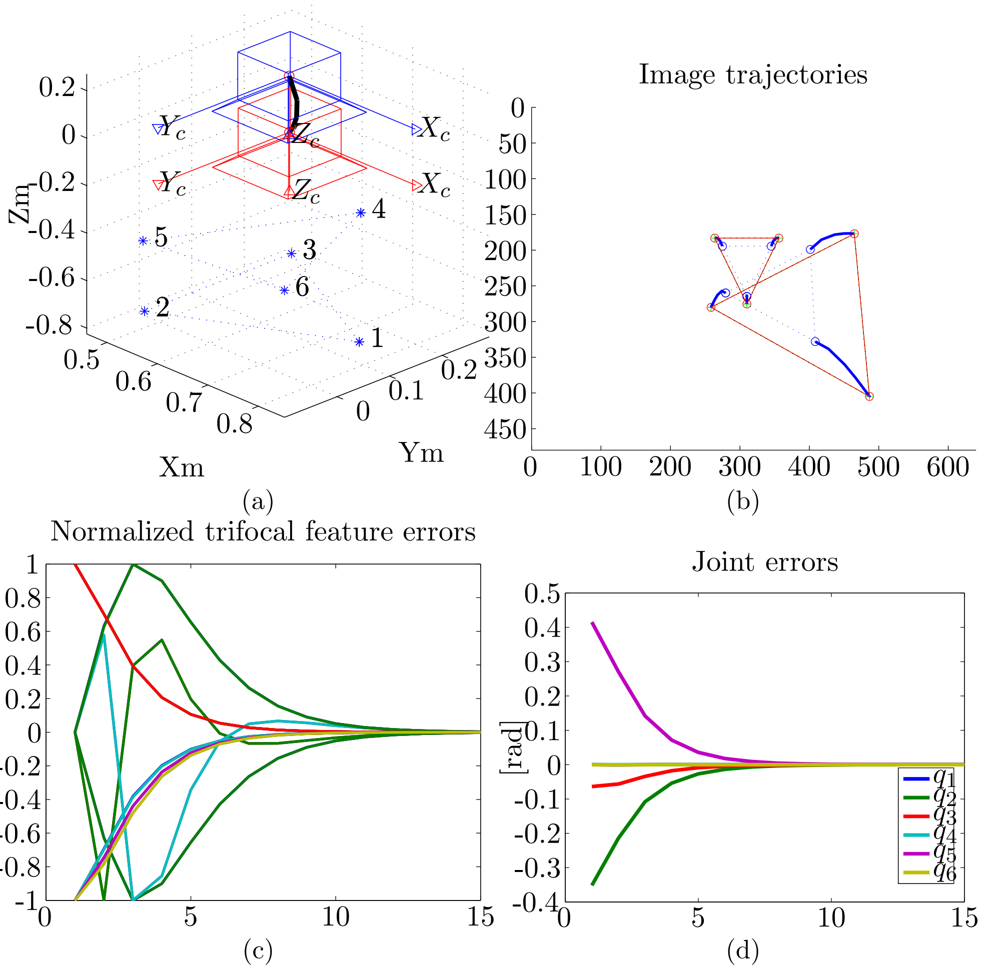


Figure 8.2: Translation Along z-axis (Depth). (a) Initial/Desired cameras and the camera trajectory. (b) Initial/Desired image coordinates and image trajectory. (c) The normalized trifocal feature errors. (d) Joint values in [rad].

six landmarks. The camera trajectory is almost linear. Control parameter $\beta = 0.5$ in this experiment. Fig. 8.2 (top-right) shows the image trajectories. Note that three landmarks have very close initial and desired values, but the trifocal features capture the geometry adequately. Fig. 8.2 (bottom-left) shows a rapid convergence of the trifocal features, and Fig. 8.2 (bottom-right) shows the convergence of the joint values.

8.3 Experiment III: Arbitrary general motion

The results for an arbitrary small general motion are presented in Fig. 8.3. The desired camera frame is rotated by $[-35.2°, -14.5°, -12.8°]$ and the translated by $[8.6, 8.6, -8.6]cm$. In this experiment, the control parameter is chosen to be $\beta = 0.5$. Other values of β also work. With a larger β the robot moves a longer trajectory and the image trajectory is more circular. Fig. 8.4 shows the image trajectory for

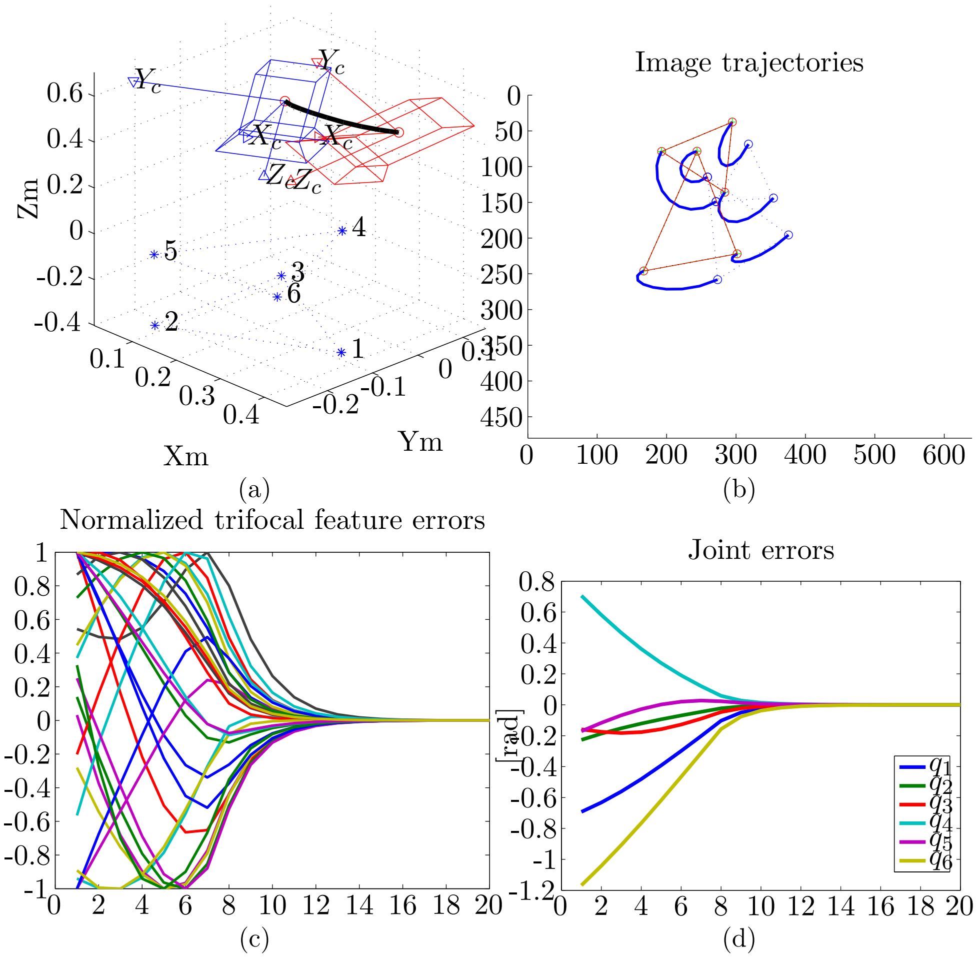


Figure 8.3: Small general motion with both translation and rotation. (a) Initial/Desired cameras and the camera trajectory. (b) Initial/Desired image coordinates and image trajectory. (c) The normalized trifocal feature error. (d) Joint values in [rad].

$\beta = 0.1$ and $\beta = 0.9$.

8.4 Experiment IV: $85°$-rotation around & translation along z-axis

This experiment includes a $85°$-rotation around the z-axis and $25cm$ motion along the z-axis. During this experiment, we noticed that some of the elements of the trifocal tensor are constantly 0. This is because of the special type of camera motion in this experiment. Specifically,

$$\mathcal{T}_1^{23} = \mathcal{T}_1^{33} = \mathcal{T}_2^{13} = \mathcal{T}_2^{33} = 0.$$

In this case, we use the remaining 23 elements of the tensor in the trifocal feature vector $\boldsymbol{\tau}$.

Fig. 8.5 summarizes this experiment. It can be seen that camera translation

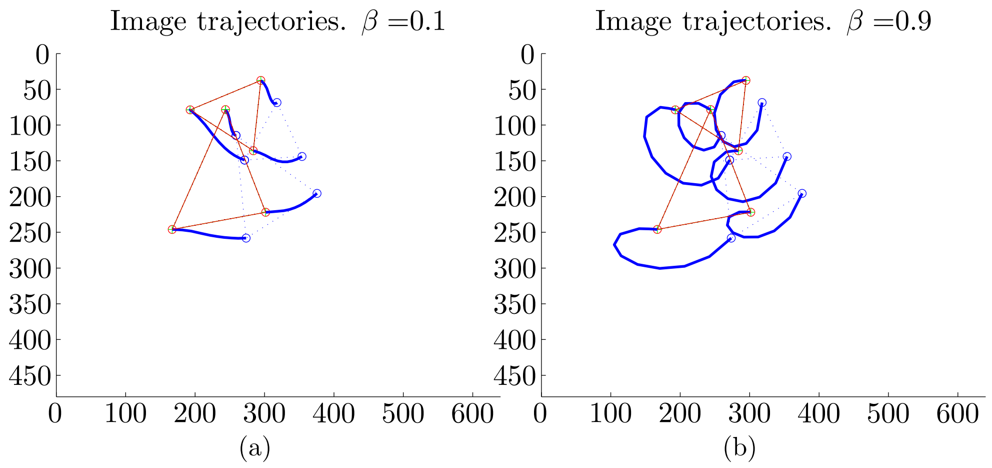


Figure 8.4: Comparison of the control parameter β in (6.45) for the experiment of Section 8.3. For $\beta = 1$ the control law uses the constant desired Jacobian $\widehat{\mathbf{J}}_{\mathbf{d}}$ and for $\beta = 0$ the control law chooses the current estimate at each iteration. (Left) $\beta = 0.1$, and (Right) $\beta = 0.9$.

is not entirely linear in the Euclidean space, however, the image trajectories are rotational, which is more desired than a linear image trajectory. Since we have made no attempt to decouple the translation from rotation, the non-linear camera trajectory is expected. Nonetheless, the rapid convergence suggests that the trifocal tensor is a suitable feature for uncalibrated visual servoing.

8.5 Experiment V: Large rotation around & translation along z-axis

One of the most challenging image-based visual servoing configurations is the 180° rotation around the z-axis [3, 11]. This is due to the nature of the image-based control law which makes the camera to retreat from the object instead of rotation around the view axis. It is important to evaluate a visual servo for large z-axis rotations, close to 180°, for example a 170° rotation [11, 3, 61].

We consider a translation of $50cm$ and a 170°-rotation to provide a common ground to compare this method against other approaches [61]. We use the same trifocal feature vector of the previous experiment. The proposed trifocal features successfully handle this case as illustrated in Fig. 8.6. Note that the desired and initial camera frames are rotated at 170° in Fig. 8.6 (top-left). The image trajectories show a spiral motion in Fig. 8.6 (top-right), which is the desired case. The results in Fig. 8.6 are obtained with control parameter $\beta = 0.05$. For $\beta = 0$, which corresponds to using the current Jacobian estimate in (6.46), control also converges, but with a slight abrupt motion at the start of the control loop. Choosing $\beta = 1$, which corresponds to using the constant value of the desired Jacobian in (6.46), was not successful. This result is expected as such a large motion is not local and the values of the Jacobian matrices, at the initial and desired states, are significantly different.

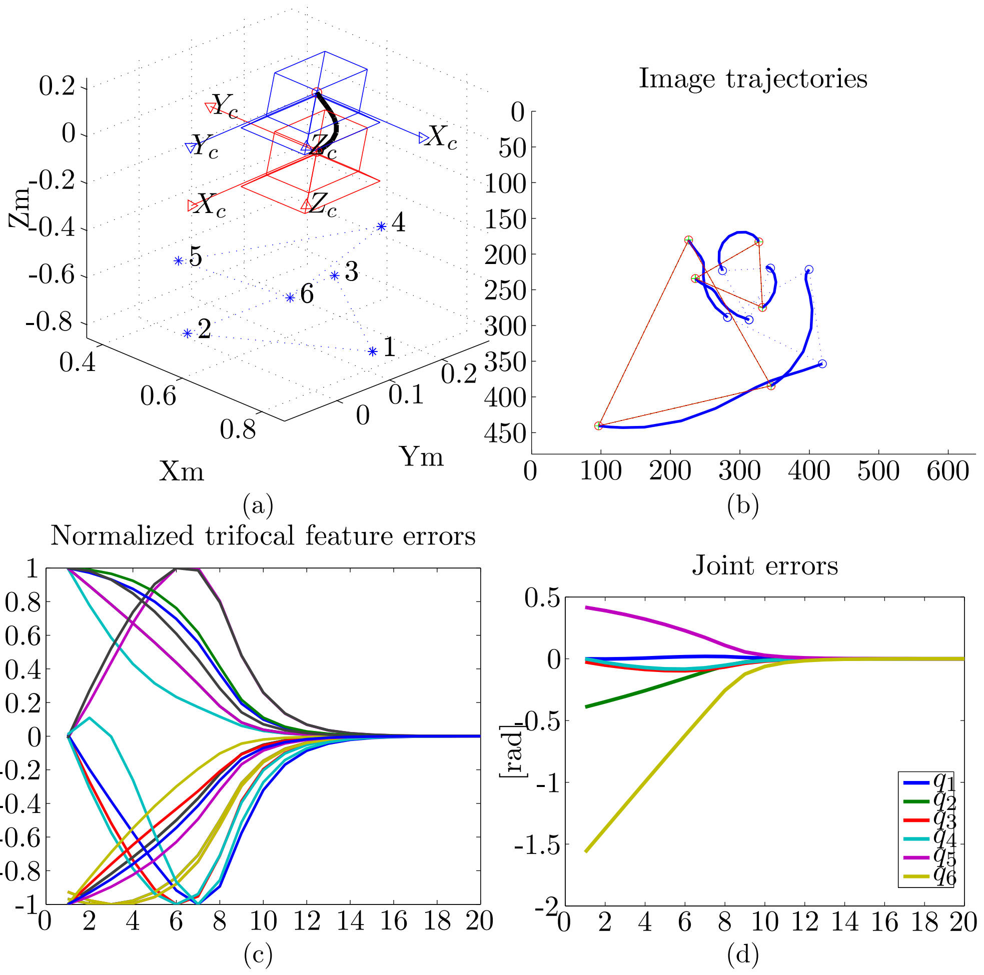


Figure 8.5: 85°-rotation around the z-axis and $25cm$ motion along the z-axis. (a) Initial/Desired cameras and the camera trajectory. (b) Initial/Desired image coordinates and image trajectory. (c) The normalized trifocal features errors. (d) Joint values in [rad].

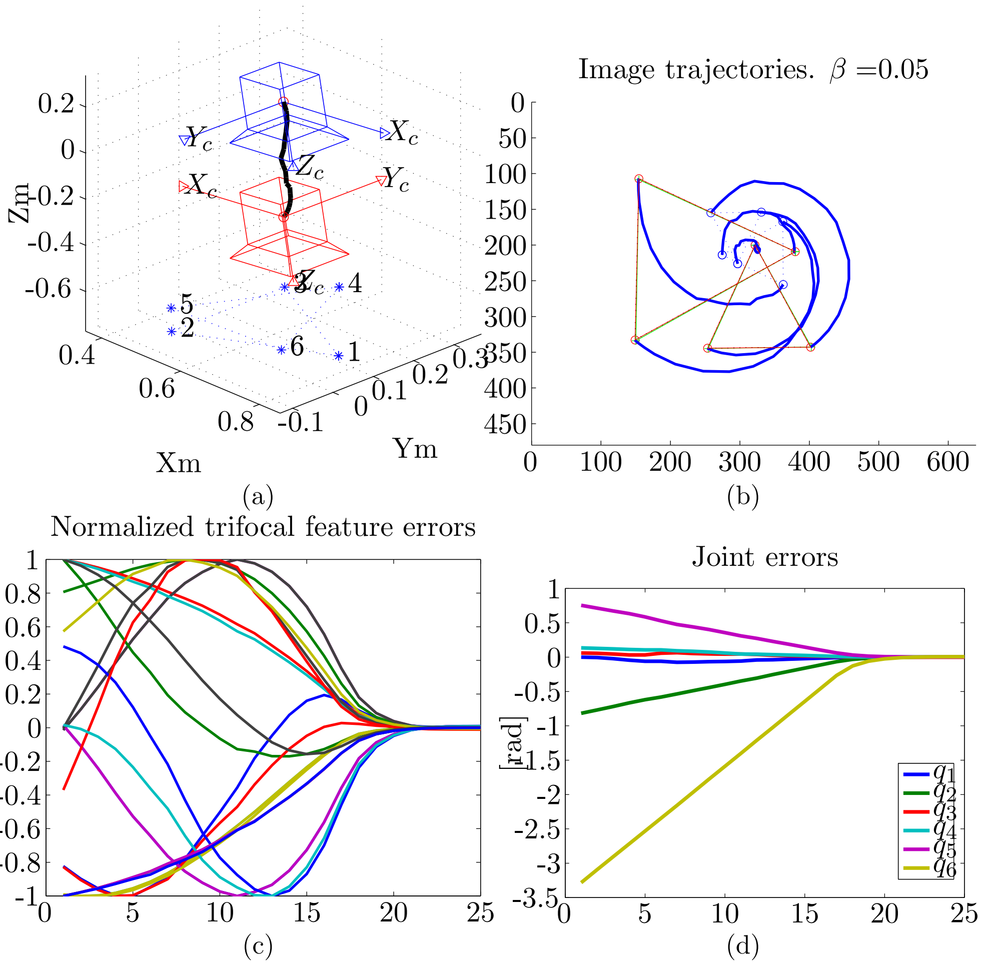


Figure 8.6: 170°-rotation around and 50cm translation along the z-axis. (a) Initial/Desired cameras and the camera trajectory. (b) Initial/Desired image coordinates and image trajectory. (c) The mean-square-error of the normalized trifocal features. (d) Joint values in [rad].

Chapter 9

Conclusions and Future Directions

9.1 Conclusions

Vision-based robotics is an active area of research with significant progress being made in developing both visual tracking and visual servoing algorithms during the past three decades. The interest in using cameras for sensing stems from the observation that images provide a natural way of perceiving the environment. Most current research efforts in visual servoing tend to focus on the control-theoretic development of controllers taking some type of explicit models of the robot, cameras, or target objects into account, while ignoring practical issues that limit the scope of visual servoing in unstructured environments. Many robotic systems, however, are required to operate in unstructured environments. There is a need to develop new algorithms for uncalibrated and model-free visual servoing.

Since robots need to act on the fly, computer vision algorithms should run in real-time. There are challenges in real-time processing of visual information. Most notably, visual measurement is often corrupted by noise and outliers. Uncertainties in the motor commands could exist as well. This imposes significant challenges on algorithm design. To perform practical tasks in unstructured settings, vision-based robots require algorithms that (i) do not depend on explicit models, (ii) are robust against sensing uncertainties and outliers, and (iii) benefit from further explorations of the scene. Consequently, this thesis has concentrated on the development of a robust framework for UVS and planning to avoid constraints without requiring explicit models in unstructured settings. This thesis is a step towards answering the question on how to robustly control the motion of a vision-based robot arm and plan occlusion-free paths in unstructured environments.

In the first two contribution chapters (Chapter 4 and Chapter 5), we have emphasized the importance of statistical robustness against outliers for vision-based robots that are constantly sensing their environment through a camera sensor. We have described how to use the robust regression machinery to reduce the effect of outliers during the motion control of an eye-in-hand robot arm. The sensory-motor space is sampled by exploration. The uncalibrated Jacobian, which relates velocities in the motor space to the velocities in the sensor space and is typically used in the visual servoing control law, is found from the sampled space. We build sensory-motor databases of the target and obstacle objects instead of constructing their explicit models. In its most complete form, the database contains the sample, the local Jacobian matrix, and whether the sample is an inlier to the sample space or an outlier. Outlier samples can be recovered by further analysis of the nearest neighbors from the database. Sampling-based planning is used to derive occlusion-free paths in the sensory-motor space. The proposed planner fits well within the uncalibrated control framework with robustness against outliers embedded in the planning algorithm.

In the third contribution chapter (Chapter 6), we have shown how the projective geometry of three views can be used in UVS. This geometry can be reconstructed (up to a projectivity) from uncalibrated images and point or line correspondences across three views.

Although we have presented only the first steps towards functional systems to be deployed in unseen environments with minimal supervisory control, hopefully we have shown the potential of the robust and uncalibrated approach.

9.2 Delimitations and Limitations

It is important to consider the delimitations and limitations of this thesis. The following aspects are addressed:

- No prior camera calibration, calibration of camera-to-robot transformation;
- No prior explicit geometric models of target objects;
- No prior explicit geometric models of obstacle objects;
- Generalization of UVS across visual features;
- Static objects and slowly-moving dynamic objects;

However, we have not addressed some important aspects, including:

- Stability proofs of UVS control law;
- Goal state outside the sampled space;

The common control-theoretic machinery used for proofs is the Lyapunov stability analysis. At this point, it seems that finding the Lyapunov candidate functions for a UVS control law with a robust Jacobian is not straightforward. The other limitation is intrinsic to the approach. Visual servoing to goal states outside the sampled space can be performed using a Broyden update of the Jacobian, but for planning we need to have sampled some of the intermediate states.

9.3 Future Directions - RUVS and Planning

The way we envision using the complete RUVS system with occlusion avoidance via planning is to be part of a larger semi-autonomous setting. A semi-autonomous system comprises a supervisory control interface and several autonomous modules. The interface enables a human operator to specify tasks and issue commands. The autonomous modules are in charge of carrying out those commands, taking the uncertainties into account. The simple examples that have been presented all fit in this paradigm. The user specifies the objective of a visual servoing task, initializes the visual trackers for objects and obstacles, and specifies rules for joint-space limits, field-of-view limits, and visual occlusions. This can be represented by atomic or *primitive* tasks. There is potential to build a diverse vocabulary for task specification to specify complex tasks via a composition of primitive tasks. As such, task specification is a promising future direction to extend this research.

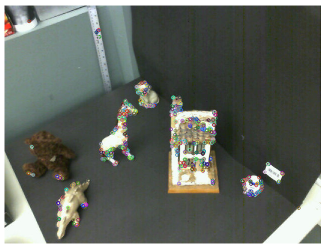
Other directions could explore methods to extract a global model of the sensory-sensor space. There are regression algorithms to build a global model from local data when outliers are not present. The most famous of these algorithms is the Receptive Field-Weighted Regression (RFWR) [125]. The RFWR algorithm is interesting because once the model is created after a training phase, it can be updated online when new data becomes available. This suits the Jacobian estimation and update routines, where the forward visual-motor map and the Jacobian model can be learned. The RFWR algorithm can only handle Gaussian noise, but the performance degrades significantly when outliers exist in the training phase. Integrating statistical robustness into online incremental learning algorithms sounds like a promising future work.

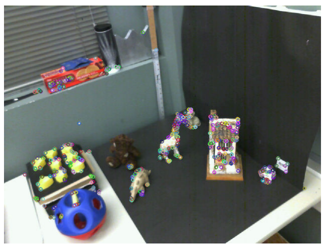
For the planning algorithm, an area that deserves attention is specification of visual occlusion. The target and obstacle objects in the planning experiments were planar and rather simple. Determining visual occlusion in the visual space with no prior models led us to simplifications. In a real settings, target objects and obstacles would not be planar. A more elaborate approach to determine visual occlusion is another future work.

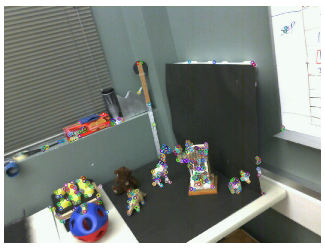
9.4 Future Directions - Trifocal Tensor

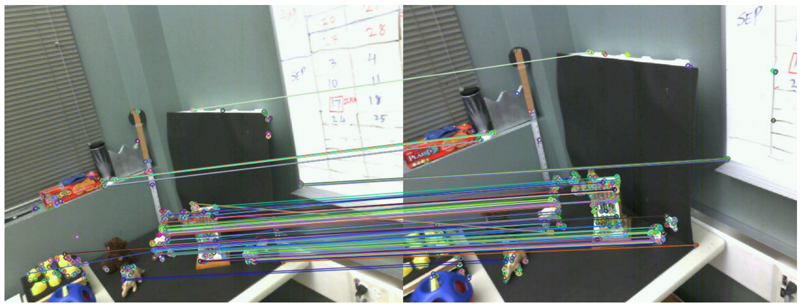
The concept of UVS from the geometry of three views, by estimation of the trifocal tensor, has several practical extensions in the future. A natural extension is to use a larger set of points correspondences to estimate the tensor, say, 500 triples as suggested by Hartley and Zisserman [110], and reduce the effect of mismatches using the RANSAC algorithm [59] or other robust matching techniques.

The Scale Invariant Feature Transform (SIFT) keypoint and descriptor [126] is a standard algorithm to find point correspondences. It provides good results but slow for most practical applications. A variant of SIFT without orientation has been used in visual servoing [127]. There is recent progress in developing computer vision algorithms for fast detection of keypoints and reliable descriptors since the introduction of SIFT. The Speed-Up Robust Feature (SURF) [128] has similar matching scores to SIFT, but has a much faster performance. The FAST algorithm [129] detects corners using machine learning and is ideal for fast keypoint detection, but corner orientation is not found. The Binary Robust Independent Elementary Feature (BRIEF) [130] is a descriptor with similar matching scores to SIFT, but it is sensitive to rotation. Recently, an open-source integration of FAST and BRIEF has been implemented as Oriented FAST and Rotated BRIEF (ORB) [131]. The ORB features seem like a natural fit to the trifocal estimation algorithm (Figure 9.1). It provides many reliable point correspondences across three views to estimate the


(a)


(b)


(c)


(d)

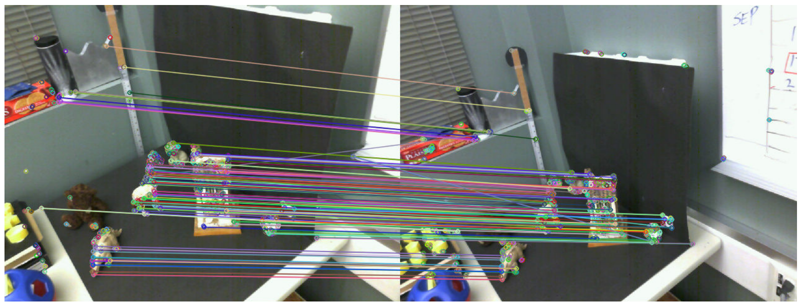

(e)

Figure 9.1: ORB features [131]. (a)-(c) The features are shown by coloured circles in three views. (d)-(e) The ORB features in view (c) are matched to ORB features from two different views.

trifocal tensor. There are mismatches in the point correspondences, of course, which can be handled by RANSAC.

Another direction for future study is to investigate particular camera motion trajectories and the specific properties of the trifocal tensor. For example, during an optical axis rotation and translation,

$$\tau_k = \begin{cases} Z & k = 3,\ 15 \\ -z_d \sin\phi & k = 9 \\ -Z & k = 27 \\ 0 & \text{otherwise} \end{cases}$$

Therefore, the rotation angle can be controlled from τ_9 (z_d is constant) and the depth variable can be controlled directly from either τ_3, τ_{15}, or τ_{27}. Other special cases can be studied to choose an optimal subset of the trifocal elements for control.

Bibliography

[1] G. R. Bradski, "Computer vision face tracking for use in a perceptual user interface," *Intel Technology Journal*, vol. 2, no. 2, Q2 1998. 2

[2] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Trans. Robot. Automat.*, vol. 12, no. 5, pp. 651–670, Oct. 1996. 3, 13

[3] F. Chaumette and S. Hutchinson, "Visual servo control. part I: Basic approaches," *IEEE Robot. Automat. Mag.*, vol. 13, no. 4, pp. 82–90, Dec. 2006. 3, 13, 14, 15, 16, 17, 18, 24, 103, 107

[4] ——, "Visual servo control. part II: Advanced approaches [tutorial]," *IEEE Robot. Automat. Mag.*, vol. 14, no. 1, pp. 109–118, Mar. 2007. 3, 13, 18, 24

[5] D. Kragic and H. I. Christensen, "Survey on visual servoing for manipulation," Royal Institute of Technology, ISRN KTH/NA/P–02/01–SE, Tech. Rep. CVAP 259, January 2002. 3

[6] K. Hosoda and M. Asada, "Versatile visual servoing without knowledge of true jacobian," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1994, pp. 186–193. 3, 9, 21, 24, 39, 57

[7] M. Jägersand, O. Fuentes, and R. Nelson, "Experimental evaluation of uncalibrated visual servoing for precision manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1997, pp. 2874–2880. 3, 9, 18, 21, 24, 57

[8] J. A. Piepmeier, G. V. McMurray, and H. Lipkin, "Uncalibrated dynamic visual servoing," *IEEE Trans. Robot. Automat.*, vol. 20, no. 1, pp. 143–147, Feb. 2004. 3, 9, 18, 21

[9] A. M. Farahmand, A. Shademan, and M. Jägersand, "Global visual-motor estimation for uncalibrated visual servoing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2007, pp. 1969–1974. 3, 9, 18, 21, 22, 24, 48, 51, 57, 79

[10] M. Kazemi, K. Gupta, and M. Mehrandezh, *Visual Servoing via Advanced Numerical Methods*, ser. Lecture Notes in Control and Information Sciences (LNCIS 401). Springer, 2010, ch. Path-Planning for Visual Servoing: A Review and Issues, pp. 189–207. 4, 33, 40, 59, 66

[11] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*, D. Kriegman, G. . Hager, and A. Morse, Eds. LNCS Series, No 237, Springer-Verlag, 1998, pp. 66–78. 4, 13, 16, 17, 18, 25, 41, 45, 103, 107

[12] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Trans. Robot. Automat.*, vol. 18, no. 4, pp. 534–549, 2002. 4, 29, 33, 35, 38, 40

[13] K. Hosoda, K. Sakamoto, and M. Asada, "Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3d reconstruction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1995, pp. 29–34. 4, 39, 41, 59

[14] F. Schramm, A. Micaelli, and G. Morel, "Calibration free path planning for visual servoing yielding straight line behaviour both in image and work space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 2216–2221. 4, 41, 59

[15] B. Allotta and D. Fioravanti, "3D motion planning for image-based visual servoing tasks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2005, pp. 2173–2178. 4, 38, 41, 59

[16] D. Kragic, "Visual servoing for manipulation: Robustness and integration issues," Ph.D. dissertation, Royal Institute of Technology, Stockholm, Sweden, 2001. 5

[17] K. Toyama and G. Hager, "Incremental focus of attention for robust visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1996, pp. 189–195. 5, 23

[18] D. Kragic and H. I. Christensen, "Cue integration for visual servoing," *IEEE Trans. Robot. Automat.*, vol. 17, no. 1, pp. 18–27, Feb. 2001. 5, 23, 58

[19] ——, "Robust visual servoing," *International Journal of Robotics Research*, vol. 22, no. 10-11, pp. 923–939, October-November 2003. 5, 23, 58

[20] P. Preisig and D. Kragic, "Robust statistics for 3D object tracking," in *Proc. IEEE International Conference on Robotics and Automation*, May 2006, pp. 2403–2408. 5, 23

[21] A. I. Comport, D. Kragic, E. Marchand, and F. Chaumette, "Robust real-time visual tracking: Comparison, theoretical analysis and performance evaluation," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 18–22, 2005, pp. 2841–2846. 5, 23

[22] T. T. H. Tran and E. Marchand, "Real-time keypoints matching: application to visual servoing," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 10–14, 2007, pp. 3787–3792. 5, 23, 52, 54

[23] A. I. Comport, E. Marchand, and F. Chaumette, "Kinematic sets for real-time robust articulated object tracking," *Image and Vision Computing*, vol. 25, pp. 374–391, April 2007. 5, 23, 52, 54

[24] E. Malis, F. Chaumette, and S. Boudet, “2-1/2-d visual servoing,” *IEEE Trans. Robot. Automat.*, vol. 15, no. 2, pp. 238–250, Apr. 1999. 5, 13, 23, 35, 37

[25] C. J. Taylor and J. P. Ostrowski, “Robust vision-based pose control,” in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 3, Apr. 24–28, 2000, pp. 2734–2740. 5, 23, 35

[26] E. Malis and F. Chaumette, “Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods,” *IEEE Trans. Robot. Automat.*, vol. 18, no. 2, pp. 176–186, Apr. 2002. 5, 23, 24, 37, 68

[27] Y.-H. Liu, H. Wang, C. Wang, and K. K. Lam, “Uncalibrated visual servoing of robots using a depth-independent interaction matrix,” *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 804–817, Aug. 2006. 5, 23, 24

[28] H. Wang, Y.-H. Liu, and D. Zhou, “Adaptive visual servoing using point and line features with an uncalibrated eye-in-hand camera,” *IEEE Trans. Robot.*, vol. 24, no. 4, pp. 843–857, Aug. 2008. 5, 23, 24

[29] G. Hu, W. MacKunis, N. Gans, W. E. Dixon, J. Chen, A. Behal, and D. Dawson, “Homography-based visual servo control with imperfect camera calibration,” *IEEE Trans. Automat. Contr.*, vol. 54, no. 6, pp. 1318–1324, June 2009. 5, 23, 24

[30] A. I. Comport, E. Marchand, and F. Chaumette, “Statistically robust 2-D visual servoing,” *IEEE Trans. Robot.*, vol. 22, no. 2, pp. 415–420, Apr. 2006. 5, 23, 52, 54, 58, 80

[31] F. Dionnet and E. Marchand, “Robust stereo tracking for space applications,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, Oct. 29 - Nov 2 2007, pp. 3373–3378. 5, 23, 52, 54

[32] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” Computer Science Department, Iowa State University, Tech. Rep. TR 98-11, 1998. 7, 30, 63

[33] M. Kazemi, K. Gupta, and M. Mehrandezh, “Global path planning for robust visual servoing in complex environments,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 326–332. 7, 41, 42, 59, 66

[34] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001. 8, 29, 30, 41, 63, 66

[35] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, “Dynamic sensor based control of robots with visual feedback,” *IEEE Trans. Robot. Automat.*, vol. 3, no. 5, pp. 404–417, October 1987. 13

[36] F. Chaumette, “Image moments: a general and useful set of features for visual servoing,” *IEEE Trans. Robot.*, vol. 20, no. 4, pp. 713–723, Aug. 2004. 13, 16, 17, 25, 45

[37] G. Hager and K. Toyama, “XVision: A portable substrate for real-time vision applications,” *Computer Vision and Image Understanding*, vol. 69, no. 1, pp. 23–37, January 1998. 13

[38] W. J. Wilson, C. C. W. Hulls, and G. S. Bell, “Relative end-effector control using cartesian position based visual servoing,” *IEEE Trans. Robot. Automat.*, vol. 12, no. 5, pp. 684–696, October 1996. 13, 14

[39] B. Espiau, F. Chaumette, and P. Rives, “A new approach to visual servoing in robotics,” *IEEE Trans. Robot. Automat.*, vol. 8, no. 3, pp. 313–326, June 1992. 13, 14, 16, 25

[40] E. Malis and P. Rives, “Robustness of image-based visual servoing with respect to depth distribution errors,” in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 1, Sept. 14–19, 2003, pp. 1056–1061. 13

[41] L. Deng, F. Janabi-Sharifi, and W. J. Wilson, “Hybrid motion control and planning strategies for visual servoing,” *IEEE Trans. Ind. Electron.*, vol. 52, no. 4, pp. 1024–1040, 2005. 13

[42] N. R. Gans and S. A. Hutchinson, “Stable visual servoing through hybrid switched-system control,” *IEEE Trans. Robot.*, vol. 23, no. 3, pp. 530–540, 2007. 13

[43] V. Kyrki, “Control uncertainty in image-based visual servoing,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 12–17, 2009, pp. 1516–1521. 13

[44] L. Deng, F. Janabi-Sharifi, and W. J. Wilson, “Stability and robustness of visual servoing methods,” in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, May 11–15, 2002, pp. 1604–1609. 13, 18

[45] N. Gans, S. A. Hutchinson, and P. I. Corke, “Performance tests for visual servo control systems, with application to partitioned approaches to visual servo control,” *Int. J. Robot. Res.*, vol. 22, no. 10–11, pp. 955–981, 2003. 13

[46] M. Ficocelli and F. Janabi-Sharifi, “Adaptive filtering for pose estimation in visual servoing,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, Oct. 29–Nov. 3, 2001, pp. 19–24. 14

[47] A. Shademan and F. Janabi-Sharifi, “Sensitivity analysis of EKF and iterated EKF pose estimation for position-based visual servoing,” in *Proc. IEEE Conference on Control Applications CCA 2005*, Aug. 28–31, 2005, pp. 755–760. 14

[48] F. Moreno-Noguer, V. Lepetit, and P. Fua, “Accurate non-iterative o(n) solution to the pnp problem,” in *Proc. IEEE 11th Int. Conf. Computer Vision ICCV 2007*, 2007, pp. 1–8. 14

[49] E. Malis, “Visual servoing invariant to changes in camera intrinsic parameters,” in *Proc. Eighth IEEE International Conference on Computer Vision ICCV 2001*, vol. 1, July 7–14, 2001, pp. 704–709. 14

[50] J. T. Feddema, C. S. G. Lee, and O. R. Mitchell, “Weighted selection of image features for resolved rate visual feedback control,” *IEEE Trans. Robot. Automat.*, vol. 1, no. 7, pp. pp. 31–47, 1991. 16

[51] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. USA: John Wiley, 2006. 17, 18, 19, 26, 28

[52] C. Samson, M. L. Borgne, and B. Espiau, *Robot Control: the Task Function Approach*. Oxford, United Kingdom: Clarendon Press, 1991. 16

[53] O. Tahri and F. Chaumette, “Point-based and region-based image moments for visual servoing of planar objects,” *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1116–1127, Dec. 2005. 17, 25

[54] O. Tahri, Y. Mezouar, F. Chaumette, and P. Corke, “Decoupled image-based visual servoing for cameras obeying the unified projection model,” *IEEE Trans. Robot.*, vol. 26, no. 4, pp. 684–697, 2010. 17

[55] R. T. Fomena, O. Tahri, and F. Chaumette, “Visual servoing from three points using a spherical projection model,” in *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, 2010, pp. 5537–5542. 17

[56] B. J. Nelson and P. K. Khosla, “Vision resolvability for visually servoed manipulation,” *Journal of Robotic Systems*, vol. 13, no. 2, pp. 75–93, 1996. 17, 104

[57] H. Sutanto, R. Sharma, and V. Varma, “The role of exploratory movement in visual servoing without calibration,” *Robotics and Autonomous Systems*, vol. 23, pp. 153–169, 1998. 20

[58] J. T. Lapreste, F. Jurie, M. Dhome, and F. Chaumette, “An efficient method to compute the inverse jacobian matrix in visual servoing,” in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 1, 2004, pp. 727–732. 21

[59] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981. 23, 51, 75, 112

[60] L. Jamone, M. Fumagalli, G. Metta, L. Natale, F. Nori, and G. Sandini, “Machine-learning based control of a human-like tendon-driven neck,” in *Proc. IEEE Int Robotics and Automation (ICRA) Conf*, 2010, pp. 859–865. 24

[61] M. Marey and F. Chaumette, “Analysis of classical and new visual servoing control laws,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 19–23, 2008, pp. 3244–3249. 25, 76, 103, 107

[62] H. Hadj-Abdelkader, Y. Mezouar, and P. Martinet, “Decoupled visual servoing based on the spherical projection of a set of points,” in *Proc. IEEE Int. Conf. Robot. Automat.*, May 12–17, 2009, pp. 1110–1115. 25

[63] J. F. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1988. 26, 27, 29

[64] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, 1996. 26, 29

[65] S. M. LaValle and J. Kuffner, J. J., "Randomized kinodynamic planning," in *Proc. IEEE Int Robotics and Automation Conf*, vol. 1, 1999, pp. 473–479. 26

[66] J. Kuffner, J. J. and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '00*, vol. 2, 2000, pp. 995–1001. 26, 30

[67] S. LaValle, *Planning algorithms.* Cambridge University Press, 2006. 26, 27

[68] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations.* Cambridge, MA: MIT Press, 2005. 26, 27, 28, 29, 30

[69] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics- Modelling, Planning, and Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer, 2009. 26, 29

[70] M. Sipser, *Introduction to the Theory of Computation*, 2nd ed. Thomson Course Technology, 2006. 27

[71] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. the 20th IEEE Annual Symp. Foundations of Computer Science*, 1979, pp. 421–427. 27

[72] J. Schwartz and M. Sharir, "On the piano movers problem: II. General techniques for computing topological properties of real algebraic manifolds," *Advances in applied Mathematics*, vol. 4, no. 1, pp. 298–351, 1983. 27

[73] J. Culberson, "Sokoban is PSPACE-complete," in *Fun With Algorithms*, vol. 4, 1999, pp. 65–76. 27

[74] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 2, pp. 224–241, 1992. 28

[75] J. Barraquand, L. E. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan, "A random samplng scheme for robot path planning," *Int. J. Robot. Res.*, vol. 16, no. 6, pp. 759–774, 1997. 28

[76] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, pp. 412–442, 1990. 29, 40

[77] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Automat.*, vol. 8, no. 5, pp. 501–518, 1992. 29

[78] N. J. Cowan, J. D. Weingarten, and D. E. Koditschek, "Visual servoing via navigation functions," *IEEE Trans. Robot. Automat.*, vol. 18, no. 4, pp. 521–533, 2002. 29, 40

[79] K. I. Tsianos and L. E. Kavraki, "Replanning: A powerful planning strategy for hard kinodynamic problems," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS 2008*, 2008, pp. 1667–1672. 30

[80] G. Chesi and H. L. Yung, "Performance limitation analysis in visual servo systems: Bounding the location error introduced by image points matching," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 12–17, 2009, pp. 695–700. 33

[81] V. Kyrki, D. Kragic, and H. I. Christensen, "New shortest-path approaches to visual servoing," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2004)*, vol. 1, 2004, pp. 349–354. 34

[82] G. Chesi and A. Vicino, "Visual servoing for large camera displacements," *IEEE Trans. Robot.*, vol. 20, no. 4, pp. 724–735, 2004. 34

[83] G. Chesi and Y. S. Hung, "Global path-planning for constrained and optimal visual servoing," *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1050–1060, 2007. 35

[84] B. Thuilot, P. Martinet, L. Cordesses, and J. Gallice, "Position based visual servoing: keeping the object in the field of vision," in *Proc. IEEE Int. Conf. Robotics and Automation ICRA '02*, vol. 2, 2002, pp. 1624–1629. 35

[85] G. Chesi, "Visual servoing path planning via homogeneous forms and lmi optimizations," *IEEE Trans. Robot.*, vol. 25, no. 2, pp. 281–291, 2009. 35, 66

[86] F. Schwarzer, M. Saha, and J.-C. Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 338–353, 2005. 36

[87] S. Leonard, E. A. Croft, and J. J. Little, "Dynamic visibility checking for vision-based motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 2283–2288. 36

[88] M. Baumann, S. Leonard, E. A. Croft, and J. J. Little, "Path planning for improved visibility using a probabilistic road map," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 195–200, 2010. 36, 59

[89] S. Leonard, E. A. Croft, and J. J. Little, "Planning collision-free and occlusion-free paths for industrial manipulators with eye-to-hand configuration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 5083–5088. 36

[90] K. Deguchi, "Optimal motion control for image-based visual servoing by decoupling translation and rotation," in *Proc. Conf. IEEE/RSJ Int Intelligent Robots and Systems*, vol. 2, 1998, pp. 705–711. 37

[91] J. A. Borgstadt and N. J. Ferrier, "Visual servoing: path interpolation by homography decomposition," in *Proc. ICRA Robotics and Automation IEEE Int. Conf*, vol. 1, 2001, pp. 723–730. 38

[92] Y. Mezouar and F. Chaumette, "Optimal camera trajectory with image-based control." *Int. J. Robot. Res.*, vol. 22, no. 10-11, pp. 781–803, 2003. 38

[93] F. Schramm and G. Morel, "Ensuring visibility in calibration-free path planning for image-based visual servoing," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 848–854, 2006. 38

[94] F. Schramm, F. Geffard, G. Morel, and A. Micaelli, "Calibration free image point path planning simultaneously ensuring visibility and controlling camera path," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 2074–2079. 39

[95] J. S. Park and M. J. Chung, "Path planning with uncalibrated stereo rig for image-based visual servoing under large pose discrepancy," *IEEE Trans. Robot. Automat.*, vol. 19, no. 2, pp. 250–258, 2003. 39

[96] ——, "Image space path planning in consideration of mechanical constraints for image-based visual servoing," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2003)*, vol. 1, 2003, pp. 755–760. 39

[97] A. Chan, E. A. Croft, and J. J. Little, "Trajectory specification via sparse waypoints for eye-in-hand robots requiring continuous target visibility," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2008, pp. 3082–3087. 40, 41, 59

[98] M. Kazemi, M. Mehrandezh, and K. Gupta, "Kinodynamic planning for visual servoing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 2478–2484. 41, 59, 66

[99] A. Shademan, A.-M. Farahmand, and M. Jägersand, "Robust jacobian estimation for uncalibrated visual servoing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010, pp. 5564–5569. [Online]. Available: http://www.cs.ualberta.ca/~azad/papers/shademan2010a.pdf 44

[100] R. Sharma and H. Sutanto, "A framework for robot motion planning with sensor constraints," *IEEE Trans. Robot. Automat.*, vol. 13, no. 1, pp. 61–73, 1997. 45

[101] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills." *IEEE Robot. Automat. Mag.*, vol. 12, no. 4, pp. 40–52, December 2005, special Issue on "Software Packages for Vision-Based Control of Motion", P. Oh, D. Burschka (Eds.). 45, 46, 100

[102] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000. 45

[103] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust Statistics: The Approach Based on Influence Functions*, ser. Wiley series in probability and mathematical statistics. New York, NY: John Wiley and Sons, Inc., 1986. 51, 52, 53, 54

[104] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection.* New York, NY: USA: John Wiley and Sons, Inc., 1987. 51, 52, 54

[105] P. J. Huber, *Robust Statistics*, ser. Wiley Series in Probability and Mathematical Statistics. New York, NY, USA: Wiley-IEEE, 1981. 52, 54

[106] C. Collewet and F. Chaumette, "Using robust estimation for visual servoing based on dynamic vision," in *Proc. IEEE Int. Conf. Robot. Automat.*, Apr. 10–14, 2007, pp. 2835–2840. 52, 54

[107] M. Black and A. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *Int. J. Comput. Vision*, vol. 36, no. 2, pp. 101–130, 1998. 52, 54

[108] A. Shademan and M. Jägersand, "Robust sampling-based planning for uncalibrated visual servoing," in *to appear in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012. [Online]. Available: http://www.cs.ualberta.ca/~azad/papers/shademan2012a.pdf 59

[109] ——, "Three-view uncalibrated visual servoing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 6234–6239. [Online]. Available: http://www.cs.ualberta.ca/~azad/papers/shademan2010c.pdf 68, 69

[110] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2003. 68, 70, 71, 75, 103, 112

[111] E. Malis and F. Chaumette, "2 1/2 d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement," *Int. J. Comput. Vision*, vol. 37, no. 1, pp. 79–97, 2000. 68

[112] G. Chesi, D. Prattichizzo, and A. Vicino, "A visual servoing algorithm based on epipolar geometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2001, pp. 737–742. 68

[113] G. Mariottini, G. Oriolo, and D. Prattichizzo, "Image-based visual servoing for nonholonomic mobile robots using epipolar geometry," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 87–100, Feb. 2007. 68, 69

[114] H. M. Becerra and C. Sagues, "A sliding mode control law for epipolar visual servoing of differential-drive robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 3058–3063. 68, 75, 77

[115] S. Benhimane and E. Malis, "Homography-based 2d visual servoing," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 15–19, 2006, pp. 2397–2402. 68

[116] G. López-Nicolás, C. Sagüés, J. J. Guerrero, D. Kragic, and P. Jensfelt, "Nonholonomic epipolar visual servoing," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2006, pp. 2378–2384. 69

[117] H. M. Becerra and C. Sagues, "Pose-estimation-based visual servoing for differential-drive robots using the 1d trifocal tensor," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 5942–5947. 69

[118] G. López-Nicolás, J. J. Guerrero, and C. Sagüés, "Visual control through the trifocal tensor for nonholonomic robots," *Robot. Auton. Syst.*, vol. 58, pp. 216–226, 2010. 69, 75, 77

[119] R. Hartley and N. Dano, "Reconstruction from six-point sequences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2000, pp. 480–486. 75

[120] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robot. Automat. Mag.*, vol. 3, no. 1, pp. 24–32, Mar. 1996. 79, 95, 103

[121] G. L. Mariottini and D. Prattichizzo, "EGT for multiple view geometry and visual servoing: robotics vision with pinhole and panoramic cameras," *IEEE Robot. Automat. Mag.*, vol. 12, no. 4, pp. 26–39, 2005. 79, 95, 103

[122] S. Leonard. openman: An open source C++ toolbox for control and simulations of manipulators. [Online]. Available: http://sourceforge.net/projects/openman/ 79

[123] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robot. Automat. Mag.*, vol. 12, no. 4, pp. 40–52, Dec. 2005. 79

[124] F. Bonsignorio, J. Hallam, and A. P. del Pobil. (March, 2010) GEM guidlines. EURON Special Interest Group on Good Experimental Methodology. Retrieved. [Online]. Available: http://www.heronrobots.com/EuronGEMSig/downloads/GemSigGuidelinesBeta.pdf 95, 103

[125] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Computing*, vol. 10, pp. 2047–2084, 1998. 112

[126] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004. 112

[127] A. Shademan and F. Janabi-Sharifi, "Using scale-invariant feature points in visual servoing," in *Poceedings of SPIE, Vol. 5603, OpticsEast 2004: Machine Vision and Its Opto-mechatronic Applications Conference*, ser. Proc. SPIE, 2004, pp. 341–346. 112

[128] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proc. European Conf. Comput. Vis. (ECCV)*, May 2006, pp. 404–417. 112

[129] E. Rosten and T. Drummond, "Machine learning for highspeed corner detection," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2006, pp. 430–443. 112

[130] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. European Conf. Comput. Vis. (ECCV)*, 2010, pp. 778–792. 112

[131] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, 2011, pp. 2564 – 2571. 112, 113