OPTIMIZATION OF POLYNOMIAL SYSTEM SOLVERS WITH APPLICATIONS TO VISUAL ODOMETRY

Oleg Naroditsky

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2012

Kostas Daniilidis, Professor
Computer and Information Science
Supervisor of Dissertation

Jianbo Shi, Associate Professor
Computer and Information Science
Graduate Group Chairperson

Dissertation committee

Camillo J. Taylor, Associate Professor
Computer and Information Science

Jean Gallier, Professor
Computer and Information Science

Ben Taskar, Assistant Professor
Computer and Information Science

Stergios I. Roumeliotis, Associate Professor
Dept. of Computer Science and Engineering
University of Minnesota

OPTIMIZATION OF POLYNOMIAL SYSTEM SOLVERS
WITH APPLICATIONS TO VISUAL ODOMETRY

Acknowledgments

First and foremost, I'd like to thank my advisor, Prof. Kostas Daniilidis, for his support over the many years we've known each other. This dissertation, my graduate and even professional career are a tribute to his unwavering belief in me, even when evidence, at times, suggested it was unwarranted. The value of his mentorship and advice is difficult to overstate in my development as a researcher.

My conversations with Prof. Jean Gallier and Prof. C.J. Taylor have yielded many valuable insights and Jean's encouragement in particular was very valuable.

It was a privilege to work with students and postdocs at the GRASP Laboratory, who made the place lively and enjoyable. They are: Kosta Derpanis, Alex Toshev, Ben Sapp, Sandy Patterson, Roy Anati, Davide Scaramuzza, Ryan Kennedy, Matthieu Lecce, Menglong Zhu and Cody Phillips. Mayank Bansal deserves a special mention for partnering with me for the Convex Optimization course.

I'd also like to acknowledge my SRI Sarnoff collaborators: Jim Bergen, Aveek Das and Supun Samarasekera. Graduate school life would have been a lot harder without them.

I gratefully acknowledge my friends, who have offered me support and encouragement throughout this process. They are: Calvin Look, Robin Kyin, Yuriko Horvath and Ameesh Makadia.

Last but not least, my aunt, Svetlana Kushnirsky, who has fed so me many delicious meals, also contributed to the timely completion of this work.

ABSTRACT

OPTIMIZATION OF POLYNOMIAL SYSTEM SOLVERS WITH APPLICATIONS TO VISUAL ODOMETRY

Oleg Naroditsky

Kostas Daniilidis

Efficient solutions to polynomial equation systems is an important topic in modern geometric computer vision. The importance stems from the fact that many minimal problems (problems that use the fewest possible number of constraints) have been formulated as polynomial systems in recent years. Minimal problems are extremely important in geometric computer vision because they guarantee the highest probability of rejecting outliers when used with robust estimation frameworks such as RANSAC. However, since many instances of the same minimal problem have to be solved in a typical live vision system, efficient solutions are paramount to reaping full benefit from their solutions. The goal of this work is to solve new and useful minimal geometry problems as well as advance the theory behind the solution methods. New solutions to two such minimal problems are offered and a new development in optimizing solutions for a class of problems using the so-called "action matrix" method is also presented. The first minimal problem that is solved is the structure from motion with a directional correspondence, where image projections of three 3D points in two cameras are combined with a common direction or a point at infinity to solve for camera motion. This algorithm can be solved in closed form or using algebraic geometry and becomes a foundation of a visual odometry algorithm that uses a four-point RANSAC hypothesis to estimate motion instead of the traditional five. The second minimal problem uses 3D point to plane correspondences to establish the motion between two coordinate systems and targets the problem of LIDAR to camera calibration. A set of six correspondence of image line (a plane in 3D)

to LIDAR points is sufficient to estimate the relative pose of the devices, including scale. This problem can be solved in closed form using the Macaulay resultant, and becomes the basis for construction of automatic calibration software. The proposed optimization to the action matrix method leads to an improvement in performance and numerical stability of existing algorithms, such as uncalibrated image stitching and three-view triangulation.

Contents

List of Figures

Algebraic Geometry Notation

- K – a field
- $\mathbb{Z}$ – the integers
- $\mathbb{C}$ – the complex numbers
- $\mathbb{R}$ – the complex numbers
- $\mathbb{Z}/p$ – the integer field modulo prime p
- $\mathbf{x}^\alpha = x_i^{\alpha_1} \dots x_n^{\alpha_n}$ – a monomial
- $f(x_1, \dots, x_n) = c_1\mathbf{x}_1 + \dots + c_m\mathbf{x}_m$ – a polynomial
- $K[x_1, \dots, x_n]$ – the ring of polynomials in $x_1, \dots, x_n$ over field K
- $K[x_1, \dots, x_n]/I$ – the quotient ring of $K[x_1, \dots, x_n]$ modulo I
- $\overline{f}^G$ – the remainder of polynomial division of f by all polynomials $g_i \in G$
- $I = \langle f_1, \dots f_k \rangle$ – a polynomial ideal generated by $\{f_1, \dots, f_k\}$
- $\deg(I)$ – degree of an ideal
- $\dim(I)$ dimension $K[x_1, \dots, x_n]/I$ as a vector space
- $\mathbf{X}$ – a vector of monomials
- $C\mathbf{X}$ – a polynomial system with coefficient matrix C

Chapter 1

Introduction

Solving geometric problems is the earliest application of the field now known as computer vision. Shortly after invention of practical photography by Louis Daguerre and Nicéphore Niépce in 1837, photogrammetry was invented to employ the new technology for the useful task of measuring objects. These efforts resulted in the first applications of mathematical methods to image interpretation. These early applications were primarily in land surveying and map creation, and brought forth the development of machines and methods for what would now be referred to as stereo vision and structure from motion. The early results largely relied on purely geometric solutions, which were precipitated by the developments in optics and projective geometry in the 19th century. Some of the pioneers in the field included a projective geometer and physicist Julius Plücker and his student Felix Klein.

Since its beginnings in the 19th century, geometry for vision has made especially rapid advances in the past two decades. Currently referred to as geometric computer vision, it was advanced by both new mathematical methods and availability of faster computers. In the last decade, the earlier approach of relying purely on geometric arguments gave way to algebraic interpretations of problems using methods of

algebraic geometry. Algebraic geometry is a branch of mathematics where geometry problems are interpreted from the perspective of commutative algebra. This is accomplished through the use of affine varieties (geometric objects), which are put in correspondence with polynomial ideals (algebraic objects). This correspondence was initially developed by Hilbert with his famous Nullstellensatz [18]. The modern field grew out of projective geometry over the course of the second half of the 20th century with Alexander Grothendieck widely recognized as making the key contributions [61].

A classic example of a problem in both photogrammetry and computer vision is the so-called calibrated five point relative pose problem. It is a minimal case of the so-called structure from motion problem for the perspective projection camera. The objective is to recover the rotation and translation of a camera between two views from five projections of 3D points into each view. In the earlest work, Kruppa proved [34] in 1939 that the five-point calibrated pose problem has at most 11 solutions relying on purely geometric arguments. It was not until Demazure in 1988 used modern algebraic geometry techniques that it was shown that the problem has, in fact, at most 10 solutions [19]. Specifically, Demazure expressed the problem as a set of polynomial constraints and constructed the corresponding projective variety. It was then matter of computing the degree of the variety, which turned out to be 10. The five-point problem, including the Demazure constraints, will be covered in more detail in Chapter 2.

Solving a new problem in geometric computer vision is in general similar to the above example. A chosen problem is first formulated as a system of polynomial equations having a finite number of solutions. Consequently, the basic tools of algebraic geometry currently used in computer vision are methods for solving systems of polynomial equations in several variables. Solving such systems is the domain of

elimination theory. One of the methods specifically used in this thesis, has its origin in Bruno Buchberger's 1965 Ph.D thesis [8] where the notion of Gröbner basis was developed [2]. This Gröbner basis is at the heart of the "action matrix" method for solving polynomial systems. The second important tool of elimination theory is the method of resultants. Specifically, we will use the so-called Macaulay resultant, which is an extension of the standard Sylvester resultant to systems of more than two polynomials.

Unlike photogrammetry, modern computer vision encompasses an entire spectrum of image interpretation, not just geometry. Important tasks include object and event recognition, object modeling and segmentation, image motion analysis and image and video enhancement. The success and maturity of geometric computer vision gave rise to numerous applications, especially in the last ten years. Among them are image stitching, real-time 3D reconstruction and visual navigation. Also known as visual odometry [56, 32], visual navigation is a task particularly important to the rapidly developing field of mobile robotics. The goal is to reconstruct a trajectory of a moving vehicle from camera input alone or in combination with other sensors [46]. In its basic form it can be thought of as structure from motion from video. Since visual odometry systems are frequently used in mobile applications [51], computational performance is extremely important and live visual odometry systems have been developed for that task. As part of this thesis, we will present an advance in the field of visual navigation which will allow for faster and more robust computation of vehicle's trajectory.

In this work we will concentrate on the so-called minimal geometry problems, which are the problems of pose estimation where only a minimal possible number of correspondences required for a finite number of solutions is available. The five-point problem mentioned above is minimal. Such problems have become extremely

useful for automated calibration, visual odometry for simultaneous localization and mapping (SLAM), augmented reality and others.

1.1 Related Work

Our work's primary application is concerned with solving and applying minimal geometry problems and adds two new methods. Minimal solvers were first introduced by Nister [54] with his famous five-point algorithm for structure from motion. Since then, minimal solutions have been found for a number of problems in geometry. Among them are the solutions to the autocalibration of radial distortion [37, 15], pose with unknown focal length [9], infinitesimal camera motion problem [64] and others. The trend in this field has been to use algebraic geometry techniques to analyze problem structure and construct solvers. This body of work was initially based on Gröbner bases techniques [63], but recently started to include other related methods for finding solutions to algebraic systems in order to improve speed and accuracy [13, 14]. These techniques have been applied non-minimal problems as well, such as three-view triangulation [65, 12]. Our optimization methods presented in Chapter 3 build directly on the fast and stable polynomial equation solving method of Byrod *et. al* [14].

Since the introduction of the Gröbner basis methods to computer vision by Stewenius [65, 63], numerous problems in geometry have been expressed as polynomial systems and solved. Minimal problems are particularly important in structure from motion, absolute pose estimation and feature-based image registration (stitching) because they construct hypotheses from minimal data, which, in turn, minimizes the probability of including an outlier in a hypothesis of a RANSAC-based process [23, 55]. Before the proliferation of these solvers researchers often relied on linear

algorithms which needed larger than minimal sets of points and hence considerably longer and less robust sampling process in RANSAC.

The first efficient solver for a minimal problem in computer vision, introduced by Nister [54] for the five-point relative pose, used a hand-crafted Gröbner basis solver. Since then Gröbner basis solvers were devised for a number of minimal problems, such as the solutions to autocalibration of radial distortion [38, 15], relative pose with unknown focal length [9], and infinitesimal camera motion [64]. Panoramic image stitching with unknown focal length has been solved [11] as well and will serve as the main demonstration of our approach. Some non-minimal problems have also been solved, such as optimal three-view triangulation [65, 12].

1.2 Contributions and Organization

We will now outline the main contributions of this work.

- We prove certain properties of current polynomial solvers used in computer vision and use them to develop a method for improving the state-of-the-art solver from Byrod *et. al* [15].

- Using the improved method above, we develop an algorithm for structure from motion problem using three image correspondences and a directional correspondence. We call this the "three-plus-one" method.

- We introduce a new method for computing visual odometry using the three-plus-one method above where four-point hypotheses are used instead of the classical five-point.

- We develop a new method for absolute pose estimation using six point on plane correspondences. It uses the Macaulay resultant to achieve a closed-form

solution. This minimal algorithm allows for accurate, automatic calibration of line scan LIDAR-camera systems from six image line to to LIDAR point correspondences.

This document is organized as follows. Chapter 2 is dedicated to a review of the basic concepts in computer vision and algebraic geometry and modern "action matrix" methods. We discuss image formation, set up several geometric problems and introduce robust estimation. In the algebraic geometry section, we describe the basic concepts such as varieties, polynomial rings and ideals as well as the Gröbner basis methods and its application to existing problems. The chapter also contains complete examples for computing the action matrix and designing a solution template for a polynomial system. It will also serve to set up the notation for the remainder of this document.

In Chapter 3 we introduce the methodological improvement to the standard method for solving polynomial systems called the action matrix method and show how it allows for significant reduction in computing time with simultaneous increase in accuracy of some of the algorithms using the method. The most significant computational step of the action matrix method is an LU or QR decomposition of a large matrix of polynomial coefficients. We show that our method allows one to significantly reduce the size of this coefficient matrix by removing certain rows and columns which turn out to be unneeded. The number of rows and columns removed has been observed to be up to 50% in one algorithm, which results in a significant improvement to the computational time as well as numerical stability. To illustrate these improvements we apply our optimization to the existing algorithms for the panoramic stitching with unknown focal length and radial distortion, and the optimal three-view triangulation. Both algorithms show reduction in the computational

time and the panoramic stitching can now be run using single precision arithmetic and in half the time per hypothesis.

In Chapter 4 we use the optimized action matrix method above to solve the structure from motion with a known directional correspondence problem. We formulate the problem in terms of motion parameters directly, not the essential matrix parameters, as in the competing approach from Fraundorfer *et. al* [24]. The action matrix solution, while not as fast as the closed-form solution from Xun and Roumeliotis, exhibits better numerical accuracy. We also give a characterization of degenerate configurations of this problem. This chapter also introduces the four-point visual odometry algorithm. It is based on the three-plus-one problem and allows us to compute up-to-scale (5DOF) visual odometry using four-point random subsets of correspondences instead of traditional five. We show that under realistic outdoor conditions our method produces similar results to the five-point method in a more robust and computationally efficient manner.

In the final chapter, we apply the Macaulay resultant to the minimal problem of pose estimation from six 3D point to plane correspondences. The result is a closed-form solution to the problem, which, as it turns out can be solved with a 4th order polynomial. We then apply this algorithm to the problem of calibration of LIDAR and camera systems, which is important in robotics. We use lines in the image and points in the LIDAR data as correspondences and develop an automatic procedure for calibrating such systems without initialization.

Chapter 2

Geometric Computer Vision and Algebraic Geometry

2.1 Introduction

Solving polynomial systems is an evolving art. Currently, there is no generic framework for solving systems of polynomial equations in a fast, numerically sound way, so every application domain has to fend for itself. Recent interest in algebraic geometry techniques in computer vision is motivated primarily by the availability of problems that can be expressed in terms of polynomials, but also the special property of these geometry problems which allows a solution template to be computed once (possibly at a high computational cost), and then applied to new instances of the same problem very quickly. The special property is that the polynomial system for a specific geometry problem is the same for each geometric configuration up to the polynomial coefficients, which are specific to the problem instance (different locations of features in the image, for example). The problem can then be solved in a generic way that is numerically stable for some large test set of configurations. The numerical issues associated with certain configurations, do not normally cause problems since the data (e.g. image point correspondences) are abundant enough to find non-degenerate configurations. The art in creating such solvers comes in manipulating the operations (for instance the monomial ordering or variable elimination sequence) to ensure a stable solution in all cases.

Many real-world problems can be expressed as systems of polynomial equations, such that the solutions to the system satisfy the original problem. In this section, we will highlight some of the recent polynomial system solving techniques developed specifically for geometric computer vision. These methods are state of the art for providing accurate solutions to a number of problems, including point triangulation, camera pose computation and camera calibration. The work in this area started with the basic hand-crafted Gröbner basis solvers [65, 64, 63]. These were followed

by automatic method for solver discovery [35] and finally by the current state-of-the-art technique using techniques which trade off the convenience of Gröbner basis for speed, numerical stability or accuracy [13, 15]. In a subsequent section we will show an improvement to the last solver, so it will be covered in the most detail.

Despite the successes in solving previously unsolved geometry problems using these methods, there is still a problem with practical usage of the algorithms, since hand-optimized solutions tend to be faster (though not necessarily more accurate). We will assess the current state of the art in speed and accuracy issues in the final section. We will also highlight some of the compromises in the different methods and offer some insight into the future directions.

Due to the complexity of the subject, it is impossible to provide the entire background on algebraic geometry within this dissertation. There are many excellent books on the subject [17, 18], but we will give a brief introduction in this chapter, referring the reader to the textbooks for proofs and details. We will also give a similarly short introduction to the field of geometric computer vision, although we recommend [27] for a complete treatment. We will sometimes present some computer vision concepts without proof or even a justification, since they are merely preliminary steps needed to obtain the initial polynomial system.

We will provide a short review of geometric computer vision, including minimal problem solving and robust estimation techniques in section 2.2. Section 2.4 will introduce some of the main concepts from algebraic geometry. In Section 2.5, we will discuss the basic Gröbner-based method developed by Stewenius and present examples of using this technique on a small polynomial system. In Section 2.6, we will describe more advanced techniques for improving performance and stability of the basic solver. In Section 2.7 we will offer some discussion on the relative merits of the methods.

2.2 Geometry and Robust Estimation

In this section we will motivate and set up some of the geometry problems that can be solved using commutative algebra techniques introduced in the remaining sections. We will introduce the basic notions from geometric computer vision and robust structure-from-motion estimation. We will also set up the polynomial systems for the 5-point pose, the 3-view triangulation and the 3-point image stitching with unknown focal length and radial distortion that will be used later to demonstrate existing techniques and our improvements.

2.2.1 Geometric Computer Vision

Geometric computer vision deals with problems of discovering motion or calibration parameters of cameras and structure of objects in a 3D world from projections of light reflected from those objects onto the 2D image plane of the cameras. Before the geometry problems can be formulated (in terms of points, lines and their correspondences), image structure must be analyzed. Low-level computer vision deals with detecting interest points (or lines), and matching them across different images. For simplicity, we will assume (unless otherwise specified) that all such features have been "normalized", which means that a camera can be though of as a rigid coordinate system in the world, with image center at $[0, 0, 1]$. We will also assume that the 2D points in the images have been detected, localized, normalized and matched to their correspondences in other images. There are two types of such correspondences: 2D-2D correspondence refers to two image points in two different cameras that are projections of light reflected from the same 3D point (object) in the scene in front of the cameras, and 2D-3D correspondence where a 3D point in the scene projects onto a 2D point in the image.

The basic problems in geometry of multiple view (perspective projection camera) geometry that can be solved via polynomial systems methods are summarized below:

1. The problem of **relative pose** refers to finding the position and orientation (pose) of a camera with respect to another camera from 2D-2D correspondences between two cameras. See Figure 2.1 for an illustration.

2. In a **resection** problem we seek the pose of camera from 2D-3D correspondences, i.e. given a 3D scene and its image, locate the camera in the scene's coordinate system.

3. In a **camera calibration** problem we want to find, in addition to relative pose, some internal parameters of the camera which make normalization possible. In this case, points are not normalized or partially normalized.

4. Solving a **triangulation** problem computes a 3D scene point from 2D-2D correspondences between two of more known cameras. See Figure 2.2

5. **Image stitching** (panorama creation) computes the 2D (image homograph) or 3D (camera rotation) parameters that align features in a set of images.

Geometric computer vision is a topic of numerous textbooks, with a complete treatment contained in Hartley and Zisserman [27].

2.2.2 Minimal Problems

Many methods have been developed to solve above problems. Some can be solved in closed form (such as 2-view triangulation), and some can be formulated as least squares problems, but without optimality guarantee. There is a class of problems which turns out to be particularly useful when dealing with noisy, real-world data.

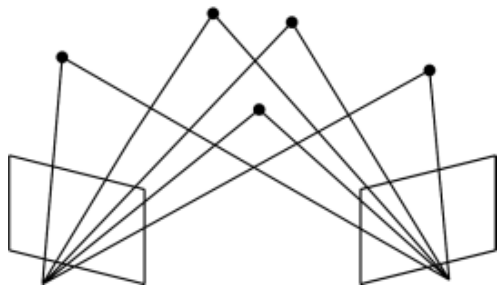


Figure 2.1: Illustration of the 5-point pose problem. We want to find the rotation and translation between cameras, given projections of five scene points.

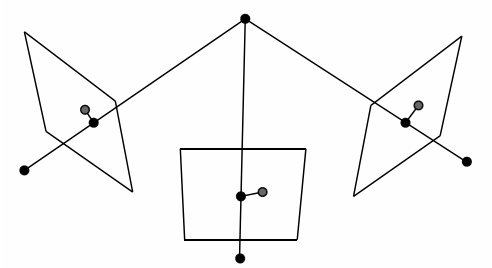


Figure 2.2: Illustration of the 3-view triangulation problem. We want to find a scene point that minimized the image reprojection error in three cameras.

Definition 2.2.1. Let $P(S)$ be a problem in geometric computer vision requiring a set of constraints S. $P(S)$ is a **minimal problem** if, in general, it has a finite number of possible geometric configurations that satisfy it, and the number of configurations for any set smaller than S is infinite.

When referring to a number of solutions of a problem, the expression "in general" in this case means that a randomly selected set of constraints will yield a given number of solutions almost surely. Of course, problems arising in practice have only one true solution corresponding to the real positions of cameras and points in the world when the measurements were taken. But since we seek not to overconstrain the problem, geometric ambiguities (such as reflections around camera center and "twisted pair" [27, 52] in the case of relative pose estimation) arise when minimal problems are solved. These ambiguities are easy to resolve by testing the recovered set of solutions against additional constraints.

Of the problems listed in the previous section, the triangulation problem is a special case. It is used to recover scene structure when camera poses are known, and as such not used in a robust framework. The minimum set of correspondences for triangulation is 2, but an optimal, Gröbner basis-based solution exists for 3

correspondences.

2.2.3 Robust Estimation

The output of a low-level vision subsystem of a real-world computer vision application is often noisy. We will first cover the possible sources of error in the feature detection and matching subsystem.

There are two sources of noise due to image processing and each one has to be dealt with by different means. The first source of noise comes from gross errors in feature detection and matching. For example, a feature that does not correspond to a stable, real-world structure might be detected by a feature detector (such as Harris of SIFT detector). This can happen purely due to image noise, but also shadows, reflections, lens flares, true features of objects that do not represent the model to be recovered and virtual features arising from intersections of projections of objects at different depths in the image plane. Errors in feature matching, which can arise when matching repetitive or weak features produce erroneous correspondences that also fall in the category of gross errors. Such gross errors are referred to as "outliers." The second source of error is feature localization. This error cannot be avoided and is inherent to all detected features, but some feature detectors are better than others (such as ones that locate features with sub-pixel accuracy).

Each category of noise needs to be dealt with separately. For problems of recovering camera pose the gross errors, or outliers, are usually eliminated using a "hypothesize-and-test," or RANSAC [23] framework. The detector noise is then minimized using least squares or iterative techniques (such as bundle adjustment), depending on the application.

The RANSAC framework requires that we produce a large number of hypotheses

from small subsets of observations (such as image correspondences), and test their fitness by scoring them against the rest of the observations (such as by computing the reprojection error of an observation in the camera). The hypothesis with the highest number of inliers is declared the winner, and is often used as an initial guess for an optimization process involving all observations (bundle adjustment).

We will demonstrate this process with a classic example. Given a set of 2D point observations $S = (x_i, y_i) \in \mathbb{R}^2$, we wish to fit a line robustly through the points. Since two points define a line, the minimal problem for a line fit needs two point observations. The steps of the RANSAC algorithm are given below.

1. Select a random subset of 2 points from the S.

2. Create a hypothesis by fitting a line through the 2 points.

3. Compute distances between the remaining points in S and the line hypothesis, and count the number of points (inliers) that have this distance within a certain threshold.

4. Repeat the steps 1 through 3 a certain number of times, keeping the hypothesis with the highest number of inliers.

5. Output the best hypothesis and its inlier set.

It is clear that the fewer observations we use, the better chance we have of getting a subset without gross errors (outliers). This is the main motivation for investigating problems involving minimal numbers of correspondences. There are many variations on the original RANSAC framework that are in use today [55, 68, 67].

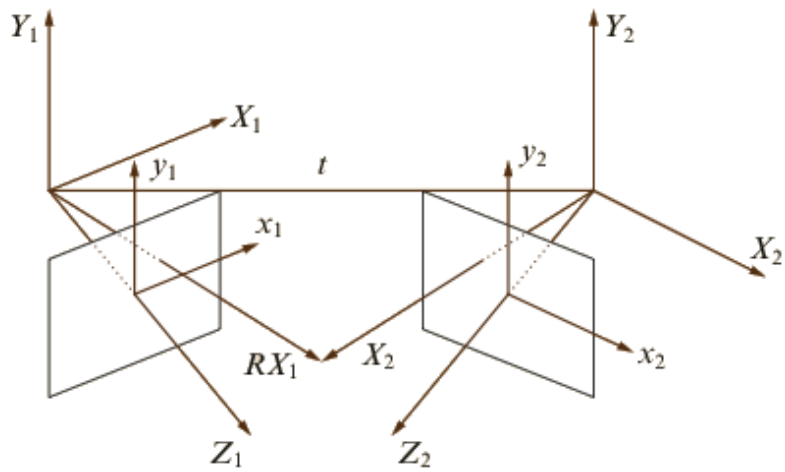


Figure 2.3: This figure illustrates the relationship between two camera coordinate frames (views) and the scene [62].

2.3 Basic Objects in Geometry

We now define some of the basic terminology used to set up the problems in the later sections.

- Image points are represented by homogeneous 3-vectors in real projective space $\mathbf{u} = (x, y, 1)^\top \in \mathbf{RP}^2$.
- Scene points are represented by homogeneous 4-vectors in real projective space $\mathbf{U} = (X, Y, Z, 1)^\top \in \mathbf{RP}^3$.
- A rigid camera motion matrix is represented by a 3×4 projection matrix P

The projection matrix can be further seen as $P = [R\, t]$, where R is a 3D rotation matrix ($R \in SO_3$), and t is the translation vector. A point $\mathbf{U}_s$ in the scene is related to a point $\mathbf{U}_c$ in the camera coordinate system by the following transformation:

$$\begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} = R \begin{pmatrix} X_s \\ Y_s \\ Z_s \end{pmatrix} + t.$$

In a camera coordinate system, the center of projection is at the origin, the Z-axis is the optical axis, and the image plane is the plane $Z = 1$. So, the projection of a point in the camera coordinate system $\mathbf{U}_c = (X_c, Y_c, Z_c, 1)$ into the image point $\mathbf{u} = (x, y, 1)$ is

$$x = \frac{X_c}{Z_c},\ y = \frac{Y_c}{Z_c}.$$

The transformation between two cameras is illustrated in Figure 2.3. We will relate the projective transformation of a scene point into the image, as defined above, as $\mathbf{u} \sim P_c \mathbf{U_s}$.

2.3.1 The 5-point Relative Pose Problem

The first problem on the list is "relative pose", also known as "structure from motion" is a problem of estimating the points in the scene and camera motion from image point correspondences. An illustration of this is shown in Figure 2.3, where a scene point is projected into two cameras, creating a 2D-2D correspondence. The minimal case for the two-view epipolar geometry estimation requires five correspondences [52]. The 5-point relative pose problem is easy to compute using the Gröbner basis [63], and it will serve as a prototype for other solutions.

The classic problem is stated as follows. Given five normalized, 2D-2D correspondences $\mathbf{u}$ to $\mathbf{u}'$ between two perspective projection cameras, compute the rigid motion between the cameras. The rigid motion matrix $[R\ \mathbf{t}]$ has 6 degrees of freedom, however, the relative pose between cameras can only be estimated up to scale or the translation vector $\mathbf{t}$.

The solution to this problem rests on the following facts in computer vision, the justification for which is outside the scope of this report An efficient solution to this

problem was introduced by [52]. We first define a matrix which relates correspondences between two calibrated cameras. This constraint is given by $\mathbf{u}'^\top E\mathbf{u} = 0$, where $E \equiv [\mathbf{t}]_\times R$, and $[\mathbf{t}]_\times$ is a 3×3 skew-symmetric matrix. The matrix E is known as the "essential matrix".

This constraint can be re-written as $\tilde{\mathbf{u}}\tilde{E} = 0$, where

$$\tilde{x} \equiv [x_1x_1' \; x_2x_1' \; x_3x_1' \; x_1x_2' \; x_2x_2' \; x_3x_2' \; x_1x_3' \; x_2x_3' \; x_3x_3']^\top,$$

$\mathbf{u} = (x_1, x_2, x_3)$ and $\tilde{E}$ is the column vector of the entries of E in row order.

We then stack the $\tilde{\mathbf{u}}^\top$ for all five points and compute the right nullspace of the resulting 5×9 matrix. The four 9-vectors that span this space, if written as $3x3$ matrices X, Y, Z and W give us the following contraint for the essential matrix:

$$E = xX + yY + zZ + wW \tag{2.3.1}$$

for some $x, y, z, w \in \mathbb{R}$. The scale ambiguity allows us to set $w = 1$ [52].

In addition, it was shown in [29] that an essential matrix satisfies the trace constraint

$$EE^\top E - \frac{1}{2}\text{trace}(EE^\top)E = 0. \tag{2.3.2}$$

The above constraint implies the determinant constraint

$$\det(E) = 0. \tag{2.3.3}$$

The two equations above are referred to as Demazure constraints [19]. If we substitute the E from equation 2.3.1 into the constraints 2.3.2 and 2.3.3, we will obtain 10 cubic polynomials in x, y and z.

This polynomial system can be approached with Gröbner basis methods, but [52] shows a way to obtain a 10th degree polynomial in x only, and solving for y and z by substitution.

2.3.2 Optimal 3-view Triangulation

The optimal triangulation problem is different from the other problems covered by the methods in this report because it requires us to find an optimum of an objective function, rather than finding solutions that (in general) fit the input data perfectly. In this case, the error function for each view is the Eucledian distance between the observed point on the image, and the projection of the triangulated 3D point into the image (see Figure 2.2). This is referred to as the reprojection error. The complete solution to the 2-view problem was first given by [26], and the 3-view problem formulation presented below was not solved until the Gröbner basis solution was published in [65].

We will now formulate the problem in terms of polynomial constraints. We are given three camera matrices $P_i = [R_i \ \mathbf{t}_i]$, and three corresponding image points $\mathbf{u}_i$. We must compute a 3D point $\mathbf{U} = [x_1, x_2, x_3]^\top$, minimizing the objective function defined as the sum of squared reprojection errors:

$$F(\mathbf{U}) = \sum_{i=1}^{3} d(P_i\mathbf{U}, \mathbf{u}_i)^2, \tag{2.3.4}$$

where $d(\mathbf{u}_i, \mathbf{u}_j)$ is the Eucledian distance (in the image) between $\mathbf{u}_i$ and $\mathbf{u}_j$.

We first simplify the problem by setting all image points $\mathbf{u}_i$ to be at the origin, and adjusting the camera matrices to compensate. This is done by rotating them around the camera's center of projection to get $P_i' = [R_{\mathbf{u}_i} R_i \ \mathbf{t}]$, where $R_{\mathbf{u}_i} \in SO_3$ is the rotation between $\mathbf{u}_i$ and $[0, 0, 1]^\top$ in the camera coordinate system. We now

rewrite the objective as

$$F(\mathbf{U}) = \sum_{i=1}^{3} F_i(\mathbf{U}), \tag{2.3.5}$$

where reprojection errors $F_i(\mathbf{U})$ are given in terms of the rows of the camera matrices $P_i' = [P_i^1\ P_i^2\ P_i^3]^\top$:

$$F_i(\mathbf{U}) = \frac{(P_i^1\mathbf{U})^2 + (P_i^2\mathbf{U})^2}{(P_i^3\mathbf{U})^2} \tag{2.3.6}$$

which is the squared length of the image vector after re-projection. We now require that

$$\frac{dF}{dx_i} = 0 \tag{2.3.7}$$

for each component x_i of $\mathbf{U}$.

Further, we can choose a coordinate system where the third row of each P_i' has a non-zero element in the ith position, placing the camera centers on the plane $z = 0$. This allows us to write the following simplification:

$$F_i(\mathbf{U}) = \frac{(P_i^1\mathbf{U})^2 + (P_i^2\mathbf{U})^2}{x_i^2} \tag{2.3.8}$$

We differentiate this function with respect to x_i. When $i \neq j$, the derivative is [63]

$$\frac{dF_i}{dx_j} = 2\frac{P_i^1(j)P_i^1\mathbf{U} + P_i^2(j)P_i^2\mathbf{U}}{x_j^2}, \tag{2.3.9}$$

where $P_i^k(j)$ is the jth element of P_i^k. When $i = j$, we have

$$\frac{dF_j}{dx_j} = 2\frac{(P_j^1(j)P_j^1\mathbf{U} + P_j^2(j)P_j^2\mathbf{U})x_j - (P_j^1\mathbf{U})^2 + (P_j^2\mathbf{U})^2}{x_j^3} \tag{2.3.10}$$

When these derivatives are written out in terms of x_j for $j = 1 \ldots 3$ and brought

under the same denominator to form polynomial equations, we end up with 3 equations of degree 6. Since the expression for the derivatives has monomials in x_j in the denominator, we require that $x_1 x_2 x_3 \neq 0$, which means that solution arising from observations in the physical world will never have $x_j = 0$. We will discuss how to deal with this constraint in section 2.5.

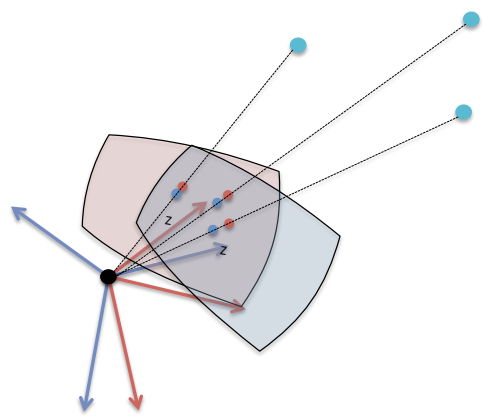


Figure 2.4: Illustration of the geometry of two-view panorama stitching with unknown common focal length and radial distortion.

2.3.3 Three-point Panorama Stitching

Another complex minimal problem is one of estimating the 3D rotation of the camera whose focal length and radial distortion are unknown. The geometry of the problem is illustrated in Figure 2.4. This problem was recently solved in [11] using the more advanced QR decomposition technique which will be covered in the next section. It required this method due to numerical stability issues associated with solving the polynomial system presented below. We will use this problem later as an example for our optimization technique where we significantly improve the performance and numerical stability of the solver. Here we will only introduce the polynomial model for

this minimal problem. The two cameras have square pixels, zero skew and the principal point in the center of the image, thus the calibration matrix $K = \mathrm{diag}(f, f, 1)$, where f is the focal length. The two camera views also share a common origin in the stitching scenario and thus $P_1 = K[I\ \mathbf{0}]$ and $P_2 = K[R\ \mathbf{0}]$, where $R \in \mathrm{SO}(3)$. We now arrive at the following relation between world point $\mathbf{U}$ and image point $\mathbf{u}$:

$$z_1\mathbf{u} = K\mathbf{U},\ z_2\mathbf{u} = KR\mathbf{U}, \tag{2.3.11}$$

where z_1 and z_2 are the depths. The dependence on the depths can be removed by rewriting the constraints as

$$\frac{\langle K^{-1}\mathbf{u}_{1j}, K^{-1}\mathbf{u}_{1k}\rangle^2}{|K^{-1}\mathbf{u}_1|^2|K^{-1}\mathbf{u}_1|^2} = \frac{\langle \mathbf{U}_j, \mathbf{U}_k\rangle}{|\mathbf{U}_j|^2|\mathbf{U}_k|^2} = \frac{\langle K^{-1}\mathbf{u}_{2j}, K^{-1}\mathbf{u}_{2k}\rangle^2}{|K^{-1}\mathbf{y}_{2j}|^2|K^{-1}\mathbf{u}_{2k}|^2} \tag{2.3.12}$$

These constrains are augmented with radial distortion model $|v| = (1 + \lambda|v|^2)|u|$. The normalized image point can now be expressed as

$$\mathbf{u} \sim \mathbf{v} + \lambda[0\ 0\ v_1^2 + v_2^2]^\top \tag{2.3.13}$$

By substituting (2.3.13) into (2.3.12), squaring to remove the square roots and multiplying through by the denominators, we obtain a polynomial of degree 3 in f^2, and degree 6 in λ. Using constraints from three point correspondences yields a system with 2 equations and 18 solutions [11].

2.3.4 Representing the System

Now that we have the polynomial system for each geometry problems presented above, we are ready to move on to solving them using techniques from algebraic geometry. For the remainder of this thesis, we will deal with the matrices of coefficients of these equations. We will think of the system as

$$C\mathbf{X} = 0, \tag{2.3.14}$$

where C is the matrix of monomial coefficients, and $\mathbf{X}$ is the vector of monomials under a certain monomial order. For example, a system of equations $\{x^2 + 2y^2 + 3x + 4 = 0, 5x + 6y + 7 = 0\}$ under lexicographic monomial order (see Section 2.4) will be given as

$$\begin{pmatrix} 1 & 0 & 3 & 2 & 0 & 4 \\ 0 & 0 & 5 & 0 & 6 & 7 \end{pmatrix} \begin{pmatrix} x^2 \\ xy \\ x \\ y^2 \\ y \\ 1 \end{pmatrix} = 0.$$

This type of representation is helpful when computing the Gröbner basis by the F4 method [22] discussed in section 2.5 and will serve as a basis for our solver optimizations proposed in Chapter 3.

2.4 Varieties, Ideals and Gröbner Basis

This section contains an introduction to the main concepts in algebraic geometry that will be used to solve systems of polynomial equations arising from minimal problems in computer vision. The presentation here is inspired by books by Cox et. al [17, 18] both of which we recommend as a formal treatment of the subject. We will approach the main concepts of algebraic geometry with an eye on solving systems of polynomial equations using Gröbner basis methods.

2.4.1 Overview

Given a set of polynomial equations in n variables with coefficients from $\mathbb{C}$, we want to find a set V in $\mathbb{C}^n$ that satisfies these equations. The main point in algebraic geometry is that we can describe V using a different set of polynomials. In fact, we can describe a set of all polynomials that V satisfies. We can now manipulate this set of polynomials, which we will call I, to choose from it polynomials which we want to solve (not the original ones), and be guaranteed that their solution set will be the same as V, the solution set we seek. The reasons we do not want to deal with the original equations may be either of the following:

- Original equations are difficult to solve (e.g. variables cannot be expressed in terms of other variables and eliminated).
- Original equations are ill-conditioned (e.g. determining the value of one variable using finite-precision arithmetic, and substituting results in a large error in another variable).

Using algebraic geometry we can try to address the above issues in a formal way, although straightforward construction of solvers is often not practical.

Algebraic geometry provides many methods for solving algebraic systems. These include direct Gröbner basis elimination, hidden variable resultant methods, multi-polynomial resultant-based techniques and eigendecomposition methods. Our overview in this chapter will be directed toward the eigendecomposition methods, which seeks to express a finite solution set V as an eigenvalue/eigenvector problem of a special matrix. This method is presented here since it is improved upon in Chapter 3 and used to solve the visual odometry problem in Chapter 4. In Chapter 5 we will give an overview of resultant-based methods, which allow us to solve another minimal problem, namely the camera-LIDAR calibration problem, in closed form. The hidden variable method has also found use in computer vision as well for solving minimal problems as well [39]. While some methods succeed better with certain problems than others, it is important to note that there is currently no generic method for solving an arbitrary system of equations under finite-precision arithmetic.

In the following sections we will introduce some of the basic objects and theorems that underpin the study of this problem.

2.4.2 Rings and Polynomials

We will begin with some formal definitions for the construction of polynomials. The choices of rings and fields are crucial when it comes to solving polynomial systems over them. A model ring is the integers, which are closed under addition, subtraction and multiplication, but not under division. A field, denoted by K, in addition to all properties of a ring, is also closed under division (except for division by the 0 element).

Since linear algebra operations on a vector space that will be defined by the coefficients of polynomials require that scalars come from a field, the exact choice

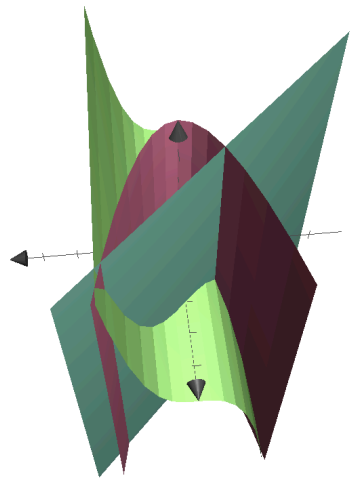


Figure 2.5: This figure illustrates a variety defined by the polynomial equations $y - x^2 = 0, z - x^3 = 0, x - y + 2 = 0$. The two points belonging to the variety can be seen at the intersection of the quadratic, cubic and planar sheets corresponding to the polynomials.

of a field is important. In geometric computer vision we will deal with the field of complex numbers $\mathbb{C}$ and its subfields the real numbers $\mathbb{R}$ and the integers modulo p denoted $\mathbb{Z}/p$, where p is a prime. The last field, called the **prime field**, is finite.

We will now formally define a monomial and polynomial of one or more variables.

Definition 2.4.1. A **monomial** is a finite product of variables $x_1, \ldots, x_n$ of the form $x_1^{k_1} x_2^{k_2} ... x_n^{k_n}$, and is denoted as $\mathbf{x}^\alpha$, where $\alpha \in \mathbb{N}^n$.

Definition 2.4.2. A **polynomial** is a finite linear combination of monomials with coefficients in a field K.

Under polynomial addition and multiplication, polynomials in $\mathbf{x} = x_1, \ldots, x_n$ form a **polynomial ring** over K is denoted as $K[x_1, \ldots, x_n]$.

2.4.3 Varieties and Ideals

A geometric description of a solution set to a polynomial system is given by an affine variety.

Definition 2.4.3. Given a set of polynomials $f_1, \ldots, f_s$ in a field $K[x_1, \ldots, x_n]$, we can define an affine variety as a set of all solutions to the system $f_1 = \ldots = f_s = 0$, and denote it as $\mathbf{V}(f_1 \ldots f_s)$.

For a geometric interpretation of the variety, let us consider a set of polynomials over a $\mathbb{R}[x, y, z]$, $\{y = x^2, z = x^3\}$. A variety defined by each polynomial generates a 2D surface in $\mathbb{R}^3$, and their intersection ($\mathbf{V}(y - x^2, z - x^3)$ defines a curve (see Figure 2.5).

We also define the object which is used to study the algebraic properties of varieties.

Definition 2.4.4. An **ideal** is a subset of $K[x_1, \ldots, x_n]$ generated by $f_1 \ldots f_s$ (and denoted by $< f_1 \ldots f_s >$) is the set

$$\{\sum_{i=1}^{s} h_i f_i : h_i, \ldots, h_s \in K[x_i, \ldots, x_n]\}. \tag{2.4.1}$$

In other words, an ideal generated by a set of polynomials $f_1 \ldots f_s$, is a set that includes the generators, and is also closed under the addition operation and multiplication by the members of the ring $K[x_1, \ldots, x_n]$.

We also define the operator $\mathbf{I}$, which as a set of all polynomials that vanish on a variety.

Definition 2.4.5. An **ideal of a variety** $V = \mathbf{V}(f_1, \ldots, f_s)$ is a set

$$\mathbf{I}(V) = \{f \in K[x_1, \ldots, x_n] : f(a_1, \ldots, a_n) = 0 \,\forall\, (a_a, \ldots, a_n) \in V\}. \tag{2.4.2}$$

It can be shown that $\mathbf{I}(V)$ is an ideal. Another helpful notion is a radical ideal:

Definition 2.4.6. An ideal $I \subset K[x_1, \ldots, x_n]$ is **radical** if $f^m \in I$ for any integer $m \geq 1$ implies that $f \in I$.

Radical ideals are denoted $\sqrt{I}$. As an example, let's see what the radical ideal of $J = \mathbb{Z}/4$ is. This is the ideal of all multiples of 4. Since 4 is in the ideal, 2 must also be in the radical ideal, and so is every multiple of 2 with other ideal members. So, $\sqrt{(J)} = \mathbb{Z}/2$. It turns out that an ideal generated by a variety (the set of all polynomials that vanish on that variety) is a radical ideal, and therefore corresponds to a set of all polynomials that vanish on that variety by a key theorem in algebraic geometry called Nullstellensatz.

Theorem 2.4.7. *Let I be an ideal in an algebraically closed field $K[x_1, \ldots, x_n]$, then*

$$\mathbf{I}(\mathbf{V}) = \sqrt{I} \tag{2.4.3}$$

In the process of solving geometry problems, we will construct an ideal from set of polynomials which vanish on the solutions to the problem (the variety), and manipulate it to find the solution.

To illustrate these concepts we will use a variety $V(x^2 - 1 = 0, y^2 - 1 = 0)$ in Figure 2.6. This variety with polynomials from $\mathbb{R}[x, y]$, has four distinct points: $(-1, -1), (-1, 1), (1, 1)$ and $(1, -1)$, plotted as yellow dots. These two equations that define the variety are plotted in red in the figure.. We create an ideal $\mathbf{I}(V)$, and plot some of its members in other colors. For example, $x^5 - x^2 + x^3 + y^2x + y^3 - x - y = 0$ is plotted in green, and $x^5 + x^2 - x^3 + 1 + y^2x - y^5 - 2y^2 - x - y^3 = 0$ is in blue. We see in the figure that each graph intersects the variety.

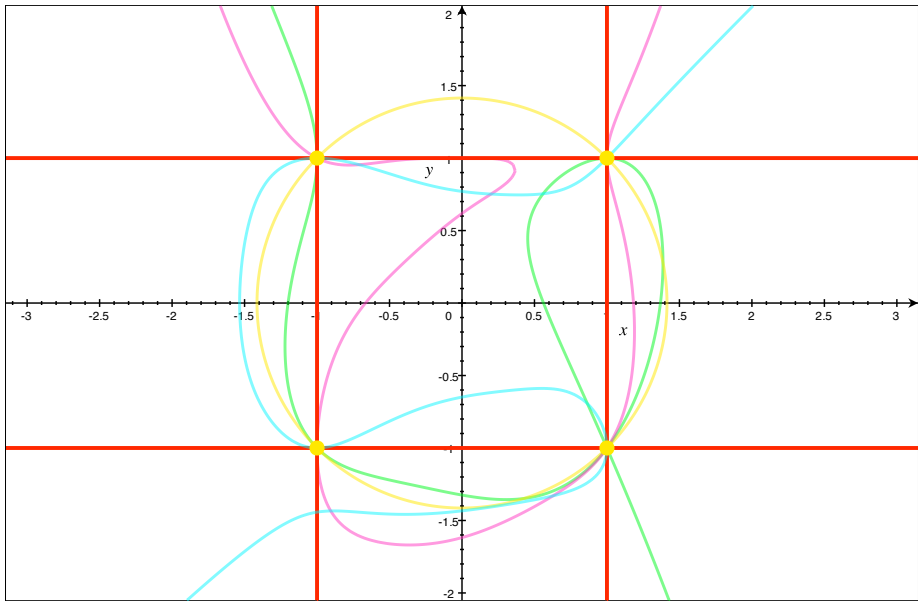


Figure 2.6: Illustration of a variety composed of the yellow points, $(-1,-1),(-1,1),(1,1)$ and $(1,-1)$, the ideal generators $\{x^2-1=0, y^2-1=0\}$ shown in red and other members of the ideal.

2.4.4 Polynomial Division and the Importance of Monomial Ordering

When performing polynomial division, the order of monomials in a polynomial affects the quotient and the remainder.

An order on monomials of the form $\mathbf{x}^\alpha = x_1^{a_1} \dots x_n^{a_n}$ is a well-ordering on tuples from $\mathbb{N}^n$. We will first define the lexicographic order.

Definition 2.4.8. Given two monomials $\mathbf{x}_1 = x_1^{a_1} \dots x_n^{a_n}$ and $\mathbf{x}_2 = x_1^{b_1} \dots x_n^{b_n}$, where variables are ordered such that $x_1 > \dots > x_n$, we say that under the **lexicographic order** $\mathbf{x}_1 >_{lex} \mathbf{x}_2$ if vector difference $(a_1 \dots a_n) - (b_1 \dots b_n)$ has the leftmost non-zero entry positive.

The order most often used when computing with Gröbner basis is the graded reverse lexicographic order.

Definition 2.4.9. Given two monomials $\mathbf{x}_1 = x_1^{a_1} \dots x_n^{a_n}$ and $\mathbf{x}_2 = x_1^{b_1} \dots x_n^{b_n}$, where variables are ordered such that $x_1 > \dots > x_n$, we say that under the **graded reverse lexicographic order** $\mathbf{x}_1 >_{grevlex} \mathbf{x}_2$ if

$$(a_1 - b_1, \dots, a_n - b_n, \sum a_i - \sum b_1) \tag{2.4.4}$$

has a negative rightmost non-zero entry.

This definition simply means that the monomials are ordered by total degree ($\sum a_i$) first, and the tie is broken in a reverse lexicographic way. In this chapter we will assume that the polynomials are ordered according to GrevLex. The GrevLex order has some nice computational properties, and is commonly used for Gröbner basis computation. Other monomial orders are convenient to use with techniques in the next chapter, and they will be described there.

We will illustrate the importance of ordering with an example, but first we will first define the quotient and the remainder of a polynomials division.

Definition 2.4.10. A **quotient** q and **remainder** r of two polynomials $f_1, f_2 \in K[x_1, \dots, x_n]$ are unique polynomials (under a chosen monomial order) such that

$$f_1 = qf_2 + r,$$

where degree$(r) <$ degree(f_2) or degree$(r) = 0$.

Let $f_1 = xy^2z$ and $f_2 = xz + x + y^2$. Using the division algorithm, we can find the quotients and the remainder of the division of f_1 by f_2. We can show that under under lexicographic order, we can express f_1 as follows:

$$f_1 = (-xy^2)f_2 - (xy^2 - y^4). \tag{2.4.5}$$

If we perform the division under the GrevLex order, we get

$$f_1 = (xz)f_2 + (-x^2z^2 - x^2z). \tag{2.4.6}$$

Once we fix the monomial order $>$, we can define the notions of leading term, leading monomial.

Definition 2.4.11. The **leading term** of a polynomial f under monomial order $>$ is the greatest term under $>$ with a non-zero coefficient. The leading term is denoted $LT(f)$.

Definition 2.4.12. The **leading monomial** of f is the monomial part of $LT(f)$. The leading monomial is denoted $LM(f)$

We will see their importance when we discuss the Gröbner basis.

2.4.5 The Gröbner Basis

Even when the monomial order is fixed, the division by two or more polynomials will not yield the same remainder in all cases. We will eventually be interested in the sets of equivalence classes under division by the generators of an ideal, and the Gröbner basis defines such a set of generators, that the remainders are unique regardless of order of polynomials in a division process.

We will use two facts from algebraic geometry to introduce the Gröbner basis.

Theorem 2.4.13. *Every ideal $I \subset K[x_1, \ldots, x_n]$ can be generated by a finite set of polynomials in $K[x_1, \ldots, x_n]$.*

This is known as the **Hilbert Basis Theorem**. The second important fact deals with uniqueness of quotients and remainders of polynomials discussed in the

last section will also help us construct quotient rings. A Gröbner basis for the ideal will uniquely define the remainders, which will allow us, eventually, to compute a set of useful equivalence classes on $K[x_1, \ldots, x_n]$ which will aid in finding the solutions to the original polynomial system.

Definition 2.4.14. For a fixed monomial order $>$ on $K[x_1, \ldots, x_n]$, a Gröbner basis for $I \subset K[x_1, \ldots, x_n]$ is a finite set of polynomials $G \subset I$ such that for all $f \in I$, $\mathrm{LT}(F)$ is divisible by $\mathrm{LT}(g_i)$ where $g_i \in G$.

This is equivalent to the following statement: $G = g_1, \ldots, g_n$ constitute a Gröbner basis of I if $\langle LT(g_1), \ldots, LT(g_n) \rangle = \langle LT(I) \rangle$.

It can be shown that G is indeed a basis for I, and that it can be computed starting with any generating set in a finite number of steps (by the Buchberger algorithm).

Note that by the Nullstellensatz this change of basis does not affect the solution set (the variety of I).

Many algorithms have been found to compute the Gröbner basis. The classic Buchberger's algorithm [7], was introduced in 1965 along with the Gröbner basis itself. This algorithm proved to be impractical due to large space requirements for some ideals [18], and due to numerical instability with finite precision arithmetic. The method for computing the Gröbner basis that is currently used in the computer vision literature was introduced in [22] as the "F4" algorithm. This algorithm improves both speed and numerical stability by representing the polynomial system as a $C\mathbf{X}$, where C matrix of coefficients and $\mathbf{X}$ is a vector of monomials in a fixed order. The algorithm iteratively performs steps of Gauss-Jordan elimination and adds more polynomials to the generating set until the basis is computed. The F4 algorithm is key for the Gröbner basis method presented in [65]. More details are

provided in Chapter 2.5.

2.4.6 Dimensions and Degrees

When deriving a solution to a new problem, it is necessary to describe a solution space of a variety. This is done by computing its **dimension**. We will illustrate it with an example. The variety $\mathbf{V}(y - x^2)$ describes a 2D manifold in $\mathbb{R}^3$, so, intuitively, its dimension is 2. On the other hand, the variety $\mathbf{V}(y - x^2, z - x^3)$ is the intersection of two 2D manifolds, and is, in general one-dimensional. Finally, the variety $\mathbf{V}(y - x^2, z - x^3, x - y + 2)$ has only two solutions, and it thus zero-dimensional. Figure 2.5 shows the graphs of all three equations. Analogously, when we talk about a zero-dimensional ideal, it is generated by a zero-dimensional variety. The algorithms for computing the Gröbner basis mentioned in the previous section only make sense on zero-dimensional ideals.

The degree of an ideal is the dimension of the vector space defined by the ring $K[x_1, \ldots, k_n]/I$, which also corresponds to the number of solutions to the polynomial system. In Chapter 2.5 we give an example of using Macaulay2 to determine that the number of solutions to the 5-point relative pose problem is 10.

2.4.7 Quotient Rings

The methods used to solve equations in this report rely heavily on properties of the quotient ring. Under division by G, a polynomial in $K[x_1, \ldots, x_n]$ by G, can be decomposed as follows:

$$f = h_1 g_1 + \ldots + h_t g_t + \overline{f}^G.$$

If G is a Gröbner basis of an ideal I, then the equation $\overline{f}^G$ is the **normal form** of f with respect to I. The equations in the ring with the same normal form with

respect to I make up a set of equivalence classes $[f]$ modulo I. We can now define the quotient ring.

Definition 2.4.15. The **quotient ring** $K[x_1, \ldots, k_n]/I$ is the set of equivalence classes for congruence modulo I:

$$K[x_1, \ldots, x_n]/I = \{[f] \in K[x_1, \ldots, x_n]\}. \tag{2.4.7}$$

Once the monomial order has been set, the class membership can be established by polynomial division by Gröbner basis G [18]. Each remainder of such division uniquely defines an equivalence class. The concept of quotient rings is crucial to solving polynomial systems. The remainders, which define the equivalence classes form a vector space with the standard basis defined by a set of monomials $B = \{\mathbf{x}^\alpha : \mathbf{x}^\alpha \notin \langle LT(I) \rangle\}$. Addition of vectors in this space is standard vector addition on the polynomial coefficients, once expressed in the basis B. The multiplication operation is multiplication modulo the Gröbner basis of I.

For a zero-dimensional ideal, this vector space is finite-dimensional [18].

2.4.8 Saturation of an Ideal

The action matrix method described below, as well as other methods based on the quotient ideal require that the input ideal is zero-dimensional (the variety vanishes at a finite number of points). For some problems, when the constraints are laid out in polynomial form, the ideal generated is not zero-dimensional, and contains families of solutions which need to be eliminated prior to solving the system. This elimination of specific solutions is called saturation. The following definition is from [63].

Definition 2.4.16. The saturation $\mathrm{sat}(I, f)$ of an ideal I with respect to a polynomial f is the set

$$\mathrm{sat}(I, f) = \{p \mid \exists k, f^k p \in I\}. \tag{2.4.8}$$

The saturation produces an ideal where the family of solutions $\mathbf{V}(f)$ has been removed from I.

2.4.9 The Action Matrix

Given a quotient ring $\mathbb{C}[x_1, \ldots, k_n]/I$ of a finite dimension, we can choose a basis B for the vector space formed by the polynomials under the division by the Gröbner basis, as described above. In this space, the multiplication by a polynomial in $\mathbb{C}[x_1, \ldots, k_n]$ turns out to be a linear operation [18]. The matrix m_f representing this operator is referred to as the **action matrix** in the computer vision literature, and "multiplication map" in [18].

Let $s = |B|$ be the size of B. The matrix m_f is a $s \times b$ matrix whose ith column is composed of the coefficients of the polynomial $\overline{f\mathbf{x}^{\alpha(i)}}^G$ where $\mathbf{x}^{\alpha(i)}$ is the ith basis monomial.

The following theorem, proved in [18], is the basis for the action matrix methods.

Theorem 2.4.17. *Let $I \in \mathbb{C}[x_1 \ldots x_n]$ be a zero-dimensional ideal, and let m_f be the action matrix for f. Then λ is an eigenvalue of m_f if and only if λ is a value of a function f on $\mathbf{V}(I)$.*

This theorem tells us that if we can extract this action matrix m_{x_i} from the vector space spanned by the basis monomials of $\mathbb{C}[x_1, \ldots, k_n]/I$, we can find the values of x_i on the variety via eigenvalue decomposition. It is also shown in [18] that, with certain restrictions, we can compute the entire solution set as the left eigenvectors of m_f.

2.4.10 Solving a Polynomial System

We give an example of solving a polynomial system using the action matrix method in an computer algebra system Maple v11. The Maple commands will be given in `typewriter` font.

Let us go back to the set of equations $f_1 = y - x^2, f_2 = z - x^3, f_3 = x - y + 2$ shown in Figure 2.5, and the zero-dimensional variety they generate. To solve this system (find the points on the variety) we first need to find the GrevLex Gröbner basis G for the ideal $I(f_1, f_2, f_3)$:

```
F := [y-x^2,z-x^3,x-y+2]:
G := Groebner[Basis](F, tdeg(x,y,z));
```

$$G := [3y - 4 - z, 3x + 2 - z, z^2 - 8 - 7z]$$

We can now find the basis monomials for $\mathbb{C}[x, y, z]/I$ with respect to the same monomial order:

```
B := Groebner[NormalSet](G,tdeg(x,y,z));
```

$$B := [1, z], table([1 = 1, z = 2])$$

We can see that the quotient ideal is spanned by two monomials, so we expect two solutions. We can now compute the action matrix m_x:

```
m_x := Groebner[MultiplicationMatrix](x, B[1], B[2], G, tdeg(x,y,z));
```

$$m_x = \begin{bmatrix} -\frac{2}{3} & \frac{1}{3} \\ \frac{8}{3} & \frac{5}{3} \end{bmatrix}.$$

Analogously, we can compute

$$m_y = \begin{bmatrix} \frac{4}{3} & \frac{1}{3} \\ \\ \frac{8}{3} & \frac{11}{3} \end{bmatrix}$$

and

$$m_z = \begin{bmatrix} 0 & 1 \\ \\ 8 & 7 \end{bmatrix}.$$

We can find the solutions via the eigenvalue decomposition of the action matrices:

```
v_x := LinearAlgebra[Eigenvalues](m_x);
v_y := LinearAlgebra[Eigenvalues](m_y);
v_z := LinearAlgebra[Eigenvalues](m_z);
```

$$v_x := \begin{bmatrix} 2 \\ \\ -1 \end{bmatrix}, \; v_y = \begin{bmatrix} 4 \\ \\ 1 \end{bmatrix}, \; v_z = \begin{bmatrix} 8 \\ \\ -1 \end{bmatrix}.$$

These are the two solutions for x, y and z for the original system. We can check them by evaluating the polynomials at those points:

```
eval(F,[x=v_x[1],y=v_y[1],z=v_z[1]]);
eval(F,[x=v_x[2],y=v_y[2],z=v_z[2]]);
```

$$[0, 0, 0]$$

$$[0, 0, 0],$$

and thus we have found the solutions.

2.5 The Gröbner Basis Method

This section introduces the methods for developing solvers for polynomial systems introduced by Stewenius [63, 65] as well as the later method by Byrod *et. al* [14]. While there are many different ways of solving a polynomial system via Gröbner basis and elimination, these methods allows one to find a stable elimination path on sample data in a finite prime field $\mathbb{Z}/p$, and then use it to solve real problems in $\mathbb{R}$. The Stewenius method is the original method for solving minimal problems using algebraic geometry, and the other methods described in this section are improvements of this general framework.

2.5.1 Method Overview

Given a problem formulation as a polynomial system $F = \{f_1, \ldots, f_k\}$, such as ones described in section 2.2, we seek to exploit the inherent structure in the ideal generated by the class of equations arising from this problem. We find a set of equations from the $I = \langle F \rangle$ (in the form of monomial multipliers of the equations in F), such that we can extract a Gröbner basis for the ideal by Gauss-Jordan elimination on the matrix of coefficients. The monomial multiples of the original equations can then be used as a template which is found once, and then applied to all other instances of the problem (same algebraic structure, but different coefficients).

It has been shown that if we pose a problem over a finite field $\mathbb{Z}/p$, we can determine the dimension of the ideal by repeatedly sampling random coefficients from the field until a stable number for the dimension is found. The stable dimension (which implies a stable basis B for $Z/p[x_1, \ldots, x_n]$), is, in general, equivalent for the dimension of the original problem with coefficients from field of rationals $\mathbb{Q}$ [69, 35]. Given a starting set of equations, we will first analyze the problem in $\mathbb{Z}/p$, and then

use it to solve problems with coefficients in $\mathbb{R}$. The exact nature of the generalization from $\mathbb{Q}$ to $\mathbb{R}$, however, is not clear from the literature.

Specifically, we perform the following steps:

1. Use computer algebra software to pose the problem as an ideal I in a finite prime field $\mathbb{Z}/p$, choosing p to be relatively large. Set the monomial order to GrevLex.

2. Compute the dimension d of the quotient ideal $\mathbb{Z}/p[x_1, \ldots, x_n]/I$ to determine if it is zero.

3. Repeat the above steps for random coefficients from $\mathbb{Z}/p$ until we can determine the dimension of an ideal. If it is 0, go to step 4.

4. If $d > 0$, find the infinite solution families, and saturate I with their solutions, obtaining a 0-dimensional ideal.

5. Compute the Gröbner basis for I by adding all monomial multiples (up to the degree of $LT(I)$) to the current generators of I, and performing Gauss-Jordan elimination on the matrix. This step is repeated until a Gröbner basis is obtained (which can be verified by the computer algebra software).

6. Repeat the above steps until a stable elimination path is found. This is essentially a probabilistic process due to the usage of random coefficients and prime fields.

7. Set up the matrix with real coefficients from $\mathbb{R}$ and perform the elimination.

8. From the resulting basis B for $\mathbb{R}[x_1, \ldots, x_n]/I$ compute the action matrix (generalized companion matrix).

9. Compute the solutions to the problem as eigenvectors of the action matrix.

Note that in this, and all other methods presented in this document, we use GrevLex monomial order for all computations of the Gröbner basis. The equations that will be added to the ideal in Step 5, along with the initial equations form an **elimination template**. This template tells us how to get polynomials needed to construct a Gröbner basis from initial polynomials. The porting of the coefficients to $\mathbb{R}$ means using the polynomials with the same monomials as in the elimination basis, but with coefficients from the actual problem. The Gauss-Jordan elimination of the template then results in a Gröbner basis for the system. Similar elimination templates are used in other methods described in later sections.

2.5.2 The 5-point Problem

For the 5-point pose problem, we start with 10 equations in 3 unknowns arising from the trace and determinant constraints on the essential matrix. We form a matrix, A, as a GrevLex-order, 10×20 random matrix in $\mathbb{Z}/p$.

The code snippet below illustrates this procedure in a computer algebra system Macaulay2:

```
-- Create a polynomial ring in x,y and z with coefficients
-- in Z/30029
R = ZZ/30029[x,y,z];


-- Create essential matrix E as a function of x, y and z with
-- random coefficients in R
t = {1_R,x,y,z};
E = fold((a,b)->a+b, apply(4, i->t_i*random(R^3,R^3)));
```

```
-- Define the trace and determinant constraints
TraceConstraint = 2*E*transpose(E)*E-trace(E*transpose(E))*E;
DeterminantConstraint = det E;

-- Define the ideal generated by these constraints
I = ideal(TraceConstraint)+ideal(DeterminantConstraint);

-- Determine the dimension and degree of I
dim I
degree I
```

Analysis with algebraic geometry software reveals the dimension of the ideal to be 0, and the degree to be 10. This means that the variety, in general, consists of 10 points.

Since the rows of A turn out to be linearly independent in general, the Gauss-Jordan elimination of A produces a Gröbner basis after the first elimination (i.e. no additional equations need to be added). From this basis, and the 10×10 action matrix is extracted, and the values for x,y and z are extracted from the 10 eigenvectors of this matrix.

2.5.3 The 3-view Triangulation Problem

When the 3 polynomial constraints for this problem given in the last section are analyzed in Macaulay2, it is revealed that the ideal formed by the polynomials is not 0-degree. This implies that there are families of solutions which need to be eliminated prior to solving the system. The solution families occur when $xyz = 0$. In [65] it

proposed that 3 partial saturation steps are used to eliminate the false solutions. The steps correspond to saturating with $x = 0$, $y = 0$ and $z = 0$, performing Gauss-Jordan elimination and updating the generators of the ideal at each step.

After the last elimination, we will get Gröbner basis for the ideal, and the 47-dimensional basis for the quotient ideal $\mathbb{R}[x, y, z]/sat(I, x, y, z)$ from which we can extract the action matrix m_x, whose eigenvectors correspond to the 47 solutions for x, y, and z.

2.5.4 Choosing the Correct Solution

Once the candidate solutions have been found, it is a relatively simple matter to choose the solution which corresponds to the physical configuration of the cameras. The technique used to disambiguate the solutions is to take an additional point correspondence constraint, and check which solution fits this constraint best. This, of course, assumes that the additional point is not an outlier, which makes the entire process less robust.

2.5.5 Comparison to Other Methods

In the case of the 5-point relative pose problem, the Stewenius method was reported to compare favorably to the original hand-designed method by Nister [52] in terms of numerical accuracy. The histogram of the numerical errors for both methods is shown in Figure 2.5.5.

Since the 3-view triangulation was first solved by this method, we will use the experiments from [13] which show that the 95th percentile distance from the true point was 15.1 units for test cases of cameras and points being distributed in a 1000 unit cube around the origin. The paper reports an improvement of a factor of 10^6

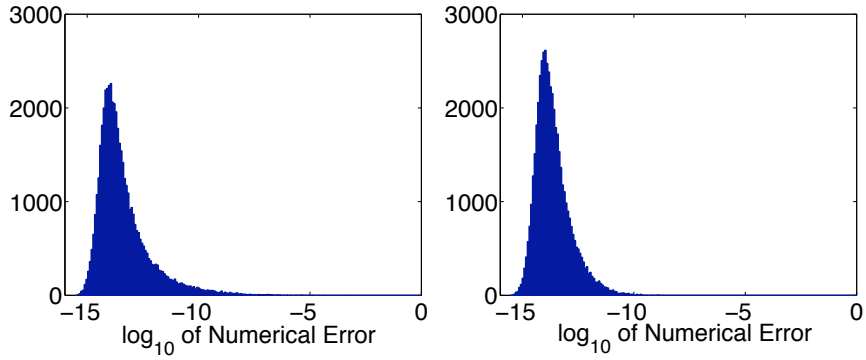


Figure 2.7: Distribution of the numerical error for the 5-point algorithm using the Nister method (left) and Stewenius method (right). The experiment has 50000 error-free, minimal cases. The numerical error was likely computed as the Frobenious norm of the difference matrix between the true and computed poses [65].

with the "change of basis" method presented in section 2.6.

2.5.6 A Fast and Stable Method for Geometry Problems

The action matrix method is one of the key ways of solving systems of polynomial equations. In this section we describe a method due to Byrod *et. al.* [13, 14]. This method was a key development in the application of algebraic methods to computer vision because it provided a stable way of generating "solution templates" for a variety of previously unaddressed problems, as well as improving accuracy of existing methods.

As we mentioned previously, a univariate polynomial can be solved using eigenvalue decomposition of a companion matrix and the action matrix is a multivariate equivalent of the companion matrix. The idea is to find a linear operator T_p for some $p \in K[\mathbf{x}]$ that represents the multiplication by p in the vector space defined by $K[\mathbf{x}]/I$, i.e., $T_p : f(\mathbf{x}) \rightarrow p(\mathbf{x})f(\mathbf{x})$. If we select a basis for this vector space, we can represent T_p as a matrix $\mathbf{m}_p$ with entries in K. It was shown in [18] that λ is

an eigenvalue of this matrix if and only if λ is a value of the function p evaluated on the variety V of the ideal. This means that if we set $p = x_k$, we can find the value of x_k which satisfies the initial system of equations. We can also determine the solutions through eigenvectors. It is known that the eigenvectors of the action matrix represent the scaled solutions to the same problem. We can also determine the scale, because the monomial 1 is always in the basis for zero-dimensional varieties.

Finding the dimensionality and basis for this vector space is the first step in recovering solutions. The dimensionality immediately tells us the number of solutions, while the basis is important in the action matrix computation. One way of obtaining these two quantities is through division of polynomials in $K[\mathbf{x}]$ by the Gröbner basis, which is a special basis for the ideal, division by which cancels out all the possible leading terms of the polynomials in the ideal.

Computing a Gröbner basis using finite precision arithmetic, however, is known to be a numerically unstable process. However, algorithms developed by Traverso in[69] allow us to analyze the ideal generated by our system using coefficients from a *prime field* $K = \mathbb{Z}/r$ (integers modulo r), where $r >> 7$ is a prime number [35]. Since this field is finite, the computation with polynomials with coefficients in $\mathbb{Z}/r$ (including Gröbner basis) is exact. The algorithms ensure that if a stable Gröbner basis is found in this field under repeated trials with random coefficients, the monomials will remain the same when we change the field to $\mathbb{Q}$ with some probability. In our case, we only need the Gröbner basis for one system, F, and it is easy to check when we have it. Once a Gröbner basis G is found, a linear basis for the quotient space can be formed by the monomials in the remainder after division by G.

In order to use efficient linear algebra techniques to manipulate the system (4.3.2)-(4.3.3), we rewrite it as follows:

$$\mathbf{CX} = 0,$$

where $\mathbf{C}$ is a matrix whose columns contain coefficients of the monomials, and $\mathbf{X}$ is the vector of monomials corresponding to the columns of $\mathbf{C}$. It should be noted that the ideal is closed under row operations on $\mathbf{C}$.

We will follow the method outlined in [14], which allows us to build the action matrix without constructing a Gröbner basis (we will still analyze the system and extract its Gröbner basis in the finite field for the purposes listed above). We briefly describe their method here. The key idea is to determine the so-called *solving basis* $\mathcal{B}$ (in our case, we use the monomial basis for the quotient ring), and the *required monomials* $\mathcal{R} = x_k\mathcal{B}\backslash\mathcal{B}$. Specifically, our objective is to find the minimum number of monomials needed to construct the action matrix, and then re-arrange the matrix such that those monomials, along with the basis monomials, occupy the last columns of the matrix. Using algebraic geometry software we can find a candidate linear basis $\mathcal{B}$ for the quotient space. For the action matrix corresponding to multiplication by x_k, the set of monomials that need to be expressed in terms of $\mathcal{B}$ is the set $\mathcal{R} = x_k\mathcal{B}\backslash\mathcal{B}$. The rest of the monomials in the system are called $\mathcal{E}$. The polynomial system with coefficients $\mathbf{C}$ can then be expressed as follows:

$$\mathbf{CX} = \begin{bmatrix} \mathbf{C}_{\mathcal{E}} & \mathbf{C}_{\mathcal{R}} & \mathbf{C}_{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0.$$

The only requirement on this coefficient matrix, after this matrix is put into the row-echelon form, is that its $|\mathcal{R}| \times |\mathcal{R}|$ submatrix corresponding to the monomials in $\mathcal{R}$ and the last $|\mathcal{R}|$ equations has full rank. This submatrix is called $\mathbf{C}_{\mathcal{R}2}$ in [14]. When we discuss our solution, we will illustrate how to use this matrix to extract the action matrix. The complete details are found in [14].

The initial set of equations F is unlikely to have a coefficient matrix $\mathbf{C}$ that meets the above requirement. This is where we will draw on the ideal members to expand the original system with additional equations, until the requirement on the action matrix construction is satisfied. The technique to generate ideal members efficiently proposed in [35] involves multiplying the original polynomials by monomials starting with the lowest orders. This operation, when applied to an equation (a row of $\mathbf{C}$), will result in the coefficients from that row to be shifted to the left in the matrix to take their places in columns corresponding to their new monomials. We will continue adding polynomials (checking for linear dependence and unneeded ones), until $\mathbf{C}$ is large enough to produce a full rank $\mathbf{C}_{\mathcal{R}2}$. We call the resulting set of polynomials (which are monomial multiples of the original system) an *elimination template*, and and matrix $\mathbf{C}$ an *elimination matrix*. This part of the process can be done with coefficients drawn randomly from $\mathbb{Z}/r$.

2.5.7 Example: Elimination Template Construction

In this section we present a complete example of the basic method from Byrod *et. al.* This particular example does not use the "solving basis" technique, where $\mathbf{X}_{\mathcal{B}}$ is larger than the basis for the quotient ring, but merely illustrates the basic flow of the algorithm. Let us assume that we have derived a polynomial system for our geometry problem and that it consists of three equations in three unknowns:

$$f_1 \equiv c_1 y + c_2 x^2 \tag{2.5.1}$$

$$f_2 \equiv c_3 z + c_4 x^3 \tag{2.5.2}$$

$$f_3 \equiv x + c_5 y + c_6, \tag{2.5.3}$$

where the coefficients c_i are given. In a computer vision geometry problem, the coefficients would come from feature locations, such as image or space points. This system is visualized in Figure 2.5 and consists of three ruled surfaces, a cubic, a quadratic and a plane. As we mentioned in Section 2.4.10, this system generates a zero-dimensional ideal and has two solutions, in general.

Given this system, we will construct a "solution template" which will solve it for x, y and z, given coefficients $c_i \in \mathbb{R}$, i.e. for any instance of the problem.

Let us once again verify that this system is zero dimensional and compute the number of solutions. To do this, we must compute the Gröbner basis of the ideal generated by f_1, f_2, f_3 using coefficients from a prime field $\mathbb{Z}_{30029}$ using Maple. We must do this multiple times using random coefficients in order to ensure that the structure of the basis is stable (the structure could be due to unwanted cancellations). The Gröbner basis is

$$y + 20018 + 20019z \tag{2.5.4}$$

$$x + 20020 + 20019z \tag{2.5.5}$$

$$z^2 + 30021 + 30022z. \tag{2.5.6}$$

From the basis we can see that the ideal is zero-dimensional (powers of all variables appear in $LT(I)$) and that there are two solutions in general (non-leading monomials span a two-dimensional vector space), as we saw in previous sections. We also observe that the basis monomials are $\mathbf{X}_{\mathcal{B}} = [z, 1]$. Let us use x as the action variable x_k above. The set of required monomials becomes $\mathbf{X}_{\mathcal{R}} = x\mathbf{X}_{\mathcal{B}} = [xz, x]$. Our goal is now to find a set of polynomials in the ideal $\langle f_1, f_2, f_3 \rangle$ that express the basis

monomials in terms of the required monomials:

$$\begin{bmatrix} I & -A \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0. \tag{2.5.7}$$

However, our original system does not have this form. Written in matrix form $C\mathbf{X} = 0$, the system is

$$\begin{bmatrix} 0 & c_2 & 0 & c_1 & 0 & 0 \\ c_4 & 0 & 0 & 0 & c_3 & 0 \\ 0 & 0 & 1 & c_5 & 0 & c_6 \end{bmatrix} \begin{bmatrix} x^3 \\ x^2 \\ x \\ y \\ z \\ 1 \end{bmatrix} = 0 \tag{2.5.8}$$

In order to gain the needed equations, we augment our original system with multiples of the original equations (since those belong to the ideal). This process must also be implemented in Maple. The steps of the template generation are as follows:

1. Add multiples of equations f_i to the system, starting with the lowest order (xf_i, yf_i, zf_i).

2. Generate the coefficient matrix C for the resulting system and fill in the coefficients from a prime field, such that $\{c_1, \ldots, c_6\} \in \mathbb{Z}_{30029}$.

3. Arrange the columns of the coefficient matrix such that $C = \begin{bmatrix} C_{\mathcal{E}} & C_{\mathcal{R}} & C_{\mathcal{B}} \end{bmatrix}$, as described in the previous section.

4. If, after Gaussian elimination, the system does not have $|\mathcal{R}|$ linearly independent equations of the form (2.5.7), repeat from Step 1.

5. If multiple equations were added in Step 1, remove equations one at a time and perform the Step 2 test.

6. Repeat Steps 1 through 4 several times with different coefficients to make sure that the results are not due to unwanted cancellations.

7. Once the smallest number of equations has been found and the stability has been verified in Step 4, output the C matrix of the resulting system, after substituting the symbolic coefficients c_i.

For the system in this example, the resulting equations are $\{f_1, f_2, f_3, f_1x, f_2x, f_3x, f_1x^2, f_3x^2\}$, from which we can construct the elimination template C_{elim} and monomial vector $\mathbf{X}_{\text{elim}}$:

$$C_{\text{elim}}\mathbf{X}_{\text{elim}} \equiv \begin{bmatrix} 0 & 0 & 0 & c_2 & 0 & c_1 & 0 & 0 & 0 & 0 \\ 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & c_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_5 & 0 & 1 & 0 & c_6 \\ 0 & c_2 & 0 & 0 & c_1 & 0 & 0 & 0 & 0 & 0 \\ c_4 & 0 & 0 & 0 & 0 & 0 & c_6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & c_5 & 0 & 0 & c_6 & 0 & 0 \\ c_2 & 0 & c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & c_5 & c_6 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^4 \\ x^3 \\ x^2y \\ x^2 \\ xy \\ y \\ xz \\ x \\ z \\ 1 \end{bmatrix}. \tag{2.5.9}$$

Notice that the basis monomials $\mathbf{X}_{\mathcal{B}}$ are the last two elements of the monomial vector (corresponding to the last two columns of the coefficient matrix) and the required monomials $\mathbf{X}_{\mathcal{R}}$ are the third and forth from last elements (corresponding to the third and forth from last columns of C).

We now have all the elements we need to solve the original system:

1. The elimination template matrix C_{elim}.

2. The list of basis monomials $\mathbf{X}_{\mathcal{B}} = [z, 1]$.

3. The list of required monomials $\mathbf{X}_{\mathcal{R}} = [xz, x]$.

Let us now demonstrate how to solve an instance of our problem using the template above by solving the system

$$f_1 \equiv 2.5y - 1.1x^2 \tag{2.5.10}$$

$$f_2 \equiv 2z - x^3 \tag{2.5.11}$$

$$f_3 \equiv x - 3y + 2. \tag{2.5.12}$$

$$\tag{2.5.13}$$

We must perform the following steps:

1. Populate the elimination template C_{elim} with given coefficients c_i:

$$\begin{bmatrix} 0 & 0 & 0 & -1.1 & 0 & 2.5 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3 & 0 & 1 & 0 & 2 \\ 0 & -1.1 & 0 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -3 & 0 & 0 & 2 & 0 & 0 \\ -1.1 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.5.14}$$

2. Perform numeric Gauss-Jordan elimination or LU decomposition on the resulting matrix. The results of Gauss-Jordan elimination are below.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2.6140 & -1.6650 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -2.0000 & 0.0000 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1.1501 & -0.7326 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -0.7252 & -1.0989 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -0.8800 & 0.0000 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -0.3191 & -0.4835 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1.3070 & -0.8325 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -0.9573 & 0.54945 \end{bmatrix} \tag{2.5.15}$$

3. Construct the action matrix from the bottom right 2×2 sub matrix, which is the negative of the action matrix:

$$A = \begin{bmatrix} 1.3070 & 0.83250 \\ 0.9573 & -0.54945 \end{bmatrix}. \tag{2.5.16}$$

4. Extract the eigenvectors of A:

$$\begin{bmatrix} 0.91800 \\ 0.39657 \end{bmatrix}, \begin{bmatrix} -0.35166 \\ 0.93612 \end{bmatrix} \tag{2.5.17}$$

5. The eigenvectors correspond to the scaled values of $\mathbf{X}_{\mathcal{B}} = [z, 1]^\top$, thus we can extract the value for z:

$$z = \{0.91800/0.39657, -0.35166/0.93612\} = \{2.31481, -0.37565\} \tag{2.5.18}$$

6. The values of x, which was our action variable, are the eigenvalues of A:

$$x = \{1.66667, -0.90909\} \tag{2.5.19}$$

7. The values of the remaining variable, y can be obtained by back-substituting the values of x and z:

$$y = \{1.22222, 0.36364\} \tag{2.5.20}$$

The above procedure makes it clear that the major computational steps of the process, as the size of the basis (number of solutions) and size of the template matrix C_{elim} rise are the Gaussian elimination of the template matrix and the eigendecomposition of the action matrix. In the next chapter, we use this method as the baseline and speed up the Gaussian elimination by constructing an equivalent but smaller elimination template.

2.5.8 Discussion

The Stewenius method set the guidelines for systematic construction of the elimination templates in computer vision. It was also the first method by which the optimal 3-view triangulation problem was revealed to have at most 64 solutions and solved. There are two main problems with this method, both of which are addressed in the next section: numerical stability, and the need to change the way the template is computed based on problem formulation.

This method does a good job of constructing a stable Gröbner basis for the problem by using the very stable Gauss-Jordan elimination. However, numerical stability of this method is not guaranteed because we have done nothing to ensure that the eigenvalue decomposition of the action matrix is numerically stable. In

fact, the implementation from the original paper has to rely on 128-bit floating point arithmetic for this computation.

That instability was addressed directly by the "solving basis" method described in [13].

The second problem is that the process of constructing an elimination template is largely manual for new problems. One has to look at the emerging basis and add polynomials if needed, and take out ones that are redundant. There is now a method in the literature to do this automatically for any zero-dimensional ideal [35], which will be presented in the next section.

2.6 Advanced Action Matrix Optimizations

In this chapter we present some of the state-of-the-art techniques aimed at improving numerical accuracy and automating the process of generating the elimination templates for computer vision problems.

2.6.1 A Gröbner-free Alternative

It turns out that the full Gröbner basis is not required for the computation of the action matrix. In this chapter we present methods that use this fact to construct more efficient elimination templates. As before, we analyze the given system of equations using coefficients from $\mathbb{Z}/p$ to obtain the Gröbner basis, and the basis $\mathcal{B}$ for the quotient ring A.

As we said before, the monomials in the basis will be the same for all instances of the problem, and only the coefficients will change. It was also noted in [13, 38] that the only elements of I needed to calculate the action matrix matrix m_f are in the set $\mathcal{R} = \{f \in I : LM(f) \in x_i \cdot \mathcal{B} \setminus \mathcal{B}\}$. These two facts allow us to compute the polynomials needed to compute the action matrix.

Given a basis $\mathcal{B} = \{\mathbf{x}^{\alpha(1)}, \ldots, \mathbf{x}^{\alpha(N)}\}$, the polynomials needed to compute the action matrix take the following form:

$$q_i = f\mathbf{x}^{\alpha(i)} + h_i, \tag{2.6.1}$$

where $h_i = \sum_{j=1}^{N} c_{ji}\mathbf{x}^{\alpha(j)} \in A$. Let the operator $T_f(f')$ be the image of f' under m_f. It was shown in [18] that to compute the action matrix, we need to compute the multiplications of the polynomial f by the basis members $T_f(\mathbf{x}^{\alpha(i)}) = \overline{f\mathbf{x}^{\alpha(i)}}^G$ for all $\mathbf{x}^{\alpha(i)} \in \mathcal{B}$.

We can pre-compute all columns of m_f corresponding to the monomials $\mathbf{x}^{\alpha(i)}$ for which $f\mathbf{x}^{\alpha(i)} \in A$. For the other monomials where $f\mathbf{x}^{\alpha(i)} \notin A$, we have to choose q_i from I such that $h_i \in A$. For the multiplication by these monomials, we have

$$T_f(\mathbf{x}^{\alpha(i)}) = \overline{f\mathbf{x}^{\alpha(i)}}^G = \overline{q_i - h_i}^G = -h_i \in A. \tag{2.6.2}$$

The polynomials q_i can be generated as combinations of the initial generators F under ideal operations (addition and multiplication by polynomials from $\mathbb{C}[x_1, \ldots, x_n]$). As a member of A, h_i must be of the form

$$h_i = \sum_{j=1}^{N} c_{ij}\mathbf{x}^{\alpha(j)} \tag{2.6.3}$$

with $c_{ij} \in \mathbb{C}$, and the coefficients c_{ij} are the corresponding elements of the action matrix. We must then construct our polynomials q_i to be of the form

$$q_i = f\mathbf{x}^{\alpha(i)} + \sum_{j=1}^{N} c_{ij}\mathbf{x}^{\alpha(j)} \tag{2.6.4}$$

for the specified subset of $\mathcal{B}$ with non-trivial remainders modulo I from polynomials in F.

We generate these polynomials by a method similar to the one we used in Chapter 2.5 to add polynomials to create a Gröbner basis, except this time we only generate the polynomials corresponding to the non-trivial columns in the action matrix.

2.6.2 Construction of an Optimized Elimination Template

We use the algorithm to construct q_i given in [35]. Given an initial set of polynomials, and the results of the analysis in $\mathbb{Z}/p$ (as described in the Chapter 2.5), including

the quotient ring basis $\mathcal{B}$, compute

1. Generate all monomial multiples $\mathbf{x}^\alpha f_i$ of degree $\leq d$ where $f_i \in F$.

2. Construct a matrix $C\mathbf{X} = 0$ of polynomials $\mathbf{x}^\alpha f_i$

3. Perform Gauss-Jordan elimination of C.

4. Test the form of all polynomials. If all polynomials q_i have been added, stop.

5. Increment d, and go to 1.

This algorithm can be used to construct the action matrices from the initial set F automatically.

2.6.3 Reducing Template Size

Adding polynomials blindly has the disadvantage of potentially adding many more polynomials than necessary. The method proposed in [35] deals with this problem by eliminating subsets of polynomials and using the sparse Gauss-Jordan elimination. If the wrong polynomials were removed, the resulting matrix will not contain polynomials q_i. Since we know the monomials in q_i from 2.6.1, we can check if the resulting system contains equations with these monomials with non-zero coefficients.

The same paper proposes a heuristic for removing subsets of polynomials. If we successfully remove k polynomials, we try to remove $2k$ polynomials on the next step. If we failed, we try to remove $\frac{1}{4}k$ polynomials.

2.6.4 Using Eigenvalues of Action Matrices

A approach described in Chapter 2.4 of Cox et. al [18] uses a single action matrix m_f to recover the solutions to the system as the left eigenvectors of m_f. However,

some solutions to the original system can be extracted from the eigenvalues of of m_f, and those eigenvalues correspond to the values of f on $\mathbf{V}(I)$ as well. So, computing eigenvalues of the n action matrices for $f_1 = x_1, \ldots, f_n = x_n$ also gives a complete set of solutions to the system (see Chapter 2.4). Byröd el. al [13] demonstrated that this method is more numerically stable, but at a small computational cost. Their observation is that since m_{f_i} are computationally simple to construct and share the same eigenvectors, we can quickly extract the eivenvalues of all m_{f_i} from an eigenvalue decomposition of m_{f_1} as follows. We perform the eigenvalue decomposition of $m_{x_1} = VD_{x_1}V^{-1}$. Then we can extract eigenvalues of m_{x_i} from the relation $m_{x_i}V = VD_{x_i}$ by matrix multiplication.

2.6.5 Discussion

In this chapter we presented some optimization techniques which improve numerical stability and accuracy of the basic Gröbner basis procedure.

The results presented in [35] show that the templates constructed using the automatic template generation techniques (Section 2.6.2) can be smaller than the corresponding Gröbner basis template, however, the computation times are unlikely to be low enough for real-time operation in a RANSAC framework for many larger problems, such as three-view triangulation.

The action matrices are only easy to construct if we compute the full Gröbner basis. We cannot extract an action matrix for an arbitrary f from the reduced basis computed via 2.6.2, which is what is required to use eigenvectors in Section 2.6.4. If we used N iterations of the reduced method, there may be a significant additional computational cost to gain the numerical advantage associated with the eigenvector method. However, this has not been evaluated in the literature.

2.7 Conclusions

The algebraic geometry solutions have found applications in geometric computer vision, however, none of the presented methods are currently in wide use if alternative formulations (such as closed-form solutions) are available owing to the high computational requirements. For example, the standard method for 3-view triangulation only requires an SVD of a 6×4 matrix, followed by local iterative refinement of the estimate. The results section of [65] compares this method to the Gröbner basis method, and finds accuracies to be similar. For the 5-point pose problem, the standard in efficiency is still the hand-derived method of [52], which does not require us to determine eigenvectors of an action matrix, but merely to find the real roots of a 10th degree polynomial in one variable (which can be done very fast using Sturm sequences). The Nister method has slightly lower accuracy, but the Frobenius norm of differences between real and estimated poses is still on the order of 10^{-10} instead of 10^{-13} for the Stewenius method.

There are, however, many problems for which alternatives are not currently available. These include various methods for estimating relative pose with unknown camera parameters (such as the 3D stitching with unknown focal length and radial distortion), as well as some problems for generalized cameras.

These methods can also be useful in the cases where just a few features can be localized and tracked very precisely, such as in augmented or virtual reality workstations and motion capture systems. In this case, solving for a minimal number of constraints with highest possible accuracy will be fast enough for real-time operation.

In the context of a hypothesize-and-test framework, the higher accuracy afforded by the methods presented is often not required, after all the answer still comes from only a few points, so it is rarely used as the final answer. Instead, an iterative

refinement which includes all inliers (as determined by RANSAC) is performed to improve the stability of the estimate using the minimal solution as the starting point. Since the minimal solution has to be computed many (sometimes hundreds) of times, speed is of the essence. When the presence of error in image feature localization is taken into account, price paid for the high accuracy of the existing Gröbner basis methods seems excessive. In the next chapter we will propose a method for further optimizing the techniques to yield not only additional computational benefits, but further improved numerical stability.

Chapter 3

Polynomial Solver Optimizations

3.1 The Need for Optimization

Since its introduction, several improvements to the action matrix method have been devised to address its numerical shortcomings [13] and ease of use [35]. While the action matrix method has been able to provide satisfactory (and in many cases the only) solutions to a range of problems, it still suffers from two main drawbacks: computational complexity and numerical accuracy. In this chapter we present a novel optimization of the state-of-the-art action matrix method by Byröd *et. al* [14] detailed in the last chapter in Section 2.5.6 with an example in Section 2.5.7 for solving geometry problems.

Once an elimination template consisting of C_{elim}, $\mathbf{X}_{\mathcal{R}}$ and $\mathbf{X}_{\mathcal{B}}$ for a problem has been constructed, the main computational steps of the action matrix approach are: matrix decomposition (such as LU or QR) of a polynomial coefficient matrix in order to express one set of monomials in terms of another and construct the action matrix, and an eigenvalue decomposition of the action matrix to find the solutions. The size of the eigenvalue problem is typically related to the number of solutions, so the

computational complexity of that part cannot be significantly reduced. The matrix used in the LU decomposition typically comes out of the structure of underlying problem and can be quite large. In this chapter we show that under some conditions, it is possible to reduce significantly the size of this matrix while guaranteeing that the algebraic structure of the problem is not affected. Since matrix decompositions are typically $O(n^3)$ operations, a reduction in its size produces a significant performance gain.

The numerical stability issues with the action matrix method are also well known. Several methods have been proposed to address this problem directly, such as using a redundant solving basis and basis selection by SVD or QR decomposition [13], but the underlying problem is the size of the LU or QR decomposition, and matrix conditioning issues associated with it. We will consider a specific problem of 3D panorama stitching with unknown focal length and radial distortion to demonstrate experimentally that after using our method to reduce the size of the matrix decomposition, numerical accuracy improves significantly. We show that the improved method is accurate even with single-precision arithmetic, which makes a fast implementation on a smartphone possible.

We will show the effectiveness of our approach on real imagery and demonstrate that in the case of RANSAC-based image stitching, our improved template can achieve good single-precision accuracy, while the state-of-the-art method fails to produce a solution due to round-off errors.

We first presented this work in [48].

3.2 The Current Action Matrix Method

We will briefly review how a polynomial template matrix is generated before proceeding to derive the conditions for its simplification. The method was described in detail in the last chapter and in [14, 36]). Let $F = \{f_1 ... f_n\}$ be a set of polynomials in variables $\mathbf{x} = x_1 \dots x_l$. The polynomials F generate an ideal I, and we assume there exists a finite-dimensional quotient space $\mathbb{R}[\mathbf{x}]/I$, thus the system F has a finite number of zeros. The aim of the action matrix method is to construct a matrix in the quotient ring space that multiplies polynomials by x_k, i.e. matrix A such that $x_k \mathbf{v}^\top \mathbf{X} = A^\top \mathbf{v}^\top \mathbf{X}_\mathcal{B}$, where $\mathbf{v}^\top \mathbf{X}_\mathcal{B} \in \mathbb{R}[\mathbf{x}]/I$ and $\mathbf{X}_\mathcal{B}$ is a monomial basis for $\mathbb{R}[\mathbf{x}]/I$. The solutions to $F = 0$ are extracted from the eigenvalues of A. Since this expression is valid for any polynomial in the quotient space we obtain

$$x_k \mathbf{X}_\mathcal{B} = A^\top \mathbf{X}_\mathcal{B}, \tag{3.2.1}$$

which suggests that we can construct the action matrix by finding a set of polynomial equations which express the monomials $x_k \mathbf{X}_\mathcal{B}$ in terms of the basis monomials.

We identify the *basis*, *required* and *extra* monomial subsets. The basis monomials $\mathcal{B}$ constitute a linear basis for the quotient ring $\mathbb{R}[\mathbf{x}]/I$ (although $\mathcal{B}$ can include other monomials when the *redundant solving basis* method is used, see [14], Section 4), and the required monomials are $\mathcal{R} = x_k \mathcal{B} \setminus \mathcal{B}$. The *extra* monomials $\mathcal{E}$ are the remaining monomials.

The solver generation usually proceeds as follows: the initial polynomial system F is extended with additional equations from the ideal $I = \langle F \rangle$. These equations are found by multiplying the equations from F by monomials, starting with lowest

order. The final system has the following form

$$C\mathbf{X} = 0, \tag{3.2.2}$$

where C is a matrix of polynomial coefficients of size $n \times m$ and $\mathbf{X}$ is the vector of all monomials in the extended system.

We rearrange the columns of the system as follows

$$C\mathbf{X} = \begin{bmatrix} C_{\mathcal{E}} & C_{\mathcal{R}} & C_{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0. \tag{3.2.3}$$

If a sufficient number of polynomials was added to the original system F, the action matrix can be found by performing LU decomposition of C. The starting point for our optimization is such a system with basis, required and extra monomials identified. We refer to this as an *elimination template*.

3.3 The Conditions for Template Simplification

First, we show under what circumstances we can separate the system $C\mathbf{X}$ into two sets of columns and stay in the ideal after elimination of one of the sets. This is a technical condition required for Lemma 2. We denote by $C_{\setminus j}\mathbf{X}_{\setminus j}$ the system $C\mathbf{X}$ with column $\mathbf{c}_j$ and the corresponding monomial $\mathbf{x}_j$ removed.

Lemma 3.3.1. *Let C be a full-rank matrix of polynomial coefficients and $\mathbf{X}$ a vector of monomials, such that $C\mathbf{X}$ is a set of n polynomials which generate an ideal I. Let $\mathbf{c}_j$ be a column of C. Let $C_{\setminus j} = LU$ be the LU decomposition of $C_{\setminus j}$. Then the*

polynomials

$$U\mathbf{X}_{\setminus j} + L^{-1}\mathbf{c}_j\mathbf{x}_j \tag{3.3.1}$$

are also in I.

Proof. The matrix L^{-1} is a product of elementary lower triangular matrices (see [44] p.142), and thus the transformation L represents only row operations. Since applying row operations to C will keep the equations in the ideal, the following statements are true:

$$\begin{aligned} C\mathbf{X} &\in I \\ C_{\setminus j}\mathbf{X}_{\setminus j} + \mathbf{c}_j\mathbf{x}_j &\in I \\ LU\mathbf{X}_{\setminus j} + \mathbf{c}_j\mathbf{x}_j &\in I \\ L^{-1}(LU\mathbf{X}_{\setminus j} + \mathbf{c}_j\mathbf{x}_j) &\in I \\ U\mathbf{X}_{\setminus j} + L^{-1}\mathbf{c}_j\mathbf{x}_j &\in I, \end{aligned} \tag{3.3.2}$$

which is the needed result. □

This lemma will be used to show that the equations stay in the ideal even after certain columns are removed. We now show that given C, it may be possible to compute the LU decomposition of a matrix smaller than C. We will now state and prove a condition under which we can remove a row and a column of C and still maintain the properties needed for successful elimination.

Lemma 3.3.2. *If there exists a column $\mathbf{c}_j$ of the polynomial coefficient matrix C that satisfies the following two properties:*

1. *It corresponds to one of the extra monomials $\mathbf{X}_{\mathcal{E}}$, i.e. $\mathbf{c}_j \in C_{\mathcal{E}}$*

2. *It can be expressed as a linear combination of the first $m - |\mathcal{R}|$ columns*

then the matrix $C_{\setminus j}$ can be used to expressing the required monomials $\mathcal{R}$ in terms of basis monomials $\mathcal{B}$ via its LU decomposition.

Proof. The need for property 1 is clear since discarding a column corresponding to either a required or basis monomial will result in the corresponding terms missing from the equations.

To show the sufficiency of property 2, let us move $\mathbf{c}_j$ to the right-hand side in the original system:

$$\begin{bmatrix} C_{\mathcal{E}\setminus j} & C_{\mathcal{R}} & C_{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}\setminus j} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = -\mathbf{c}_j \mathbf{x}_j. \tag{3.3.3}$$

After LU factorization of the left-hand side matrix $C_{\setminus j} = LU$ the system becomes

$$L \begin{bmatrix} U_{\mathcal{E}\setminus j} & C_{\mathcal{R}1} & C_{\mathcal{B}1} \\ 0 & U_{\mathcal{R}} & C_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}\setminus j} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = -\mathbf{c}_j \mathbf{x}_j, \tag{3.3.4}$$

where $U_{\mathcal{E}\setminus j}$ and $U_{\mathcal{R}}$ are upper-triangular matrices. Multiplying by L^{-1} from the left, we obtain

$$\begin{bmatrix} U_{\mathcal{E}\setminus j} & C_{\mathcal{R}1} & C_{\mathcal{B}1} \\ 0 & U_{\mathcal{R}} & C_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}\setminus j} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = -L^{-1}\mathbf{c}_j \mathbf{x}_j. \tag{3.3.5}$$

Where the polynomials formed by the rows belong to the original ideal by Lemma 1.

Since we need exactly $|\mathcal{R}|$ equations where the basis monomials are expressed in terms of required monomials, the last $|\mathcal{R}|$ elements of the vector $L^{-1}\mathbf{c}_j\mathbf{X}_{\mathcal{E}j}$ must be 0 (then the corresponding equations will not include any terms with $\mathbf{x}_j$). Let us show

that this is implied by property 2.

Let us rewrite property 2 as

$$\mathbf{c}_j = a_1\mathbf{c}_1 + ... + a_{j-1}\mathbf{c}_{j-1} + a_{j+1}\mathbf{c}_{j+1} + ... + a_{m-|\mathcal{R}|}\mathbf{c}_{m-|\mathcal{R}|}, \tag{3.3.6}$$

where $\mathbf{c}_k$ are the first $m - |\mathcal{R}|$ columns of $C_{\setminus j}$. The lower triangular matrix L^{-1} acts on $\mathbf{c}_k$ as follows: it annihilates elements below k (since $L^{-1}C_{\setminus j}$ is an upper triangular matrix). Thus the right hand side of the equation is

$$L^{-1}\mathbf{c}_j\mathbf{x}_j = a_1 \begin{bmatrix} \bullet \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{x}_j + \ldots + a_{m-|\mathcal{R}|} \begin{bmatrix} \bullet \\ \vdots \\ \bullet \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{x}_j, \tag{3.3.7}$$

where the last vector has zeros in the last $|\mathcal{R}|$ positions. □

The above lemma allows construction of a smaller matrix on the left-hand side of the system (3.2.2) by moving terms to the right-hand side and guarantees that the right-hand side will be zero in the last $|\mathcal{R}|$ equations.

While removing whole rows (equations) will keep the remainder in I, the ideal generated by the resulting set will be smaller if the lowest order generators are removed. We can now define excess columns as well as excess row-column pairs, removal of which will not affect the elimination.

Definition 3.3.3. An *excess column* $\mathbf{c}_j$ of C is a column that satisfies Lemma 2, and an *excess pair* of C is a row $\mathbf{r}_i$ of C and a column $\mathbf{c}_j$, such that $\mathbf{c}_j$ is a column

of C with $\mathbf{r}_i$ removed that satisfies Lemma 2.

The proposed optimization of elimination templates will make use of the Lemma 2 to find excess columns and pairs. At this point it is clear how we should proceed: we will first test each column to see if it can be safely moved to the right-hand side of the equation $C\mathbf{X} = 0$ and then, when no excess columns are found, we test all row-column combinations. The progressively smaller matrix will retain the full left and right hand sides, such that at any point all the equations in it are in the ideal. Since we will be operating with specific instances of the problems (symbolic elimination is impossible for most templates), we will have to verify the resulting template thoroughly for numerical stability and repeat the process as necessary to achieve a good template.

We now proceed to illustrate this process step-by-step with an example.

3.4 Example: Three-point Panorama Stitching

As an example, we demonstrate how to reduce the size of the elimination template for the problem of estimating panoramic stitching parameters in the case of unknown focal length and radial distortion [11]. In this problem we estimate the rotation between two views and the common focal length and radial distortion coefficient. The formulation of this problem as a polynomial system was presented in the previous chapter and the details can be found in the original paper as well as in Section 2.3.3 of this dissertation. As part of this work, we implemented the optimization as a modification of an existing implementation available from Lund University. [1]

[1] The original code is found at http://www.maths.lth.se/vision/downloads/data/stitching3pt.zip. Our optimized code is at http://www.cis.upenn.edu/~narodits/Site_3/Solver_Optimization.html.

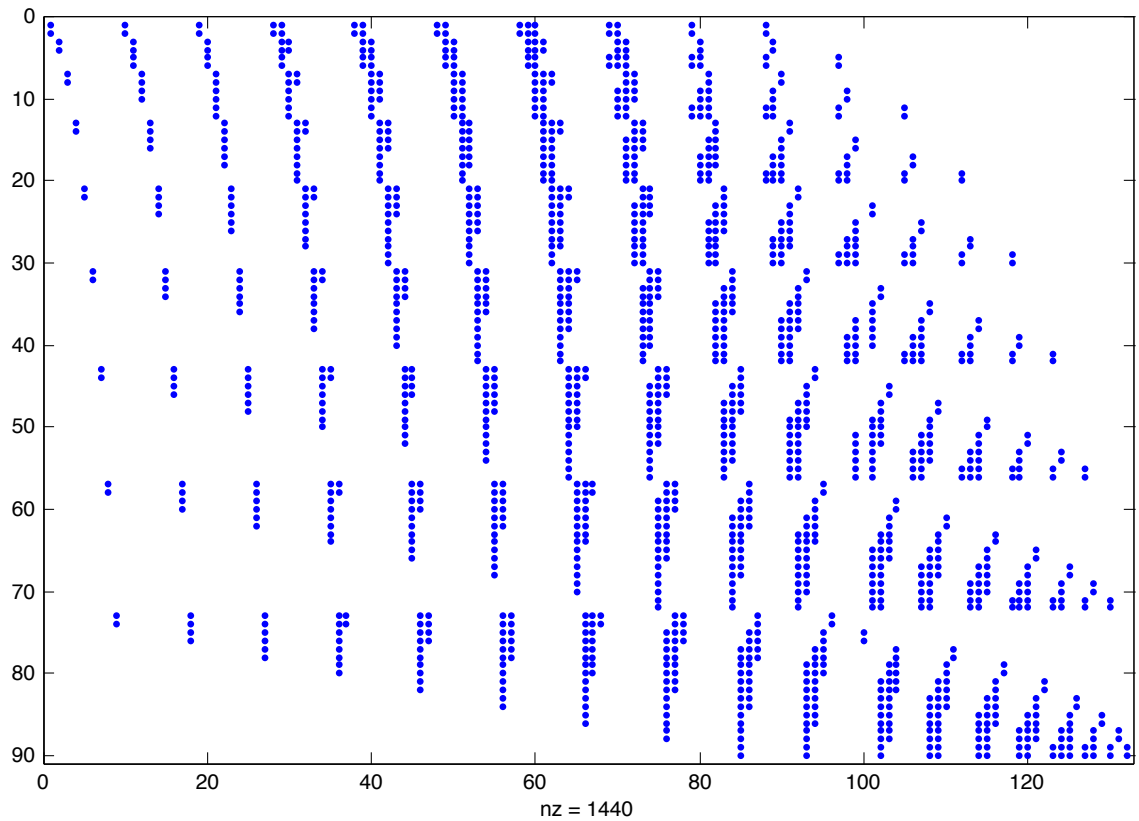


Figure 3.1: Structure of the template as described in [11].

3.4.1 Eliminating Excess Columns

The template for this problem consists of 90 equations in 132 monomials, and is solved using the redundant solving basis method and QR decomposition of a 90×100 matrix in order to create a numerically stable basis. The structure of the template is shown in Figure 3.1. It is arranged such that $C = \begin{bmatrix} C_{\mathcal{E}} & C_{\mathcal{R}} & C_{\mathcal{B}} \end{bmatrix}$, where $|\mathcal{E}| = 100$, $|\mathcal{R}| = 7$ and $|\mathcal{B}| = 25$. Thus the system has 25 solutions and will be solved via eigenvalue decomposition of a 25×25 action matrix.

We will now show how to use the results in Section 3.3 to reduce the size of this template. One way to identify excess columns present in the initial template is via Gaussian elimination with partial pivoting. The eliminated matrix has the

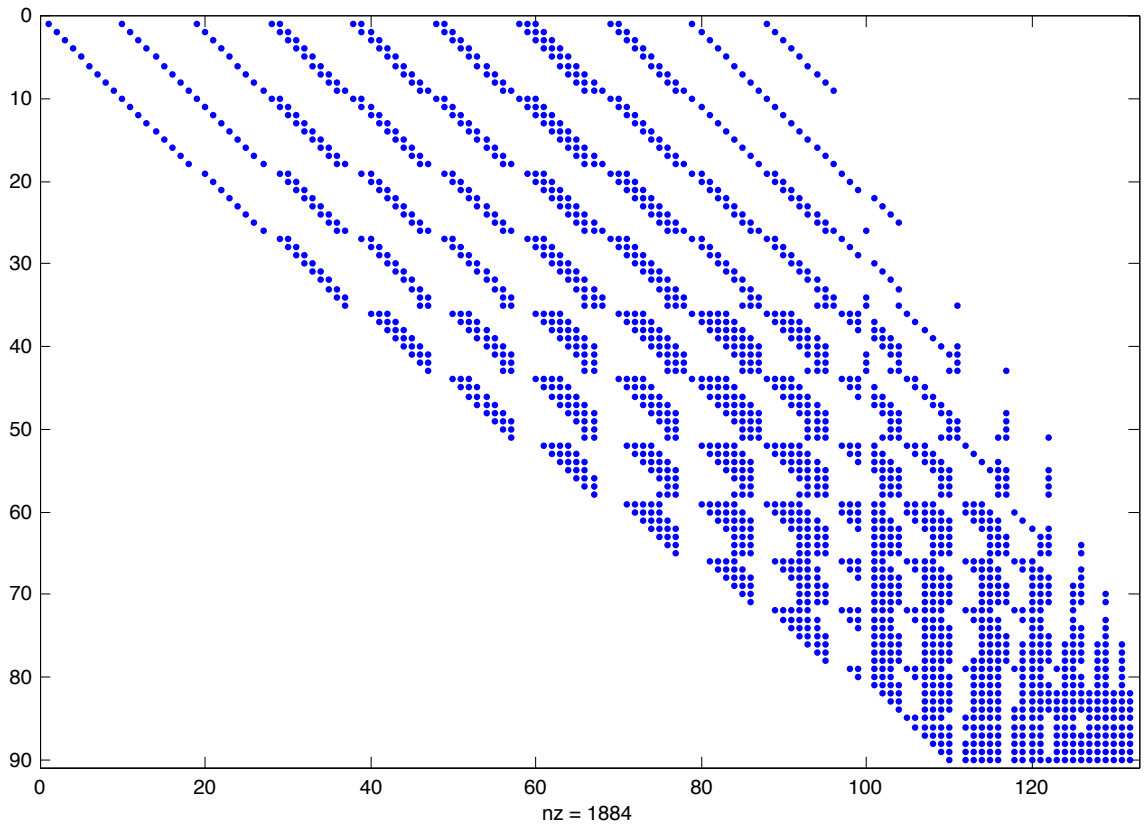


Figure 3.2: Structure of elimination template after Gaussian elimination.

structure shown in Figure 3.2. It is important to note at this point that since symbolic elimination of such a large matrix is not practical, the matrix structure above is from a "typical" numeric example. While cancellation pattern and structure of a symbolic Gaussian elimination will always be the same (since the relationships between all matrix entries are fixed), a numeric elimination may encounter problems with precision (especially in the lower right part of the matrix), and thus give different structure for different instances of the problem. We must try many instances until a stable row-echelon form of C is found.

In the structure above we can immediately observe columns which have zero pivots. These columns, namely $[19, 28, 38, 39, 48, 49, 58, 59, 60, 69, 70, 79, 80, 88]$, correspond to excess columns and can each be expressed as a linear combination of the

previous, extra columns, and thus by Lemma 2, they can be eliminated from the matrix. At this point the template matrix is 118×90.

3.4.2 Eliminating Excess Pairs

For the second round of optimization, we will find excess pairs by testing each row and column. This is accomplished by systematically removing a row of C, and for each column corresponding to an $\mathcal{E}$ monomial, computing the vector $L^{-1}\mathbf{c}_j x_j$, as in equation (3.3.5) and checking if the $\mathcal{R}$ monomials have zero coefficients. If the number of zeros is greater than or equal to $|\mathcal{R}|$, then the row is removed and the column is moved to the right-hand side. We repeat this process on the resulting left and right hand side matrices until a smallest size left-hand side matrix is found. Once again, the numerical stability of this process is an issue, and a sufficient number of instances must be tried in order to ascertain the quality of the resulting template. This optimization round allowed us to create a stable template of size 54×77 with QR decomposition being performed on a 54×45 matrix. The structure of the optimized template is shown in Figure 3.3. We now list the complete set of excess rows and columns which can be used to reproduce our results. The original 90×132 matrix is constructed in the function setup_3pt.m of the aforementioned code. After arranging the matrix as $C = \begin{bmatrix} C_{\mathcal{E}} & C_{\mathcal{R}} & C_{\mathcal{B}} \end{bmatrix}$, we remove the columns $[1\ldots5,\ 9\ldots14,\ 18\ldots23,\ 27\ldots32,\ 37\ldots42,\ 47\ldots52,\ 57\ldots62\ 68\ldots72,\ 78\ldots80,$ $88]$ and rows $[1\ldots30,\ \ 73\ldots78]$. After this, the elimination proceeds as before, except we must adjust the indices `m_ind` to compensate for a smaller matrix.

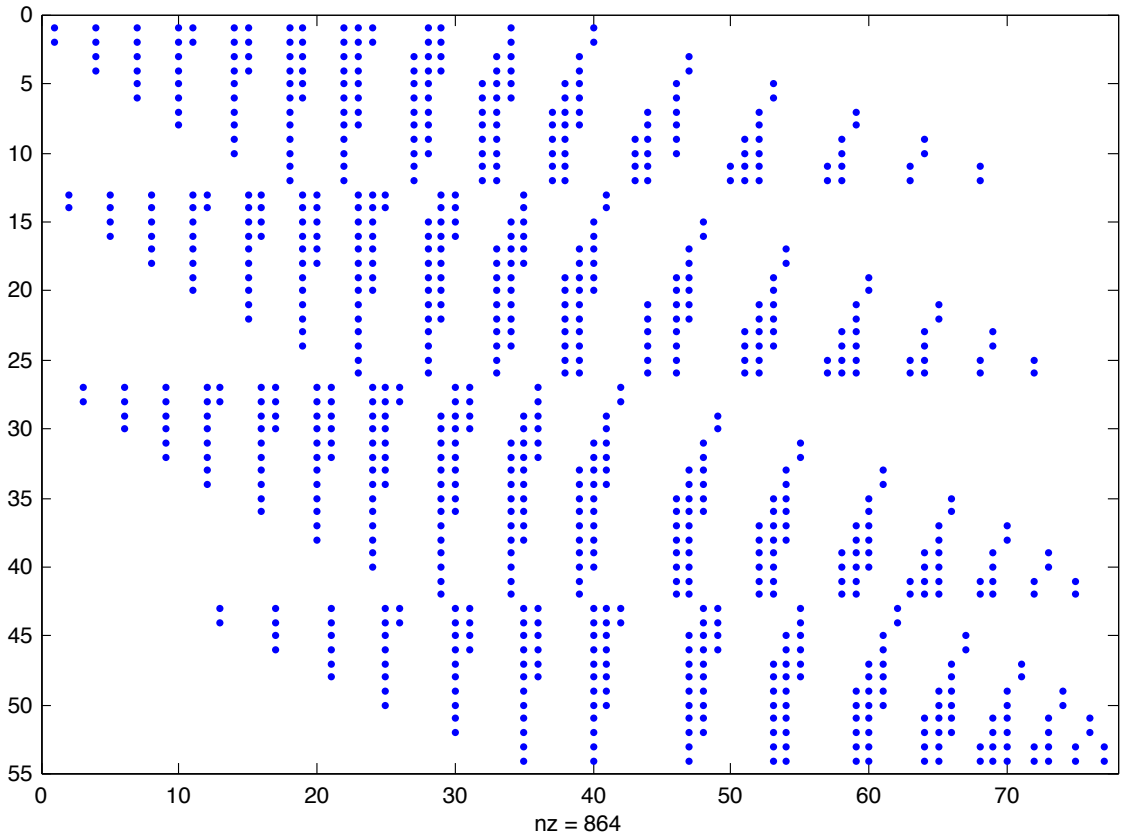


Figure 3.3: Structure of the template after optimization.

3.4.3 Numerical Stability

The numerical stability of the resulting template is verified in Figures 3.4 and 3.5. We observe that the performance is almost identical in both the noise-free and noisy cases on a large set of randomly generated configurations. In terms of computational performance, the average QR decomposition time decreased from 1.1ms to 0.24ms, which translates into a doubling of overall performance when matrix inversion and eigenvalue decomposition are taken into account.

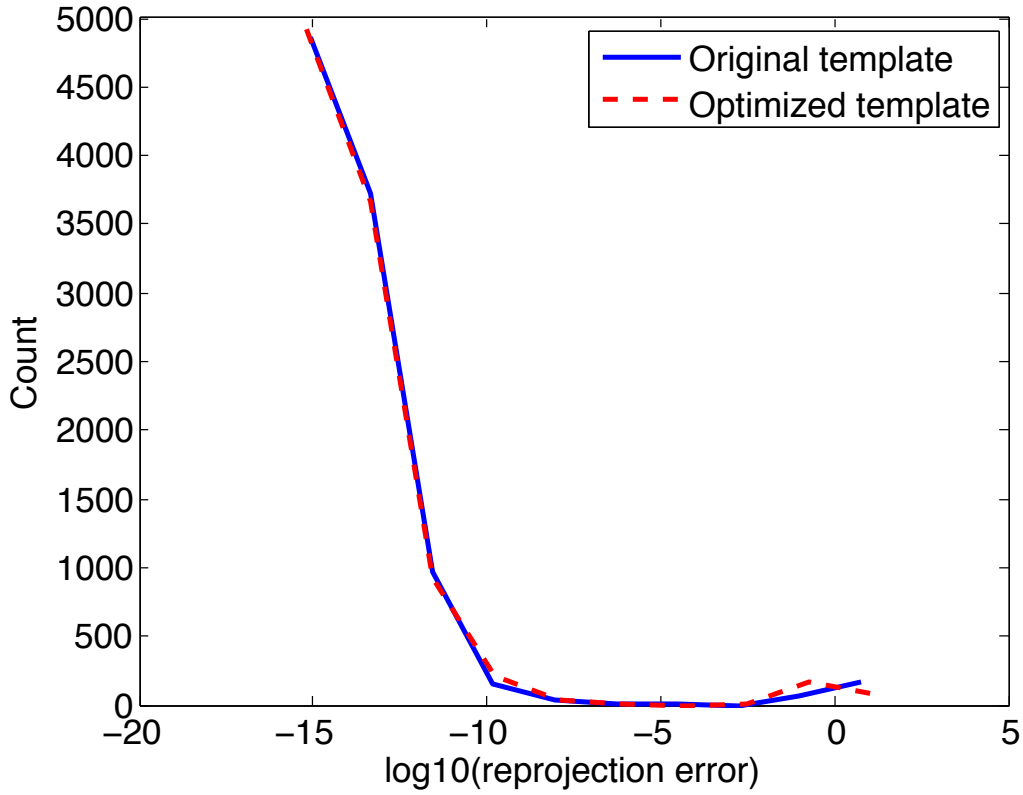


Figure 3.4: Noise free, double precision experiment for the three-point stitch problem. Comparison of the orders of magnitude of reprojection errors between solutions to 10^5 noise-free instances. Reduced template is dashed red and the original template is solid blue.

Aside from speed, the other significant advantage of our new template is its numerical stability under single precision arithmetic. Due to the smaller size of our

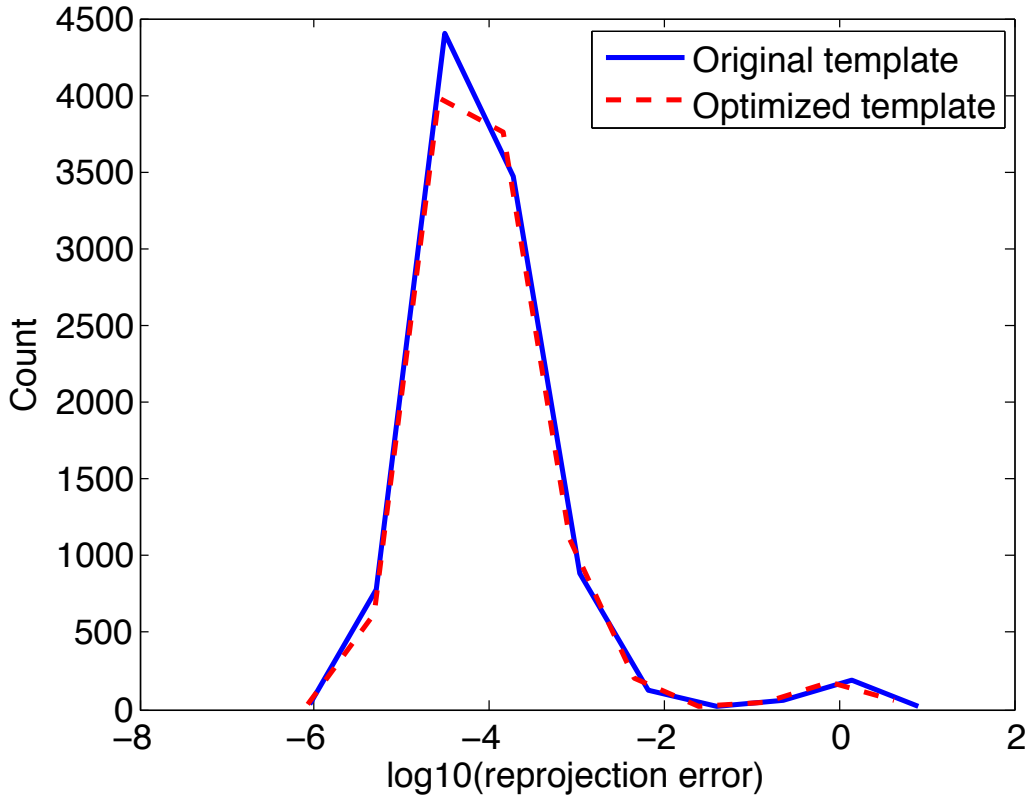


Figure 3.5: Noisy, double precision experiment for the three-point stitch problem. Comparison of the orders of magnitude of reprojection errors between solutions to 10^5 instances. The points are contaminated with 0.01 standard deviation noise in the normalized image plane. The dashed, red plot was generated with the reduced template and solid, blue with the original. The solutions obtained by both templates are very close, which shows that we did not compromise numerical stability of the algorithm by reducing the size of the template.

template, the round-off errors are not as significant. The comparison of the two templates in single precision is shown in Figure 3.6. The original template is not usable in single-precision arithmetic with a 4300 of 10000 cases failing to produce the correct solution. In contrast, the reduced template failed in only 224 cases. The single precision implementation used single precision for QR decomposition, matrix division and eigenvalue decomposition. It is clear that for the original template, it is the 90×100 QR decomposition that turned out to be unstable.

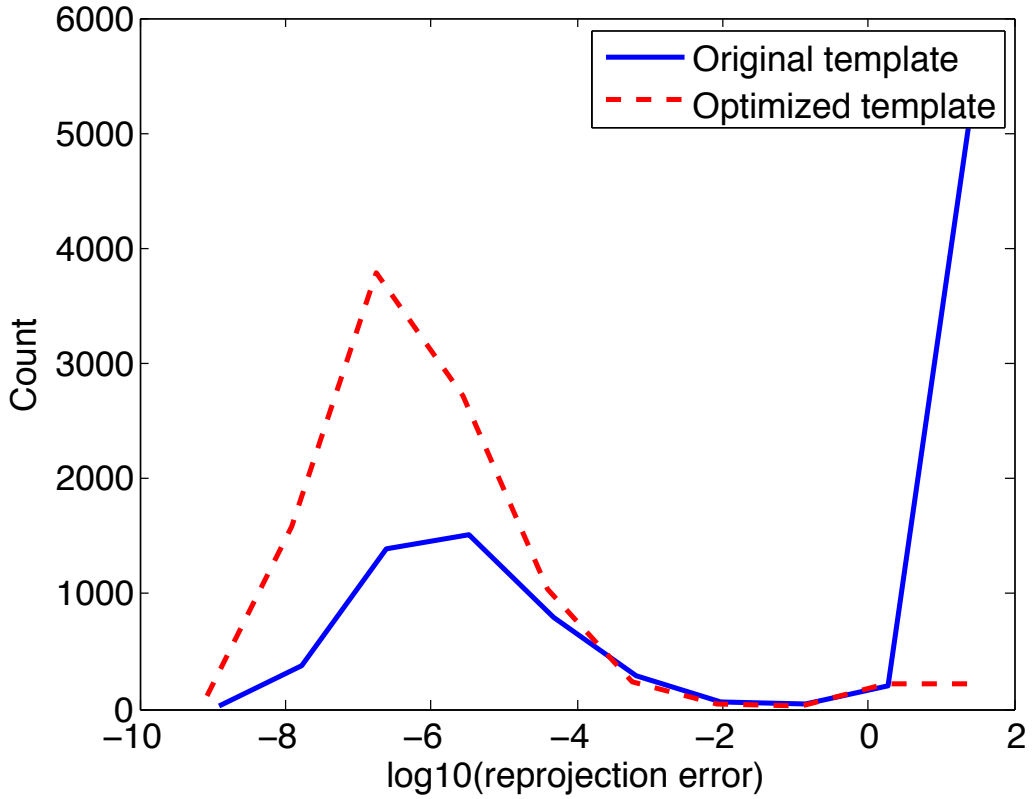


Figure 3.6: A noise-free, single precision experiment for the three-point stitch problem. The graph shows the comparison of the orders of magnitude of reprojection errors between solutions to 10^5 instances of the problem. The dashed, red plot was generated with the reduced template and solid, blue with the original. This plot demonstrates that only our reduced template can be used with single precision since in about 4300 cases the original template fails completely.

3.4.4 Experimental Results

Finally, we verify performance of the reduced template on real images. In Figures 3.7 and 3.8 we show the panoramas generated with our new template with single precision arithmetic. The procedure to generate these results was was follows: 1) detect SIFT [41] features on the two images and match them, 2) run a RANSAC process on the subsets of the matches and output the best hypothesis, and 3) warp the images using the hypothesis.

On the feature points shown, both double precision implementations produced similar results, however, the single precision implementations differ dramatically. Of the 300 generated hypotheses, the original template failed to find any solutions in 297 cases (the remaining 3 cases gave wrong solutions) and the reduced template only had 43 such failures (for the images in Figure 3.8).

Overall, we can conclude that the reduced template is stable and can be applied to real-world images, even on single precision hardware, such as a smartphone.

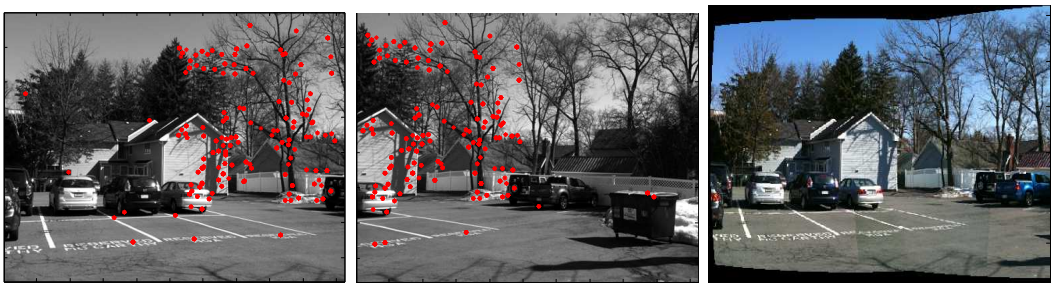


Figure 3.7: The matched SIFT features on the cell phone images and the resulting panorama generated with single precision arithmetic using the reduced template and RANSAC. The performance was similar for the double precision in both templates, however, the single precision original template did not produce a valid hypothesis after 200 iterations. No bundle adjustment or image blending was applied, hence the panorama is generated with a single three-point hypothesis.

Figure 3.8: The matched SIFT features on high quality images and the resulting panorama generated with single precision arithmetic using the reduced template and RANSAC.

3.5 Example: Optimal Three-view Triangulation

The problem of L2-optimal, three-view triangulation is a classic problem that is solved by the action matrix method. First solved by Stewenius *et. al* in [65] in 256-bit arithmetic, this problem was revisited by Byrod *et. al* [12] and solved with double precision arithmetic. With its 47-dimensional linear basis, this problem is difficult to optimize, however, we still manage to improve performance slightly. We base our implementation on the freely available version from Byrod. [2]

3.5.1 Polynomial Model

As we mentioned in Chapter 2, the optimal three-view triangulation problem is not a minimal problem (minimal number of views is 2), but it is nonetheless a classic problem solved by the action matrix method. We briefly review the model here. Given three camera matrices $\mathbf{P}_i = [R_i \ t_i]$, and three corresponding image points $\mathbf{u}_i$, the 3D point $\mathbf{U} = [u_1, u_2, u_3]^\top$ must be computed such that it minimizing the objective function defined as the sum of squared reprojection errors:

$$F(\mathbf{U}) = \sum_{i=1}^{3} d(\mathbf{P}_i\mathbf{U}, \mathbf{u}_i)^2. \tag{3.5.1}$$

The problem is simplified by setting all image points $\mathbf{u}_i$ to be at the origin, and adjusting the camera matrices to compensate. This is done by rotating them around the camera's center of projection to get $\mathbf{P}'_i = [R_{\mathbf{u}_i} R_i \ t]$, where $R_{\mathbf{u}_i} \in SO_3$ is the rotation between $\mathbf{u}_i$ and $[0, 0, 1]^\top$ in the camera coordinate system. The objective is now $F(\mathbf{U}) = \sum_{i=1}^{3} F_i(\mathbf{U})$, where reprojection errors $F_i(\mathbf{U})$ are given in terms of the

[2] The code is at http://www.maths.lth.se/vision/downloads/data/optimal_tvt.zip

rows of the camera matrices $\mathbf{P}'_i = [P_i^1\ P_i^2\ P_i^3]^\top$:

$$F_i(\mathbf{U}) = \frac{(P_i^1\mathbf{U})^2 + (P_i^2\mathbf{U})^2}{(P_i^3\mathbf{U})^2} \tag{3.5.2}$$

which is the squared length of the image vector after re-projection.

Optimality requires that $\frac{dF}{du_i} = 0$ for each component u_i of $\mathbf{U}$. After differentiating with respect to u_i. When $i \neq j$, the derivative is [63]

$$\frac{dF_i}{du_j} = 2\frac{P_i^1(j)P_i^1\mathbf{U} + P_i^2(j)P_i^2\mathbf{U}}{u_j^2}, \tag{3.5.3}$$

where $P_i^k(j)$ is the jth element of P_i^k. When $i = j$, we have

$$\frac{dF_j}{du_j} = 2\frac{(P_j^1(j)P_j^1\mathbf{U} + P_j^2(j)P_j^2\mathbf{U})u_j - (P_j^1\mathbf{U})^2 + (P_j^2\mathbf{U})^2}{u_j^3} \tag{3.5.4}$$

When these derivatives are written out in terms of u_j for $j = 1 \ldots 3$ and brought under the same denominator to form polynomial equations, we end up with 3 equations of degree 6.

3.5.2 Template Optimization

After the addition of a sufficient number of equations, the elimination template for this problem has size 225×209. We reduced the size to 154×204. We begin by rearranging the columns of the coefficient matrix such that $C = \begin{bmatrix} C_{\mathcal{E}} & C_{\mathcal{R}} & C_{\mathcal{B}} \end{bmatrix}$. We then remove rows, one-by-one and test the template for stability, as suggested

in [35]. This allows us to remove rows

$$[1 \ldots 43, 49, 122, 124, 125, 127, 128, 130,$$
$$133, 167, 169, 170, 171, 172, 199, 200, 202, 204,$$
$$206, 209, 210, 213, 214, 218, 222].$$

We now apply our method to remove five additional excess pairs from the resulting matrix. We remove rows [21 , 79 , 80 , 81, 117], and columns [1, 2, 3, 55 , 56]. The number of pairs removed is only 5, however, this problem has 50 basis monomials and 31 required monomials, so the task of choosing Lemma 2 compliant columns is difficult. It is still possible that a smaller template lurks inside this one, but finding it might be a difficult combinatorial problem.

We use simulated data to show once again that this removal has little effect on the results in double precision arithmetic. We plot the results in Figures 3.9 and 3.10. Single precision appears to be far out of reach for this problem.

3.6 Conclusions

In this chapter we developed a new method for reducing the size of the action matrix template and proved some properties of the columns of the template matrix. We showed that under some conditions, rows and columns of the matrix can be removed a priori, resulting in improved speed.

We produced a real example of algorithms which benefits substantially from our approach both in terms of speed and numerical stability, namely the three-point panoramic stitching and the "3+1" algorithm. We believe that this generic approach can be used on other algorithms as well, making the action matrix method an even

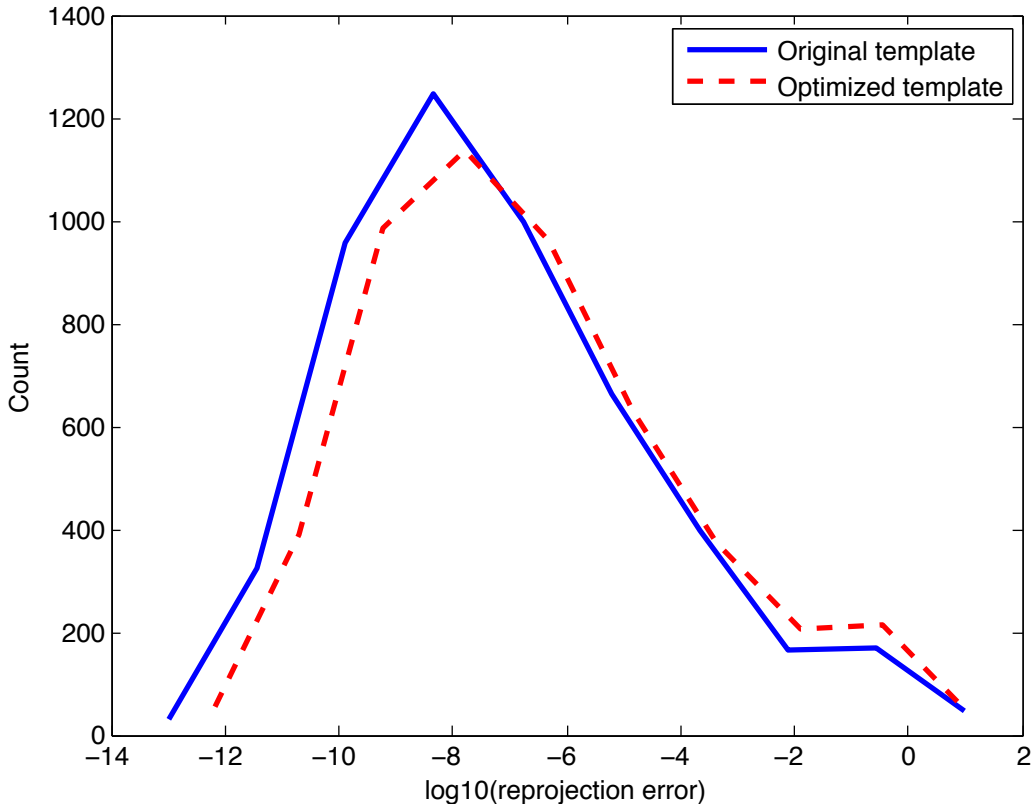


Figure 3.9: Optimal three-view triangulation experiment with no noise. We compare the orders of magnitude of error in triangulated 3D point $\log_{10}(\|X - \hat{X}\|_2)$. We used the standard basis (tvt_solve_std.m script), and not the QR method. Note that due to the nature of the problem, the error is much higher than in the case of 3D panorama stitching.

more attractive way to solve geometry problems in computer vision.

As a possible further improvement to this optimization, the method can be applied without relying on potentially numerically deficient examples, as we did in this here. By applying Lemmas 1 and 2 at the template generation stage with random coefficients drawn from $\mathbb{Z}/p$, it may be possible to produce an exact solution.

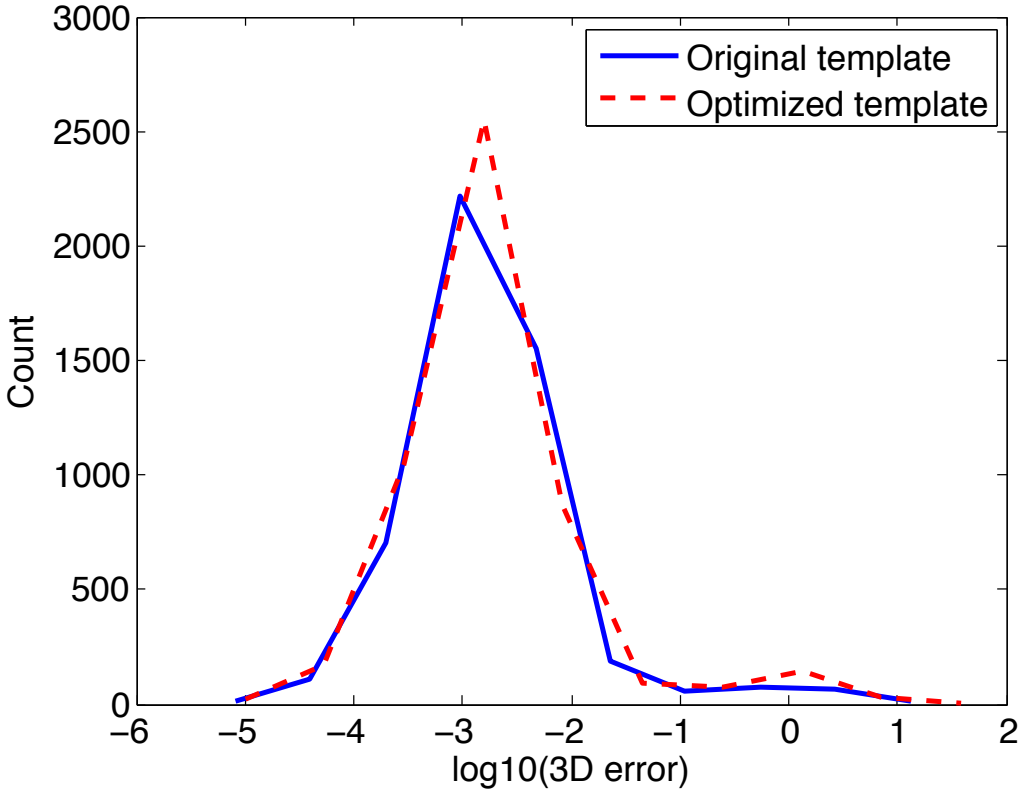


Figure 3.10: Optimal three-view triangulation experiment with noise. We compare the orders of magnitude of error in triangulated 3D point $\log_{10}(\|X - \hat{X}\|_2)$ between the two templates. The standard deviation of the noise in the points in camera coordinates was 0.01. The results are for the standard basis (tvt_solve_std.m script), and not the QR method. The results are again very similar despite being computed with a smaller template.

Chapter 4

Structure from Motion Using Directional Correspondence

4.1 Introduction

Data association has been identified as one of the two main challenges in visual odometry next to observation noise (see special issue to the workshop [1]). Cluttered environments with independently moving objects yield many erroneous feature correspondences which have to be detected as outliers. It has been shown [56] that Random Sample Consensus (RANSAC) provides a stable framework for the treatment of outliers in monocular visual odometry. For RANSAC it is highly desirable to have a hypothesis generator that uses the minimal number of data points to generate a finite set of solutions, since this minimizes the probability of choosing an outlier as part of the data. For example, in minimal cases, absolute pose estimation requires three correspondences between the world and image points, and relative pose requires five image to image correspondences. In this chapter we propose a new minimal method for computing relative pose for monocular visual odometry that uses

three image correspondences and a common direction in the two camera coordinate frames, which we call a "directional correspondence". We call this the "three-plus-one" method. The main motivation for using the three-plus-one method is to enable visual odometry using RANSAC with a four-point minimal solver (instead of the traditional five), as long as the fourth point is at infinity (and thus provides a directional correspondence). However, with the advent of robots and mobile devices with inertial measurement units (IMU), the three-plus-one algorithm becomes an attractive way to solve the visual odometry problem where the directional correspondence is provided by an IMU (e.g. the gravity vector).

The main goal in this chapter is to introduce two efficient algorithms for the three-plus-one problem. It has been known [71] and it is straightforward to deduce that the knowledge of the directional correspondence reduces the number of rotation unknowns to one, yielding a system of three quadratic equations in three unknowns. In Section 4.3, we show how to formulate the full relative pose problem as a system of four polynomial equations. We then present two methods for solving this system. The first method is a direct, closed-form solution, leading to a quartic polynomial of one variable, and is found in Section 4.4. Before introducing the second method, we give a brief review of the algebraic geometry techniques in Section 4.5 and a more complete treatment was found in Chapter 2. The method based on these techniques, and specifically on the "action matrix" method from Byrod *et al* [14], is presented in Section 4.6. In this case, eigenvalue decomposition of a 4×4 matrix recovers three of the four variables simultaneously.

Our second goal is to investigate the properties of both solutions, exploring performance under noise and adverse imaging conditions. In Section 4.8, we demonstrate that both methods solve the problem correctly and highlight the differences in

numerical properties and computational requirements. Since we envision that this algorithm may be implemented on mobile devices and low-power CPUs that frequently lack the hardware to process double precision (64-bit) floating point numbers, we investigate the numerical properties of both solutions in single and double precision arithmetic, and show that except for specific imaging conditions, both algorithms are stable in both implementations.

Our third goal is to demonstrate that the three-plus-one method can be used in place of the five-point method in visual odometry applications. To that end, we compare the accuracy and the computational requirements of the five-point algorithm with the closed-form algorithm for the three-plus-one problem using simulated data, and show it to be superior in estimating camera rotation in noisy conditions without sacrificing translation estimation accuracy.

The final goal is to demonstrate that the three-plus-one method is a reliable alternative to the five-point method in a real-world visual odometry application, if an ample supply of distant points is present (such as in outdoor environments). In Section 4.9.1, we use our method in a RANSAC framework to compute monocular visual odometry on outdoor video data. When used with RANSAC, our visual odometry does not require any knowledge about which points are at infinity, because we simply let RANSAC choose the inlier hypothesis from all available image correspondences. We will show that since we only require four correspondences, our method leads to more robust visual odometry than the five-point method. Moreover, in cases where only few nearby point correspondences can be found, having it as another hypothesis generator reduces the probability of failure of a five-point-based navigation algorithm. In Section 4.9.2, in order to demonstrate the potential of our method for vision-inertial fusion, we present the results of a real experiment, comparing the three-plus-one and the five-point method when an IMU is available.

We first presented this work in [50].

4.2 Related Work

Our work places itself at the intersection of minimal solvers for geometric vision problems with approaches using motion constraints.

There have been two previous papers dedicated to solving the three-plus-one problem. However, both fall short of providing an efficient, closed-form solution. The solution in Kalantari *et al.* [31] uses a Gröbner basis method, but due to suboptimal formulation, ends up with 12 solutions. A related problem was solved in the work of Lobo and Dias [40] who use a general formulation of a given reference direction (vertical in their case) to solve several geometric vision problems by using vanishing points and/or inertial measurements.

Structure from motion has benefited from attitude measurements. When all 3 DOF of rotation are known, either from vanishing points [33, 5] or inertial measurements, the problem can be reduced to a tractable estimation of the focus of expansion. Vieville [71] demonstrated the first approach where only the gravity vector is used to simplify structure from motion. The relation between the use of gravity vector and the lack of knowledge in correspondences has been studied in [21, 42]. When multiple frames are used, inertial measurements have been naturally integrated along visual features as measurements in nonlinear Kalman filtering [58, 47, 66, 30, 25]. Partial information like altitude has been used [59] to eliminate the unknown scale of monocular vision. Diel *et al.* [20] presented how a new epipolar constraint based on inertial information can be added in the visual odometry estimation process. In augmented reality, Azuma [4] and You *et al.* [72] proposed hybrid inertial-vision trackers where vision-based algorithms refine orientation estimates provided by an

inertial sensor. Burschka and Hager developed a vision approach to SLAM [10], which circumvents the problems caused from drift in the inertial measurements by using a vision algorithm to estimate directly the relative pose of the camera between frames.

4.3 Problem Formulation and Notation

We now introduce notation for the basic geometric objects we will use to formulate the problem. Image points are represented by homogeneous 3-vectors $\mathbf{q} = (x, y, 1)^\top$. Scene (world) points are represented by homogeneous 4-vectors $\mathbf{Q} = (X, Y, Z, 1)^\top$. Given image point correspondences $\mathbf{q}$ and $\mathbf{q}'$ in two calibrated views, it is known that the "essential matrix" constraint relating them is $\mathbf{q}'^\top E\mathbf{q} = 0$, where $E \equiv \hat{\mathbf{t}}S$ where the rotation matrix $S \in \mathbf{SO}(3)$ and $\hat{\mathbf{t}}$ is a 3×3 skew-symmetric matrix corresponding to the translation vector $\mathbf{t}$, which is known only up to scale. The essential matrix thus has five parameters.

We will now define and formulate the three-plus-one problem. We are given three image correspondences $\mathbf{q}_i \leftrightarrow \mathbf{q}'_i, i = 1, .., 3$ from calibrated cameras, and a single directional correspondence in the form of two unit vectors $\mathbf{d} \leftrightarrow \mathbf{d}'$ (see Figure 4.1a). Our goal is to find the essential matrix E which relates the two cameras, and thus find the rigid transformation between them up to a scale factor. We will first show that this problem is equivalent to finding the translation vector $\mathbf{t}$ and a rotation angle θ around an arbitrary rotation axis.

Let us choose the arbitrary rotation axis to be $\mathbf{e}_2 = [0, 1, 0]^\top$. We can now compute the rotation matrices R and R' that coincide $\mathbf{d}$ and $\mathbf{d}'$ with $\mathbf{e}_2$, and apply them to the respective image points, yielding $\mathbf{p_i} = R\mathbf{q_i}$ and $\mathbf{p_i}' = R'\mathbf{q_i}'$ for each $i = 1, .., 3$. This operation aligns the directional correspondence in the two views

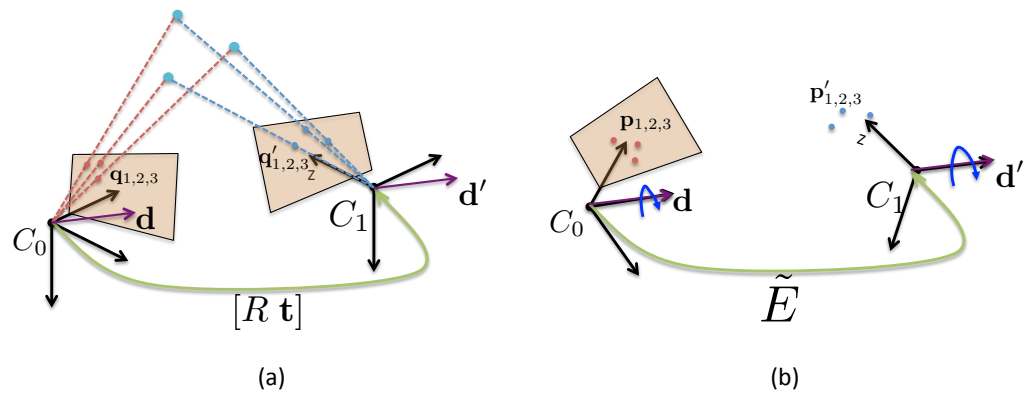


Figure 4.1: (a) Illustration of the geometry of the three-plus-one problem. The three image correspondences $\mathbf{q}_i \leftrightarrow \mathbf{q}'_i$ and a directional correspondence $\mathbf{d} \leftrightarrow \mathbf{d}'$ between coordinate frames C_0 and C_1 are given. (b) Illustration of the geometry after applying the rotation matrices R and R' which align the vectors $\mathbf{d}$ and $\mathbf{d}'$ with the y-axis respectively.

with $\mathbf{e}_2$. Once the axis is chosen, we only need to estimate the rotation angle around it and the translation vector in order to reconstruct the essential matrix.

After taking the directional constraint into account, from the initial five parameters in the essential matrix, we now only have to estimate three (see Figure 4.1b). This three-parameter essential matrix $\tilde{E}$ relates the points $\mathbf{p}$ and $\mathbf{p}'$ as follows:

$$\mathbf{p}_i'^T \tilde{E} \mathbf{p}_i = 0, \tag{4.3.1}$$

Since the rotation is known to be around $\mathbf{e}_2$, we can use the axis-angle parameterization of a rotation matrix to parametrize $\tilde{E}$ as follows:

$$\tilde{E} = \hat{\tilde{\mathbf{t}}}(I + \sin\theta \hat{\mathbf{e}}_2 + (1 - \cos\theta)\hat{\mathbf{e}}_2^2),$$

where $\tilde{\mathbf{t}} = R'\mathbf{t}$.

Each image point correspondence gives us one such equation of the form (4.3.1),

for a total of three equations in four unknowns (elements of $\tilde{\mathbf{t}}$ and θ). To create a polynomial system, we set $s = \sin\theta$ and $c = \cos\theta$, and add the trigonometric constraint $s^2 + c^2 - 1 = 0$, for a total of five equations in four unknowns. In order to reduce the number of unknowns and take care of the scale ambiguity in $\tilde{E}$, we choose the direction of the epipole by assuming that the translation vector $\tilde{\mathbf{t}}$ has the form $[x, y, 1]^\top$. This means that for each $\tilde{\mathbf{t}}$ that we recover, $-\tilde{\mathbf{t}}$ will also need to be considered as a possible solution.

Once we substitute for $\tilde{E}$ in equation (4.3.1), the resulting system of polynomial equations has the following form:

$$a_{i1}xs + a_{i2}xc + a_{i3}ys + a_{i4}yc + a_{i5}x - a_{i2}s + a_{i1}c + a_{i6} = 0 \tag{4.3.2}$$

for $i = 1, .., 3$, and the equation

$$s^2 + c^2 - 1 = 0. \tag{4.3.3}$$

We will refer to these equations as $F = \{f_i(x, y, s, c), i = 1, ..., 4\}$ in the rest of the chapter. The coefficients a_{ij} are expressed in terms of image correspondences as follows:

$$\begin{aligned}
a_{i1} &= p'_{iy}p_{ix} \\
a_{i2} &= -p'_{iy} \\
a_{i3} &= -p'_{ix}p_{ix} - 1 \\
a_{i4} &= p'_{ix} - p_{ix} \\
a_{i5} &= p_{iy} \\
a_{i6} &= -p'_{ix}p_{iy},
\end{aligned} \tag{4.3.4}$$

where p_{ix} and p_{iy} (p'_{ix} and p'_{iy}) are the first and second components of the rotated image points p_i (p'_i). In the next section we will analyze and solve this system in closed form and show that it has up to four solutions. The total number of possible pose matrices arising from our formulation is therefore at most 8, when we take into account the fact that we have to consider the sign ambiguity in $\tilde{\mathbf{t}}$. When the motion of the camera in the z direction (after the rotation by R and R') is extremely small, the parametrization $\tilde{\mathbf{t}} = [x, y, 1]^\top$ is numerically unstable. We deal with this rare instability by formulating and solving a system for the parametrizations $\tilde{\mathbf{t}} = [x, 1, z]^\top$ and $\tilde{\mathbf{t}} = [1, y, z]^\top$, which can be easily done using the methods we describe below, but omitted for the purposes of this presentation.

4.4 Closed-form Solution

We first review the closed-form solution proposed by Xun and Roumeliotis in their technical report []. This solution will be used to reason about the degenerate configurations later in this chapter. They first show that the system has four solutions, and that it can be solved analytically by elimination and back-substitution. Specifically, we first present an elimination procedure to obtain a 4th-order univariate polynomial in c, which can be solved in closed-form. Subsequently, we determine the remaining three variables through by back-substitution, where each solution of c returns exactly one solution for the other three variables. Therefore, we have a total of 4 solutions for the relative rotation matrix and translation vector.

The main steps of the elimination procedure are listed as follows.

1. Solve for x and y as a function of c and s using the first two equations in (4.3.2). The variables x and y can be expressed as quadratic functions of c and s.

2. Substitute x and y in the third equation in (4.3.2). This yields again a cubic polynomial in c and s, which is reduced into a quadratic by exploiting the relationship between its coefficients and the trigonometric constraint.

3. Finally, using the Sylvester resultant (see Chapter 3, §5 in [17]), we can eliminate one of the remaining two unknowns, say s, and obtain a 4th-order polynomial in c.

Now, we describe the details of our approach. Rewrite the first two equations in (4.3.2) as linear functions of c and s as follows:

$$\begin{bmatrix} a_{11}s + a_{12}c + a_{15} & a_{13}s + a_{14}c \\ a_{21}s + a_{22}c + a_{25} & a_{23}s + a_{24}c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{12}s - a_{11}c - a_{16} \\ a_{22}s - a_{21}c - a_{26} \end{bmatrix}, \tag{4.4.1}$$

and solve the above linear system for x and y:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{d} \begin{bmatrix} a_{23}s + a_{24}c & -(a_{13}s + a_{14}c) \\ -(a_{21}s + a_{22}c + a_{25}) & a_{11}s + a_{12}c + a_{15} \end{bmatrix} \begin{bmatrix} a_{12}s - a_{11}c - a_{16} \\ a_{22}s - a_{21}c - a_{26} \end{bmatrix}, \tag{4.4.2}$$

where the determinant

$$d = (a_{11}s + a_{12}c + a_{15})(a_{23}s + a_{24}c) - (a_{21}s + a_{22}c + a_{25})(a_{13}s + a_{14}c). \tag{4.4.3}$$

Substituting the expression for x and y into the third equation in (4.3.2) and multiplying both sides of the equation by d, yields a cubic equation in s and c:

$$g_1s^3 + g_2cs^2 + g_1sc^2 + g_2c^3 + g_3s^2 + g_4sc + g_5c^2 + g_6s + g_7c = 0.$$

The coefficients g_i for $i = 1, ..., 6$ are derived symbolically and are found in Section

4.11, equation (4.11.1). By using the fact that $s^2+c^2=1$, and exploiting the relation between the coefficients of the first four terms, we can reduce this equation to the following quadratic

$$g_1 s + g_2 c + g_3 s^2 + g_4 sc + g_5 c^2 + g_6 s + g_7 c = 0. \tag{4.4.4}$$

In the final step, we employ the Sylvester resultant to eliminate one of the two remaining variables from equations (4.3.3) and (4.4.4). The resultant of the two polynomials is the determinant of the Sylvester matrix

$$\begin{bmatrix} g_3 & g_4 c + g_1 + g_6 & g_5 c^2 + g_2 c + g_7 c & 0 \\ 0 & g_3 & g_4 c + g_1 + g_6 & g_5 c^2 + g_2 c + g_7 c \\ 1 & 0 & c^2 - 1 & 0 \\ 0 & 1 & 0 & c^2 - 1 \end{bmatrix}, \tag{4.4.5}$$

which leads to a 4th-order polynomial equation

$$\sum_{i=0}^{4} h_i c^i = 0, \tag{4.4.6}$$

with coefficients h_i given in Section 4.11, equation (4.11.2). This shows that in general, the system has four solutions for c. Back-substituting the solutions of c into equation (4.4.4), we compute the corresponding solutions for s. Note that each solution for c corresponds to one solution for s because we can reduce the order of equation (4.4.4) to linear in s, once c is known, by replacing the quadratic terms s^2 with $1-c^2$. After s and c are determined, we compute the corresponding solutions for x and y using (4.4.2) for a total of four solutions. We will describe how to recover the pose matrix from x, y, s and c in Section 4.6.4.

In the next section, we will solve the polynomial system in (4.3.2) and (4.3.3) using algebraic geometry techniques.

4.5 Algebraic Geometry Background

In this section, we review the algebraic geometry concepts that have been applied to solving geometry problems in computer vision. The definitive introduction to these concepts can be found in textbooks on algebraic geometry [18, 17].

One of the textbook ways [18] of solving algebraic systems is via the so-called *action matrix*. We will give a brief overview of this method. Consider a system of polynomial equations in m variables $f_1(\mathbf{x}) = ... = f_n(\mathbf{x}) = 0$, where $\mathbf{x} = (x_1, ..., x_m)$, and coefficients from a field K. A geometric description of the solution set to a polynomial system is given by an affine *variety* V. In the case where there are finitely many solutions, the variety V is zero-dimensional which includes a finite number of points in K^m when K is an algebraically closed field. The ring of all polynomials in $\mathbf{x}$ is denoted by $K[\mathbf{x}]$.

The polynomials f_i are the *generators* of the polynomial *ideal* $I = \{\sum_{i=1}^{m} h_i f_i : h_1, ...h_m \in K[\mathbf{x}]\}$. In other words, an ideal generated by f_i is a set that includes the generators, and is also closed under addition and multiplication by other polynomials in $K[\mathbf{x}]$. It is easy to show that the polynomials in the ideal vanish on the same variety as the generating set. The problem of solving the system now becomes a problem of finding a subset of equations in the ideal with properties that make them easy to solve.

The methods used to solve polynomial systems in computer vision rely heavily on the properties of a set of equivalence classes for polynomial division (the remainders) of members of $K[\mathbf{x}]$ by members of the ideal I. This set of equivalence classes is called

the *quotient ring*, and is denoted as $K[\mathbf{x}]/I$. If the variety is zero-dimensional (i.e., the system has finitely many solutions), the quotient ring is a vector space whose dimension equals the number of solutions. On this vector space, we can define linear maps, which are often represented in matrix forms and called action matrices.

The action matrix is the key for solving systems of polynomial equations. A univariate polynomial can be solved using eigenvalue decomposition of a companion matrix. The action matrix is a multivariate equivalent of the companion matrix. The idea is to find a linear operator T_p for some $p \in K[\mathbf{x}]$ that represents the multiplication by p in the vector space defined by $K[\mathbf{x}]/I$, i.e., $T_p : f(\mathbf{x}) \rightarrow p(\mathbf{x})f(\mathbf{x})$. If we select a basis for this vector space, we can represent T_p as a matrix $\mathbf{m}_p$ with entries in K. It was shown in [18] that λ is an eigenvalue of this matrix if and only if λ is a value of the function p evaluated on the variety V of the ideal. This means that if we set $p = x_k$, we can find the value of x_k which satisfies the initial system of equations. We can also determine the solutions through eigenvectors. It is known that the eigenvectors of the action matrix represent the scaled solutions to the same problem. We can also determine the scale, because the monomial 1 is always in the basis for zero-dimensional varieties.

Finding the dimensionality and basis for this vector space is the first step in recovering solutions. The dimensionality immediately tells us the number of solutions, while the basis is important in the action matrix computation. One way of obtaining these two quantities is through division of polynomials in $K[\mathbf{x}]$ by the Gröbner basis, which is a special basis for the ideal, division by which cancels out all the possible leading terms of the polynomials in the ideal. A good introduction to Gröbner base is found in [17].

Computing a Gröbner basis using finite precision arithmetic is known to be a numerically unstable process. However, algorithms developed by Traverso in[69]

allow us to analyze the ideal generated by our system using coefficients from a *prime field* $K = \mathbb{Z}/r$ (integers modulo r), where $r >> 7$ is a prime number [35]. Since this field is finite, the computation with polynomials with coefficients in $\mathbb{Z}/r$ (including Gröbner basis) is exact. The algorithms ensure that if a stable Gröbner basis is found in this field under repeated trials with random coefficients, the monomials will remain the same when we change the field to $\mathbb{Q}$ with some probability. In our case, we only need the Gröbner basis for one system, F, and it is easy to check when we have it. Once a Gröbner basis G is found, a linear basis for the quotient space can be formed by the monomials in the remainder after division by G.

In order to use efficient linear algebra techniques to manipulate the system (4.3.2)-(4.3.3), we rewrite it as follows:

$$\mathbf{CX} = 0,$$

where $\mathbf{C}$ is a matrix whose columns contain coefficients of the monomials, and $\mathbf{X}$ is the vector of monomials corresponding to the columns of $\mathbf{C}$. It should be noted that the ideal is closed under row operations on $\mathbf{C}$.

In solving our problem, we will follow the method outlined in [14], which allows us to build the action matrix without constructing a Gröbner basis (we will still analyze the system and extract its Gröbner basis in the finite field for the purposes listed above). We briefly describe their method here. The key idea is to determine the so-called *solving basis* $\mathcal{B}$ (in our case, we use the monomial basis for the quotient ring), and the *required monomials* $\mathcal{R} = x_k\mathcal{B}\backslash\mathcal{B}$. Specifically, our objective is to find the minimum number of monomials needed to construct the action matrix, and then re-arrange the matrix such that those monomials, along with the basis monomials, occupy the last columns of the matrix. Using algebraic geometry software we can find a candidate linear basis $\mathcal{B}$ for the quotient space. For the action matrix corresponding

to multiplication by x_k, the set of monomials that need to be expressed in terms of $\mathcal{B}$ is the set $\mathcal{R} = x_k\mathcal{B}\backslash\mathcal{B}$. The rest of the monomials in the system are called $\mathcal{E}$. The polynomial system with coefficients $\mathbf{C}$ can then be expressed as follows:

$$\mathbf{CX} = \begin{bmatrix} \mathbf{C}_{\mathcal{E}} & \mathbf{C}_{\mathcal{R}} & \mathbf{C}_{\mathcal{B}} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0.$$

The only requirement on this coefficient matrix, after this matrix is put into the row-echelon form, is that its $|\mathcal{R}| \times |\mathcal{R}|$ submatrix corresponding to the monomials in $\mathcal{R}$ and the last $|\mathcal{R}|$ equations has full rank. This submatrix is called $\mathbf{C}_{\mathcal{R}2}$ in [14]. When we discuss our solution, we will illustrate how to use this matrix to extract the action matrix. The complete details are found in [14].

The initial set of equations F (see (4.3.2)-(4.3.3)) is unlikely to have a coefficient matrix $\mathbf{C}$ that meets the above requirement. This is where we will draw on the ideal members to expand the original system with additional equations, until the requirement on the action matrix construction is satisfied. The technique to generate ideal members efficiently proposed in [35] involves multiplying the original polynomials by monomials starting with the lowest orders. This operation, when applied to an equation (a row of $\mathbf{C}$), will result in the coefficients from that row to be shifted to the left in the matrix to take their places in columns corresponding to their new monomials. We will continue adding polynomials (checking for linear dependence and unneeded ones), until $\mathbf{C}$ is large enough to produce a full rank $\mathbf{C}_{\mathcal{R}2}$. We call the resulting set of polynomials (which are monomial multiples of the original system) an *elimination template*, and and matrix $\mathbf{C}$ an *elimination matrix*. This part of the process can be done with coefficients drawn randomly from $\mathbb{Z}/r$.

4.6 Action Matrix Solution

In this section, we will follow our implementation of the action-matrix method due to [14, 35]. The elimination template should be computed using a symbolic mathematics software, such as Maple or Macaulay2. We found Maple to be a more convenient choice.

We have already showed that the system has four solutions, but this can also be verified using Maple's Groebner package. Let J be the ideal generated by F, where coefficients a_{ij} were chosen at random from $\mathbb{Z}/30029$. We then computed the GrevLex-order Gröbner basis for J. Since this ideal is zero-dimensional, and the vector space spanned by the polynomials of the quotient ring was four-dimensional, there are in general four solutions to the system in the field of real numbers.

In the next two subsections, we describe the details that are specific to the three-plus-one problem, and thus the set of polynomials formed by (4.3.2) and (4.3.3). The choices of variable order and action monomial (c in our case) that we made below are not arbitrary. Other choices can produce much larger elimination templates or may be less favorable numerically. The entire process of elimination template generation was repeated for several variable orderings and action monomials to ensure stability and small size of the template. This process paralleled the automated method proposed in [35]. (We did not use that method directly due to its use of full Gröbner basis and a requirement for reduced row-echelon form, instead of row-echelon form, for the coefficient matrix which causes numerical instability.)

4.6.1 Finding the Bases

We used the Gröbner basis of J in a finite field (such as the one computed to verify the number of solutions) to determine the solving basis monomials. We chose to

order the variables (x, y, c, s). After computing the GrevLex Gröbner basis with that order, we chose the solving basis $\mathcal{B}$ to be $[y, c, s, 1]$, the same as the quotient ring basis monomials. The set of required monomials $\mathcal{R}$, which is the set of monomials that need to be expressed in terms of $\mathcal{B}$ for the action matrix $\mathbf{m}_c$, is thus $[yc, c^2, cs]$.

4.6.2 Constructing the Elimination Template

Once we know the solving basis and the required monomials, we must extend the initial set of four polynomials with other polynomials from the ideal J such that the rank condition on $\mathbf{C}_{\mathcal{R}2}$, after elimination, is satisfied. We multiplied the four original polynomials by the monomials in (x, y, c, s) of degrees 1 and 2 and added them to the system, put the resulting coefficient matrix in row-echelon form and check the rank of $\mathbf{C}_{\mathcal{R}2}$. We then eliminated the redundant polynomials from the template.

The result of this process is elimination template, which consists of a set of 21 monomial multipliers and corresponding polynomials from the initial system:

$$\{f_1, f_2, f_3, f_4, f_1s, f_2s, f_3s, f_4s, f_1c, f_2c, f_3c, f_4y, f_4x, \\ f_1s^2, f_2s^2, f_3s^2, f_1cs, f_2cs, f_3cs, f_4ys, f_4xs\} \quad (4.6.1)$$

as well as, a vector of 25 monomials:

$$[\ yc^2s \quad ys^2c \quad xs^3 \quad ys^3 \quad xc^2 \quad yc^2 \quad xcs \quad ycs \quad c^2s \quad xs^2 \quad ys^2 \quad s^2c \quad s^3 \\ xc \quad xs \quad ys \quad s^2 \quad x \quad yc \quad c^2 \quad cs \quad y \quad c \quad s \quad 1\ ]. \quad (4.6.2)$$

The coefficients from the equations in (4.6.1) will form the entries in the 21×25 elimination matrix with columns corresponding to the monomials in the vector (4.6.2).

The exact arrangement of coefficients is given in Appendix 4.12.

4.6.3 Reduction and Action Matrix Extraction

With the coefficient matrix at hand, we leave Maple and $\mathbb{Z}/r$. The template will remain the same across all instances of the problem. We construct the 21×25 matrix from the coefficients a_{ij} (see (4.3.4)) for the particular instance of the problem, and perform Gaussian elimination with partial pivoting or LU decomposition. The elimination can be stopped 3 rows early for added efficiency. We then extract the 3×3 matrix $\mathbf{C}_{\mathcal{R}2}$ representing the monomials in $\mathcal{R}$ (columns 19, 20 and 21) in the last three rows of the upper triangular matrix. We invert this matrix and multiply it with the matrix $\mathbf{C}_{\mathcal{B}}$ representing the monomials in $\mathcal{B}$ (columns 22 through 25) in the last three rows. The rows of the resulting 3×4 matrix $\mathbf{C}_{\mathcal{R}2}^{-1}\mathbf{C}_{\mathcal{B}}$ become the first three columns of the 4×4 action matrix $\mathbf{m}_c$. The last column has a 1 in the third position, indicating that c (a required monomial $1 \cdot c$) is already expressed in the basis as a vector $[0, 1, 0, 0]^\top$. The solutions are extracted as the real eigenvectors of this action matrix which can be computed in closed form. Since the value of a constant polynomial evaluated at any point is also constant, we set the scale of our solutions by dividing each element of the eigenvector by the last element, which corresponds to the monomial 1.

We have recovered up to four sets of values for y, c and s, and must now find the corresponding values for x by solving one of the equations from (4.3.2) for each set of values. These equations are linear in x.

4.6.4 Back Substitution and Pose Recovery

We will now describe how to find the pose matrices from solutions to the system. We recover the rotation as

$$R_{\mathbf{e}_2} = \exp(\operatorname{atan2}(s, c)\hat{\mathbf{e}}_2),$$

and translation as

$$\tilde{\mathbf{t}} = \pm[x, y, 1]^\top.$$

Finally, we reconstruct each pose as follows:

$$P = \left[\begin{array}{c|c} S & \mathbf{t} \end{array}\right] = \left[\begin{array}{c|c} R'^\top R_{\mathbf{e}_2} R & R'^\top \tilde{\mathbf{t}} \end{array}\right].$$

There are up to 8 such pose matrices for each instance of the problem. Point triangulation and chirality checks are used to eliminate false solutions. Since this solution method is designed to be used in robust estimation frameworks (such as RANSAC), any remaining false hypotheses can be eliminated by triangulating an additional point and choosing the P with the minimum reprojection error.

4.7 Degenerate Configurations

It was pointed out in [24] that the three-plus-one algorithm is not degenerate for collinear world points, except for the line parallel to the translation direction. It turns out that this degeneracy is only a special case of two additional degenerate configurations.

The first one occurs when all world points lie on the horopter [3], i.e. their

projections are the same in the first and second images up to a projective transformation, which is an ambiguous configuration for two views ([27], Result 22.21). Algebraically, this configuration causes the coefficients a_{i4} from (4.3.4) to vanish, removing the terms of the form $a_{i4}yc$ from the equations. The resulting polynomial system no longer generates a zero-dimensional ideal, and thus has an infinite number of solutions.

The second degenerate configuration occurs when the determinant d in (4.4.2) is zero. When this occurs, the translation $\tilde{\mathbf{t}}$ cannot be estimated from the point correspondences using the equation (4.4.2). We can derive the geometric condition that causes the determinant to vanish as follows. After projecting two generic 3D points $\mathbf{x}_i = [X_i, Y_i, Z_i]^\top$ for $i = 1, 2$ into the camera frames, we get $\mathbf{p}_i = \mathbf{x}_i$ and $\mathbf{p}'_i = R_{\mathbf{e}_2}\mathbf{x}_i + \tilde{\mathbf{t}}$. We compute the corresponding coefficients (4.3.4), and substitute them into equation (4.4.3). After using the fact that $s^2 + c^2 = 1$, the determinant condition becomes

$$(Z_2Y_1 - Z_1Y_2)(cx - s) + (Z_1X_2 - X_1Z_2)y + (X_1Y_2 - X_2Y_1)(sx + c) = 0, \quad (4.7.1)$$

which we can rewrite as $(-R_{\mathbf{e}_2}^\top \tilde{\mathbf{t}})^\top \hat{\mathbf{x}}_1 \mathbf{x}_2 = 0$. This condition means that the second camera's center of projection, expressed in the first camera's coordinate system, is orthogonal to the vector formed by the cross product of the world points. In other words, the degeneracy occurs when the world points are coplanar with the translation vector. This is a more general case than the three points parallel to the translation direction discussed in [24]. A geometric argument can be made via Chasles' Theorem ([27], Thrm. 22.3) since five points on a plane (three world points and two camera centers) generally define a conic. The theorem proves that in this case there exist alternative reconstructions. This configuration is shown in Figure 4.2. We note that

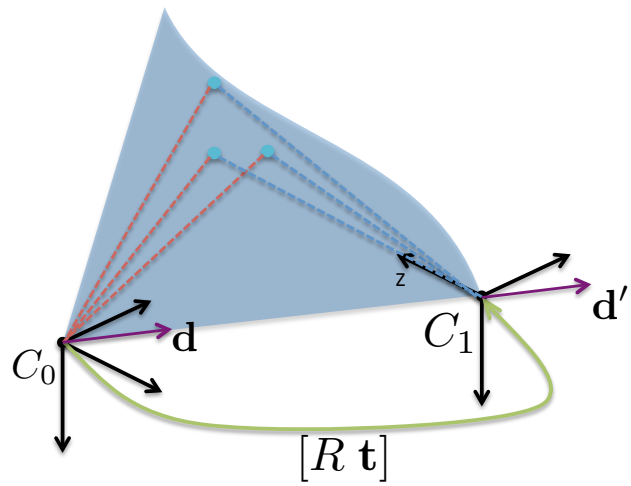


Figure 4.2: A degenerate configuration occurs when all world points are on the same plane as the cameras' centers of projection C_0 and C_1.

while the closed form solution is degenerate when the two correspondences $i = 1, 2$ in equation (4.3.4) are coplanar with the baseline, the action matrix method only fails when all three points are coplanar with $\mathbf{t}$.

In practice these degeneracies do not cause a large number of errors, as seen in the experiments.

4.8 Simulation Results

In this section we establish the expected performance level of our algorithms in noise-free and noisy conditions, comparing them first to each other and then to the five-point relative pose estimation algorithm. This is accomplished with simulated data. We study both single and double precision arithmetic implementations for the action-matrix and closed-form algorithms, and look for numerical differences between them. In the comparison with the five-point method, we expect that the more constrained three-plus-one method will give better accuracy in noisy conditions.

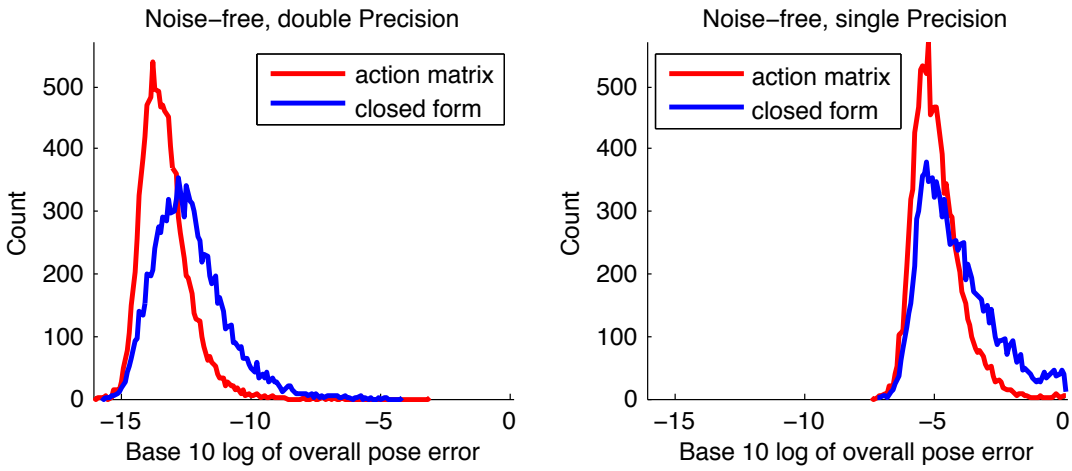


Figure 4.3: Distribution of numerical errors in recovered poses for 10^4 random configurations with single and double precision implementations. The error measured is the Frobenius norm of the difference between the true and estimated pose matrices. The median errors for double precision are $3.9 \cdot 10^{-14}$ for the action matrix and $3.1 \cdot 10^{-13}$ for the closed form method. For single precision the errors are $9.3 \cdot 10^{-6}$ and $3.5 \cdot 10^{-5}$, respectively.

In each figure where single and double precision versions of the three-plus-one algorithm are compared, the legend is as follows: C and A refer to the "closed form" and "action matrix" methods, respectively, and 32f and 64f refer to the floating point precision, single and double, respectively.

The input data was generated as follows. The pose of the first camera was defined to be the identity pose $[I|0]$, and the reference direction was generated as a random unit vector. The pose of the second camera was generated uniformly at random as a unit translation vector $\mathbf{t}$ and three Euler angles corresponding to roll, pitch and yaw of the second camera within the limits specified by the experiment. The Euler angles were converted to a rotation matrix R, which together with $\mathbf{t}$ formed the camera pose $[R|\mathbf{t}]$. Sets of five three dimensional world points were generated within a spatial volume defined by the parameters of the experiment, so

as to be visible by both cameras. The world points were then projected into the image planes of the two cameras (with identical intrinsic calibration defined by the experiment) to form image correspondences, and contaminated with Gaussian noise with standard deviation in pixels defined by the experiment. The second camera's reference direction was then computed, and the directional correspondence vectors were contaminated by Gaussian rotational noise with standard deviation in degrees defined by the experiment. The sets of image and directional correspondences were then used to compute pose with the three-plus-one and the five-point algorithms. Each method produces a set of pose hypotheses for each input set. The error reported for each input set is the minimum error for all hypotheses. All comparisons between algorithms were run on identical input data.

4.8.1 Perfect Data

First, we establish the correctness and numerical stability of our algorithms. In these experiments, the pose was allowed to vary over the entire range of rotations, and the translation and directional correspondence vectors were generated uniformly at random and normalized to length 1. Figure 4.3 shows errors in pose recovery on perfect, simulated data. The noise metric is the minimum Frobenius norm of the differences between the true pose matrix and each computed pose matrix (up to eight per instance). The error distribution shows that both algorithms perform as expected, with the action matrix method exhibiting better numerical stability. We will discuss this difference in the next section.

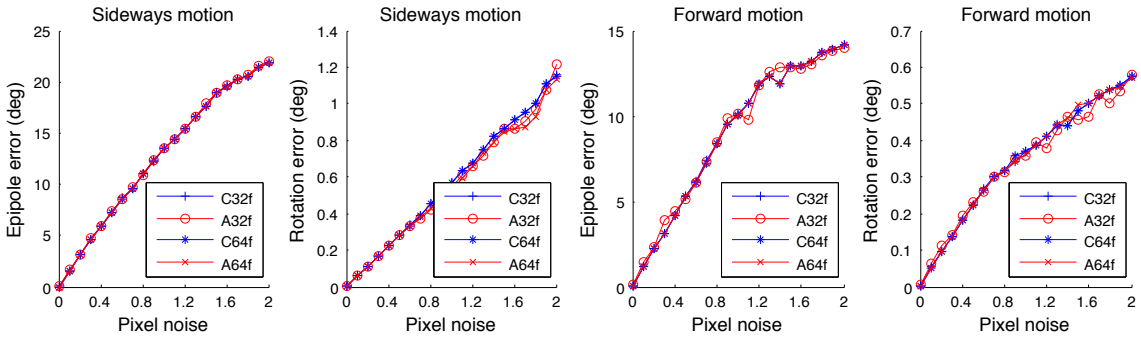


Figure 4.4: Median translation and rotation errors for the sideways and forward motion of the baseline camera against noise standard deviation. As with other motion estimation methods, the sideways motion gives significantly worse performance than forward motion on the same data.

4.8.2 Image Noise

In subsequent simulated results we examine the performance for "standard" imaging conditions, which we define as a 640x480 camera with a 60° FOV, where structure points are found between 10 and 40 baselines away, where one baseline is the distance between camera centers. We will first consider only pixel noise, and deal with directional correspondence noise later. Figure 4.4 compares performance for forward and sideways motion of the camera under different pixel noise conditions. It comes as no surprise that forward motion is generally better numerically, and that the rotation estimate (1DOF) is significantly better than the estimate of the epipole. The plots also conclusively demonstrate numerical stability of both single and double precision implementations under "standard" imaging conditions. These experiments show that once realistic noise is added, the numerical precision of either algorithm is sufficient for implementation on single precision processors.

The seemingly large errors in epipole estimation are due to the fact that the scene points are located far from the cameras, but we believe it to be a realistic (if difficult) configuration.

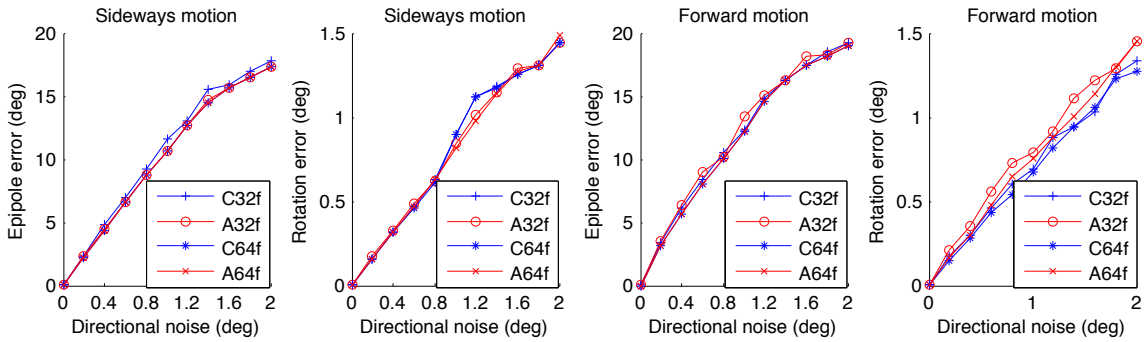


Figure 4.5: Median translation and rotation errors for varying levels of noise in directional correspondences for the "standard" camera. The noise standard deviation varies from 0° to 2°.

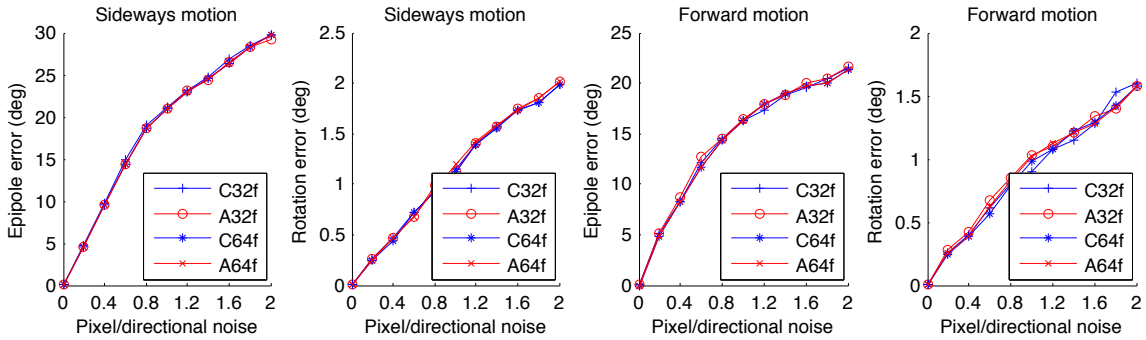


Figure 4.6: Median translation and rotation errors for varying levels of noise in both directional and image correspondences for the "standard" camera. The noise standard deviation varies from 0° to 2°, and from 0 to 2 pixels for image correspondences.

4.8.3 Directional Correspondence Noise

In this section, we investigate the performance impact of errors in directional correspondences. The directional noise was simulated as a rotation of the direction vector around a random axis with an angle magnitude drawn from a normal distribution. The standard deviation of the noise is plotted on the x-axis. The effect of directional noise only for a range of errors between 1 and 2 degrees can be seen in Figure 4.5. Performance under both pixel and directional noise is presented in Figure 4.6.

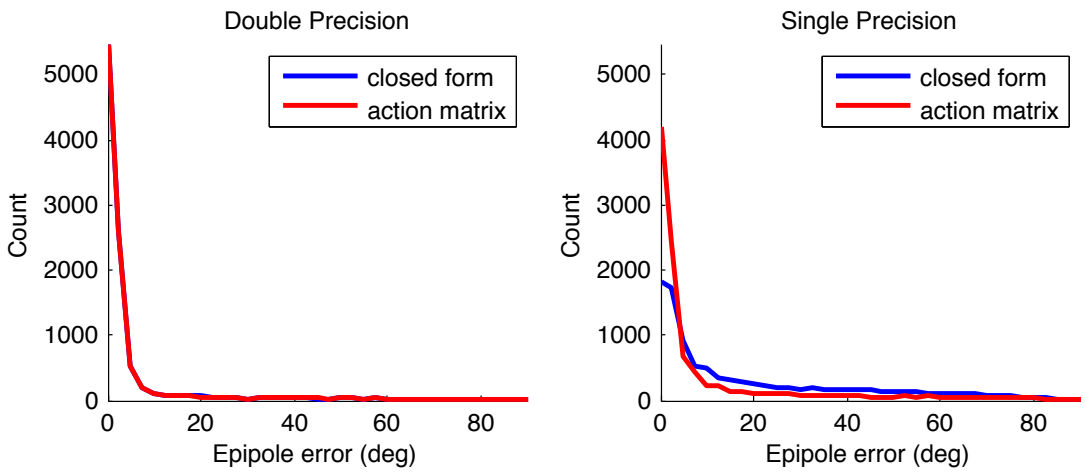


Figure 4.7: Distribution of errors in epipole orientations in degrees for 10^4 trials under forward motion with pixel error standard deviation of 0.3 and field of view of 10°. The median errors for double precision are 1.1° for both methods. For single precision the errors are 8.5° for the closed form and 1.9° for the action matrix method.

4.8.4 Numerical Stability

With noise-free data, the closed-form, single precision algorithm has noticeably worse performance than the action matrix algorithm (see Figure 4.3), however, there is no noticeable difference when the noise is added for "standard" camera, as we saw in the previous sections. Figure 4.7 demonstrates that under more difficult imaging conditions of 10° field of view with 0.3 pixel noise, the median errors in the epipole direction are the same in double precision, but in single precision the error is 4.5 times greater for the closed-form solution. This demonstrates that under some conditions, it is advantageous to use the action-matrix solution because of its superior numeric properties.

4.8.5 Comparison with the Five-point Method

We also compare the three-plus-one method to the classic five-point method. While they are not equivalent (since the five-point method does not require a specific point to be at infinity), they can be used interchangeably in some real situations, as described in the next section.

Since both closed-form and action-matrix-based algorithms exhibit similar performance, we only compare the double precision implementations of our closed-form algorithm and the five-point algorithm. Figure 4.8 demonstrates the effect of the field of view on the algorithms. The rotation estimation is generally better with the three-plus-one algorithm, while translation error does not decrease as quickly with the field of view in the three-plus-one case as in the five-point case. In Figure 4.9 we plot errors for several levels of directional noise, while varying the pixels noise. It is clear from the graphs that the three plus one algorithm is better at estimating rotations than the five point algorithm, even under significant error in the directional correspondence, but the five-point method is better at estimating sideways translation, even in the cases of small error in the directional correspondence.

In real experiments, we will use our method to compute vision-only relative pose, when points at infinity are present. But first, we compare the performance of the five-point and the three-plus-one methods in this scenario in simulation. The directional correspondence in this case is generated as a projection of a point at infinity, contaminated with noise and made unit-length. The directional correspondence noise can now be measured in pixels, putting the two methods on equal footing.

The test data were generated differently for this experiment. The first three correspondences were projected into the image from a range of 10 to 40 baselines, as before. An additional point at infinity was randomly generated within the field

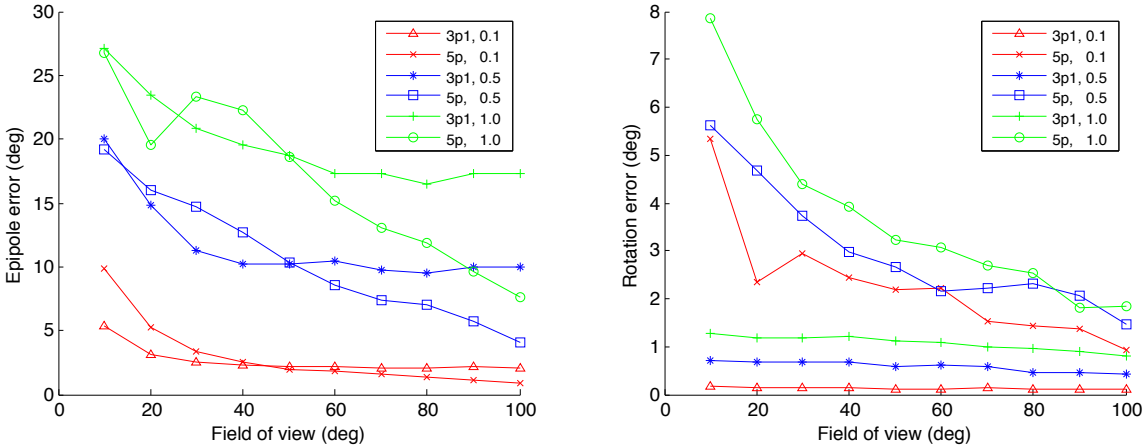


Figure 4.8: Median translation and rotation errors for varying fields of view of the baseline camera and random poses. In the legend, the three-plus-one algorithm is labeled "3p1", and the five-point algorithm is "5p". The number after the algorithm name indicates the standard deviation of pixel and directional (for three-plus-one method only) error standard deviations levels in pixels and degrees. The colors also correspond to noise levels: red is 0.1 pixel and 0.1°, red is 0.5 pixel and 0.5°, and green is 1.0 pixel and 1.0°.

of view of the camera and projected into the images. This correspondence was unitized and contaminated with pixel noise of the same standard deviation as the first three image points, giving us a directional constraint from a point at infinity. This experiment assumes that a real camera would have a sufficiently wide depth of field to keep nearby and distant features in focus simultaneously, which is expected when its field of view is wide, or its aperture is sufficiently small. From the results shown in Figure 4.10 we conclude that our method outperforms the five-point method, while using only four image points, in estimating rotation in forward and sideways motion, and translation in forward motion. Our method does a slightly worse job estimating translation in the sideways motion.

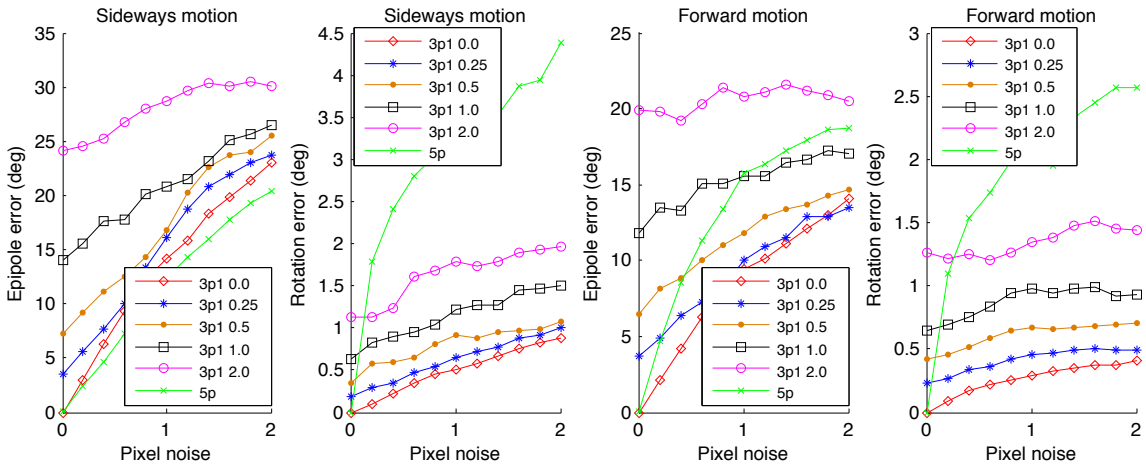


Figure 4.9: Comparison of the median errors of the three-plus-one algorithm with the five-point algorithm for the cases of forward and sideways motion for different directional noise levels. In the legend, the three-plus-one algorithm is labeled "3p1", and the five-point algorithm is labeled "5p". The number after algorithm name indicates the standard deviation of the directional noise in degrees. The green sequence with 'x' marker corresponds are the median errors in the five-point algorithm.

4.8.6 Computational Considerations

When using RANSAC, we can estimate the probability of success in getting an outlier-free hypothesis based on the number of elements in the minimal data set. When we estimate the epipolar geometry using image correspondences only, there are two sets of inliers: the set that can be used as directional correspondences and a set that can be used as a point correspondences. Both inlier ratios have to be taken into account when computing the RANSAC stopping criterion. If the number of distant points is sufficiently large (such as in outdoor scenes), we can realize a significant performance gain with our method since fewer hypotheses will need to be considered [23] due to smaller model size.

Since the hypothesis generator will run hundreds of times per frame in RANSAC-based visual odometry schemes, it is important to compare the computational requirements for the five point algorithm with the proposed methods. Computing

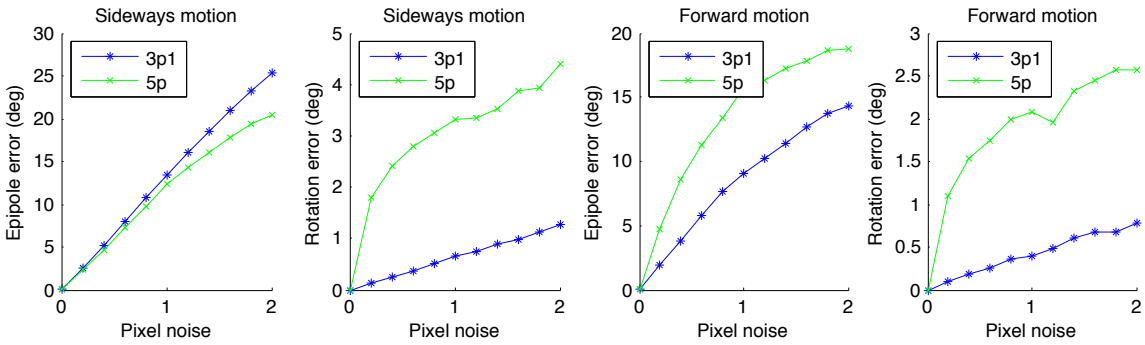


Figure 4.10: Comparison of the three-plus-one algorithm with the five-point algorithm where directional constraints are derived from image points at infinity. The plots show median errors in pose estimation. The green sequences with the 'x' marker show performance of the five-point algorithm and the blue sequences with the '*' marker correspond to the three-plus-one algorithm. The directional correspondences are derived from points at infinity and contaminated with the same pixel noise as the other image points. This graph shows the superior performance of the three-plus-one method in rotation estimation for forward and sideways motion, as well as translation estimation in forward motion.

the coefficients a_{ij} requires 9 multiplications. The closed-form solution requires 125 multiplications before arriving at the quartic polynomial. The real roots of the 4th degree polynomial can be extracted in closed form by computing and solving the depressed quartic and two quadratics for a total of about 40 operations and six square roots. The computation of the remaining variables takes additional 144 operations. The main computational step of the action matrix algorithm is Gaussian elimination (LU decomposition) of a 21×25 matrix. While theoretically taking $O(2n^3/3)$, or about 9000 operations, the elimination of our sparse matrix only requires about 500 multiplications. The eigenvalue decomposition of a 4×4 matrix is done by solving a quartic equation and eigenvectors are extracted with an inverse power iteration, which costs 88 multiplications.

On the other hand, the main computational steps in the classic five-point algorithm [54] are: extraction of the null space of a dense 5×9 matrix, requiring 280

operations, Gauss-Jordan elimination of a dense 10×20 matrix, requiring about 1300 operations, and real root isolation of a 10th degree polynomial, which can be accomplished as eigenvalue decomposition of a 10×10 sparse companion matrix or as an iterative root isolation process. From these observations we can conclude that both the closed-form and the action-matrix forms of the three-plus-one algorithm are significantly more efficient than the five-point algorithm. In real experiments, the performance of the C implementation of the closed-form algorithm outperformed an optimized implementation of the five-point method, on average, by a factor of 5 ($2.6\mu s$ compared to $13.0\mu s$ on a 3GHz laptop).

4.9 Experimental Results

In the introduction we specified as one of the main goals of this work the demonstration of monocular, RANSAC-based visual odometry with a four-correspondence hypothesis generator. We used our C++, double-precision implementation of the action-matrix-based method to test the algorithm in this context. We used a hand-held, 640×480 pixel, black and white camera with a $50°$ field of view lens to record an 825-frame, outdoor video sequence. The sequences included motion exercising all degrees of freedom, and was a realistic representation of a robot localization task (see sample images in Figure 4.11).

Harris corners and correlation matching was used to obtain image correspondences. The matches were used to estimate camera motion following the monocular scheme similar to the one described in [56]. The experiments consisted of using the correspondences to estimate camera motion with the standard five point algorithm and the new three point plus direction algorithm. We computed 200 hypotheses for each image pair. The correspondences themselves, the number of hypotheses

and the other system parameters remained the same, and only the pose hypothesis generator was changed between experiments. In the structure from motion experiment without an IMU, the directional correspondence was simply a unitized image point correspondence. Since most outdoor scenes have no shortage of faraway feature matches, RANSAC had no trouble choosing the right hypothesis with our method.

We will briefly describe the steps of the monocular visual odometry scheme:

1. Track features between consecutive frames. Estimate relative poses between two frames using preemptive RANSAC [53] with hypotheses generated either by the five point algorithm or the three-plus-one algorithm.

2. Use iterative refinement to polish the estimated pose with respect to all the inlier points.

3. Triangulate the feature matches in the two frames into 3D points. If this is not the first pair of frames, estimate the common scale between the current and last pose estimate using a 1-point preemptive RANSAC procedure.

4. Set the scale for the current pose estimate and attach it to the last estimate.

5. Repeat from step 1.

This procedure produces a robust, monocular visual navigation solution. If the features cannot be tracked for some reason or the preemptive RANSAC fails to produce a correct hypothesis, the scale estimate will fail, and the pose will jump.

4.9.1 Structure from Motion Results

In Figure 4.13 we stitched together the poses and highlighted the places where breaks in the path occurred. Since we know that we have enough points to track, the failures

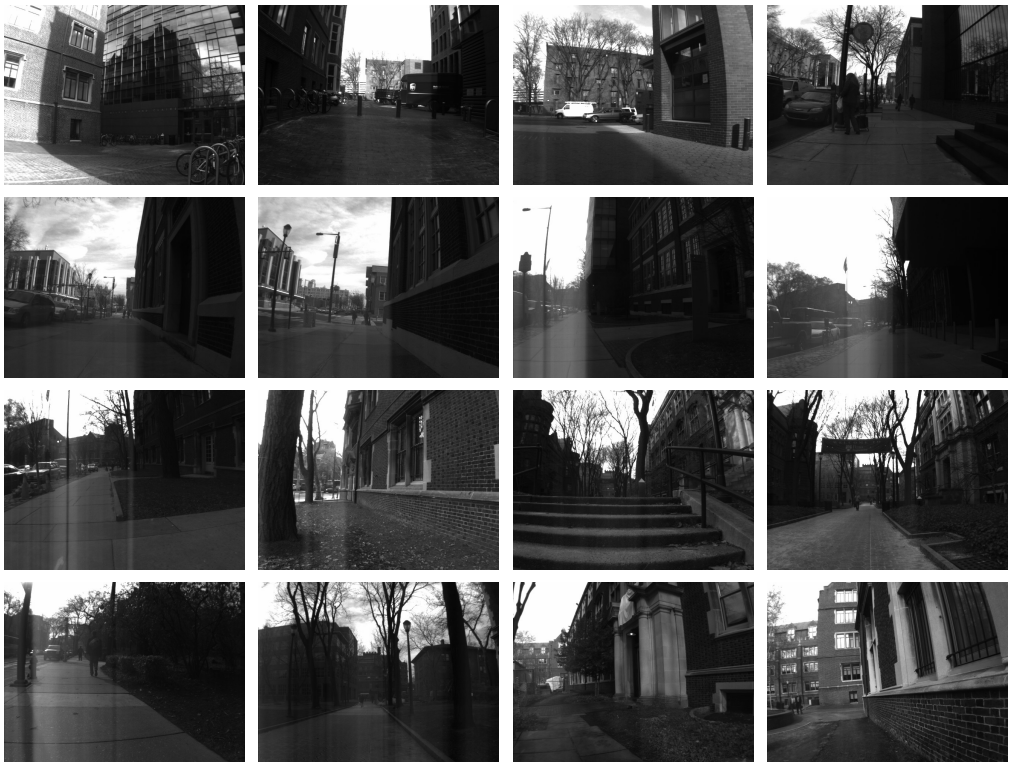


Figure 4.11: Sample images from our outdoor data set used to produce results in Figure 4.13. The bottom image illustrates the difficult lighting conditions.

are due to RANSAC-based pose estimation or RANSAC-based scale estimation, and is a result of a failure to choose an inlier subset. It is interesting to note that the failures happened in different places with different algorithms due to randomness of sampling. We expect more robust results (fewer breaks) from the three-plus-one method, and we found it to be the case due to the limited number of hypothesis.

In order to further quantify the real-world performance, we also computed relative pose for each consecutive pair of frames in the data set with each algorithm. Figure 4.12 shows the histogram of relative errors between the three-plus-one and five-point algorithm. For each algorithm, the pose was computed as follows: corner features were matched between two consecutive frames, and RANSAC-based relative pose estimation, followed by iterative refinement was performed on all matches. The

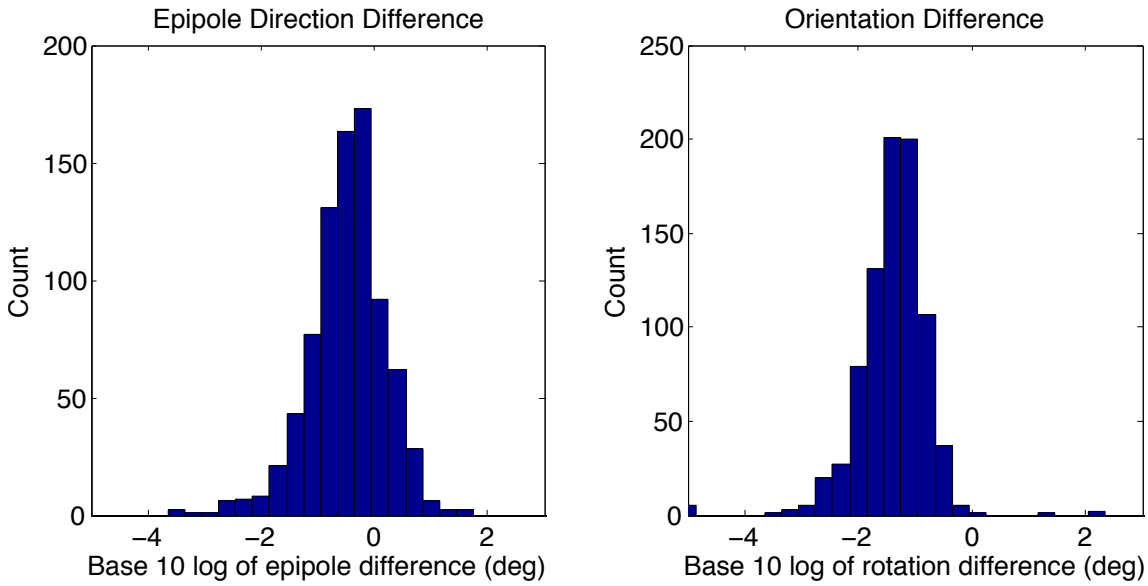


Figure 4.12: Differences in recovered relative pose between our algorithm and five-point for the experiment in Figure 4.13. The differences are small, except the few frames where one of the methods failed to find the right hypothesis. Those places are marked on Figure 4.13. The differences for a total of 824 consecutive frame pairs were computed. This shows that we can count on RANSAC to choose correctly the points at infinity and that in a realistic scenario there are enough such points to enable the three plus one visual odometry.

error between poses is, as before, the logarithm of the Frobenius norm of the difference between the pose matrices. The algorithms give very similar results, except in a handful of frames, where pose was computed incorrectly due to a failure to select an inlier point set in one or the other algorithm.

To further demonstrate the real-world performance, we collected a 2582-frame video from a mobile robot, where the position of the robot was tracked with a Topcon tracking total station. In this experiment the hypotheses were generated with the closed form method. The resulting trajectory is plotted in Figure 4.15(b). The histogram of differences between poses computed with the five-point method and our method is shown in Figure 4.14.

4.9.2 Structure from Motion Results with a Camera and an IMU

We investigated using our algorithm to combine visual and inertial data by introducing the gravity vector in the camera coordinate system as the directional correspondence. For this data collection, the camera was rigidly mounted on a rig with a Microstrain 3DM-GX1 IMU, and data was synchronously acquired from both devices. We collected an indoor data set and used the visual odometry setup described above to compare the five-point method with our three-points-plus-gravity method. There were no visible points at infinity in this data set, and the reference direction was set to the gravity vector of the IMU in the camera coordinate system. The camera and IMU rig was moved by hand in all six degrees of freedom. The results are presented in Figure 4.16. In this data set, RANSAC with the five-point hypothesis generator generally performed similarly to our method, but failed to accurately recover relative pose for one of the frames, resulting in a jump near the bottom left of the trajectory, and failure to close the loop.

4.10 Conclusion

In this chapter we presented two efficient algorithms to determine relative pose from three image point correspondences and a directional correspondence. We showed that the algorithm based on algebraic geometry yields better numerical performance than the simpler, closed-form algorithm, but the differences are not significant in most realistic settings. However, we also demonstrated that in certain difficult imaging configurations, the action matrix method can perform better in single precision implementation, and is therefore recommended for processors with 32-bit floating-point

arithmetic where extra accuracy is required.

In our comparison with the five-point method, we showed that the more constrained three-plus-one method does a better job of estimating rotations than the five-point method, as expected. We also showed that both the closed-form and action-matrix implementations are faster than the five-point method, making it even more attractive for real-time applications.

Another attribute of our algorithm is its non-degeneracy for colinear world points in general, however, we have identified and confirmed experimentally three degenerate configurations: when all world points lie on the horopter [3], and when the three world points are lie on a line parallel to the direction of motion or the reference direction.

We also demonstrated that the three-plus-one algorithm can provide accurate and robust results in real-world settings when used with RANSAC and bundle adjustment, and can be used to perform complete six degree of freedom visual odometry for outdoor scenes with or without aid from an IMU. We demonstrated that in those settings, our method exhibits better robustness then the five-point method when used with RANSAC due to having a smaller minimal data set. We believe that the real power of this algorithm is that it can be used as a complement to the five-point algorithm to increase the reliability of visual navigation systems, while improving speed.

4.11 Closed-form Coefficients

In this section we list the coefficients for the closed-form solution presented in Section 4.4. The coefficients a_{ij} are found in equation (4.3.4). The coefficients g_i in the

polynomial (4.4.4) are as follows:

$$
\begin{aligned}
g_1 &= -a_{11}a_{23}a_{32} + a_{13}a_{21}a_{32} - a_{12}a_{21}a_{33} + a_{11}a_{22}a_{33} + a_{23}a_{12}a_{31} - a_{13}a_{22}a_{31} \\
g_2 &= -a_{24}a_{11}a_{32} + a_{14}a_{21}a_{32} - a_{12}a_{21}a_{34} + a_{11}a_{22}a_{34} + a_{12}a_{24}a_{31} - a_{14}a_{22}a_{31} \\
g_3 &= -a_{23}a_{16}a_{31} + a_{13}a_{26}a_{31} + a_{23}a_{12}a_{35} - a_{13}a_{22}a_{35} - a_{11}a_{26}a_{33} + a_{15}a_{22}a_{33} \\
&\quad + a_{21}a_{16}a_{33} - a_{25}a_{12}a_{33} - a_{15}a_{23}a_{32} + a_{13}a_{25}a_{32} + a_{11}a_{23}a_{36} - a_{13}a_{21}a_{36} \\
g_4 &= -a_{23}a_{16}a_{32} - a_{24}a_{16}a_{31} + a_{13}a_{26}a_{32} + a_{14}a_{26}a_{31} - a_{11}a_{23}a_{35} + a_{12}a_{24}a_{35} \\
&\quad + a_{13}a_{21}a_{35} - a_{14}a_{22}a_{35} - a_{11}a_{26}a_{34} - a_{12}a_{26}a_{33} + a_{15}a_{22}a_{34} - a_{15}a_{21}a_{33} \\
&\quad + a_{21}a_{16}a_{34} + a_{22}a_{16}a_{33} - a_{25}a_{12}a_{34} + a_{25}a_{11}a_{33} + a_{15}a_{23}a_{31} - a_{15}a_{24}a_{32} \\
&\quad - a_{13}a_{25}a_{31} + a_{14}a_{25}a_{32} + a_{24}a_{11}a_{36} + a_{23}a_{12}a_{36} - a_{13}a_{22}a_{36} - a_{14}a_{21}a_{36} \\
g_5 &= -a_{24}a_{16}a_{32} + a_{14}a_{26}a_{32} - a_{24}a_{11}a_{35} + a_{14}a_{21}a_{35} - a_{12}a_{26}a_{34} - a_{15}a_{21}a_{34} \\
&\quad + a_{22}a_{16}a_{34} + a_{25}a_{11}a_{34} + a_{15}a_{24}a_{31} - a_{14}a_{25}a_{31} + a_{12}a_{24}a_{36} - a_{14}a_{22}a_{36} \\
g_6 &= -a_{23}a_{16}a_{35} + a_{13}a_{26}a_{35} - a_{15}a_{26}a_{33} + a_{25}a_{16}a_{33} + a_{15}a_{23}a_{36} - a_{13}a_{25}a_{36} \\
g_7 &= -a_{24}a_{16}a_{35} + a_{14}a_{26}a_{35} - a_{15}a_{26}a_{34} + a_{25}a_{16}a_{34} + a_{15}a_{24}a_{36} - a_{14}a_{25}a_{36},
\end{aligned}
\tag{4.11.1}
$$

where a_{ij} come from (4.3.4). The coefficients of the quartic polynomial in c are

$$
\begin{aligned}
h_0 &= -g_1^2 - 2g_1g_6 - g_6^2 + g_3^2 \\
h_1 &= 2g_3g_2 - 2g_4g_6 + 2g_3g_7 - 2g_4g_1 \\
h_2 &= -g_4^2 + g_1^2 + g_6^2 + g_2^2 + g_7^2 - 2g_3^2 + 2g_1g_6 + 2g_2g_7 + 2g_3g_5 \\
h_3 &= 2g_4g_1 + 2g_4g_6 + 2g_5g_2 + 2g_5g_7 - 2g_3g_2 - 2g_3g_7 \\
h_4 &= g_4^2 + g_5^2 + g_3^2 - 2g_3g_5.
\end{aligned}
\tag{4.11.2}
$$

The quartic equation built from the coefficients h_i yields the solution for c.

4.12 Elimination Template Matrix

In this section we list the coefficients of the 21×25 elimination matrix described in Section 4.6.2. The coefficients a_{ij} are found in equation (4.3.4). The first fifteen rows are arranged in five sets of three rows that are the coefficients of equations in (4.3.2) after multiplication by variables listed in equation (4.6.1). Each row below is repeated for each $i = 1, 2, 3$:

$$\begin{bmatrix} a_{i4} & a_{i3} & 0 & 0 & 0 & 0 & a_{i5} & 0 & a_{i1} & 0 & 0 & -a_{i2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{i6} & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & a_{i4} & a_{i1} & a_{i3} & 0 & 0 & 0 & 0 & 0 & a_{i5} & 0 & a_{i1} & -a_{i2} & 0 & 0 & 0 & a_{i6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & a_{i2} & a_{i4} & a_{i1} & a_{i3} & 0 & 0 & 0 & 0 & 0 & a_{i5} & 0 & 0 & 0 & 0 & 0 & a_{i1} & -a_{i2} & 0 & a_{i6} & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & a_{i2} & a_{i4} & 0 & a_{i1} & a_{i3} & 0 & 0 & 0 & a_{i5} & 0 & -a_{i2} & 0 & 0 & 0 & a_{i1} & 0 & 0 & a_{i6} & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{i2} & a_{i1} & a_{i3} & 0 & a_{i5} & a_{i4} & 0 & 0 & 0 & a_{i1} & -a_{i2} & a_{i6} \end{bmatrix}$$

The last six rows of the matrix come from the coefficients of equation (4.3.3) after multiplication:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

This matrix can now be used for the reduction and action matrix extraction, as described in Section 4.6.3.

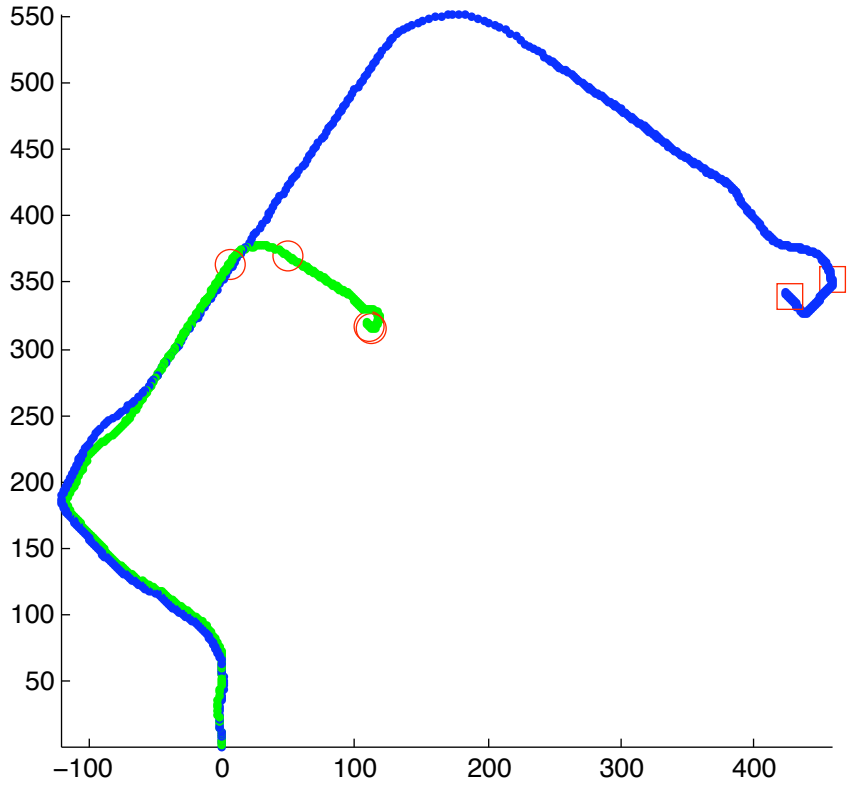


Figure 4.13: Estimated camera trajectories for the outdoor data set using our three-plus-one method (blue) and the five-point algorithm (green). The red squares and circles indicate places where scale was lost, and trajectory was manually stitched together. Given the same input, our method jumped twice, while the five-point method jumped four times. The scale was not reset after stitching, so each piece of the trajectory has a different scale. Since the translation is up to scale, the translation units are set arbitrarily. The total length of the track in the real world was about 430m, of which we were able to travel about 230m before the first break under challenging imaging conditions. Sample frames are shown in Figure 4.11.

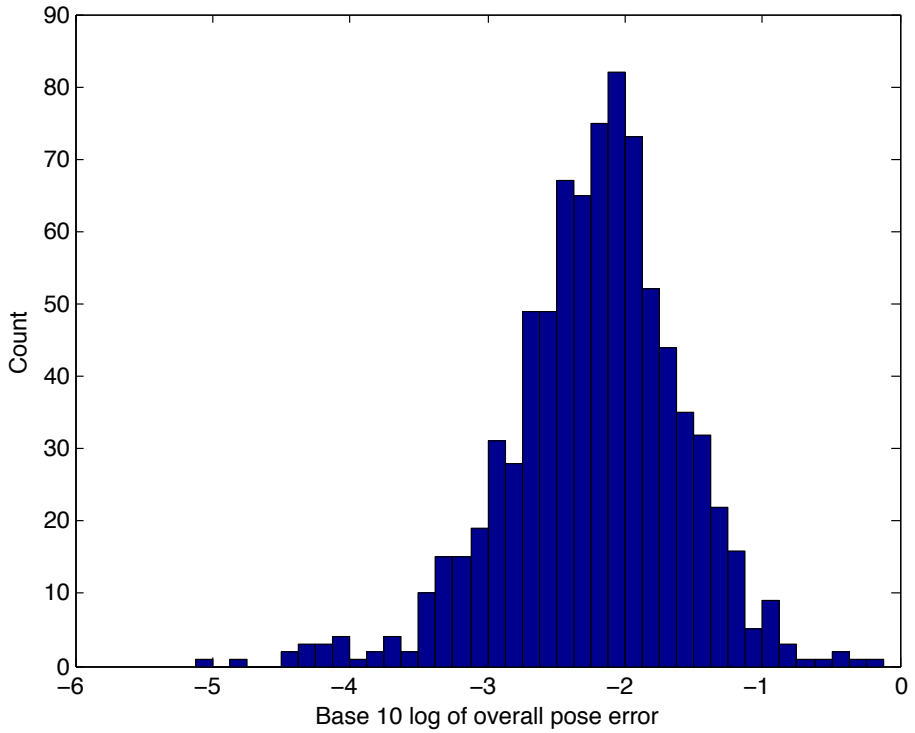
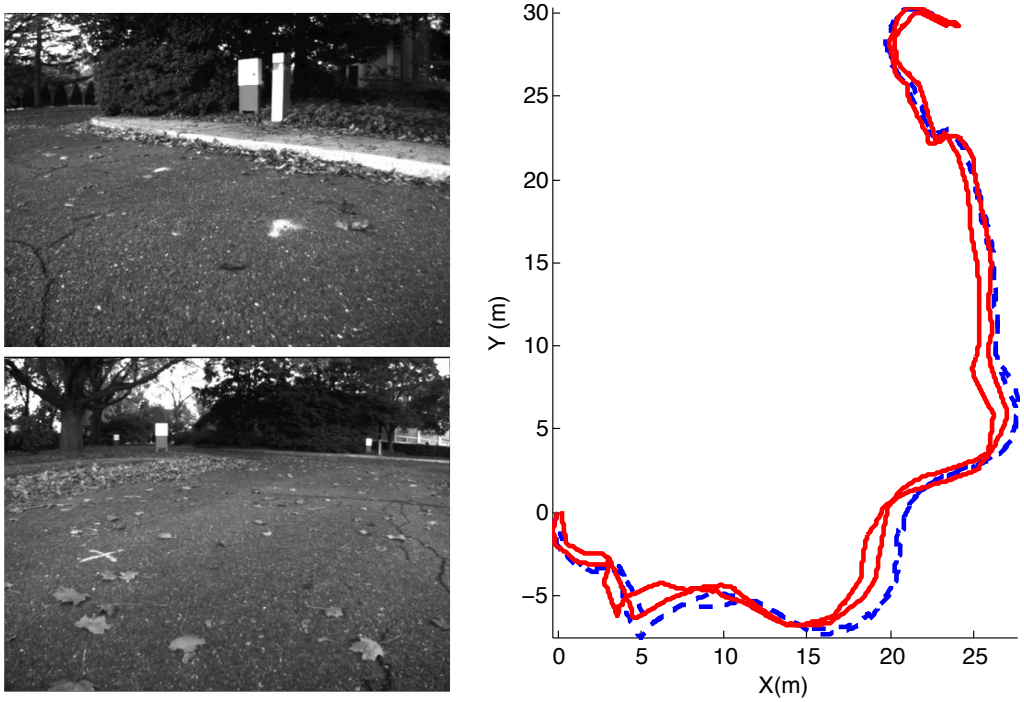


Figure 4.14: The histogram of relative error in the visual odometry relative pose estimates between the three-plus-one and the five-point algorithms. The error measured is the Frobenius norm of the difference between the three-plus-one and five-point estimated pose matrices. The poses for a total of 824 consecutive frame pairs (see Figure 4.11) were computed.



(a) (b)

Figure 4.15: (a) Sample images from the 2582-frame mobile robot data set. (b) Trajectories obtained using visual odometry with the proposed hypothesis generator and the ground truth collected using a Topcon tracking total station. The three-plus-one visual odometry (solid red) was manually scaled (with a single overall scale and a correction factor for scale drift) and aligned with the ground truth (dashed blue). The results demonstrate that the algorithm performs correctly in outdoor scenes.

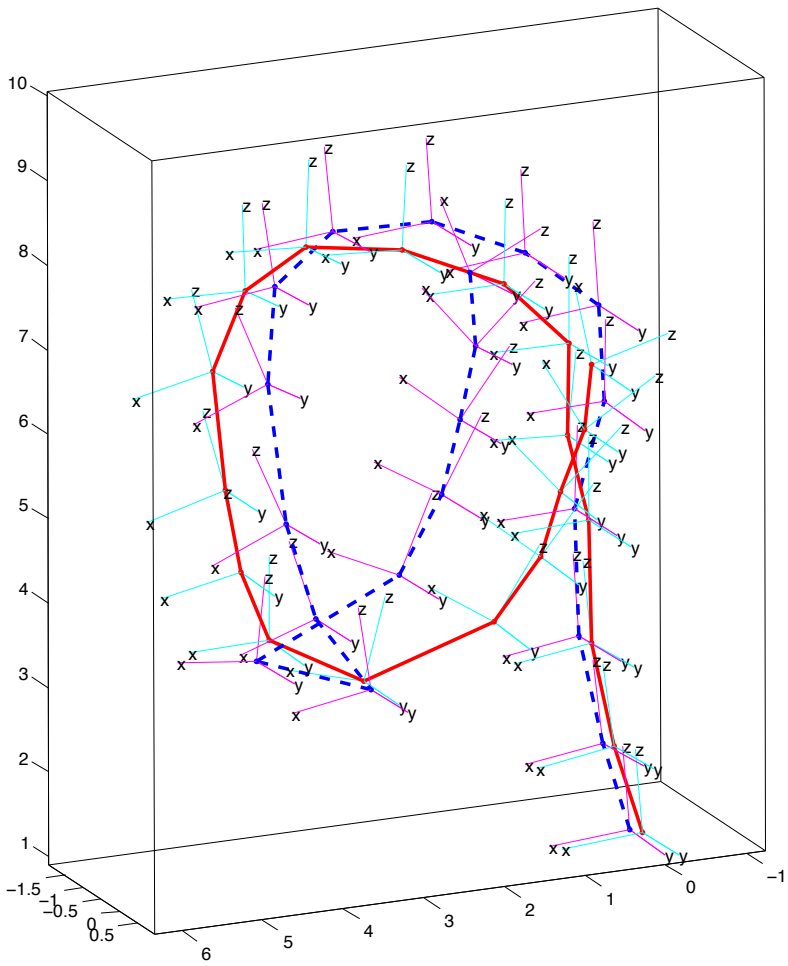


Figure 4.16: Camera trajectories from a short segment of an indoor dataset where the reference direction was provided by the IMU. The red solid lines and dashed blue lines connect the centers of projection determined with our method, the five-point method, respectively. The coordinate axes attached to each point show the rig's relative orientation in space. The motion was approximately a loop, produced by hand, while exercising all six degrees of freedom, as seen by the orientation of the coordinate axis. The five point method jumps at one place, and fails to close the loop. The translation units are arbitrary since the translation was estimated up to scale, but the total length of the track was about 2m.

Figure 4.17: A sample frame from the sequence which was used to generate results in Figure 4.16. The IMU was used to provide directional reference (gravity), as opposed to the outdoor data where points at infinity were used.

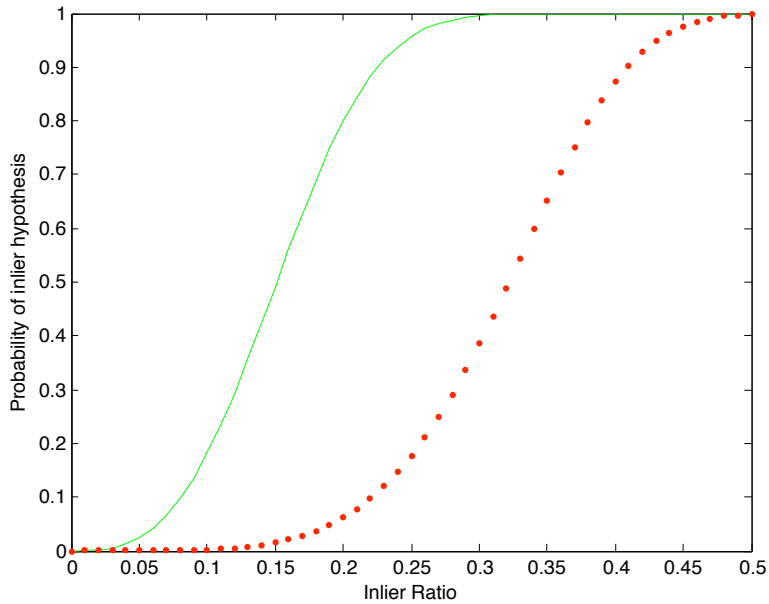


Figure 4.18: The probability p of catching only inliers in an iteration as a function of the inlier ratio w is $p = 1 - (1 - w^n)^k$ for k iterations and $n = 3, 5$ the size of minimal data sets. We plot for $k = 100$ and $k = 200$ and observe the significant difference when inliers are less than 50% of the data [23].

Chapter 5

A Minimal Problem for Camera-LIDAR Alignment

5.1 Introduction

With a widespread use of range sensors such as LIDAR in computer vision and robotics, the need to calibrate them with respect the other sensors and the vehicle becomes essential. In this chapter we will look at a problem of automatically calibrating a LIDAR-Camera rig by developing a solution to a minimal problem which takes as input correspondences between image features and LIDAR samples. A camera provides a dense, color image of the environment, and LIDAR gives a sparse, but accurate, depth map. Fusion of visual and distance information is challenging when there is parallax between the two sensors and when distance consists of a single line scan. Unlike in structured light techniques, the laser is not visible in the image, hence we have to invent a way to associate features on the line scan with features in the image if we want to eliminate a manual selection of the features.

This chapter gives a complete solution to this problem, allowing automatic,

initialization-free calibration of the sensors, assuming only overlapping fields of view. We demonstrate via a series of synthetic and real experiments that our method, which is based on minimal solutions and RANSAC [23], is numerically stable and accurate for realistic calibration scenarios.

This method, first presented at [49] can be used with automatically computed correspondences between lines in the image and 3D points returned by the LIDAR. We formulate and solve the minimal problem using these correspondences. We show that the minimum number of constraints is six, and that there are at most four distinct solutions. We derive a closed-form solution to the problem using variable elimination and resultants.

5.2 Related Work

The closest and most cited work to ours is by Zhang and Pless [73] who match a scanline to a checkerboard. When moving a checkerboard, traditional camera calibration [6] can extract the normal to the checkerboard with respect to a global camera reference, while detection of a line in the laser profile enables association of 3D-points with the calibration plane. The algorithm starts with a linear initialization, suffering under the well known effects of linearization like finding 3x3 matrices satisfying the data equation and then finding the closest special orthogonal matrix. Mei and Rives [43] have applied the same principle to catadioptric images but they exploit the association of a 3D-line (in terms of direction and an offset) to a calibration plane. It is worth noticing that the equation associating the plane normal to the 3D-line direction is of the form $\mathbf{n}^\top R\mathbf{d} = 0$, is algebraically the same as ours after eliminating the translation. However, the authors use, similarly to [73], the association of points.

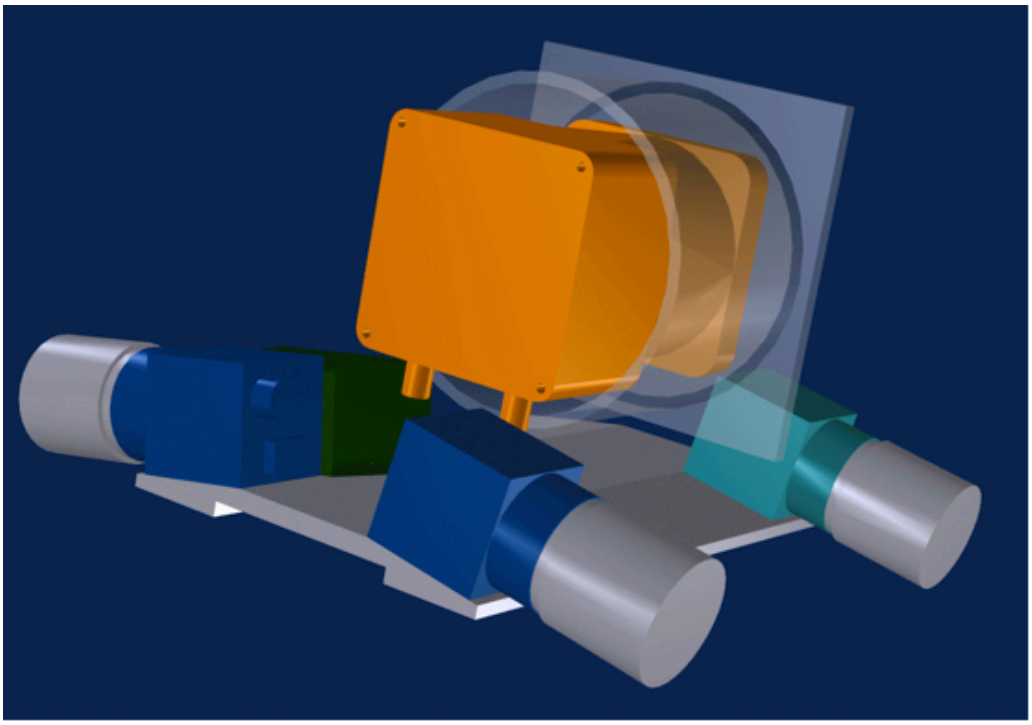


Figure 5.1: A capture rig incorporating four cameras and a Hokuyo LIDAR. Our method is intended to automatically calibrate such systems.

When a laser system produces a full depth map at once, the only challenge is associating features. In [70], a similar to [73] association of 3D points to camera planes is used. In [57] an IMU enables the registration of line scans into a 3D LIDAR and the relative transformation is found via hand-eye calibration [28]. Scaramuzza [60] uses the association of hand-clicked points in a full 3D map with points in catadioptric images.

5.3 Problem Description

Let us formally define the problem of camera-to-LIDAR calibration. We are given a rig consisting of a calibrated camera (intrinsic parameters are known) and scan line LIDAR that are rigidly mounted with respect to each other. Our goal is to find the

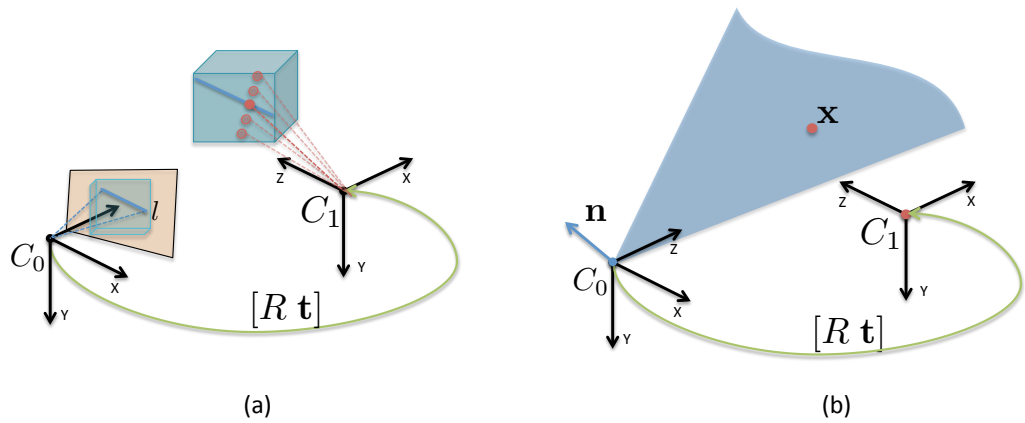


Figure 5.2: (a) The correspondence between the line l in the image (coordinate frame C_0) and 3D point produced by the LIDAR (coordinate frameC_1). (b) A geometric interpretation of the correspondence. A 3D plane through the origin with a normal $\mathbf{n}$ in C_0 corresponds to a 3D point $\mathbf{x}$ in coordinate frame C_1.

rigid transformation $[R|\mathbf{t}]$ such that given a 3D point $\mathbf{x} = [x_1, x_2, x_3]^\top$ obtained by the LIDAR, we can compute the corresponding point $\mathbf{y}$=$[y_1, y_2, y_3]^\top$ in the camera's coordinate system (and then in the image via the intrinsic calibration) as $\mathbf{y} = R\mathbf{x}+\mathbf{t}$.

A single LIDAR datum consists of a depth and angle at which the depth was sensed. We define the coordinate system for the LIDAR as follows. The origin is the center of laser sensor rotation, and the plane of laser rotation is the Y-Z plane. Consequently, we can always express a 3D LIDAR point as $\mathbf{x} = [0, x_2, x_3]^\top$. The geometry of the correspondence is illustrated in Figure 5.2.

As with any calibration problem from sensor data, we must collect correspondences between the readings of different sensors. In this case, we construct a calibration target containing a single black to white transition (see Figure 5.3), which we detect as a line segment in the image and a point in the LIDAR's luminance output for a single line scan (see Figure 5.4). We discuss feature detection in detail in Section 5.6.2. A line in the image corresponds to a plane in the world containing the line and the center of projection of the camera. We now have a correspondence

between a 3D point in the LIDAR coordinate system and a plane in the camera's coordinate system. Thus our constraint is that the LIDAR point, taken into the camera's coordinate system, must lie on the corresponding plane. We express this constraint as

$$\mathbf{n}^\top(R\mathbf{x} + \mathbf{t}) = 0, \tag{5.3.1}$$

where $\mathbf{n}$ is the normal to the plane in the camera coordinate system and $\mathbf{x}$ is the LIDAR point.

Figure 5.3: A single camera frame from the calibration data set showing the calibration object. The object consists of a black line on a white sheet of paper. We detect the white-to-black transition looking from the top of the image.

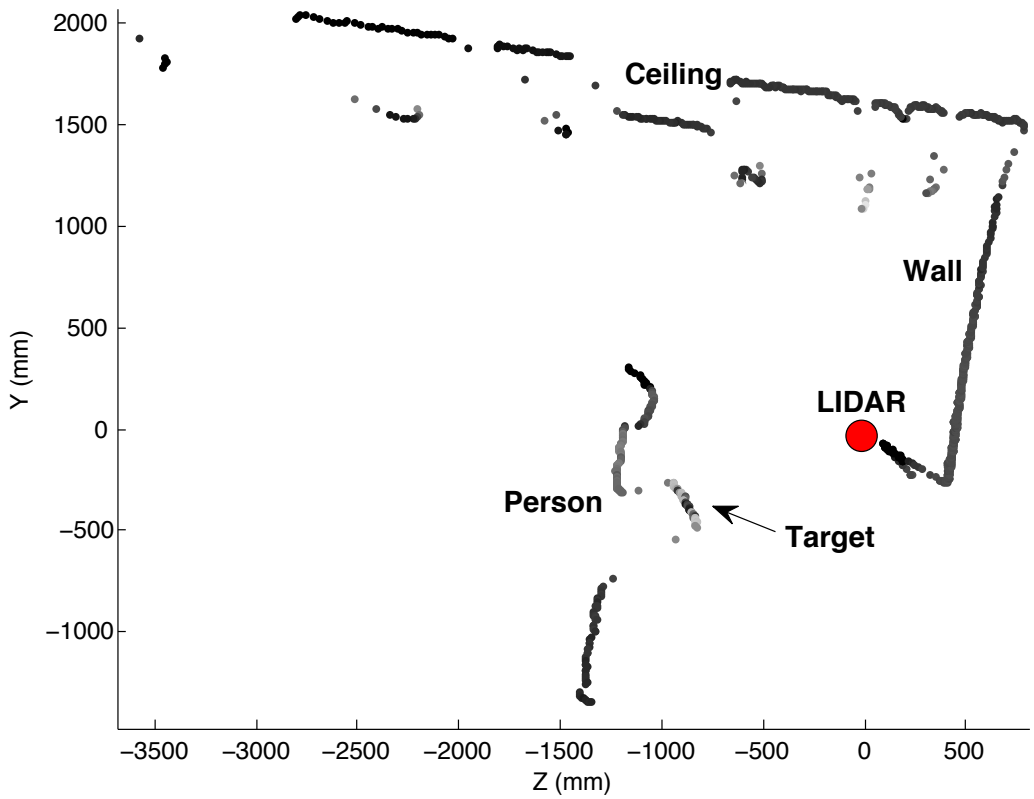


Figure 5.4: A portion of a LIDAR scan showing a person holding the calibration target. The points are colored by their intensity returns. The LIDAR's scan plane is close to vertical, and its origin is marked by a circle.

5.4 The Polynomial System

In this section we show how to formulate the problem as a set of polynomial equations. Our original point correspondence constraints have the form

$$\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t}) = 0 \tag{5.4.1}$$

for $i = 1, \ldots, 6$, where $R \in \mathbf{SO}(3)$ and $\mathbf{t}$, $\mathbf{n}_i$, $\mathbf{x}_i$ and $\mathbf{t}$ are 3-vectors. Let $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ be the columns of R. Since we set up the LIDAR coordinate system in such a way that the first component of each $\mathbf{x}_i$ is zero, we do not have an explicit dependence

on $\mathbf{r}_1$. By taking the cross product

$$\mathbf{r}_1 = \mathbf{r}_2 \times \mathbf{r}_3, \tag{5.4.2}$$

we can recover the first column of R from the other two. Since R is orthonormal, we can add the following constraints

$$\begin{aligned} \mathbf{r}_3^\top \mathbf{r}_3 &= 1 \\ \mathbf{r}_2^\top \mathbf{r}_2 &= 1 \\ \mathbf{r}_3^\top \mathbf{r}_2 &= 0. \end{aligned} \tag{5.4.3}$$

We expand the constraints in equation (5.4.1), to obtain linear equations of the form

$$\begin{bmatrix} \mathbf{a}_i^\top & \mathbf{a}_i'^\top \end{bmatrix} \begin{bmatrix} \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix} + \mathbf{n}_i^\top \mathbf{t} = 0 \tag{5.4.4}$$

for $i = 1, \ldots, 6$, where the constant vectors $\mathbf{a}_i$ and $\mathbf{a}_i'$ are expressed in terms of input data in the following way:

$$\begin{bmatrix} \mathbf{a}_i \\ \mathbf{a}_i' \end{bmatrix} = \begin{bmatrix} \mathbf{n}_i x_{i2} \\ \mathbf{n}_i x_{i3} \end{bmatrix}. \tag{5.4.5}$$

Our system now has 6 linear, homogenous equations in 9 unknowns corresponding to the point-to-plane correspondences, and three second order equations constraining the rotation matrix. In the next two sections we will present a closed-form solution to this polynomial system.

5.5 The Closed-form Solution

Since we have linear constraints (5.4.1), and require a six-degree-of-freedom solution, it is not surprising that we need a minimum of six correspondences. It may seem surprising at first, however, that this polynomial system can be solved in closed-form, despite having eight solutions. First, we briefly outline the following steps to eliminate eight of the variables from the system, solve a univariate polynomial and then back substitute:

1. Symbolically solve for the translation component $\mathbf{t}$ using the first three linear equations, and substitute into the three remaining linear equations.

2. Symbolically solve three of the six remaining equations for the components of $\mathbf{r}_2$ and substitute into remaining equations.

3. Use the Macaulay resultant (see Chapter 7, §6 in [17]) to symbolically eliminate two of the three remaining variables from the three remaining equations, and solve the resulting quartic equation in closed form for r_{33}.

4. Substitute the solutions into the three equations, and use the Sylvester resultant (see Chapter 3, §5 in [17]) of two of them to eliminate one of the two remaining variables, and solve the resulting quartic equation in closed form.

5. Test the solutions to obtain the one satisfying all equations.

6. Back-substitute for the variables in $\mathbf{r}_2$ and $\mathbf{t}$.

The symbolic templates generated by this process will remain the same across all instances of the problem, so they only need to be computed once, so that solving an instance of a problem can be accomplished by substitution of the data into the expressions for the solution.

We will now describe the solution in detail. Let us first eliminate $\mathbf{t}$ from the first three constraints of the form (5.4.1). These are linear equations, so we can express $\mathbf{t}$ as

$$\mathbf{t} = -\begin{bmatrix} \mathbf{n}_1^\top \\ \mathbf{n}_2^\top \\ \mathbf{n}_3^\top \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{n}_1^\top R\mathbf{x}_1 \\ \mathbf{n}_2^\top R\mathbf{x}_2 \\ \mathbf{n}_3^\top R\mathbf{x}_3 \end{bmatrix}. \tag{5.5.1}$$

In the above system, the components of $\mathbf{t}$ are expressed in terms of the remaining variables $\mathbf{r}_2$ and $\mathbf{r}_3$. Let us call the coefficients of the remaining variables b_{ij}. The three equations in (5.5.1) then become

$$t_i = \begin{bmatrix} \mathbf{b}_i^\top & \mathbf{b}_i'^\top \end{bmatrix} \begin{bmatrix} \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix}, \tag{5.5.2}$$

for $i = 1, \ldots, 3$ which is linear in the the components of $\mathbf{r}_2$ and $\mathbf{r}_3$.

We now substitute for $\mathbf{t}$ in the three remaining constraints of the form (5.4.1) for $i = 4, \ldots, 6$ to get three linear constraints in $\mathbf{r}_2$ and $\mathbf{r}_3$.

$$\begin{bmatrix} \mathbf{c}_4^\top \\ \mathbf{c}_5^\top \\ \mathbf{c}_6^\top \end{bmatrix} \mathbf{r}_2 + \begin{bmatrix} \mathbf{c}_4'^\top \\ \mathbf{c}_5'^\top \\ \mathbf{c}_6'^\top \end{bmatrix} \mathbf{r}_3 = 0, \tag{5.5.3}$$

where c_{ij} and c'_{ij} are the coefficients (expressed once again entirely in terms of problem data)

$$\begin{aligned} \mathbf{c}_i &= \mathbf{a}_i + \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \end{bmatrix} \mathbf{n}_i \\ \mathbf{c}_i' &= \mathbf{a}_i' + \begin{bmatrix} \mathbf{b}_1' & \mathbf{b}_2' & \mathbf{b}_3' \end{bmatrix} \mathbf{n}_i. \end{aligned} \tag{5.5.4}$$

Let C and C' be the 3×3 matrices of coefficients c_{ij} and c'_{ij}. Using the linear system (5.5.3), we can express $\mathbf{r}_2$ as

$$\mathbf{r}_2 = -C^{-1}C'\mathbf{r}_3. \tag{5.5.5}$$

We can now eliminate $\mathbf{r}_2$ from the system by substituting equation (5.5.5) into the three second order constraints (5.4.3), arriving (after full expansion) at the system

$$r_{13}^2 + r_{23}^2 + r_{33}^2 = 1 \tag{5.5.6}$$

$$e_{11}r_{13}^2 \;+\; e_{12}r_{13}r_{23} \;+\; e_{13}r_{23}^2 \;+\; e_{14}r_{13}r_{33} \;+\; e_{15}r_{23}r_{33} \;+\; e_{16}r_{33}^2 \;=\; 1 \tag{5.5.7}$$

$$e_{21}r_{13}^2 \;+\; e_{22}r_{13}r_{23} \;+\; e_{23}r_{23}^2 \;+\; e_{24}r_{13}r_{33} \;+\; e_{25}r_{23}r_{33} \;+\; e_{26}r_{33}^2 \;=\; 0, \tag{5.5.8}$$

where the constants e_{ij} are computed in closed form in terms of entries of C^{-1} and C' after the substitution. These coefficients are given explicitly in the supplemental material[1]. We can observe that given a set of values that satisfy this system, the negative values will also satisfy it. Thus, we expect to solve this system via a quartic polynomial in squares of the variables instead of an eighth degree.

While we cannot directly solve for any of the variables in the system formed by the last three equations, nor can we reduce the number of variables further by rearranging the system, we can still obtain a univariate polynomial in r_{33} by using the Macaulay resultant of the three equations. This multivariate resultant is the quotient of the determinants of the numerator matrix (5.5.11) and the denominator

[1]The MATLAB function, which includes all coefficients, can be obtained from www.cis.upenn.edu/~narodits/SolveLidarCalibrationOpt.m.

$$
\begin{bmatrix}
e_{11} & 0 & 0 & 0 & 0 & 0 & e_{13} & 0 & 0 & 0 & 0 & e_{16}r_{33}^2-1 & e_{15}r_{33} & e_{14}r_{33} & e_{12} \\
0 & e_{11} & 0 & 0 & 0 & 0 & e_{15}r_{33} & e_{13} & 0 & e_{12} & 0 & 0 & e_{16}r_{33}^2-1 & 0 & e_{14}r_{33} \\
0 & 0 & e_{11} & 0 & 0 & 0 & e_{16}r_{33}^2-1 & e_{15}r_{33} & e_{13} & e_{14}r_{33} & e_{12} & 0 & 0 & 0 & 0 \\
e_{14}r_{33} & e_{12} & 0 & e_{11} & 0 & 0 & 0 & 0 & 0 & e_{13} & 0 & 0 & 0 & e_{16}r_{33}^2-1 & e_{15}r_{33} \\
0 & e_{14}r_{33} & e_{12} & 0 & e_{11} & 0 & 0 & 0 & 0 & e_{15}r_{33} & e_{13} & 0 & 0 & 0 & e_{16}r_{33}^2-1 \\
e_{16}r_{33}^2-1 & e_{15}r_{33} & e_{13} & e_{14}r_{33} & e_{12} & e_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & r_{33}^2-1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & r_{33}^2-1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & r_{33}^2-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & r_{33}^2-1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & r_{33}^2-1 \\
e_{31} & 0 & 0 & 0 & 0 & 0 & e_{33} & 0 & 0 & 0 & 0 & e_{36}r_{33}^2 & e_{35}r_{33} & e_{34}r_{33} & e_{32} \\
0 & e_{31} & 0 & 0 & 0 & 0 & e_{35}r_{33} & e_{33} & 0 & e_{32} & 0 & 0 & e_{36}r_{33}^2 & 0 & e_{34}r_{33} \\
e_{34}r_{33} & e_{32} & 0 & e_{31} & 0 & 0 & 0 & 0 & 0 & e_{33} & 0 & 0 & 0 & e_{36}r_{33}^2 & e_{35}r_{33} \\
0 & e_{34}r_{33} & e_{32} & 0 & e_{31} & 0 & 0 & 0 & 0 & e_{35}r_{33} & e_{33} & 0 & 0 & 0 & e_{36}r_{33}^2
\end{bmatrix}
\tag{5.5.11}
$$

matrix

$$
\begin{bmatrix}
e_{13} & 0 & e_{16} \\
0 & e_{13} & e_{11}r_{13}^2-1 \\
1 & 0 & 1
\end{bmatrix}
\tag{5.5.9}
$$

When we set this resultant to 0, we obtain the following polynomial equation:

$$
g_1 r_{33}^8 + g_2 r_{33}^6 + g_3 r_{33}^4 + g_4 r_{33}^2 + g_5 = 0. \tag{5.5.10}
$$

The coefficients g_i can be computed in closed form using a symbolic mathematics program, such as Maple [45]. They only have seven thousand terms due to the sparsity of the matrix (5.5.11), and are explicitly given in the supplemental material.

The equation (5.5.10) can be solved in closed form as a quartic equation in r_{33}^2, thus giving us up to four real solutions and eight possibilities for r_{33}. We note that since both $[R\,|\,\mathbf{t}]$ and $[-R\,|\,-\mathbf{t}]$ are the solutions to the original system, we only need to back-substitute the positive, real solutions of (5.5.10), and there are up to four of them. Once we obtain the corresponding $[R\,|\,\mathbf{t}]$, simply choose the sign of the overall solution that makes the determinant of R positive.

$$\begin{bmatrix} e_{13} & e_{12}r_{13} + e_{15}\tilde{r}_{33} & e_{11}r_{13}^2 + e_{16}\tilde{r}_{33}^2 - 1 + e_{14}r_{13}\tilde{r}_{33} & 0 \\ 0 & e_{13} & e_{12}r_{13} + e_{15}\tilde{r}_{33} & e_{11}r_{13}^2 + e_{16}r_{33}^2 - 1 + e_{14}r_{13}\tilde{r}_{33} \\ 1 & 0 & -1 + r_{13}^2 + \tilde{r}_{33}^2 & 0 \\ 0 & 1 & 0 & -1 + r_{13}^2 + \tilde{r}_{33}^2 \end{bmatrix} \tag{5.5.12}$$

While we know that each solution for r_{33} corresponds to a single solution for the rest of the variables in the original system, when we substitute the numeric values $\tilde{r}_{33}$ in the equations (5.5.7), (5.5.8), and (5.5.6), we are faced with a system which may have false solutions due to numerical error. While it is possible to substitute the closed-form solutions to the quartic, it is not practical due to the radicals in the quartic formula. Instead, we compute the Sylvester resultant of (5.5.7) and (5.5.6) with respect to r_{23}, which is the determinant of the Sylvester matrix (5.5.12). The resultant is thus a quartic polynomial in r_{13}, and its coefficients are computed symbolically and are given in the supplemental material.

We set the resultant to 0 and obtain the real solutions $\tilde{r}_{13}$ to the equation. We substitute the values into (5.5.8) and solve for the remaining variable r_{23}. It now becomes a simple matter of testing the solutions using equation (5.5.6) to see which solution indeed lies on the sphere. We repeat this process with the remaining real solutions to (5.5.10).

Having obtained up to four sets of solutions for $\mathbf{r}_3$, it is simple to recover solutions for $\mathbf{r}_2$ from (5.5.3), $\mathbf{r}_1$ from (5.4.2), and $\mathbf{t}$ from (5.5.1). As we mentioned before, we must choose the sign of each solution to make the determinant of R positive. We can find the unique solution to the problem by using a 7th correspondence to disambiguate among the four solution. To do this, we substitute the $\mathbf{n}_7$ and $\mathbf{x}_7$ into the constraint equation (5.4.1) for each candidate R and $\mathbf{t}$ and choose the solution with the lowest absolute residual.

The steps described above produce a symbolic template for this problem. By this we mean that given any particular set of six correspondences, we can arrive at the solutions for R and $\mathbf{t}$ only by substituting the data into pre-computed expressions and solving univariate polynomials (in this case of degree 4).

5.6 Results

In this section we will first establish the correctness of our algorithm and explore its sensitivity to noise using simulated data, and then perform a real calibration of a LIDAR-camera rig mounted on a mobile robot for the purpose of coloring the 3D LIDAR points with pixels from the camera and show the results.

5.6.1 Simulations

Our first experiment establishes the correctness and numerical stability of our symbolic template. We generate, uniformly at random, roll, pitch and yaw of the LIDAR with respect to the camera in the range of $\pm 30°$, and translation vectors with uniformly random components from 0 to 30cm. For each of these ground truth calibrations, we generate six noise-free correspondences.

We generate these correspondences by simulating a camera looking at a target. Specifically, we choose two random points in sampling volumes in front of the camera, one on the left and one on the right. This closely models what happens in the real system where the 3D line must intersect the LIDAR scan plane which is close to the vertical plane separating the left and right halves of the image. These two points define a line which is then intersected with the LIDAR plane to obtain the point $\mathbf{x}$. The points, along with the camera center, define the plane and its normal $\mathbf{n}$.

The histogram of errors for 10^5 such configurations is shown in Figure 5.5. The

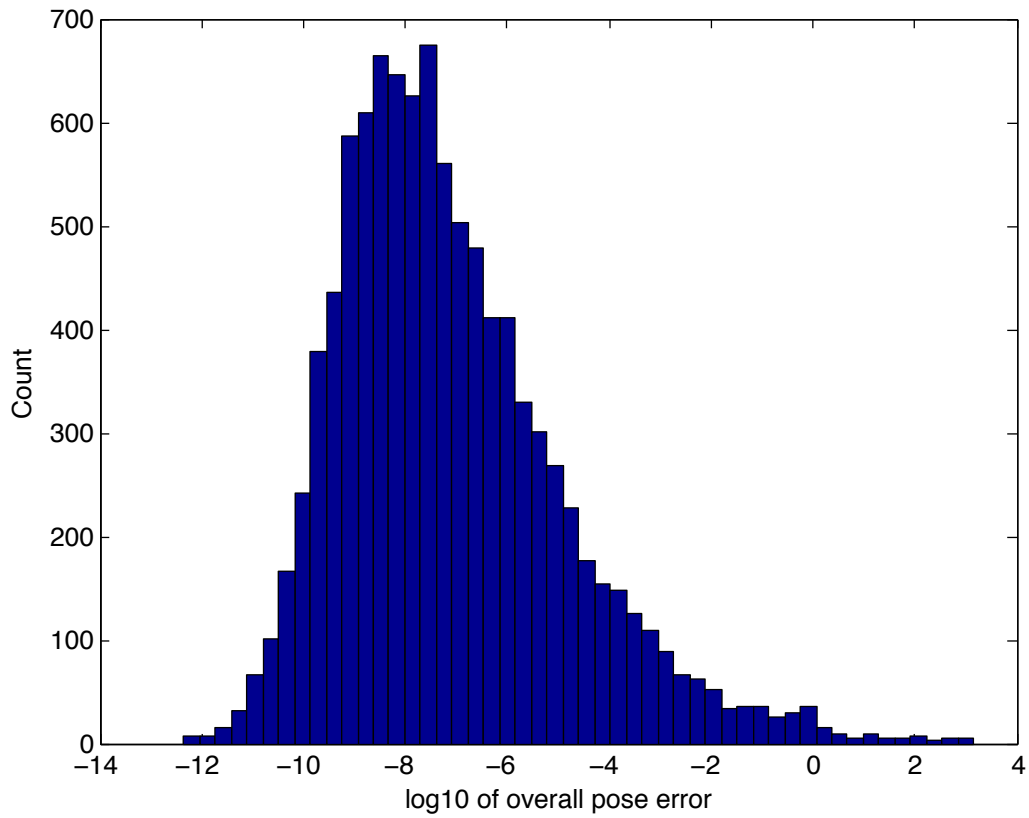


Figure 5.5: The histogram of numerical errors for 10^5 random, noise-free instances of the problem. The error is defined as the $\log_{10} e$, where e of the Frobenius norm of the difference between the ground truth and the computed matrices (see (5.6.1)). Since the points were not checked for degeneracy (such as collinearity), some failures are observed. If we consider a failure to be $\log_{10} e > -1$, then the method fails 1.97% of the time.

error metric is the the following:

$$e = \min_i(\|[R_{gt} \,|\, \mathbf{t}_{gt}] - [R_i \,|\, \mathbf{t}_i]\|_{\mathcal{F}}), \tag{5.6.1}$$

for i from 1 to the number of solutions, R_{gt} and $\mathbf{t}_{gt}$ comprise the ground truth calibration and $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm. The figure demonstrates that our solution correctly solves the calibration problem. The accuracy varies due to the random nature of correspondences and lack of degeneracy checking during sampling.

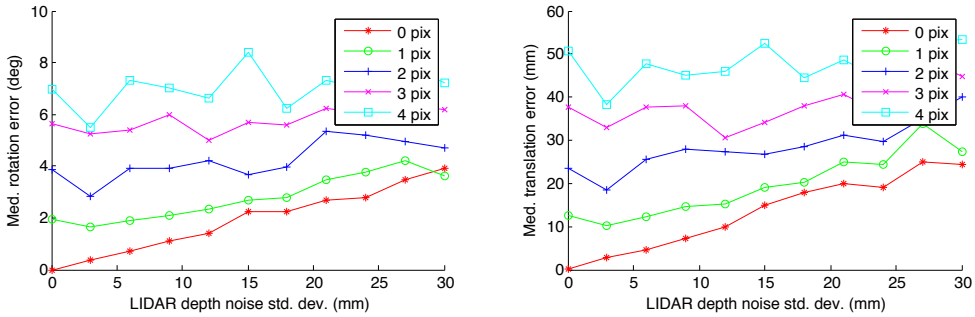


Figure 5.6: Errors in rotation and translation estimation for a simulated rig with a 100mm distance between the camera and LIDAR. Each point shows median error for 200 random configurations of LIDAR-image correspondences. Each sequence corresponds to different levels of image noise plotted against LIDAR noise. The noise values are the standard deviations. The image errors are line translation error (pix) for the baseline camera described in Section 5.6.1.

We now profile the algorithm with respect to noise in sensor data. We consider three sources of error: depth uncertainty in the LIDAR points and misestimation of position and rotation of the corresponding lines in the image, both of which lead to the 3D point being some distance off the plane through the line and center of the camera. For the LIDAR depth error, the Hokuyo UTM-30LX specifies the standard deviation of 30mm for ranges less than 1m. We will study the accuracy for noise standard deviations of 0mm to 30mm. The image processing accuracy will be in pixels with respect to a baseline calibrated camera, which we define as a 640×480 pixel sensor with a $60°$ field of view. While in the real data the errors in line extraction will depend on the length of the edge segments extracted, we will use the range of 0 to 4 pixel standard deviations in the simulated results. The results for different error levels are shown in Figure 5.6, and demonstrate a greater sensitivity to image noise than depth noise.

5.6.2 A Real Calibration

The sensor rig for the real-world calibration experiment consists of a calibrated 640×480 Flea2 camera with a $77°$ field of view and a Hokuyo UTM-30LX line scanner, with angular resolution of 0.25 deg (see Figure 5.1).

Images of the calibration target (see Figures 5.3 and 5.4) are captured synchronously by the two sensors at various positions. We detect the target as follows. For the LIDAR calibration target detection we use the line scan, including intensity, returned from the device in the region of interest for calibration. First, the derivatives of the intensity vector of the line scan are computed using a difference of gaussians filter. The peaks of this derivative signal are detected by non-maximal suppression. This detects both rising and falling edges in the intensity signal. We then improve the estimate of the edges by fitting a line to all the neighboring 3D points and projecting the intensity edge sample point onto that line to give us the final LIDAR feature point $\mathbf{x}_i$. The discrete sampling of the angle could cause an angular error, but since we can control the target's location during calibration, this error can be controlled. Even if the LIDAR and camera are further apart as on a larger robot, a long target can be used which could keep the target close to both sensors.

The first two steps in image line extraction are radial distortion removal and edge detection [16]. The edgels are then combined using the Hough transform to output line segments. We define "linescore" as is a measure of how well the gradient information in the image fits with the proposed line. We compute this on the line segments to orient them and prune out ones with poor support. In order to define linescore, first define Q_j as the set of pixels which lie within 1 pixel of the line segment j to score, and define $\mathbf{g}_i$ to be the gradient at pixel i, and $\mathbf{m}_j$ to be the normal to

the line segment j which we are scoring. $linescore_j = \sum_{i \in Q_j} \mathbf{g}_i^\top \mathbf{m}_j$ measures the support in the gradient image for each line.

Next, the candidate edges are pruned more using the following heuristics, which use the fact that our target contains a single black stripe on a white background. The heuristics look for pairs of line segments which contain a white to black transition followed by a black to white transition. To do this, we look at the normal direction combined with the linescore to propose candidate pairings respecting this. For each candidate pairing we perform a number of tests:

1. Check that the candidate pair's normals are close to parallel while accounting for possible perspective distortions.

2. The lines are required to be close together because the target never fills up too much of the field of view.

3. Check that the ratio of width the height of the rectangle created by the pair of line segments is appropriate. This is a check that the target has the correct aspect ratio.

4. Overlap is computed and thresholded to rule out line segments which don't occur next to each other.

Once these criteria are passed we take the two endpoints of each line, and record the normal to the plane $\mathbf{n}_i$ passing through these points and the camera center. Thus the vector $\mathbf{n}_i$ corresponding to the LIDAR point $\mathbf{x}_i$ become the input for RANSAC process. We then compute a robust calibration solution using around 400 correspondences in the RANSAC framework. The process chooses the best calibration hypothesis based on six correspondences which we then iteratively refine using all of the inliers.

We tested the quality of our calibration. Our rig is equipped with stereo cameras that we calibrated using the Camera Calibration Toolbox [6]. We computed the error in the LIDAR-camera calibration by observing that we can compute the left-to-right camera calibration by combining the left camera and right camera to LIDAR calibrations. We define an error metric as follows. Let us denote a calibration transformation $P_q = \left[\begin{smallmatrix} R_q & \mathbf{t}_q \\ 0\,0\,0 & 1 \end{smallmatrix}\right]$. We can define the error matrix

$$P_{err} = P_{lr} P_{rh} P_{lh}^{-1},$$

where P_{lr} is the left-to-right calibration computed using the calibration toolbox, and P_{rh} and P_{lh} are the left camera and right camera to Hokuyo calibrations, respectively, computed with our method. If both of our calibrations were accurate, R_{err} would be close to identity and $\|\mathbf{t}_{err}\|_2$ would be close to 0. The real rotation error was 0.26°, and the translation error was 1.9mm (the distance between the LIDAR and each camera was approximately 140mm).

This calibration was used to color the LIDAR points with the corresponding pixels from the camera. Our system automatically acquired the images and registered LIDAR scans using visual odometry. The registered LIDAR point cloud is shown in Figure 5.7. The same point cloud colored by the camera image pixels is shown in Figure 5.8.

5.7 Conclusion

We overcome the difficulty of calibrating LIDAR-camera rigs by introducing a new algorithm based on the minimal solution to the calibration from line to 3D point

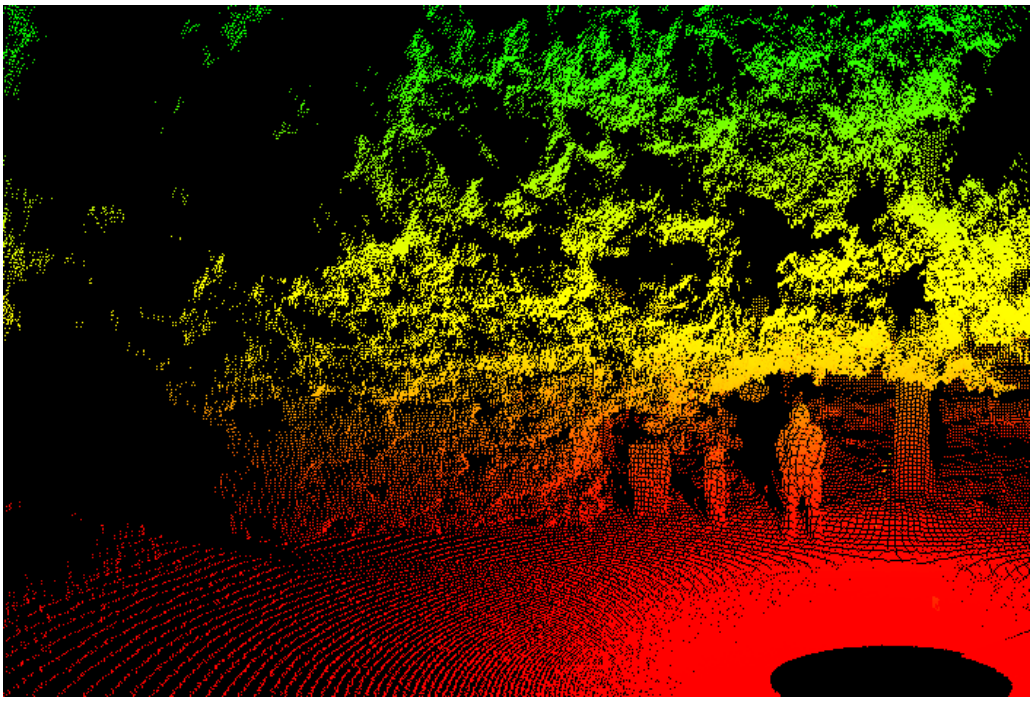


Figure 5.7: A sample LIDAR scan acquired by the mobile robot colored by height.

correspondences, used in a robust framework and without initialization. Using automatic feature detection described, the algorithm can be used to automatically calibrate a variety of sensor platforms. Our experiments with simulated and real data indicate that this method is both correct and practical.

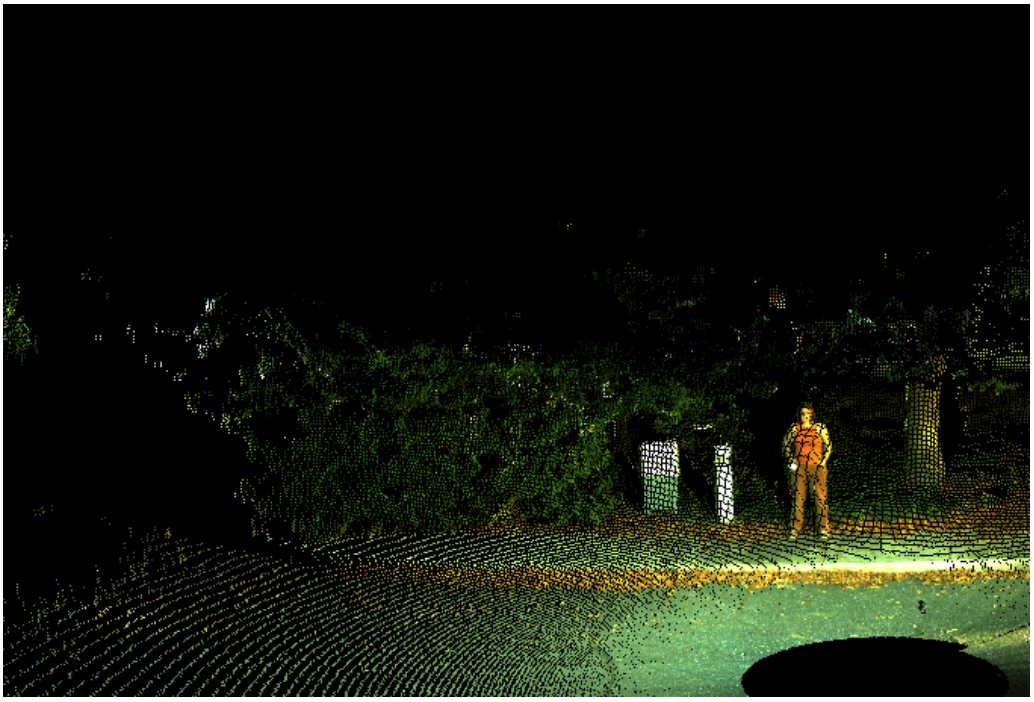


Figure 5.8: A LIDAR scan colored using camera pixels.

Chapter 6

Conclusions

In this dissertation we explored the methods of algebraic geometry as they pertain to solving systems of polynomial equations in the context of geometric computer vision. The work on the solver optimization presented in Chapter 3 opens the door to further study of algebraic properties of polynomial system coefficient matrices in search of faster and more numerically stable representations. While we only proved some properties that rely on linear-algebraic properties of the matrices, exploiting algebraic-geometric structure and sparsity are interesting opportunities for future work. While applications presented are limited to computer vision, the methods developed have potential in any field where a problem can be formulated as a polynomial system with a static algebraic structure.

The three-plus-one visual odometry algorithm in Chapter 4 constitutes an improvement on the conventional techniques both in terms of accuracy and computational efficiency. The formulation of the minimal problem makes it easy to integrate it with existing systems using cameras and/or inertial sensors for navigation. The domains of application and the extent to which the three-plus-one improves systems

employing different combinations of sensors and different visual navigation techniques (such as Kalman filtering, local bundle adjustment and a variety of other SLAM techniques) are exciting areas of research as well.

The LIDAR-camera calibration algorithm from Chapter 5 is a ready-to-use solution to calibration of commonly used robotics platforms. The ease of implementation (due to closed form) and ability to operate without initialization make this algorithm and the system described in the chapter a complete solution for calibrating this type of a system. In addition, the minimal problem itself (six 3D-plane-to-3D-point correspondence problem) may find applications elsewhere, such as in motion recovery given point to plane correspondences derived from other sensors.

Bibliography

[1] Inside data association. *Robotics and Autonomous Systems*, 57(12):1155 – 1156, 2009.

[2] M.P. Abramson. Historical background to Gröbner's paper. *ACM Communications in Computer Algebra*, 43(1/2):22–23, 2009.

[3] M. Armstrong, A. Zisserman, and R. Hartley. Self-calibration from image triplets. *Proc. Fourth European Conference on Computer Vision*, pages 1–16, Cambridge, UK, April 14-18, B. Buxton (Ed.), Springer LNCS 1064 1996.

[4] R. Azuma. The challenge of making augmented reality work outdoors. In Y. Ohta and H. Tamura, editors, *In Mixed Reality: Merging Real and Visual Worlds*, pages 379–390. Springer Verlag, New York et al., 1999.

[5] M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. *The International Journal of Robotics Research*, 23(12):1113, 2004.

[6] Jean-Yves Bouguet. Camera calibration toolbox for matlab. *www.vision.caltech.edu*, 2006.

[7] B. Buchberger. An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Ph.D. dissertation, University of Innsbruck*, Jun 1965.

[8] B. Buchberger. Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of symbolic computation*, 41(3-4):475–511, 2006.

[9] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the P4P problem for camera with unknown focal length. *IEEE Conf. Computer Vision and Pattern Recognition*, Anchorage, AK, June 24-26 2008.

[10] D. Burschka and G.D. Hager. V-GPS (SLAM): Vision-based inertial system for mobile robots. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 409–415, 2004.

[11] M Byröd, M Brown, and K Åström. Minimal solutions for panoramic stitching with radial distortion. *Proc. British Machine Vision Conference*, London, UK, Sept. 24-27 2009.

[12] M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. *Asian Conference on Computer Vision*, Jan 2007.

[13] M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of grobner basis polynomial equation solver. *Proc. Int. Conf. on Computer Vision*, 2007.

[14] M. Byröd, K. Josephson, and K. Åström. Fast and stable polynomial equation solving and its application to computer vision. *International Journal of Computer Vision*, Jan 2009.

[15] M. Byröd, Z. Kukelova, K. Josephson, and T. Pajdla. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. *IEEE Conf. Computer Vision and Pattern Recognition*, Jan 2008.

[16] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[17] D Cox, J Little, and D O'Shea. *Ideals, varieties, and algorithms*. Springer, Jan 1997.

[18] D. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*. Springer, Jan 2005.

[19] M. Demazure. Sur deux problemes de reconstruction. *INRIA Tech. Rep.*, 1988.

[20] D. Diel, P. DeBitetto, and S. Teller. Epipolar constraints for vision-aided inertial navigation. In *IEEE Workshop on Motion and Video Computing*, pages 221–228, 2005.

[21] J. Domke and Y. Aloimonos. Integration of visual and inertial information for egomotion: a stochastic approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 2053–2059. Citeseer, 2006.

[22] J. Faugere. A new efficient algorithm for computing Gröbner bases (F4). *Journal of pure and applied algebra*, pages 1–29, Oct 1999.

[23] M. Fischler and R. Bolles. Random sample consensus. *Communications of the ACM*, Jan 1981.

[24] F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. *Proc. Eleventh European Conference on Computer Vision*, pages 269–282, Sept. 2010.

[25] M. George and S. Sukkarieh. Inertial navigation aided by monocular camera observations of unknown features. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3558–3564, 2007.

[26] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, Jan 1997.

[27] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2004.

[28] R. Horaud and F. Dornaika. Hand-eye calibration. *Intl. J. Robot. Res.*, 1995.

[29] T. Huang and O. Faugeras. Some properties of the E matrix in two-view motion estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Jan 1989.

[30] E. Jones, A. Vedaldi, and S. Soatto. Inertial structure from motion with autocalibration. In *Proc. of the IEEE Int. Conf. on Computer Vision Workshop on Dynamical Vision, Rio de Janeiro, Brazil*, 2007.

[31] M. Kalantari, A. Hashemi, F. Jung, and J. Guedon. A new solution to the relative orientation problem using only 3 points and the vertical direction. *arXiv*, cs.CV, May 2009.

[32] K. Konolige, M. Agrawal, and J. Sola. Large-scale visual odometry for rough terrain. *Robotics Research*, pages 201–212, 2011.

[33] J. Kosecka and W. Zhang. Video compass. *Proc. Seventh European Conference on Computer Vision*, pages 476–490, Copenhagen, Denmark 2002.

[34] E. Kruppa. Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung. *Sitz.-Ber. Akad.Wiss.,Wien, Math. Naturw. Kl., Abt. IIa.*, 122:1939–1948, 1913.

[35] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. *Proc. Tenth European Conference on Computer Vision*, Jan 2008.

[36] Z. Kukelova, M. Byröd, K. Josephson, and T. Pajdla. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. *Computer Vision and Image Understanding*, Jan 2008.

[37] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. *IEEE Conf. Computer Vision and Pattern Recognition*, Minneapolis, MN, June 18-23 2007.

[38] Z Kukelova and T Pajdla. A minimal solution to the autocalibration of radial distortion. *IEEE Conf. Computer Vision and Pattern Recognition*, Minneapolis, MN, June 18-23 2007.

[39] H. Li and R. Hartley. Five-point motion estimation made easy. *Proc. Int. Conf. on Pattern Recognition*, 2006.

[40] J. Lobo and J. Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(12):1597–1608, 2003.

[41] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, Jan 2004.

[42] A. Makadia and K. Daniilidis. Correspondenceless ego-motion estimation using an IMU. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2005.

[43] C. Mei and P. Rives. Calibration between a central catadioptric camera and a laser range finder for robotic applications. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 532 –537, may. 2006.

[44] C. Meyer. *Matrix analysis and applied linear algebra*. SIAM, Jan 2000.

[45] M Minimair. MR: Macaulay resultant package for Maple. *ACM SIGPLAN*, 39(4):26–29, 2004.

[46] AI Mourikis and SI Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3565–3572, 2007.

[47] A.I. Mourikis, N. Trawny, S.I. Roumeliotis, A. Johnson, and L. Matthies. Vision aided inertial navigation for precise planetary landing: Analysis and experiments. *Proc. Robotics Systems and Science Conference*, 2007.

[48] O. Naroditsky and K. Daniilidis. Optimizing polynomial solvers for minimal geometry problems. *Proc. Int. Conf. on Computer Vision*, pages 975–982, 2011.

[49] O. Naroditsky, A. Patterson, and K. Daniilidis. Automatic alignment of a camera with a line scan LIDAR system. *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 3429–3434, 2011.

[50] O. Naroditsky, X. Zhou, J. Gallier, S. Roumeliotis, and K. Daniilidis. Two efficient solutions for visual odometry using directional correspondence. *IEEE Trans. Pattern Analysis and Machine Intelligence*, (99):1–1, 2011.

[51] O. Naroditsky, Z. Zhu, A. Das, T. Oskiper, S. Samarasekera, and R. Kumar. Videotrek: A vision system for a tag-along robot. *IEEE Conf. Computer Vision and Pattern Recognition*, Miami, FL, June 20-25 2009.

[52] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Conf. Computer Vision and Pattern Recognition*, Wisconsin, June 16-22 2003.

[53] D. Nister. Preemptive RANSAC for live structure and motion estimation. *IEEE Conf. Computer Vision and Pattern Recognition*, pages 199–206 vol.1, Wisconsin, June 16-22 2003.

[54] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Jan 2004.

[55] D. Nister. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005.

[56] D. Nister, O. Naroditsky, and J. Bergen. Visual odometry. *IEEE Conf. Computer Vision and Pattern Recognition*, 1:I–652 – I–659 Vol.1, Washington, DC, Jun 29 - Jul 1 2004.

[57] P. Nunez, P. Drews, R. Rocha, and J. Dias. Data fusion calibration for a 3D laser range finder and a camera using inertial data. *European Conference on Mobile Robots '09*, page 9, 2009.

[58] G. Qian, R. Chellappa, and Q. Zheng. Robust structure from motion estimation using inertial data. *Journal of the Optical Society of America A*, 18(12):2982–2997, 2001.

[59] S.I. Roumeliotis, A.E. Johnson, and J.F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *Proc. IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 4326–4333. Citeseer, 2002.

[60] D. Scaramuzza, A. Harati, and R. Siegwart. Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4164–4169, 2007.

[61] W. Scharlau. Who is Alexander Grothendieck. *Notices of the American Mathematical Society*, "55"("8"):"930–941", "2008".

[62] B. Siciliano and O. Khatib. *Springer handbook of robotics*. Springer, Jan 2008.

[63] H. Stewenius. Gröbner basis methods for minimal problems in computer vision. *maths.lth.se*, Jan 2005.

[64] H. Stewenius, C. Engels, and D. Nister. An efficient minimal solution for infinitesimal camera motion. *IEEE Conf. Computer Vision and Pattern Recognition*, Jan 2007.

[65] H. Stewenius, F. Schaffalitzky, and D. Nister. How hard is 3-view triangulation really? *Proc. Int. Conf. on Computer Vision*, Jan 2005.

[66] D. Strelow and S. Singh. Motion estimation from image and inertial measurements. *The International Journal of Robotics Research*, 23(12):1157, 2004.

[67] P. Torr and C. Davidson. IMPSAC: Synthesis of importance sampling and random sample consensus. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Jan 2003.

[68] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, Jan 2000.

[69] C. Traverso. Gröbner trace algorithms. *Symbolic and algebraic computation: ISSAC'88*, Jan 1988.

[70] R. Unnikrishnan and M. Hebert. Fast extrinsic calibration of a laser rangefinder to a camera. *Tech. Rep. CMU Robotics Institute*, page 339, 2005.

[71] T. Vieville, E. Clergue, and P. Facao. Computation of ego-motion and structure from visual and inertial sensors using the vertical cue. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 591–598, 1993.

[72] S. You and U. Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *IEEE Virtual Reality*, pages 71–78, 2001.

[73] Q. Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 2301 – 2306 vol.3, sep. 2004.