

FAST AND STABLE POLYNOMIAL EQUATION  
SOLVING AND ITS APPLICATION TO COMPUTER  
VISION

MARTIN BYRÖD



LUND UNIVERSITY

Faculty of Engineering  
Centre for Mathematical Sciences  
Mathematics

Mathematics  
Centre for Mathematical Sciences  
Lund University  
Box 118  
SE-221 00 Lund  
Sweden  
<http://www.maths.lth.se/>

Licentiate Theses in Mathematical Sciences 2008:3  
ISSN 1404-028X

ISBN 978-91-633-2583-0  
LUTFMA-2028-2008

© Martin Byröd, 2008

Printed in Sweden by Media-Tryck, Lund 2008

## Preface

The central theme of this thesis is the problem of finding all zeros of a system of polynomial equations in several variables. The driving motivation for studying this topic has been to use the knowledge to solve real world computer vision problems. For the reader not acquainted with computer vision and in particular the geometric problems that arise in the interplay between the projected image and the three dimensional world, it is probably far from obvious how polynomial equations and computer vision are related. To bridge this gap, after a brief introduction, the thesis starts off with a chapter on basic concepts of computer vision and camera geometry as well as an introduction to some concepts of algebraic geometry, which is the theory of polynomial equations. After this the remainder of the thesis is devoted to contributions made during the course of work up to this point.

During the thesis work, two types of contributions have been made: (i) Contributions related to the theory and techniques of numerical methods for polynomial equation solving [7, 6, 8] and (ii) the application of these techniques to previously unsolved problems in computer vision [29, 6, 10]. The paper [6] is mainly an application, but occurs in both categories since it also contains a preliminary version of the redundant basis method presented in Chapter 4.

In summary, the thesis is based on material from the following papers:

- [8] M. Byröd, K. Josephson, K. Åström, A Column-Pivoting Based Strategy for Monomial Ordering in Numerical Gröbner Basis Calculations, Submitted, 2008.
- [10] M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, K. Åström, Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion, Accepted for publication at *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, Alaska, 2008.
- [6] M. Byröd, K. Josephson, K. Åström, Fast Optimal Three View Triangulation, *Proc. Asian Conference on Computer Vision (ACCV)*, Tokyo, Japan, 2007.
- [7] M. Byröd, K. Josephson, K. Åström, Improving Numerical Accuracy of Gröbner Basis Polynomial Equation Solvers, *Proc. International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil, 2007.
- [29] K. Josephson, M. Byröd, F. Kahl, K. Åström, Image-Based Localization Using Hybrid Feature Correspondences, *ISPRS Workshop BenCOS at CVPR*, Minneapolis, USA, 2007.

In addition to the above, the following papers have also been written during the thesis work:

- [35] Z. Kukelova, M. Byröd, K. Josephson, T. Pajdla, K. Åström, Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion, Submitted to *Computer Vision and Image Understanding*, 2008.
- [9] M. Byröd, K. Josephson, K. Åström, Optimal Three View Triangulation By Polynomial Equation Solving, Submitted to *IPSJ Transactions on Computer Vision and Applications*, 2008.

## Organization of the Thesis

In more detail, the thesis is organized as follows. After an introduction in Chapter 1, we give an overview of some basic concepts of geometric computer vision and the classical theory of algebraic geometry in Chapter 2. Thereafter, in Chapter 3, we present the theoretical underpinnings of a set of new numerical techniques introduced in Chapter 4. These first four chapters constitute the first part of the thesis and for completeness, Chapter 4 also contains a section with some experimental results to highlight the relative benefits and drawbacks of the various introduced techniques. In part two of the thesis we set out to use these techniques to solve a set of real world computer vision problems. Chapter 5 starts off by looking at a specific instance of the triangulation problem. Thereafter, in Chapter 6, we investigate how relative camera motion can be computed in the presence of potentially heavy radial distortion. Finally, in Chapter 7, we study the mathematics of camera pose estimation when a mixture of world coordinate to camera and camera to camera correspondences are given.

## Acknowledgements

I would like to acknowledge the contribution of a number of people to the completion of this thesis. First of all, I would like to thank my supervisor, Kalle Åström, who is a pleasure to work with; always engaged, wise, friendly and ready to answer any question at any time. A thank you also goes to the rest of the Mathematical Imaging Group and in particular to Klas Josephson for energy, ideas and collaborative efforts in most of my work as a phd student so far. Furthermore I would like to thank Zuzana Kukelova and Tomas Pajdla at the Czech Technical University in Prague for interesting discussions, useful comments and collaboration on two papers. I would also like to thank our very helpful secretary Ann-Kristin Ottosson for aid in countless practical issues. Finally, lots of thanks to friends and family for everything else.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Related Work . . . . .	9
<b>I</b>	<b>Methods</b>	<b>11</b>
<b>2</b>	<b>Preliminaries</b>	<b>13</b>
2.1	Geometric Computer Vision . . . . .	13
2.1.1	The Calibration Matrix . . . . .	14
2.1.2	Epipolar Geometry . . . . .	15
2.1.3	Structure from Motion . . . . .	16
2.1.4	Minimal Problems . . . . .	16
2.2	Algebraic Geometry for Equation Solving . . . . .	17
2.2.1	The Action Matrix . . . . .	18
2.2.2	Gröbner Bases . . . . .	19
2.2.3	A Note on Algebraic and Linear Bases . . . . .	21
2.2.4	Floating Point Gröbner Basis Computations . . . . .	21
<b>3</b>	<b>Theoretical Contributions</b>	<b>23</b>
3.1	A New Approach to the Action Matrix Method . . . . .	23
3.1.1	Solving Bases . . . . .	24
3.1.2	Solving Basis Computations . . . . .	28
<b>4</b>	<b>Techniques</b>	<b>31</b>
4.1	Using Redundant Solving Bases - The Truncation Method . . . . .	31
4.2	Basis Selection . . . . .	32
4.2.1	The QR Method . . . . .	33
4.2.2	The SVD Method . . . . .	34
4.2.3	Basis Selection and Adaptive Truncation . . . . .	36
4.3	Other Techniques . . . . .	37
4.3.1	A Single Elimination Step . . . . .	37
4.3.2	Using Eigenvalues Instead of Eigenvectors . . . . .	38
4.4	Experimental Validation . . . . .	38
4.4.1	Optimal Three View Triangulation . . . . .	39
4.4.2	Relative Pose with Unknown Focal Length . . . . .	45
4.4.3	Relative Pose for Generalized Cameras . . . . .	47
4.5	Discussion . . . . .	47

<b>II</b>	<b>Applications</b>	<b>49</b>
<b>5</b>	<b>Optimal Triangulation</b>	<b>51</b>
5.1	Introduction . . . . .	51
5.2	Three View Triangulation . . . . .	53
5.3	A Numerical Solution to the Three View Triangulation Problem	54
5.4	Experimentas . . . . .	55
5.4.1	Synthetic Data . . . . .	55
5.4.2	A Real Example . . . . .	56
5.5	Conclusions . . . . .	57
<b>6</b>	<b>Epipolar Geometry Under Radial Distortion</b>	<b>59</b>
6.1	Introduction . . . . .	59
6.2	Uncalibrated Case . . . . .	61
6.2.1	Details on the Expansion Step for the Uncalibrated Case	62
6.3	Calibrated Case . . . . .	62
6.4	Experiments . . . . .	63
6.4.1	Tests on Synthetic Images . . . . .	63
6.4.2	Time Consumption . . . . .	67
6.4.3	Tests on Real Images . . . . .	67
6.5	Conclusions . . . . .	69
<b>7</b>	<b>Hybrid Minimal Problems</b>	<b>71</b>
7.1	Introduction . . . . .	71
7.2	Problem Formulation . . . . .	72
7.3	Minimal Hybrid Correspondence Sets . . . . .	72
7.4	Solving Hybrid Minimal Cases with Algebraic Geometry . . . . .	74
7.4.1	Calibrated Cameras . . . . .	75
7.4.2	Experimental Results for the (2,2) Case . . . . .	77
7.4.3	Unknown Focal Length . . . . .	78
7.4.4	Experimental Results for the (1,3) Case . . . . .	79
7.5	Conclusions . . . . .	79
<b>8</b>	<b>Conclusions</b>	<b>81</b>

# Chapter 1

## Introduction

Numerous geometric problems in computer vision involve the solution of systems of polynomial equations. This is particularly true for so called minimal structure and motion problems [12, 34, 53]. Solutions to minimal structure and motion problems can often be used in RANSAC algorithms to find inliers in noisy data [20, 54, 55]. For such applications one needs to solve a large number of minimal structure and motion problems as fast as possible in order to find the best set of inliers. The minimal solutions then typically also serve as an initial estimate to be able to deploy a more sophisticated optimization algorithm, which relies on inlier free data and good initialization. There is thus a need for fast and numerically stable algorithms for solving particular systems of polynomial equations.

Another area of recent interest is global optimization used for *e.g.* optimal triangulation, resectioning and fundamental matrix estimation. Global optimization is a promising, but difficult pursuit and different lines of attack have been tried, *e.g.* branch and bound [1],  $L_\infty$ -norm methods [24, 30] and methods using linear matrix inequalities (LMIs) [32]. An alternative way to find the global optimum is to calculate stationary points directly (usually by solving some polynomial equation system) [25, 52]. So far, this has been an approach of limited applicability since calculation of stationary points is numerically difficult for larger problems. By using the new methods for polynomial equation solving presented in this thesis it should be possible to handle a somewhat larger class of problems, thus offering an alternative to the above mentioned optimization methods. An example of this is optimal three view triangulation which has previously not been solved in a practical way [52]. We show in Chapter 5 that using the new techniques presented in this thesis, this problem can now be solved in a reasonably efficient way with an algorithm implemented in standard IEEE double precision.

The state-of-the-art method for numerical solution of polynomial equations is based on calculations with Gröbner bases [48] and has many applications in computer vision, but also in other fields such as cryptology [19] and robotics [2]. A typical outline of such algorithms is that one first studies a specific geometric problem and finds out what structure the Gröbner basis of the ideal  $I$  has for that problem, how many solutions there are and what the degrees of monomials occurring in the Gröbner basis elements are. For each instance of the problem with numerical data, the process of forming the Gröbner basis follows the same

steps and the construction of the Gröbner basis can be written down as a sequence of pre determined elimination steps using numerical linear algebra. The Gröbner basis can then be used to construct an *action matrix*, which represents multiplication in the quotient space  $\mathbb{C}[\mathbf{x}]/I$ . The solution to the problem is then obtained through an eigenvalue decomposition of the action matrix.

Currently, the limiting factor in using these methods for larger and more difficult cases is numerical problems. For example in [52] it was necessary to use emulated 128 bit numerics to make the system work, which made the implementation very slow. This thesis improves on the state of the art of these techniques making it possible to handle larger and more difficult problems in a practical way.

In the thesis we pin-point the main source of these numerical problems (the conditioning of a crucial elimination step) and propose a range of techniques for dealing with this issue. The main novelty is a new approach to the action matrix method for equation solving, relaxing the need of adhering to a properly defined monomial order and a complete Gröbner basis. This unlocks substantial freedom, which is used in a number of different ways to improve stability.

Firstly, we show how the sensitive elimination step can be avoided by using an overly large/redundant linear basis for  $\mathbb{C}[\mathbf{x}]/I$  to construct the action matrix. This method yields the right solutions along with a set of false solutions that can then easily be filtered out by evaluation in the original equations.

Secondly, we show how a true linear basis for  $\mathbb{C}[\mathbf{x}]/I$  can be constructed from a redundant basis in such a way that good numerical precision is retained. This is done by attempting to find an optimal reordering or even linear combination of the monomials and we investigate what conditions such a reordering/linear combination needs to satisfy. We develop the tools needed to compute the action matrix in a general linear basis for  $\mathbb{C}[\mathbf{x}]/I$  and propose two strategies for selecting this basis which enhances the stability of the solution procedure.

The first of these is a fast strategy based on QR factorization with column pivoting. The Gröbner basis like computations employed to solve a system of polynomial equations can essentially be seen as matrix factorization of an under-determined linear system. Based on this insight, we combine the robust method of QR factorization from numerical linear algebra with the Gröbner basis theory needed to solve polynomial equations. More precisely, we employ QR factorization with column pivoting in the above mentioned elimination step and obtain a simultaneous selection of linear basis and triangular factorization.

Factorization with column pivoting is a very well studied technique and there exist highly optimized and reliable implementations of these algorithms in *e.g* LAPACK [38], which makes this technique accessible and relatively straightforward to implement.

The second technique for basis selection goes one step further and employs singular value decomposition (SVD) to select a general linear basis of polynomials for  $\mathbb{C}[\mathbf{x}]/I$ . This technique is computationally more demanding than the QR method, but yields even better stability.

Finally, we show how a redundant linear basis for  $\mathbb{C}[\mathbf{x}]/I$  can be combined with the above basis selection techniques. In the QR method, since the pivot elements are sorted in descending order, we get an adaptive criterion for where to truncate the Gröbner basis like structure by setting a maximal threshold for the quotient between the largest and the smallest pivot element. When the quotient exceeds this threshold we abort the elimination and move the remaining

columns into the basis. This way, we expand the basis only when necessary.

## 1.1 Related Work

The area of polynomial equation solving is currently very active. See *e.g.* [11] and references therein for a comprehensive exposition of the state of the art in this field.

One of the oldest and still used methods for non-linear equation solving is the Newton-Raphson method which is fast and easy to implement, but depends heavily on initialization and finds only a single zero for each initialization. In the univariate case, a numerically sound procedure to find the complete set of roots is to compute the eigenvalues of the companion matrix. However, if only real solutions are needed, the fastest way is probably to use Sturm sequences [28].

In several variables a first method is to use resultants [14], which using a determinant construct enables the successive elimination of variables. However, the resultant grows exponentially in the number of variables and is in most cases not practical for more than two variables. An alternative way of eliminating variables is to compute a lexicographical Gröbner basis for the ideal generated by the equations which can be shown to contain a univariate polynomial representing the solutions [14]. This approach is however often numerically unstable.

A radically different approach is provided by homotopy continuation methods [57]. These methods typically work in conjunction with mixed volume calculations by constructing a simple polynomial system with the same number of zeros as the actual system that is to be solved. The simple system with known zeros is then continuously deformed into the actual system. The main drawback of these methods is the computational complexity with computation times ranging in seconds or more.

At present, the best methods for geometric computer vision problems are based on eigendecomposition of a certain matrices (action matrices) representing multiplication in the quotient space  $\mathbb{C}[\mathbf{x}]/I$ . The action matrix can be seen as a direct generalization of the companion matrix in the univariate case. The factors that make this approach attractive is that it (i) is fast and numerically feasible, (ii) handles more than two variables and reasonably high degrees (up to around 10) and (iii) is well suited to tuning for specific applications. To the authors best knowledge, this method was first used in the context of computer vision by Stewenius *et al* [48] even though Gröbner basis methods were mentioned in [27].



Part I  
Methods



## Chapter 2

# Preliminaries

This chapter introduces some background knowledge to facilitate the understanding of the remainder of the thesis. We start by presenting the basics of geometric computer vision including the linear pin-hole camera and the fundamental and essential matrices. We then give some elements of algebraic geometry used for polynomial equation solving.

### 2.1 Geometric Computer Vision

The general field of computer vision deals with the problem of making a computer “see”. This thesis, however, treats only geometric computer vision, which refers to extracting geometric information about the world (scene) and the observer (camera) from a sequence of two or more images. This forms the basis for many applications; Stereo, 3D reconstruction, panoramic stitching, augmented reality, robotics, etc. See [26] for a thorough introduction to the subject. The fundamental entity in this process is the *camera*, which needs to be modeled in some sensible manner. The most popular way of doing this is to adopt the central projection principle which yields the *pin-hole camera* model. In geometric language, the pin-hole camera consists of a camera center  $t$  and a plane  $\pi$  (the image plane) with a local coordinate system. Projection of a world point  $\mathbf{X}$  is done by intersecting the ray from  $t$  through  $\mathbf{X}$  with  $\pi$  and the projected point  $\mathbf{x}$  is simply obtained as the intersection, see Figure 2.1.

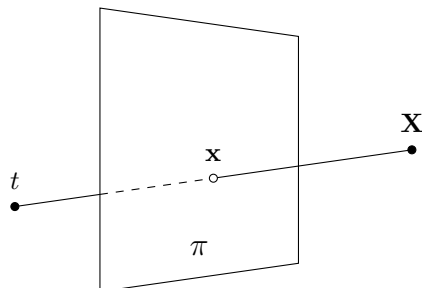


Figure 2.1: Projection of the point  $\mathbf{X}$  onto the image plane  $\pi$  using the pin-hole camera model.

We now choose coordinate system in the world so that the camera is at the origin and place the origin  $\mathcal{O}$  of  $\pi$  so that the axis from the camera center to  $\mathcal{O}$  is perpendicular to  $\pi$  and then align the camera axis with the world coordinate  $Z$ -axis producing the schematic setup illustrated in Figure 2.2.

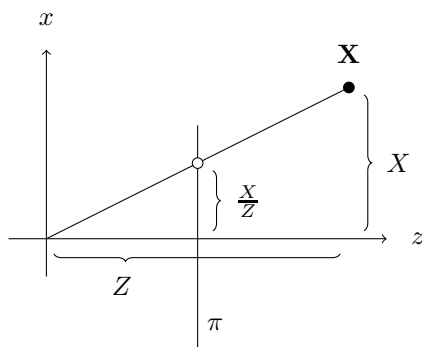


Figure 2.2: The setup with the camera axis aligned with the  $z$ -axis is convenient for deriving the central projection equations.

With this setup, we can use the top-triangle theorem of Euclidean geometry to derive the projection of a world point  $\mathbf{X} = [X, Y, Z]^T$ . From Figure 2.2 we easily see that we get the image coordinates

$$\begin{aligned} x &= \frac{X}{Z} \\ y &= \frac{Y}{Z}. \end{aligned} \quad (2.1)$$

Now consider the more general case with a camera center  $t \neq \mathbf{0}$  and a camera axis which is not aligned with the  $Z$ -axis (but still intersects the origin of the image plane). This can be brought back to the situation in Equation 2.1 by a matrix multiplication

$$\mathbf{X}' = [R \quad -Rt] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}, \quad (2.2)$$

where  $R$  is a  $3 \times 3$  rotation matrix yielding  $x = X'/Z'$  and  $y = Y'/Z'$ . We denote the matrix in equation 2.2 by  $P$  and switch to homogeneous coordinates which means that we extend the image coordinates  $x$  and the world coordinates  $\mathbf{X}$  with a 1 as in Equation 2.2. We thus get the familiar pin-hole projection equation

$$\lambda x = P\mathbf{X}, \quad (2.3)$$

where the depth  $\lambda$  is now instead put on the left hand side.

### 2.1.1 The Calibration Matrix

The camera matrix  $P$  derived above is a  $3 \times 4$  matrix with a special structure. If we let  $P$  be any  $3 \times 4$  matrix we get a general projective camera. Using a variant of QR decomposition of matrices we can write  $P$  as

$$P = K [R \quad t], \quad (2.4)$$

where  $R$  is an orthogonal matrix and  $K$  is upper triangular. It is common to write

$$K = \begin{bmatrix} f & fs & x_0 \\ 0 & f\gamma & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.5)$$

which can then be interpreted as the focal length  $f$ , the principal point  $[x_0, y_0]^T$  (the point of intersection between camera axis and image plane), the aspect ratio  $\alpha$  (scale ratio between the  $y$ -axis and the  $x$ -axis in the image) and the skew  $s$  which models non-orthogonal coordinate axes in the image. Of these parameters, the focal length  $f$  is the only parameter which is explicitly used in this thesis.

Typical assumptions are: (i) the camera is calibrated which means that  $K$  is known and we can then multiply the image coordinates with  $K^{-1}$  and assume that  $K$  is the identity matrix in (2.4), (ii) the camera is calibrated up to an unknown focal length  $f$  which means that we can assume

$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.6)$$

or (iii) the camera is uncalibrated which means we have a general camera matrix  $P$ .

Algorithmically, as we will see, the uncalibrated case is often the easiest to work with since any partial or full calibration means that we have to introduce non-linear constraints which complicate the situation.

### 2.1.2 Epipolar Geometry

In the setting of two uncalibrated cameras  $P_1$  and  $P_2$ , image coordinates  $x_1$  and  $x_2$  corresponding to a common world point  $X$  obey a bilinear constraint known as the epipolar constraint

$$x_1^T F x_2 = 0. \quad (2.7)$$

We work with homogeneous coordinates and  $F$  is thus a  $3 \times 3$  matrix, which is known as the fundamental matrix and is uniquely determined by the camera matrices  $P_1$  and  $P_2$ . An important property of  $F$  is that we always have  $\det(F) = 0$ . Conversely, any  $3 \times 3$  matrix  $F$  with  $\det(F) = 0$  is a fundamental matrix of some cameras  $P_1$  and  $P_2$ .

Consider now two calibrated cameras  $P_1$  and  $P_2$  on the form (2.2). These uniquely determine a matrix  $E$ , called the essential matrix, which satisfies (2.7) for any corresponding points as well as  $\det(E) = 0$ . Moreover, since  $E$  is computed from two calibrated cameras it can be shown that the two nonzero singular values of  $E$  are equal which can be expressed as

$$2EE^T E - \text{tr}(EE^T)E = 0, \quad (2.8)$$

known as the trace constraint for the essential matrix.

### 2.1.3 Structure from Motion

One of the main goals of geometric computer vision can be formulated as solving the *structure from motion* problem. The term structure from motion comes

from the idea of inferring the structure (3D configuration) of an observed scene solely from motion as captured in a sequence of images. In the general setting nothing is assumed to be known about the cameras. Algebraically formulated, we are given  $m \cdot n$  image points  $\mathbf{x}_{ij}$  captured by  $m$  unknown cameras  $P_i$  from  $n$  unknown world points  $\mathbf{X}_j$ . The problem is to determine the unknown cameras and world points satisfying

$$\lambda_{ij}\mathbf{x}_{ij} = P_i\mathbf{X}_j \quad (2.9)$$

for all  $i$  and  $j$ .

A typical structure from motion system consists of the following steps

1. Establish tentative point correspondences across views using some local patch descriptor. SIFT [41] is a popular choice. This step typically produces a large set of true as well as false correspondences.
2. Repeatedly compute fundamental/essential matrices from pairwise views using randomly selected small subsets of points and save the sets which are consistent with many of the other points (the RANSAC algorithm [20]).
3. Fine tune the reconstruction by employing a large scale optimization algorithm to minimize *e.g.* the sum of squares of reprojection errors over all views for the points selected in the previous step.

The structure from motion problem is by no means solved and each of the steps above constitutes an active sub-field in its own right.

#### 2.1.4 Minimal Problems

In the structure from motion system sketched in the previous section, step 2 involved computing camera geometries from small numbers of correspondences. The motivation for this is that a small set of correspondences is less likely to contain incorrect matches. It is therefore interesting to investigate what the minimal number of point correspondences is for a given geometric problem and to devise algorithms for solving them. Such problems are usually referred to as *minimal problems* or *minimal cases* and will occur frequently throughout this thesis.

Understanding the geometry and the number of solutions of minimal structure and motion problems has a long history. For instance, computing the fundamental matrix in the uncalibrated case requires a minimal set of seven points in two views and with this setup the problem has three solutions. This problem was studied and solved already in 1855, *cf.* [12]. The corresponding calibrated case was in principle solved in 1913 [34] and later corrected by Demazure [15]. However, it was only recently that a practical numerical algorithm for solving this problem was given [45, 49]. As mentioned above the study of minimal cases has got increased attention with its use in RANSAC algorithms to solve both for geometry and correspondence in numerous applications [26].

## 2.2 Algebraic Geometry for Equation Solving

In this section we review some of the classical theory of multivariate polynomials. We consider the following problem

*Problem 1.* Given a set of  $m$  polynomials  $f_i(\mathbf{x})$  in  $s$  variables  $\mathbf{x} = (x_1, \dots, x_s)$ , determine the complete set of solutions to

$$\begin{aligned} f_1(\mathbf{x}) &= 0, \\ &\vdots \\ f_m(\mathbf{x}) &= 0. \end{aligned} \tag{2.10}$$

We denote by  $V$  the zero set of (2.10). In general  $V$  need not be finite, but in this work we will only consider zero dimensional  $V$ , i.e.  $V$  is a point set.

The general field of study of multivariate polynomials is algebraic geometry. See [14] and [13] for a nice introduction to the field and for proofs of all claims made in this section. In the language of algebraic geometry,  $V$  is an affine algebraic variety and the polynomials  $f_i$  generate an *ideal*  $I = \sum_i h_i(\mathbf{x})f_i(\mathbf{x})$ , where  $h_i \in \mathbb{C}[\mathbf{x}]$  are any polynomials and  $\mathbb{C}[\mathbf{x}]$  denotes the set of all polynomials in  $\mathbf{x}$  over the complex numbers.

The motivation for studying the ideal  $I$  is that it is a generalization of the set of equations (2.10). A point  $\mathbf{x}$  is a zero of (2.10) iff it is a zero of  $I$ . Being even more general, we could ask for the complete set of polynomials which vanish on  $V$ . If  $I$  is equal to this set, then  $I$  is called a radical ideal.

We say that two polynomials  $f, g$  are equivalent modulo  $I$  iff  $f - g \in I$  and denote this by  $f \sim g$ . With this definition we get the quotient space  $\mathbb{C}[\mathbf{x}]/I$  of all equivalence classes modulo  $I$  and let  $[\cdot]$  denote the natural projection  $\mathbb{C}[\mathbf{x}] \mapsto \mathbb{C}[\mathbf{x}]/I$ , i.e. by  $[f_i]$  we mean the set  $\{g_i : f_i - g_i \in I\}$  of polynomials equivalent to  $f_i$  modulo  $I$ .

A related structure is  $\mathbb{C}[V]$ , the set of equivalence classes of polynomial functions on  $V$ . We say that a function  $F$  is polynomial on  $V$  if there is a polynomial  $f$  such that  $F(\mathbf{x}) = f(\mathbf{x})$  for  $\mathbf{x} \in V$  and equivalence here means equality on  $V$  (see Figure 2.3). If two polynomials are equivalent modulo  $I$ , then they are obviously also equal on  $V$ . If  $I$  is radical, then conversely two polynomials which are equal on  $V$  must also be equivalent modulo  $I$ . This means that for radical ideals,  $\mathbb{C}[\mathbf{x}]/I$  and  $\mathbb{C}[V]$  are isomorphic. Now, if  $V$  is a point set, then any function on  $V$  can be identified with a  $|V|$ -dimensional vector and since the unisolvence theorem for polynomials guarantees that any function on a discrete set of points can be interpolated exactly by a polynomial, we get that  $\mathbb{C}[V]$  is isomorphic to  $\mathbb{C}^r$ , where  $r = |V|$ .

### 2.2.1 The Action Matrix

Turning to equation solving, our starting point is the companion matrix which arises for polynomials in one variable. For a third degree polynomial

$$p(x) = x^3 + a_2x^2 + a_1x + a_0, \tag{2.11}$$

the companion matrix is

$$\begin{bmatrix} -a_2 & 1 & 0 \\ -a_1 & 0 & 1 \\ -a_0 & 0 & 0 \end{bmatrix}. \tag{2.12}$$

The eigenvalues of the companion matrix are the zeros of  $p(x)$  and for high degree polynomials, this provides a numerically stable way of calculating the roots.

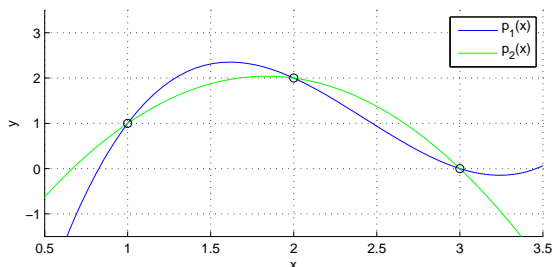


Figure 2.3: Given  $V = \{1, 2, 3\}$ , the two polynomials  $p_1(x)$  and  $p_2(x)$  shown in the figure are equivalent and hence represent the same equivalence class in  $\mathbb{C}[V]$ .

With some care, this technique can be extended to the multivariate case as well, which was first done by Lazard in 1981 [39]. For  $V$  finite, the space  $\mathbb{C}[\mathbf{x}]/I$  is finite dimensional. Moreover, if  $I$  is radical, then the dimension of  $\mathbb{C}[\mathbf{x}]/I$  is equal to  $|V|$ , *i.e.* the number of solutions [14]. For some  $p \in \mathbb{C}[\mathbf{x}]$  consider now the operation  $T_p : f(\mathbf{x}) \mapsto p(\mathbf{x})f(\mathbf{x})$ . The operator  $T_p$  is linear and since  $\mathbb{C}[\mathbf{x}]/I$  is finite dimensional, we can select a linear basis  $\mathcal{B}$  of polynomials for  $\mathbb{C}[\mathbf{x}]/I$  and represent  $T_p$  as a matrix  $\mathbf{m}_p$ . This matrix is known as the action matrix and is precisely the generalization of the companion matrix we are looking for. In fact, in the example above, we can let the set  $\{[x^2], [x], [1]\}$  be a basis for  $\mathbb{C}[x]/\langle p(x) \rangle$ , where  $\langle p(x) \rangle$  denotes the ideal generated by  $p(x)$ . Representing  $T_x : f(x) \mapsto xf(x)$  in this basis yields exactly the matrix in Equation 2.12. The eigenvalues of  $\mathbf{m}_p$  are  $p(\mathbf{x})$  evaluated at the points of  $V$ . Moreover, the eigenvectors of  $\mathbf{m}_p^T$  equals the vector of basis elements evaluated on  $V$ . Briefly, this can be understood as follows: Consider an arbitrary polynomial  $p(\mathbf{x}) = c^T \mathbf{b}$ , where  $c$  is a vector of coefficients and  $\mathbf{b}$  is a vector of polynomials forming a basis of  $\mathbb{C}[\mathbf{x}]/I$ . We then have

$$[p \cdot c^T \mathbf{b}] = [(\mathbf{m}_p c)^T \mathbf{b}] = [c^T \mathbf{m}_p^T \mathbf{b}]. \quad (2.13)$$

This holds for any coefficient vector  $c$  and hence it follows that  $[p\mathbf{b}] = [\mathbf{m}_p^T \mathbf{b}]$ , which can be written  $p\mathbf{b} = \mathbf{m}_p^T \mathbf{b} + \mathbf{g}$  for some vector  $\mathbf{g}$  with components  $g_i \in I$ . Evaluating the expression at a zero  $\bar{\mathbf{x}} \in V$  we get  $\mathbf{g}(\bar{\mathbf{x}}) = 0$  and thus obtain

$$p(\bar{\mathbf{x}})\mathbf{b}(\bar{\mathbf{x}}) = \mathbf{m}_p^T \mathbf{b}(\bar{\mathbf{x}}), \quad (2.14)$$

which we recognize as an eigenvalue problem on the matrix  $\mathbf{m}_p^T$  with eigenvectors  $\mathbf{b}(\bar{\mathbf{x}})$ . In other words, the eigenvectors of  $\mathbf{m}_p^T$  yield  $\mathbf{b}(\mathbf{x})$  evaluated at the zeros of  $I$  and the eigenvalues give  $p(\mathbf{x})$  at the zeros. The conclusion we can draw from this is that zeros of  $I$  corresponds to eigenvectors and eigenvalues of  $\mathbf{m}_p$ , but not necessarily the opposite, *i.e.* there can be eigenvectors/eigenvalues that do not correspond to actual solutions. If  $I$  is radical, this is not the case and we have an exact correspondence.

Note here that in a strict sense, a set of monomials  $\mathcal{B}$  cannot form a basis for  $\mathbb{C}[\mathbf{x}]/I$  since  $\mathbb{C}[\mathbf{x}]/I$  is a space of equivalence classes. What we mean is that a set of monomials  $\mathcal{B}$  are *representatives* of equivalence classes forming a basis of  $\mathbb{C}[\mathbf{x}]/I$  or alternatively that the natural projection  $[\cdot]$  of the monomials onto

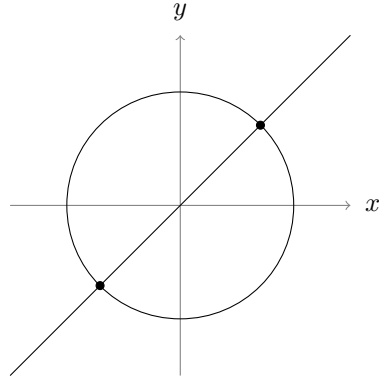


Figure 2.4: The intersection of a line and a circle can be formulated as a system of two polynomial equations. See Example 2.

$\mathbb{C}[\mathbf{x}]/I$  form a basis. In the following we will, however, typically use the slightly incorrect but more readable terminology of referring to a set of monomials as a basis.

### 2.2.2 Gröbner Bases

We have seen theoretically that the action matrix  $\mathbf{m}_p$  provides the solutions to a corresponding system of polynomial equations. The main issue is now how to compute  $\mathbf{m}_p$ . This is in general done by selecting a linear basis  $\mathcal{B}$  for  $\mathbb{C}[\mathbf{x}]/I$  and then computing  $[p \cdot b_i]$  for each  $b_i \in \mathcal{B}$ . To do actual computations in  $\mathbb{C}[\mathbf{x}]/I$  we need to represent each equivalence class  $[f]$  by a well defined representative polynomial. The idea is to use multivariate polynomial division and represent  $[f]$  by the remainder under division of  $f$  by  $I$ . Fortunately, for any polynomial ideal  $I$ , this can always be done and the tool for doing so is a Gröbner basis  $G$  for  $I$  [14]. The Gröbner basis for  $I$  is a canonical set of generators for  $I$  with the property that multivariate division by  $G$ , denoted  $\overline{f}^G$ , always yields a well defined remainder. By well defined we mean that for any  $f_1, f_2 \in [f]$ , we have  $\overline{f_1}^G = \overline{f_2}^G$ . The Gröbner basis is computed relative a monomial order and will be different for different monomial orders. As a consequence, the set of representatives for  $\mathbb{C}[\mathbf{x}]/I$  will be different, whereas the space itself remains the same.

The linear basis  $\mathcal{B}$  should consist of elements  $b_i$  such that the elements  $\{[b_i]\}_{i=1}^r$  together span  $\mathbb{C}[\mathbf{x}]/I$  and  $\overline{b_i}^G = b_i$ . Then all we have to do to get  $\mathbf{m}_p$  is to compute the action  $\overline{pb_i}^G$  for each basis element  $b_i$ , which is easily done if  $G$  is available.

**Example 2.** The following two equations describe the intersection of a line and a circle as illustrated in Figure 2.

$$\begin{aligned} x^2 + y^2 - 1 &= 0 \\ x - y &= 0. \end{aligned} \tag{2.15}$$

A Gröbner basis for this system is

$$\begin{aligned} y^2 - \frac{1}{2} &= 0 \\ x - y &= 0, \end{aligned} \tag{2.16}$$

from which we trivially see that the solutions are  $\frac{1}{\sqrt{2}}(1, 1)$  and  $\frac{1}{\sqrt{2}}(-1, -1)$ . However, it is nevertheless instructive to construct the action matrix. In this case  $\mathcal{B} = \{y, 1\}$  are representatives for a basis for  $\mathbb{C}[\mathbf{x}]/I$  and we have  $T_x[1] = [x] = [y]$  and  $T_x[y] = [xy] = [y^2] = [\frac{1}{2}]$ , which yields the action matrix

$$\mathbf{m}_x = \begin{bmatrix} 0 & 1 \\ \frac{1}{2} & 0 \end{bmatrix}, \tag{2.17}$$

with eigenvalues  $\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}$ . □

### 2.2.3 A Note on Algebraic and Linear Bases

At this point there is a potentially confusing situation since there are two different types of bases at play. There is the linear basis  $\mathcal{B}$  of the quotient space  $\mathbb{C}[\mathbf{x}]/I$  and there is the algebraic basis (Gröbner basis)  $G$  of the ideal  $I$ . To make the subsequent arguments as transparent as possible for the reader we will emphasize this fact by referring to the former as a *linear basis* of  $\mathbb{C}[\mathbf{x}]/I$  and the latter as an *algebraic basis* of  $I$ .

### 2.2.4 Floating Point Gröbner Basis Computations

The well established Buchberger's algorithm is guaranteed to compute a Gröbner basis in finite time and works well in exact arithmetic [14]. However, due to round-off errors, it easily becomes unstable in floating point arithmetic and except for very small examples it becomes practically useless. The reason for this is that in the Gröbner basis computation, leading terms are successively eliminated from the generators of  $I$  by pairwise subtraction of polynomials, much like Gaussian elimination. This leads to cancellation effects where it becomes impossible to tell whether a certain coefficient should be zero or not.

A technique introduced by Faugere *et al* in [18] is to write the system of equations on matrix form

$$\mathbf{C}\mathbf{X} = 0, \tag{2.18}$$

where  $\mathbf{X} = [\mathbf{x}^{\alpha_1} \ \dots \ \mathbf{x}^{\alpha_n}]^T$  is a vector of monomials with the notation  $\mathbf{x}^{\alpha_k} = x_1^{\alpha_{k1}} \cdots x_s^{\alpha_{ks}}$  and  $\mathbf{C}$  is a matrix of coefficients. Elimination of leading terms now translates to matrix operations and we then have access to a whole battery of techniques from numerical linear algebra allowing us to perform many eliminations at the same time with control on pivoting *etc.*

This technique takes us further, but for larger more demanding problems it is necessary to study a particular class of equations and use knowledge of what the structure of the Gröbner basis should be to design a special purpose Gröbner basis solver [48]. Typical examples where this method can be applied are: relative orientation for omnidirectional cameras [22], fundamental matrix estimation with radial distortion [36], optimal three view triangulation [52], *etc.* The typical work flow has been to study the particular problem at hand with the aid of a computer algebra system such as Maple or Macaulay2 [4] and extract information such as the leading terms of the Gröbner basis, the monomials to use as a basis for  $\mathbb{C}[\mathbf{x}]/I$ , the number of solutions, *etc* and work out a specific

set of larger (Gauss-Jordan) elimination steps leading to the construction of a Gröbner basis for  $I$ .

Although, these techniques have permitted the solution to a large number of previously unsolved problems, many difficulties remain. Most notably, the above mentioned elimination steps (if at all doable) are often hopelessly ill conditioned [52, 37]. This is in part due to the fact that one has focused on computing a complete and correct Gröbner basis respecting a properly defined monomial order, which we show is not necessary.

In this work we move away from the goal of computing a Gröbner basis for  $I$  and focus on finding a representative of  $f$  in terms of a linear combination of a basis  $\mathcal{B}$ , since this is the key to constructing  $\mathbf{m}_p$ . We denote this operation  $\overline{f}$  for a given  $f \in \mathbb{C}[\mathbf{x}]$ . Specifically, it is not necessary to be able to compute  $\overline{f}$  for any  $f \in \mathbb{C}[\mathbf{x}]$ . To construct  $\mathbf{m}_p$ , we only need to worry about finding  $\overline{f}$  for  $f \in p\mathcal{B} \setminus \mathcal{B}$ , which is an easier task. It should however be noted that the computations we do much resemble those necessary to get a Gröbner basis.

A further advantage of not having to compute a complete Gröbner basis is that we are not bound by any particular monomial order which as we will see, when used right, buys considerable numerical stability. In addition to this we introduce an object which generalizes the action matrix and can be computed even when a true linear basis for  $\mathbb{C}[\mathbf{x}]/I$  cannot be used.

Drawing on these observations, we investigate in detail the exact matrix operations needed to compute  $\overline{f}$  and thus obtain a procedure which is both faster and more stable, enabling the solution of a larger class of problems than previously possible. The theory behind these statements is explored in Chapter 3 and subsequently used in Chapter 4 to derive new stable algorithms for equation solving.



# Chapter 3

## Theoretical Contributions

In this chapter we present a new way of looking at the action matrix method for polynomial equation solving. The advantage of the new formulation is that it yields more freedom in how the action matrix is computed allowing us to derive numerically more stable algorithms.

### 3.1 A New Approach to the Action Matrix Method

We start with a few examples that we will use to clarify the ideas of this chapter.

**Example 3.** In the five point relative orientation problem for calibrated cameras, [34, 15, 44, 49], the calculation of the essential matrix using 5 image point correspondences leads to 10 equations of degree 3 in 3 unknowns. These equations involve 20 monomials. By writing the equations as in (2.18) and using a total degree ordering on the monomials we get a coefficient matrix  $\mathbf{C}$  of size  $10 \times 20$  and a monomial vector  $\mathbf{X} = [\mathbf{x}^{\alpha_1} \dots \mathbf{x}^{\alpha_n}]^T$  with 20 monomials. It turns out that the first  $10 \times 10$  block  $\mathbf{C}_1$  of  $\mathbf{C} = [\mathbf{C}_1 \ \mathbf{C}_2]$  is in general of full rank and thus the first 10 monomials  $\mathbf{X}_1$  can be expressed in terms of the last 10 monomials  $\mathbf{X}_2$  as

$$\mathbf{X}_1 = -\mathbf{C}_1^{-1}\mathbf{C}_2\mathbf{X}_2. \quad (3.1)$$

This makes it possible to regard the monomials in  $\mathbf{X}_2$  as representatives of a linear basis for  $\mathbb{C}[\mathbf{x}]/I$ . It is now straightforward to calculate the action matrix for  $T_x$  (the multiplication operator for multiplication by  $x$ ) since monomials in the linear basis are either mapped to monomials in the basis or to monomials in  $\mathbf{X}_1$ , which can be expressed in terms of the basis using (3.1).  $\square$

In this example the linear basis  $\mathbf{X}_2$  can be thought of as a basis for the space of remainders after division with a Gröbner basis for one choice of monomial order and this is how these computations have typically been viewed. However, the calculations above are not really dependent on any properly defined monomial order and it seems that they should be meaningful irrespective of whether a true monomial order is used or not. Moreover, we do not use all the Gröbner basis properties.

Based on these observations we emphasize two important facts: (i) We are not interested in finding the Gröbner basis or a basis for the remainder space

relative to some Gröbner basis *per se*; it is enough to get a well defined mapping  $\bar{f}$  and (ii) it suffices to calculate  $\bar{f}$  on the elements  $x \cdot \mathbf{x}^{\alpha_i}$ , *i.e.* we do not need to be able to compute  $\bar{f}$  for all  $f \in \mathbb{C}[\mathbf{x}]$ . These statements and their implications will be made more precise further on.

**Example 4.** Consider the equations

$$\begin{aligned} f_1 &= xy + x - y - 1 = 0, \\ f_2 &= xy - x + y - 1 = 0, \end{aligned} \tag{3.2}$$

with solutions  $(-1, -1), (1, 1)$ . Now let  $\mathcal{B} = \{x, y, 1\}$  be a set of representatives for the equivalence classes in  $\mathbb{C}[\mathbf{x}]/I$  for this system. The set  $\mathcal{B}$  does not constitute a proper basis for  $\mathbb{C}[\mathbf{x}]/I$  since the elements of  $\mathcal{B}$  represent linearly dependent equivalence classes. They do however span  $\mathbb{C}[\mathbf{x}]/I$ . Now study the operator  $T_y$  acting on  $\mathcal{B}$ . We have  $T_y(1) = y$ ,  $T_y(x) = xy \sim x - y + 1$  and  $T_y(y) = y^2 \sim xy \sim x - y + 1$  which gives a multiplication matrix

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

An eigendecomposition of this matrix yields the solutions  $(-1, -1), (1, 1), (-1, 0)$ . Of these the first two are true solutions to the problem, whereas the last one does not satisfy the equations and is thus a false zero.  $\square$

In this example we used a set of monomials  $\mathcal{B}$  whose corresponding equivalence classes spanned  $\mathbb{C}[\mathbf{x}]/I$ , but were not linearly independent. However, it was still possible to express the image  $T_y(\mathcal{B})$  in terms of  $\mathcal{B}$ . The elements of the resulting action matrix are not uniquely determined. Nevertheless we were able to use it to find the solutions to the problem. In this section we give general conditions for when a set  $\mathcal{B}$  can be used to construct a multiplication matrix which produces the desired set of zeros, possibly along with a set of false zeros, which need to be filtered out.

More generally this also means that the chosen representatives of the linear basis of  $\mathbb{C}[\mathbf{x}]/I$  need not be low order monomials given by a Gröbner basis. In fact, they need not be monomials at all, but could be general polynomials.

Drawing on the concepts illustrated in the above two examples we define a *solving basis*, similar to  $\mathcal{B}$  in Example 4. The overall purpose of the definition is to rid our selves of the need of talking about a Gröbner basis and properly defined monomial orders, thus providing more room to derive numerically stable algorithms for computation of the action matrix and similar objects.

In the following we will also provide techniques for determining if a candidate basis  $\mathcal{B}$  constitutes a solving basis and we will give numerically stable techniques for basis selection in too large (linearly dependent) solving bases, here referred to as redundant bases.

### 3.1.1 Solving Bases

We start off with a set of polynomial equations as in (2.10) and a (point) set of zeros  $V(f_1, \dots, f_m)$  and make the following definition

**Definition 5.** Consider a finite subset  $\mathcal{B} \subset \mathbb{C}[\mathbf{x}]$  of the set of polynomials over the complex numbers. If for each  $b_i \in \mathcal{B}$  and some  $p \in \mathbb{C}[x]$  we express  $pb_i$  as a linear combination of basis elements as

$$p(\mathbf{x})b_i(\mathbf{x}) = \sum_j m_{ij}b_j(\mathbf{x}), \quad (3.3)$$

for some (not necessarily unique) coefficients  $m_{ij}$  and where equality means equality on  $V$ , then we call  $\mathcal{B}$  a *solving basis* for (2.10) w.r.t  $p$ .  $\square$

We now get the following for the matrix  $\mathbf{m}_p$  made up of the coefficients  $m_{ij}$ .

**Theorem 6.** *Given a solving basis  $\mathcal{B}$  for (2.10) w.r.t  $p$ , the evaluation of  $p$  on  $V$  is an eigenvalue of the matrix  $\mathbf{m}_p$ . Moreover, the vector  $\mathbf{b} = (b_1, \dots, b_r)^T$  evaluated on  $V$  is an eigenvector of  $\mathbf{m}_p$ .*

*Proof.* By the definition of  $\mathbf{m}_p$ , we get

$$p(\mathbf{x})\mathbf{b}(\mathbf{x}) = \begin{bmatrix} pb_1 \\ \vdots \\ pb_r \end{bmatrix} = \begin{bmatrix} \sum_j m_{1j}b_j \\ \vdots \\ \sum_j m_{rj}b_j \end{bmatrix} = \mathbf{m}_p\mathbf{b}(\mathbf{x}) \quad (3.4)$$

for  $\mathbf{x} \in V$ .  $\square$

As will become clear further on, when  $\mathcal{B}$  is a true basis for  $\mathbb{C}[\mathbf{x}]/I$ , then the matrix  $\mathbf{m}_p$  defined here is simply the transposed action matrix for multiplication by  $p$ .

Given a solving basis, the natural question to ask is now under which circumstances all solutions to the related system of equations can be obtained from an eigenvalue decomposition of  $\mathbf{m}_p$ . We next explore some conditions under which this is possible. A starting point is the following definition

**Definition 7.** A solving basis  $\mathcal{B}$  is called a *complete solving basis* if the inverse image of the mapping  $x \mapsto \mathbf{b}(x)$  from variables to monomial vector is finite for all points.  $\square$

A complete solving basis allows us to recover all solutions from  $\mathbf{m}_p$  as shown in the following theorem.

**Theorem 8.** *Let  $\mathcal{B}$  be a complete solving basis for (2.10) and  $\mathbf{m}_p$  as above and assume that for all eigenvalues  $\lambda_i$  we have  $\lambda_i \neq \lambda_j$  for  $i \neq j$ . Then the complete set of solutions to (2.10) can be obtained from the set of eigenvectors  $\{v_i\}$  of  $\mathbf{m}_p$ .*

*Proof.* The vector  $\mathbf{b}(\mathbf{x})$  evaluated on  $V$  is an eigenvector of  $\mathbf{m}_p$ . The number of eigenvectors and eigenvalues of  $\mathbf{m}_p$  is finite so we can compute all eigenvectors  $\{v_i\}$  and get  $\{\mathbf{b}(\bar{\mathbf{x}})\}$  for all  $\bar{\mathbf{x}} \in V$  as a subset of these. Applying  $\mathbf{b}^{-1}$  to  $v_i$  for all  $i$  thus yields a finite set of points containing  $V$ . Evaluation in (2.10) allows us to filter out the points of this set which are not solutions to (2.10).  $\square$

If on the other hand the inverse image is not finite for some  $v_i$  so that we get a parameter family  $\mathbf{x}$  corresponding to this eigenvector, then the correct solution can typically not be obtained without further use of the equations (2.10) as illustrated in the following example.

**Example 9.** Consider the polynomial system

$$\begin{aligned} y^2 - 2 &= 0 \\ x^2 - 1 &= 0 \end{aligned} \tag{3.5}$$

with  $V = \{(1, \sqrt{2}), (-1, \sqrt{2}), (1, -\sqrt{2}), (-1, -\sqrt{2}), \}$ . Clearly,  $\mathcal{B} = \{x, 1\}$  with monomial vector  $\mathbf{b}(x, y) = [x \ 1]^T$ , is a solving basis w.r.t  $x$  for this example since  $1 \cdot x = x$  and  $x \cdot x = x^2 = 1$  on  $V$ . Hence,  $\mathbf{b}(x, y)$  evaluated on  $V$  is an eigenvector of

$$\mathbf{m}_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{3.6}$$

which is easily confirmed. However, these eigenvectors do not provide any information about the  $y$ -coordinate of the solutions. We could try adding  $y$  to  $\mathcal{B}$  but this would not work since the values of  $xy$  on  $V$  cannot be expressed as a linear combination of  $x$  and  $y$  evaluated on  $V$ . A better choice of solving basis would be  $\mathcal{B} = \{xy, x, y, 1\}$ .  $\square$

At a first glance, Theorem 8 might not seem very useful since solving for  $x$  from  $\mathbf{b}(x) = v_i$  potentially involves solving a new system of polynomial equations. However, it provides a tool for ruling out choices of  $\mathcal{B}$  which are not practical to work with. Moreover, there is usually much freedom in the choice of  $\mathcal{B}$ . In general,  $\mathcal{B}$  can be set of polynomials. However, it is often practical to work with a basis of monomials. For each  $b_i$  we then get the following result

**Corollary 10.** *If  $\mathcal{B}$  consists of monomials  $b_i$  on the form  $b_i(x) = x_1^{\alpha_{i1}} \dots x_s^{\alpha_{is}}$  and the  $r \times s$  matrix  $A$  with  $A_{ij} = \alpha_{ij}$  is of rank  $s$ , then all solutions to (2.10) can be obtained from the eigenvectors of  $\mathbf{m}_{x_k}$ .*

*Proof.* Taking the logarithm of  $b_j(\mathbf{x})$  we get component wise

$$\log(b_i(\mathbf{x})) = \sum_j \alpha_{ij} \log(\tilde{x}_j), \tag{3.7}$$

where  $\tilde{x}_j = \pm x_j$  if necessary. Using the matrix  $A$ , this can be written

$$\log(\mathbf{b}(\mathbf{x})) = A \begin{bmatrix} \log(\tilde{x}_1) \\ \vdots \\ \log(\tilde{x}_s) \end{bmatrix}. \tag{3.8}$$

If  $\text{rank}(A) = s$  then we can solve linearly for  $\log(\tilde{\mathbf{x}})$  and theorem 8 yields the conclusion.  $\square$

We get an even more convenient situation if the right monomials are included in  $\mathcal{B}$ :

**Corollary 11.** *If  $\{1, x_1, \dots, x_s\} \subset \mathcal{B}$ , then all solutions to (2.10), can be directly read off from the eigenvectors of  $\mathbf{m}_{x_k}$ .*

*Proof.* Since the monomials  $\{1, x_1, \dots, x_s\}$  occur in  $\mathcal{B}$ , they enter in the vector  $\mathbf{b}(x)$  and hence the mapping in Definition 7 is injective with a trivial inverse.  $\square$

The purpose of Theorem 6 and Corollaries 10 and 11 are to provide guarantees for when all information about the solutions can be obtained from the multiplication matrices. The idea behind these results is to consider the relation between a solution point  $\bar{\mathbf{x}}$  and monomial vector  $\mathbf{b}(\bar{\mathbf{x}})$ . We know that a solution point  $\bar{\mathbf{x}}$  *always* corresponds to a vector  $\mathbf{b}(\bar{\mathbf{x}})$  which is an eigenvector of the corresponding multiplication matrix. The only way we could miss some solutions would hence be if two different zeros  $\bar{\mathbf{x}}_1$  and  $\bar{\mathbf{x}}_2$  map to the same monomial vector. However, if the mapping  $\mathbf{x} \mapsto \mathbf{b}(\mathbf{x})$  is injective, this cannot happen and we are safe.

The situation in Corollary 11 is certainly the most convenient one. However, even if not all variables are included as elements in  $\mathcal{B}$ , we can often still express each variable  $x_k$  as a linear combination of the basis elements  $b_i(\mathbf{x})$  for  $\mathbf{x} \in V$  by making use of the original equations. We thus again obtain a well defined inverse to the mapping in Definition 7.

**Example 12.** Consider the polynomial system (3.2) from Example 4. Subtracting  $f_1$  and  $f_2$  and dividing by 2 we get a third polynomial  $f_3 = x - y$ . Thus  $\mathcal{B} = \{y, 1\}$  constitutes a solving basis w.r.t  $x$  since  $T_x(1) = x = y$  (on  $V$ ) and  $T_x(y) = xy = x - y + 1 = 1$  (on  $V$ ). The vector of monomials  $\mathbf{b}(x, y) = [y \ 1]^T$  is not invertible since it does not give any information about the  $x$  coordinate. However, we can use  $f_3 = x - y = 0$  to get the solutions from from the eigenvectors.  $\square$

Finally, we show how the concept of solving basis connects to the standard theory of action matrices in the quotient space  $\mathbb{C}[x]/I$ .

**Theorem 13.** *If the ideal  $I$  generated by (2.10) is radical, then a complete solving basis  $\mathcal{B}$  w.r.t to  $p$  for (2.10) with the property that all eigenvalues of  $\mathbf{m}_p$  are distinct spans  $\mathbb{C}[x]/I$ .*

*Proof.* Since  $I$  is radical,  $\mathbb{C}[x]/I$  is isomorphic to  $\mathbb{C}[V]$ , the ring of all polynomial functions on  $V$ . Moreover, since  $V$  is finite, all functions on  $V$  are polynomial and hence  $\mathbb{C}[V]$  can be identified with  $\mathbb{C}^r$ , where  $r = |V|$ . Consider now the matrix  $B = [\mathbf{b}(x_1), \dots, \mathbf{b}(x_r)]$ . Each row of  $B$  corresponds to a (polynomial) function on  $V$ . Hence, if we can show that  $B$  has row rank  $r$ , then we are done. Due to theorem 6, all  $\mathbf{b}(x_i)$  are eigenvectors of  $\mathbf{m}_p$  corresponding to eigenvalues  $p(x_i)$ . By the assumption of distinct eigenvalues we have  $p(x_i) \neq p(x_j)$  whenever  $\mathbf{b}(x_i) \neq \mathbf{b}(x_j)$ . Since  $\mathcal{B}$  is a complete solving basis we have  $\mathbf{b}(x_i) \neq \mathbf{b}(x_j)$  whenever  $x_i \neq x_j$ . This means that the  $r$  points in  $V$  correspond to distinct eigenvalues and hence, since eigenvectors corresponding to different eigenvalues are linearly independent,  $B$  has column rank  $r$ . For any matrix row rank equals column rank and we are done.  $\square$

The above theorem provides a correspondence between solving bases and linear bases for  $\mathbb{C}[\mathbf{x}]/I$  and in principle states that under some extra restrictions, a solving basis is simply a certain choice of linear basis for  $\mathbb{C}[\mathbf{x}]/I$  and then the matrix  $\mathbf{m}_p$  turns into the action matrix. However, relaxing these extra restrictions we get something which is not necessarily a basis for  $\mathbb{C}[\mathbf{x}]/I$  in the usual sense, but can still be used to construct a matrix  $\mathbf{m}_p$  which encodes the solutions. This is what we call a solving basis. Using the concept of a solving basis provides two distinctive advantages:

(i) For a radical polynomial system with  $r$  zeros,  $\mathbb{C}[\mathbf{x}]/I$  is  $r$ -dimensional, so a basis for  $\mathbb{C}[\mathbf{x}]/I$  contains  $r$  elements. This need not be the case for a solving basis, which could well contain more than  $r$  elements, but due to Theorem 8 still provides the right solutions. This fact is exploited in Section 4.1.

(ii) Typically, the arithmetic in  $\mathbb{C}[\mathbf{x}]/I$  has been computed using a Gröbner basis for  $I$ , which directly provides a monomial basis for  $\mathbb{C}[\mathbf{x}]/I$  in form of the set of monomials which are not divisible by the Gröbner basis. In this work we move focus from Gröbner basis computation to the actual goal of expressing the products  $pb_i$  in terms of a set of linear basis elements and thus no longer need to adhere to the overly strict ordering rules imposed by a particular monomial order. This freedom is exploited in Sections 4.2.1 and 4.2.2.

Finally, (i) and (ii) are combined in Section 4.2.3.

### 3.1.2 Solving Basis Computations using Numerical Linear Algebra

We now describe the most straightforward technique for deciding whether a candidate basis  $\mathcal{B}$  w.r.t one of the variables  $x_k$ , can be used as a solving basis and simultaneously calculate the action of  $T_{x_k}$  on the elements of  $\mathcal{B}$ .

We start by generating more equations by multiplying the original set of equations by a hand crafted (problem dependent) set of monomials. This yields additional equations, which are equivalent in terms of solutions, but hopefully linearly independent from the original ones. In Example 9, we could multiply by *e.g.*  $\{x, y, 1\}$ , yielding  $xy^2 - 2x, x^3 - x, y^3 - 2y, x^2y - y, y^2 - 2, x^2 - 1$ .

Given a candidate for a linear basis  $\mathcal{B}$  of monomials one then partitions the set of all monomials  $\mathcal{M}$  occurring in the equations in to three parts  $\mathcal{M} = \mathcal{E} \cup \mathcal{R} \cup \mathcal{B}$ , where  $\mathcal{R} = x_k \mathcal{B} \setminus \mathcal{B}$  is the set of monomials that need to be expressed in terms of  $\mathcal{B}$  to satisfy the definition of a solving basis and  $\mathcal{E} = \mathcal{M} \setminus (\mathcal{R} \cup \mathcal{B})$  is the set of remaining (excessive) monomials. Each column in the coefficient matrix represents a monomial, so we reorder the columns and write

$$\mathbf{C} = [\mathbf{C}_{\mathcal{E}} \quad \mathbf{C}_{\mathcal{R}} \quad \mathbf{C}_{\mathcal{B}}], \quad (3.9)$$

reflecting the above partition. The  $\mathcal{E}$ -monomials are not used in the action matrix computation so we eliminate them by putting  $\mathbf{C}_{\mathcal{E}}$  on row echelon form using LU factorization

$$\begin{bmatrix} \mathbf{U}_{\mathcal{E}1} & \mathbf{C}_{\mathcal{R}1} & \mathbf{C}_{\mathcal{B}1} \\ \mathbf{0} & \mathbf{C}_{\mathcal{R}2} & \mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = \mathbf{0}. \quad (3.10)$$

We now discard the top rows and provided that enough linearly independent equations were added in the first step so that  $\mathbf{C}_{\mathcal{R}2}$  is of full rank, we multiply by  $\mathbf{C}_{\mathcal{R}2}^{-1}$  from the left producing

$$[\mathbf{I} \quad \mathbf{C}_{\mathcal{R}2}^{-1} \mathbf{C}_{\mathcal{B}2}] \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = \mathbf{0}, \quad (3.11)$$

or equivalently

$$\mathbf{X}_{\mathcal{R}} = -\mathbf{C}_{\mathcal{R}2}^{-1} \mathbf{C}_{\mathcal{B}2} \mathbf{X}_{\mathcal{B}}, \quad (3.12)$$

which means that the  $\mathcal{R}$ -monomials can be expressed as a linear combination of the basis monomials. Thus  $\mathcal{B}$  is a solving basis and the matrix  $\mathbf{m}_{x_k}$  can easily be constructed as in (3.3). In other words, given an enlarged set of equations and a choice of linear basis  $\mathcal{B}$ , the full rank of  $\mathbf{C}_{\mathcal{R}2}$  is sufficient to solve (2.10) via eigendecomposition of  $\mathbf{m}_{x_k}$ . The above method is summarized in Algorithm 1 and given the results of Section 3.1.1 we now have the following

**Result 14.** *Algorithm 1 yields the complete set of zeros of a polynomial system, given that the pre- and postconditions are satisfied.*

*Proof.* The postcondition that  $\mathbf{C}_{\mathcal{R}2}$  is of full rank ensures that  $\mathcal{B}$  is a solving basis and together with the preconditions, Theorem 8 and Corollary 11 then guarantees the statement.  $\square$

**Example 15.** Consider the equations from Example 2. Multiplying the second equation by  $x$  and  $y$  yields the enlarged system

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{bmatrix} = 0, \quad (3.13)$$

with  $\mathcal{M} = \{x^2, xy, y^2, x, y, 1\}$  and since we chose  $\mathcal{B} = \{y, 1\}$ , we get  $\mathcal{R} = \{xy, x\}$  and  $\mathcal{E} = \{x^2, y^2\}$ . After Step 11 and 12 of Algorithm 1 we have  $\mathbf{C}_{\mathcal{R}2} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$  and  $\mathbf{C}_{\mathcal{B}2} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$  and inserting into (4.11) we obtain

$$\begin{bmatrix} xy \\ x \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{2} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y \\ 1 \end{bmatrix}, \quad (3.14)$$

which then allows us to construct  $\mathbf{m}_x$  for this example.  $\square$

A typical problem that might occur is that some eigenvalues of  $\mathbf{m}_{x_k}$  are equal, which means that two or more zeros have equal  $x_k$ -coordinate. Then the corresponding eigenvectors can not be uniquely determined. This problem can be resolved by computing  $\mathbf{m}_{x_k}$  for several  $k$  and then forming a random linear combination  $\mathbf{m}_{a_1x_1+\dots+a_sx_s} = a_1\mathbf{m}_{x_1} + \dots + a_s\mathbf{m}_{x_s}$ , which then with very small probability has two equal eigenvalues.

As previously mentioned, computing  $\mathbf{m}_p$  for a larger problem is numerically very challenging and the predominant issue is expressing  $p\mathcal{B}$  in terms of  $\mathcal{B}$ , via something similar to (3.12). The reason for this is that without proper care,  $\mathbf{C}_{\mathcal{R}2}$  tends to become very ill conditioned (condition numbers of  $10^{10}$  or higher are not uncommon). This was also the reason that extremely slow emulated 128 bit numerics had to be used in [52] to get a working algorithm.

In the next chapter we investigate techniques to circumvent this problem and produce a well conditioned  $\mathbf{C}_{\mathcal{R}2}$ , thus drastically improving numerical stability.

---

**Algorithm 1** Compute a solving basis w.r.t  $x_k$  and use it to solve a polynomial system.

---

**Require:** List of equations  $F = \{f_1, \dots, f_m\}$ , set of basis monomials  $\mathcal{B}$  containing the coordinate variables  $1, x_1, \dots, x_s$ ,  $m$  lists of monomials  $\{L_i\}_{i=1}^m$ .

**Ensure:**  $\mathbf{C}_{\mathcal{R}2}$  is of full rank, eigenvalues of  $\mathbf{m}_{x_k}$  are distinct.

- 1:  $F_{\text{ext}} \leftarrow F$
  - 2: **for all**  $f_i \in F$  **do**
  - 3:   **for all**  $\mathbf{x}^{\alpha_j} \in L_i$  **do**
  - 4:      $F_{\text{ext}} \leftarrow F_{\text{ext}} \cup \{\mathbf{x}^{\alpha_j} \cdot f_i\}$
  - 5:   **end for**
  - 6: **end for**
  - 7: Construct coefficient matrix  $\mathbf{C}$  from  $F_{\text{ext}}$ .
  - 8:  $\mathcal{M} \leftarrow$  The set of all monomials occurring in  $F_{\text{ext}}$ .
  - 9:  $\mathcal{R} \leftarrow x_k \cdot \mathcal{B} \setminus \mathcal{B}$
  - 10:  $\mathcal{E} \leftarrow \mathcal{M} \setminus (\mathcal{R} \cup \mathcal{B})$
  - 11: Reorder and partition  $\mathbf{C}$ :  $\tilde{\mathbf{C}} = [\mathbf{C}_{\mathcal{E}} \ \mathbf{C}_{\mathcal{R}} \ \mathbf{C}_{\mathcal{B}}]$ .
  - 12: LU-factorize to obtain  $\mathbf{C}_{\mathcal{R}2}$  and  $\mathbf{C}_{\mathcal{B}2}$  as in (3.10).
  - 13: Use (3.12) to express  $x_k \cdot \mathbf{x}^{\alpha_i}$  in terms of  $\mathcal{B}$  and store the coefficients in  $\mathbf{m}_{x_k}$ .
  - 14: Compute eigenvectors of  $\mathbf{m}_{x_k}$  and read off the tentative set of solutions.
  - 15: Evaluate in  $F$  to filter out possible false zeros.
-

# Chapter 4

## Techniques

Drawing on the ideas introduced in the previous chapter, this chapter presents a range of techniques for improving the numerical stability of algorithms which rely on eigenvalue decomposition of a multiplication matrix. These techniques are based on efficient and numerically sound methods from numerical linear algebra. A benefit of this is that such routines have a relatively long history and are very well studied. Moreover, there exist highly optimized implementations in free code libraries such as LAPACK [38].

### 4.1 Using Redundant Solving Bases - The Truncation Method

As mentioned in Section 3.1.2, the sub matrix  $\mathbf{C}_{\mathcal{R}2}$  which appears in Equation 3.10 is a large cause of numerical problems in the equation solving process. A typical situation with an ill conditioned or rank deficient  $\mathbf{C}_{\mathcal{R}2}$  is that there are a few problematic monomials where the corresponding columns in  $\mathbf{C}$  are responsible for the deteriorated conditioning of  $\mathbf{C}_{\mathcal{R}2}$ . A straightforward way to improve the situation is to simply include the problematic monomials in  $\mathcal{B}$ , thus avoiding the need to express these in terms of the other monomials. In practice this means that some columns of  $\mathbf{C}_{\mathcal{R}}$  are moved into  $\mathbf{C}_{\mathcal{B}}$ . This technique is supported by Theorem 8, which guarantees that we will find the original set of solutions among the eigenvalues/eigenvectors of the larger  $\mathbf{m}_p$  found using this redundant basis. The price we have to pay is performing an eigenvalue decomposition on a larger matrix.

Not all monomials from  $\mathcal{M}$  can be included in the basis  $\mathcal{B}$  while still enabling the calculation of the necessary multiplication matrices. In general it is a difficult question exactly which monomials can be used or even if there *exists* a set  $\mathcal{B}$  among  $\mathcal{M}$ , which can be used as a solving basis. One can however easily see that  $\mathcal{B}$  has to be a subset of the following set, which we denote the *permissible* monomials:

**Definition 16.** The set of *permissible* monomials is the set

$$\mathcal{P} = \{b \in \mathcal{M} : pb \in \mathcal{M}\} \tag{4.1}$$

of monomials which stay in  $\mathcal{M}$  under multiplication by  $p$ .

An example of how the redundant solving basis technique can be used is provided by the problem of  $L_2$ -optimal triangulation from three views [52]. The optimum is found among the up to 47 stationary points, which are zeros of a polynomial system in three variables. In this example an enlarged set of 255 equations in 209 monomials were used to get a Gröbner basis. Since the solution dimension  $r$  is 47 in this case, the 47 lowest order monomials were used as a basis for  $\mathbb{C}[\mathbf{x}]/I$  in [52], yielding a numerically difficult situation. In fact, as will be shown in more detail in the experiments section, this problem can be solved by simply including more elements in  $\mathcal{B}$ . In this example, the complete permissible set  $\mathcal{P}$  contains 154 monomials. By including all of these in  $\mathcal{B}$  leaving 55 monomials to be expressed in terms of  $\mathcal{B}$ , we get a much smaller and in this case better conditioned elimination step. As mentioned above, this leads to a larger eigenvalue decomposition, but all true solutions can still be found among the larger set of eigenvalues/eigenvectors. This is illustrated in Figure 4.1, where the set of eigenvalues computed from  $\mathbf{m}_{x_k}$  for one instance are plotted in the complex plane together with the actual solutions of the polynomial system.

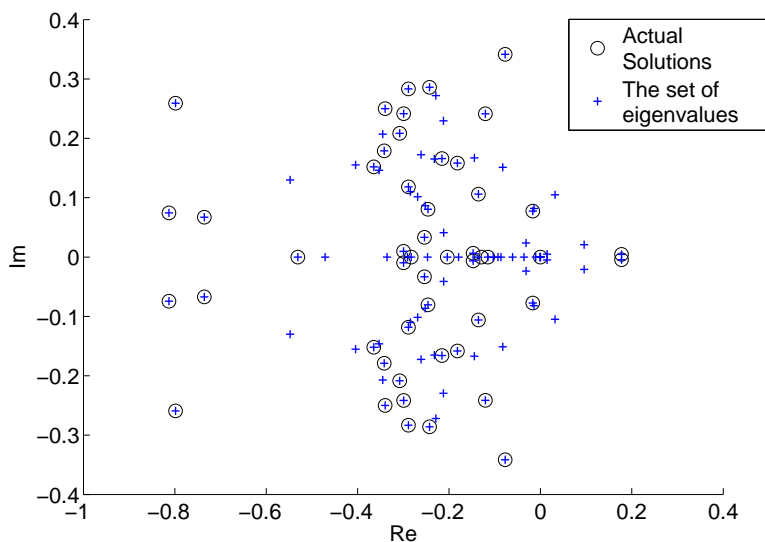


Figure 4.1: Eigenvalues of the action matrix using the redundant basis method and actual solutions to the polynomials system plotted in the complex number plane. The former are a strict superset of the latter.

## 4.2 Basis Selection

In the previous section we saw how it is possible to pick a “too large” ( $> r$  elements) linear basis  $\mathcal{P}$  and still use it to solve the equations. In this section we show how one can select a true (linearly independent) basis as a subset of  $\mathcal{P}$  in a numerically stable way and thus gain both speed and stability. In the following,  $\mathcal{P}$  denotes any subset of  $\mathcal{M}$  with the property that the obtained  $\mathbf{C}_{\mathcal{R}2}$  is of full rank, thus making  $\mathcal{P}$  a solving basis.

Since the set  $V$  of zeros of (2.10) is finite with  $r$  points,  $\mathcal{P}$  seen as a set of functions on  $V$  contains at most  $r$  linearly independent elements. It should therefore be possible to choose a subset  $\mathcal{P}' \subset \mathcal{P}$  such that the elements in  $\mathcal{P}'$  can be expressed as linear combinations of elements in  $\mathcal{P} \setminus \mathcal{P}'$ . By dropping  $\mathcal{P}'$  from the solving basis, the set  $\mathcal{B} = \mathcal{P} \setminus \mathcal{P}'$  would thus constitute a new tighter solving basis w.r.t the same multiplier  $p$  and ideal  $I$  as  $\mathcal{P}$ .

We now present two numerically stable techniques for constructing a true basis  $\mathcal{B}$  from a redundant solving basis  $\mathcal{P}$ .

### 4.2.1 The QR Method

We start by selecting  $\mathcal{P}$  as large as possible, still yielding a full rank  $\mathbf{C}_{\mathcal{R}2}$  and form  $[\mathbf{C}_{\mathcal{E}} \ \mathbf{C}_{\mathcal{R}} \ \mathbf{C}_{\mathcal{P}}]$ . Any selection of basis monomials  $\mathcal{B} \subset \mathcal{P}$  will then correspond to a matrix  $\mathbf{C}_{\mathcal{B}}$  consisting of a subset of the columns of  $\mathbf{C}_{\mathcal{P}}$ .

By performing Gaussian elimination we again obtain (3.10), but with  $\mathcal{B}$  replaced by  $\mathcal{P}$ , letting us get rid of the  $\mathcal{E}$ -monomials by discarding the top rows. Furthermore, the  $\mathcal{R}$ -monomials will all have to be expressed in terms of the  $\mathcal{P}$ -monomials so we continue the elimination putting  $\mathbf{C}_{\mathcal{R}2}$  on triangular form, obtaining

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1} \\ \mathbf{0} & \mathbf{C}_{\mathcal{P}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \quad (4.2)$$

At this point we could simply continue the Gaussian elimination, with each new pivot element representing a monomial expressed in terms of the remaining basis monomials. However, this typically leads to poor numerical performance since, as previously mentioned, the elimination might be very ill conditioned. This is where the basis selection comes to play.

As noted above we can choose which of the  $p$  monomials in  $\mathcal{P}$  to put in the basis and which to reduce. This is equivalent to choosing a permutation  $\Pi$  of the columns of  $\mathbf{C}_{\mathcal{P}2}$ ,

$$\mathbf{C}_{\mathcal{P}2}\Pi = [c_{\pi(1)} \ \dots \ c_{\pi(p)}] \quad (4.3)$$

and then proceed using standard elimination. The goal must thus be to make this choice so as to minimize the condition number  $\kappa([c_{\pi(1)} \ \dots \ c_{\pi(p-r)}])$  of the first  $p-r$  columns of the permuted matrix. In its generality, this is a difficult combinatorial optimization problem. However, the task can be approximately solved in an attractive way by QR factorization with column pivoting [23]. With this algorithm,  $\mathbf{C}_{\mathcal{P}2}$  is factorized as

$$\mathbf{C}_{\mathcal{P}2}\Pi = \mathbf{Q}\mathbf{U}, \quad (4.4)$$

where  $\mathbf{Q}$  is orthogonal and  $\mathbf{U}$  is upper triangular. By solving for  $\mathbf{C}_{\mathcal{P}2}$  in (4.4) and substituting into (4.2) followed by multiplication from the left with  $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^T \end{bmatrix}$  and from the right with  $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \Pi \end{bmatrix}$ , we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1}\Pi \\ \mathbf{0} & \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \Pi^T \mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \quad (4.5)$$

We observe that  $\mathbf{U}$  is in general not quadratic and write  $\mathbf{U} = [\mathbf{U}_{\mathcal{P}'2} \ \mathbf{C}_{\mathcal{B}2}]$ , where  $\mathbf{U}_{\mathcal{P}'2}$  is quadratic upper triangular. We also write  $\mathbf{C}_{\mathcal{P}1}\Pi = [\mathbf{C}_{\mathcal{P}'1} \ \mathbf{C}_{\mathcal{B}1}]$

and  $\Pi^T \mathbf{X}_{\mathcal{P}1} = [\mathbf{X}_{\mathcal{P}'1} \quad \mathbf{X}_{\mathcal{B}}]^T$  yielding

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}'1} & \mathbf{C}_{\mathcal{B}1} \\ \mathbf{0} & \mathbf{U}_{\mathcal{P}'2} & \mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = \mathbf{0}. \quad (4.6)$$

Finally

$$\begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \end{bmatrix} = - \begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}'1} \\ \mathbf{0} & \mathbf{U}_{\mathcal{P}'2} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{C}_{\mathcal{B}1} \\ \mathbf{C}_{\mathcal{B}2} \end{bmatrix} \mathbf{X}_{\mathcal{B}} \quad (4.7)$$

is analogous to (3.12) and amounts to solving  $r$  upper triangular equation systems which can be efficiently done by back substitution.

The reason why QR factorization fits so nicely within this framework is that it simultaneously solves the two tasks of reduction to upper triangular form and numerically sound column permutation and with comparable effort to normal Gaussian elimination.

Furthermore, QR factorization with column pivoting is a widely used and well studied algorithm and there exist free, highly optimized implementations, making this an accessible approach.

Standard QR factorization successively eliminates elements below the main diagonal by multiplying from the left with a sequence of orthogonal matrices (usually Householder transformations). For matrices with more columns than rows (under-determined systems) this algorithm can produce a rank-deficient  $\mathbf{U}$  which would then cause the computations in this section to break down. QR with column pivoting solves this problem by, at iteration  $k$ , moving the column with greatest 2-norm on the last  $m - k + 1$  elements to position  $k$  and then eliminating the last  $m - k$  elements of this column by multiplication with an orthogonal matrix  $Q_k$ .

## 4.2.2 The SVD Method

By considering not only monomial bases, but more general polynomial bases it is possible to further improve numerical stability. We now show how singular value decomposition (SVD) can be used to construct a basis for  $\mathbb{C}[\mathbf{x}]/I$  as  $r$  linearly independent linear combinations of elements in a solving basis  $\mathcal{P}$ .

As in Section 4.2.1 we start out by selecting an as large as possible (redundant) solving basis and perform preliminary matrix operations forming (4.2), where the aim is now to construct a linearly independent basis from  $\mathcal{P}$ . We now do this by performing an SVD on  $\mathbf{C}_{\mathcal{P}2}$ , writing

$$\mathbf{C}_{\mathcal{P}2} = \mathbf{U}\Sigma\mathbf{V}^T, \quad (4.8)$$

where  $\mathbf{U}$  and  $\mathbf{V}$  are orthogonal and  $\Sigma$  is diagonal with typically  $r$  last elements zero  $\Sigma = \begin{bmatrix} \Sigma' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$  for a system with  $r$  solutions.

Now multiplying from the left with  $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}^T \end{bmatrix}$  and from the right with  $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{V} \end{bmatrix}$  in (4.2), we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1}\mathbf{V} \\ \mathbf{0} & \Sigma \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{V}^T \mathbf{X}_{\mathcal{P}} \end{bmatrix} = \mathbf{0}. \quad (4.9)$$

The matrix  $\mathbf{V}$  induces a change of basis in the space spanned by  $\mathcal{P}$  and we write  $\tilde{\mathbf{X}}_{\mathcal{P}} = \mathbf{V}^T \mathbf{X}_{\mathcal{P}} = [\mathbf{x}'_{\mathcal{P}} \quad \mathbf{x}_{\mathcal{B}}]^T$ , where  $\mathcal{P}'$  and  $\mathcal{B}$  are now sets of polynomials.

Using this notation we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{0} & \tilde{\mathbf{C}}_{\mathcal{P}1} \\ \mathbf{0} & \Sigma' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = \mathbf{0}, \quad (4.10)$$

where  $\Sigma'$  is diagonal with  $n - r$  nonzero diagonal entries. The zeros above  $\Sigma'$  enter since  $\Sigma'$  can be used to eliminate the corresponding elements without affecting any other elements in the matrix. In particular this means that we have

$$\begin{cases} \mathbf{X}_{\mathcal{P}'} & = & \mathbf{0} \\ \mathbf{X}_{\mathcal{R}} & = & -\mathbf{U}_{\mathcal{R}}^{-1} \tilde{\mathbf{C}}_{\mathcal{P}1} \mathbf{X}_{\mathcal{B}} \end{cases} \quad (4.11)$$

on  $V$ , which allows us to express any elements in  $\text{span}(\mathcal{M})$  in terms of  $\mathbf{X}_{\mathcal{B}}$ , which makes  $\mathcal{B}$  a solving basis.

Computing the action matrix is complicated slightly by the fact that we are now working with a polynomial basis rather than a monomial one. To deal with this situation we introduce some new notation. To each element  $e_k$  of  $\tilde{\mathcal{P}} = \mathcal{P}' \cup \mathcal{B}$  we assign a vector  $v_k = [0 \dots 1 \dots 0]^T \in \mathbb{R}^{|\tilde{\mathcal{P}}|}$ , with a one at position  $k$ . Similarly, we introduce vectors  $u_k \in \mathbb{R}^{|\mathcal{M}|}$ ,  $w_k \in \mathbb{R}^{|\mathcal{B}|}$  representing elements of  $\mathcal{M}$  and  $\mathcal{B}$  respectively. Further we define the linear mapping  $R : \text{span}(\mathcal{M}) \mapsto \text{span}(\mathcal{B})$ , which using (4.11) associates an element of  $\text{span}(\mathcal{M})$  with an element in  $\text{span}(\mathcal{B})$ . We represent  $R$  by a  $|\mathcal{B}| \times |\mathcal{M}|$  matrix

$$\mathbf{R} = [-\tilde{\mathbf{C}}_{\mathcal{P}'}^T, \mathbf{U}_{\mathcal{R}}^{-1T} \quad \mathbf{0} \quad \mathbf{I}], \quad (4.12)$$

acting on the space spanned by the vectors  $u_k$ .

We also introduce the mapping  $M_p : \text{span}(\mathcal{P}) \mapsto \text{span}(\mathcal{M})$  given by  $M_p(f) = p \cdot f$  with the representation

$$(\mathbf{M}_p)_{ij} = I(x^{\alpha_i} = p \cdot x^{\alpha_j}), \quad (4.13)$$

where  $I(\cdot)$  is the indicator function.

$\mathbf{M}_p$  represents multiplication by  $p$  on  $\mathcal{P}$ . In the basis  $\tilde{\mathcal{P}}$  induced by the change of basis  $\mathbf{V}$  we thus get

$$\tilde{\mathbf{M}}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}^T \end{bmatrix} \mathbf{M}_p \mathbf{V}. \quad (4.14)$$

Finally, we get a representation of the multiplication mapping from  $\mathcal{B}$  to  $\mathcal{B}$  as

$$\tilde{\mathbf{m}}_p = \mathbf{R} \tilde{\mathbf{M}}_p \mathbf{L}, \quad (4.15)$$

where  $\mathbf{L} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}$  simply interprets the  $w_k \in \mathbb{R}^{|\mathcal{B}|}$  vectors as  $\mathbb{R}^{|\tilde{\mathcal{P}}|}$ -vectors. The matrix  $\tilde{\mathbf{m}}_p$  derived here is the transpose of the corresponding matrix in Section 3.1.1.

An eigendecomposition of  $\tilde{\mathbf{m}}_p^T$  yields a set of eigenvectors  $\tilde{v}$  in the new basis. It remains to inverse transform these eigenvectors to obtain eigenvectors of  $\mathbf{m}_p^T$ . For this last step we need to construct the change of basis matrix  $\mathbf{V}_q$  in the quotient space. Using  $\mathbf{R}$  and  $\mathbf{L}$ , we get

$$\mathbf{V}_q^{-1} = \tilde{\mathbf{P}} \mathbf{V}^T \mathbf{L}. \quad (4.16)$$

Assume now that  $\tilde{v}$  is an eigenvector of  $\tilde{\mathbf{m}}_p^T$ . Using  $\mathbf{V}_q$  we have

$$\tilde{\mathbf{m}}_p^T = \mathbf{V}_q^T \mathbf{m}_p \mathbf{V}_q^{-T}. \quad (4.17)$$

We can see directly that  $\tilde{v} = \mathbf{V}_q^T v$  is an eigenvector for this matrix iff  $v$  is an eigenvector of  $\mathbf{m}_p$ . This yields

$$v = \mathbf{V}_q^{-T} \tilde{v} \quad (4.18)$$

and hence we have a way of going back to our original basis where we can read off the solutions to our equations.

As will be seen in the experiments, the SVD method is somewhat more stable than the QR method, but significantly slower due to the costly SVD factorization.

### 4.2.3 Basis Selection and Adaptive Truncation

We have so far seen three techniques for dealing with the case when the sub matrix  $\mathbf{C}_{\mathcal{P}_2}$  is ill conditioned. By the method in Section 4.1 we avoid operating on  $\mathbf{C}_{\mathcal{P}_2}$  altogether. Using, the QR and SVD methods we perform elimination, but in a numerically much more stable manner. One might now ask whether it is possible to combine these methods. Indeed it turns out that we can combine either the QR or the SVD method with a redundant solving basis to get an adaptive truncation criterion yielding even better stability in some cases. The way to do this is to choose a criterion for early stopping in the factorization algorithms. The techniques in this section are related to truncation schemes for rank-deficient linear least squares problems, cf [33].

A neat feature of QR factorization with column pivoting is that it provides a way of numerically estimating the conditioning of  $\mathbf{C}_{\mathcal{P}_2}$  simultaneously with elimination. By design, the QR factorization algorithm produces an upper triangular matrix  $\mathbf{U}$  with diagonal elements  $u_{ii}$  of decreasing absolute value. The factorization proceeds column wise, producing one  $|u_{ii}|$  at a time. If  $\text{rank}(\mathbf{U}) = r$ , then  $|u_{rr}| > 0$  and  $u_{r+1,r+1} = \dots = u_{nn} = 0$ . However, in floating point arithmetic, the transition from finite  $|u_{ii}|$  to zero is typically gradual passing through extremely small values and the rank is consequently hard to determine. For robustness it might therefore be a good idea to abort the factorization process early. We do this by setting a threshold  $\tau$  for the ratio  $|\frac{u_{11}}{u_{ii}}|$  and abort the factorization once the value exceeds this threshold. A value of  $\tau \approx 10^8$  has been found to yield good results<sup>1</sup>. Note that this produces an equivalent result to carrying out the full QR factorization and then simply discarding the last rows of  $\mathbf{U}$ . This is practical since off-the-shelf packages as LAPACK only provide full QR factorization, even though some computational effort could be spared by modifying the algorithm so as not to carry out the last steps.

Compared to setting a fixed (redundant) basis size, this approach is beneficial since both rank and conditioning of  $\mathbf{C}_{\mathcal{P}_2}$  might depend on the data. By the above method we decide adaptively where to truncate and *i.e* how large the linear basis for  $\mathbb{C}[\mathbf{x}]/I$  should be.

---

<sup>1</sup>Performance is not very sensitive to the choice of  $\tau$  and values in the range  $10^6$  to  $10^{10}$  yield similar results.

In the context of the SVD we get a similar criterion by looking at the singular values instead and set a threshold for  $\frac{\sigma_1}{\sigma_i}$ , which for  $i = \text{rank}(\mathbf{C}_{\mathcal{P}_2})$  is exactly the condition number of  $\mathbf{C}_{\mathcal{P}_2}$ .

## 4.3 Other Techniques

We end the part on techniques in this chapter with two less involved but still useful ideas.

### 4.3.1 A Single Elimination Step

In previous works which have been more closely connected to classical algebraic geometry using properly defined monomial orders *etc*, a Gröbner basis for the particular ideal has typically been obtained by successive elimination and addition of equations [48, 36]. This is also more similar to how the original Buchberger's algorithm for computing a Gröbner basis works. We strongly advocate avoiding this and instead first adding all equations and then doing the full elimination in one go. The reason for this is that, as mentioned often in this text, the eliminations tend to be ill conditioned. If several elimination steps are interleaved with addition of new equations, numerical errors accumulate and the algorithms easily become unstable.

### 4.3.2 Using Eigenvalues Instead of Eigenvectors

In the literature, the preferred method of extracting solutions using eigenvalue decomposition is to look at the eigenvectors. It is also possible to use the eigenvalues, but for a problem with  $s$  variables, this seemingly requires us to solve  $s$  eigenvalue problems since each eigenvalue only gives the value of one variable. However, there can be an advantage with using the eigenvalues instead of eigenvectors. If there are multiple eigenvalues (or almost multiple eigenvalues) the computation of the corresponding eigenvectors will be numerically unstable. However, the eigenvalues can usually be determined with reasonable accuracy. In practice, this situation is not uncommon with the action matrix.

Fortunately, we can make use of our knowledge of the eigenvectors to devise a scheme for quickly finding the eigenvalues of any action matrix on  $\mathbb{C}[\mathbf{x}]/I$ . From Section 2.2 we know that the right eigenvectors of an action matrix is the vector of basis elements of  $\mathbb{C}[\mathbf{x}]/I$  evaluated at the zeros of  $I$ . This holds for *any* action matrix and hence all action matrices have the same set of eigenvectors. Consider now a problem involving the two variables  $x_i$  and  $x_j$ . If we have constructed  $\mathbf{m}_{x_i}$ , the construction of  $\mathbf{m}_{x_j}$  requires almost no extra time. Now perform an eigenvalue decomposition  $\mathbf{m}_{x_i} = \mathbf{V}\mathbf{D}_{x_i}\mathbf{V}^{-1}$ . Since  $\mathbf{V}$  is the set of eigenvectors for  $\mathbf{m}_{x_j}$  as well, we get the eigenvalues of  $\mathbf{m}_{x_j}$  by straightforward matrix multiplication and then element wise division from

$$\mathbf{m}_{x_j}\mathbf{V} = \mathbf{V}\mathbf{D}_{x_j}. \quad (4.19)$$

This means that with very little extra computational effort over a single eigenvalue decomposition we can obtain the eigenvalues of all action matrices we need.

## 4.4 Experimental Validation

In this section we evaluate the numerical stability of the proposed techniques on a range of typical geometric computer vision problems. The experiments are mainly carried out on synthetic data since we are interested in the intrinsic numerical precision of the solver. By intrinsic precision we mean precision under perfect data. The error under noise is of course interesting for any application, but for minimal data this is an effect of the problem formulation and *not* of the particular equation solving technique.

In Section 4.4.1 all the main methods (standard, truncated, SVD and QR) are tested on the problem of optimal triangulation from three different views. This problem was first studied in [52] where emulated 128 bit arithmetics was necessary to get usable results. Later, with the techniques presented in this thesis the problem was given an efficient implementation in standard IEEE double precision and the details of this are given in Chapter 5 in the applications part of the thesis. However, this example provides such a nice illustration of the relative benefits and drawbacks of the different techniques so we take the liberty of borrowing some of the results and present them already in this section.

Apart from the triangulation example, the improved methods are tested on the problems of relative pose with unknown but common focal length [50] and relative pose for generalized cameras [51]. Significant improvements in stability are shown in all cases.

### 4.4.1 Optimal Three View Triangulation

The triangulation problem is formulated as finding the world point that minimizes the sum of squares of the reprojection errors in the three views. We do this by computing the gradient of the sum of squares error and setting it to zero. This yields three sixth degree polynomial equations in three variables (the  $X$ ,  $Y$  and  $Z$  coordinates of the unknown point) and using *e.g.* a computer algebra system one can check that the system has 47 (real and complex) zeros. After some preliminary manipulations described in Chapter 5 we expand the set of equations up to degrees 9 (see the beginning of Section 3.1.2) yielding 225 equations in 209 different monomials.

The synthetic data used in the validation was generated with three randomly placed cameras at a distance around 1000 from the origin and a focal length of around 1000. The unknown world point was randomly placed in a cube with side length 1000 centered at the origin. The methods have been compared on 100,000 test cases.

### Numerical Experiments

The first experiment investigates what improvement can be achieved by simply avoiding the problematic matrix elimination using the techniques of Section 4.1. For this purpose we choose the complete set of permissible monomials  $\mathcal{P}$  as a redundant basis and perform the steps of Algorithm 1. In this case we thus get a redundant basis of 154 elements and a  $154 \times 154$  multiplication matrix to perform eigenvalue decomposition on. In both cases the eigenvectors are used to find the solutions. The results of this experiment are shown in Figure 4.2.

As can be seen, this relatively straightforward technique already yields a large improvement in numerical stability.

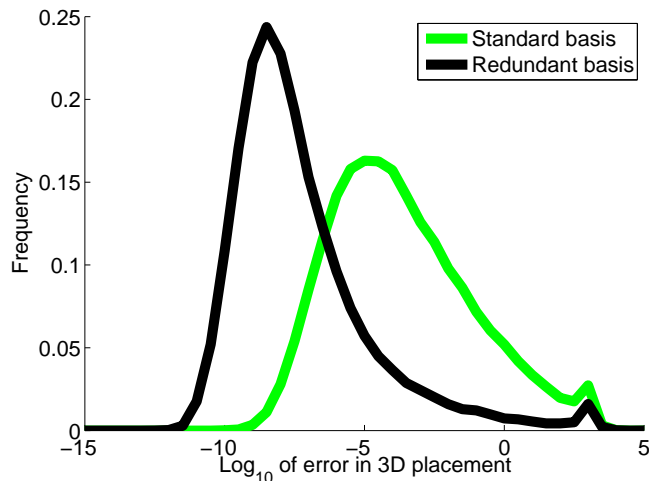


Figure 4.2: Histogram of errors over 100,000 points. The improvement in stability using the redundant basis renders the algorithm feasible in standard IEEE double precision.

Looking closely at Figure 4.2 one can see that even though the general stability is much improved, a small set of relatively large errors remain. It is unclear what causes these errors. However, by doing some extra work using the QR method of Section 4.2.1 to select a true basis as a subset of  $\mathcal{P}$ , we improve stability further in general and in particular completely resolve the issue with large errors, *cf* Figure 4.3. Moreover, we get a smaller eigenvalue decomposition and hence reduce computational complexity.

In Figure 4.4, the performance of the QR method is compared to the slightly more stable SVD method which selects a polynomial basis for  $\mathbb{C}[\mathbf{x}]/I$  from the monomials in  $\mathcal{P}$ . In this case, errors are typically a factor  $\sim 5$  smaller for the SVD method compared to the QR method.

The reason that a good choice of basis improves the numerical stability is that the condition number in the elimination step can be lowered considerably. Using the basis selection methods, the condition number is decreased by about a factor  $10^5$ . Figure 4.5 shows a scatter plot of error versus condition number for the three view triangulation problem. The SVD method displays a significant decrease and concentration in both error and condition number. It is interesting to note that to a reasonable approximation we have a linear trend between error and condition number. This can be seen since we have a linear trend with slope one in the logarithmic scale. Moreover, we have a  $y$ -axis intersection at about  $10^{-13}$ , since the coordinates are around 1000 in magnitude this means that we have a relative error  $\approx 10^{-16}\kappa = \epsilon_{mach}\kappa$ . This observation justifies our strategy of minimizing the condition number.

As mentioned in Section 4.3.2, it might be beneficial to use the eigenvalues instead of eigenvectors to extract solutions.

When solving this problem using eigenvalues there are two extra eigenvalue problems of size  $50 \times 50$  that need to be solved. The impact of the switch from

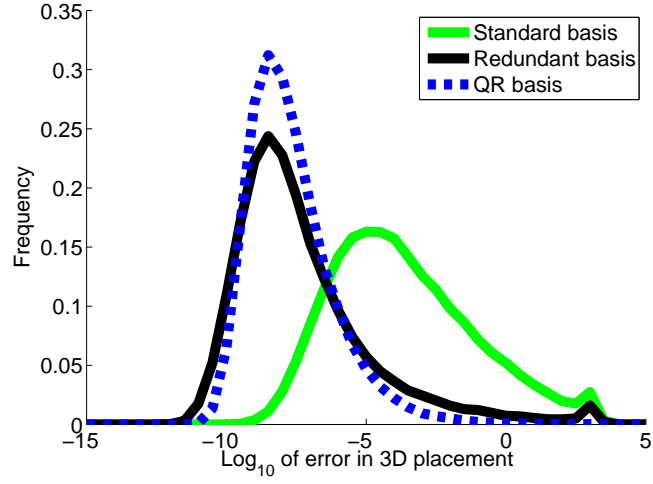


Figure 4.3: Histogram of errors for the standard, redundant basis and QR methods. The QR method improves stability in general and in particular completely removes the small set of large errors present in both the standard and redundant basis methods.

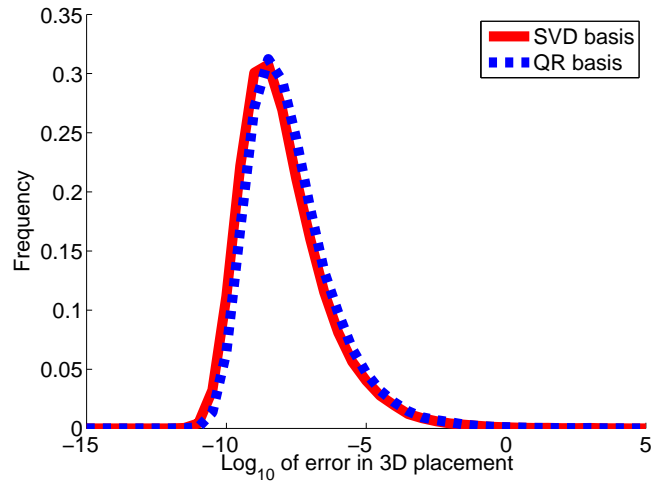


Figure 4.4: Comparison between the SVD and QR methods. The SVD method improves somewhat over the QR method at the cost of the computationally more demanding SVD factorization.

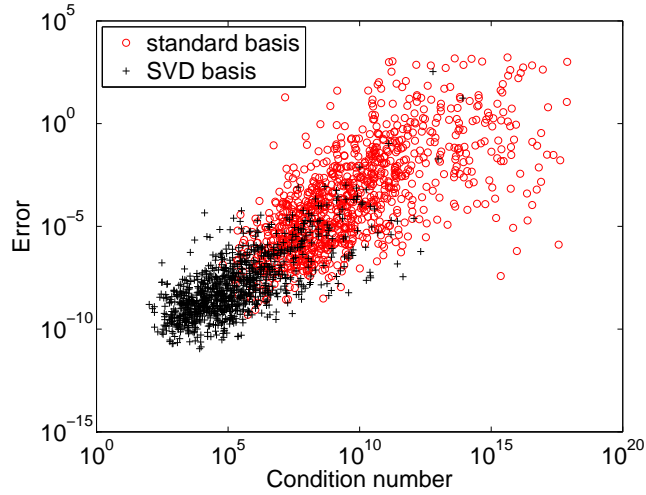


Figure 4.5: Error versus condition number for the part of the matrix which is inverted in the solution procedure.

eigenvectors to eigenvalues is shown in Figure 4.6. For this example we gain some stability at the cost of having to perform three eigenvalue decompositions (one for each coordinate) instead of only one. Moreover, we need to sort the eigenvalues using the eigenvectors to put together the correct triplets.

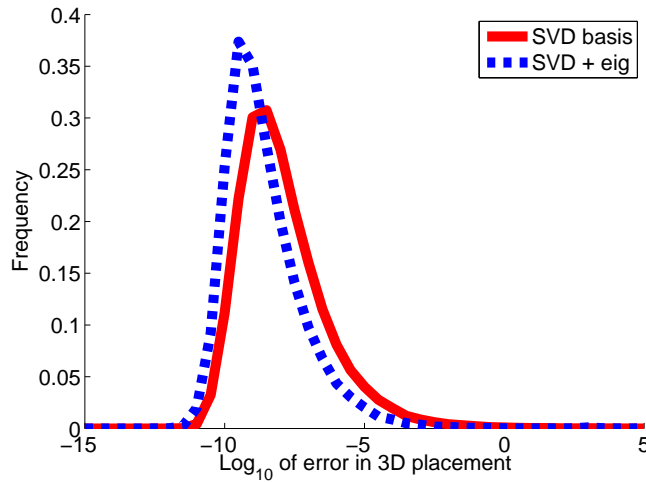


Figure 4.6: Error histograms showing the difference in precision between the eigenvalue and eigenvector methods.

However, we can use the trick of Section 4.3.2 to get nearly the same accuracy using only a single eigenvalue decomposition. Figure 4.7 shows the results of this method. The main advantage of using the eigenvalues is that we push down the number of large errors considerably.

Finally we study the combination of basis selection and early stopping which

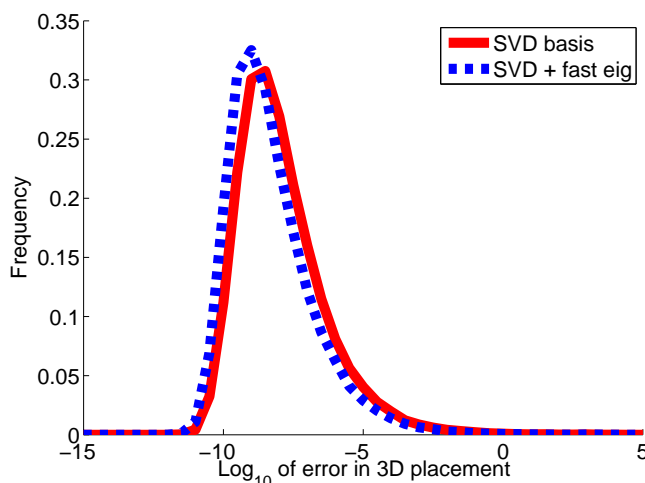


Figure 4.7: This graph shows the increase in performance when the fast eigenvalue method is used instead of the eigenvector method.

yields a redundant solving basis for the three view triangulation problem. The basis size was determined adaptively as described in Section 4.2.3 with a threshold  $\tau = 10^8$ . Table 4.1 shows the distribution of basis sizes obtained when this method was used. Since the basis is chosen minimal in 94% of the cases for the SVD-method and 95% for the QR method the time consumption is almost identical to the original basis selection methods, but as can be seen in Table 4.2 the number of large errors are reduced. This is probably due to the fact that truncation is carried out only when the matrices are close to being singular.

	50	51	52	53	54	$\geq 55$
SVD	94.0	3.5	0.8	0.4	0.3	1.0
QR	95.0	3.0	0.7	0.3	0.2	0.8

Table 4.1: Basis sizes for the QR and SVD methods with variable basis size. The table shows the percentage of times certain basis sizes occurred during 100,000 experiments.

To conclude the numerical experiments on three view triangulation two tables with detailed error statistics are given. The acronyms *STD*, *QR*, *SVD* and *TRUNC* respectively denote the standard method, QR method, SVD method and redundant basis method. The suffixes *eig*, *fast* and *var* respectively denote the eigenvalue method, the fast eigenvalue method (Section 4.3.2) and the use of a variable size basis (Section 4.2.3). Table 4.2 shows how many times the error gets larger the some given levels for several solvers. This is interesting for example when RANSAC is used. As can be seen, the QR-method with adaptive basis size is the best method for reducing the largest errors but the SVD-method with use of the eigenvalues is the best in general. Table 4.3 shows the median and the 95:th percentile errors for the same methods as in the previous table. Notable in here is that the 95:th percentile is improved with as much as factor  $10^7$  and the median with a factor  $10^5$ . The SVD-method with eigenvalues is

shown to be the best but the QR-method gives almost as good results.

Method	$> 10^{-3}$	$> 10^{-2}$	$> 10^{-1}$	$> 1$
STD	35633	24348	15806	9703
STD:eig	29847	19999	12690	7610
SVD	1173	562	247	119
SVD:eig	428	222	128	94
SVD:fast	834	393	178	94
SVD:var+fast	730	421	245	141
TRUNC	6712	4697	3339	2384
TRUNC:fast	5464	3892	2723	2015
QR	1287	599	269	127
QR:eig	517	250	149	117
QR:fast	1043	480	229	106
QR:var+fast	584	272	141	71

Table 4.2: Number of errors out of 100,000 experiments larger than certain levels. The QR-method with adaptive basis size yields the fewest number of large errors.

Method	95th	50th
STD	$1.42 \cdot 10^1$	$9.85 \cdot 10^{-5}$
STD:eig	$5.30 \cdot 10^0$	$3.32 \cdot 10^{-5}$
SVD	$1.19 \cdot 10^{-5}$	$6.09 \cdot 10^{-9}$
SVD:eig	$1.20 \cdot 10^{-6}$	$1.29 \cdot 10^{-9}$
SVD:fast	$4.37 \cdot 10^{-6}$	$2.53 \cdot 10^{-9}$
SVD:var+fast	$2.34 \cdot 10^{-6}$	$2.50 \cdot 10^{-9}$
TRUNC	$6.55 \cdot 10^{-3}$	$1.40 \cdot 10^{-8}$
TRUNC:fast	$1.87 \cdot 10^{-3}$	$3.27 \cdot 10^{-9}$
QR	$1.78 \cdot 10^{-5}$	$1.06 \cdot 10^{-8}$
QR:eig	$1.70 \cdot 10^{-6}$	$2.08 \cdot 10^{-9}$
QR:fast	$6.97 \cdot 10^{-6}$	$3.64 \cdot 10^{-9}$
QR:var+fast	$3.41 \cdot 10^{-6}$	$3.61 \cdot 10^{-9}$

Table 4.3: The 95th percentile and the median error for various methods. The improvement in precision is up to a factor  $10^7$ . The SVD method gives the best results, but the QR-method is not far off.

### Speed Comparison

The main motivation for using the QR-method rather than the SVD-method is that the QR-method is computationally less expensive. To verify this the standard, SVD and QR-methods were run and the time was measured. Since the implementations were done in Matlab it was necessary to take care to eliminate the effect of Matlab being an interpreted language. To do this only the time after construction of the coefficient matrix was taken into account. This is because the construction of the coefficient matrix essentially amounts to copying

coefficients to the right places which can be done extremely fast in *e.g.* a C language implementation.

In the routines that were measured no subroutines were called that were not built-in functions in Matlab. The measurements were done with the Matlab profiler.

The time measurements were done on an Intel Core 2 2.13 GHz machine with 2 GB memory. Each algorithm was executed with 1000 different coefficient matrices constructed from the same type of scene setups as previously. The same set of coefficient matrices was used for each method. The result is given in Table 4.4. Our results show that the QR-method is approximately three times faster than the SVD-method but 50% slower than the standard method. The reason that the redundant basis method is more than twice as slow as the QR method is the larger eigenvalue decomposition which dominates the computation time.

Method	Time per call / ms	Relative time
SVD	66.89	1
TRUNC	55.84	0.83
QR	24.45	0.37
STD	16.44	0.25

Table 4.4: Time consumption in the solver part for the three different methods. The time is an average over 1000 function calls.

#### 4.4.2 Relative Pose with Unknown Focal Length

Relative pose for calibrated cameras is a well known problem and the standard minimal case for this is five points in two views. There are in general ten solutions to this problem. For the same problem but with unknown focal length, the corresponding minimal case is six points in two views, which was solved by Stewénius *et al* using Gröbner basis techniques [50].

Following the same recipe as Stewénius *et al* it is possible to express the fundamental matrix as a linear combination,

$$F = F_0 + F_1 l_1 + F_2 l_2. \quad (4.20)$$

Then putting  $f^{-2} = p$  one obtains nine equations from the constraint on the essential matrix [46]

$$2EE^T E - \text{tr}(EE^T)E = 0. \quad (4.21)$$

A 10th equation is then obtained by making use of the fact that the fundamental matrix is singular, *i.e.*  $\det(F) = 0$ . These equations involve the unknowns  $p$ ,  $l_1$  and  $l_2$  and are of total degree 5. The problem has 15 solutions in general.

We set up the coefficient matrix  $\mathbf{C}$  by multiplying these ten equations by  $p$  so that the degree of  $p$  reaches a maximum of four. This gives 34 equations in a total of 50 monomials.

The validation data was generated with two cameras of equal focal length of around 1000 placed at a distance of around 1000 from the origin. The six points were randomly placed in a cube with side length 1000 centered at the origin. The standard, SVD, and QR-methods have been compared on 100,000

test cases and the errors in focal length are shown in Figure 4.8. In this case the QR-method yields slightly better results than the SVD-method. This is probably due to loss in numerical precision when the solution is transformed back to the original basis.

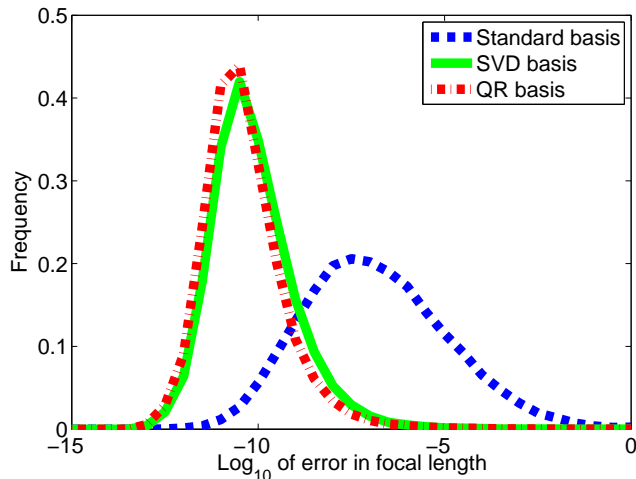


Figure 4.8: The error in focal length for relative pose with two semi calibrated cameras with unknown but common focal length.

#### 4.4.3 Relative Pose for Generalized Cameras

Generalized cameras provide a generalization of the standard pin-hole camera in the sense that there is no common focal point through which all image rays pass, *cf* [47]. Instead the camera captures arbitrary image rays or lines. Solving for the relative motion of a generalized camera can be done using six point correspondences in two views. This is a minimal case which was solved in [51] with Gröbner basis techniques. The problem equations can be set up using quaternions to parameterize the rotation, Plücker representation of the lines and a generalized epipolar constraint which captures the relation between the lines. After some manipulations one obtains a set of sixth degree equations in the three quaternion parameters  $v_1, v_2$  and  $v_3$ . For details, see [51]. The problem has 64 solutions in general.

To build our solver including the change of basis we multiply an original set of 15 equations with all combinations of  $1, v_1, v_2, v_3$  up to degree two. After this we end up with 101 equations of total degree 8 in 165 different monomials.

We generate synthetic test cases by drawing six points from a normal distribution centered at the origin. Since the purpose of this investigation is not to study generalized cameras under realistic conditions we have not used any particular camera rig. Instead we use a completely general setting where the cameras observe six randomly chosen lines each through the six points. There is also a random relative rotation and translation relating the two cameras. It is the task of the solver to calculate the rotation and translation.

The methods have been compared on a data set of 10,000 randomly generated test cases. The results from this experiment are shown in Figure 4.9. As can

be seen, a good choice of basis yields drastically improved numerical precision over the standard method.

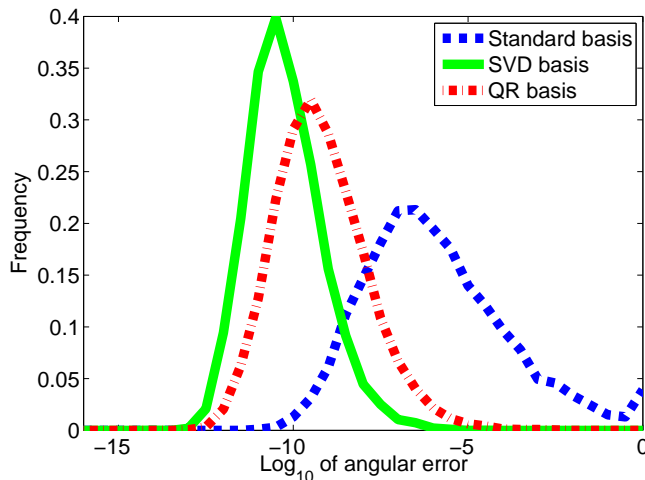


Figure 4.9: The angular error for relative pose with generalized cameras.

## 4.5 Discussion

We have introduced some new theoretical ideas as well as a set of techniques designed to overcome numerical problems encountered in state-of-the-art methods for polynomial equation solving. We have shown empirically that these techniques in many cases yield dramatic improvements in numerical stability and further permits the solution of a larger class of problems than previously possible.

The techniques for solving polynomial equations that are used in this work can be summarized as follows. The original equations are first expanded by multiplying the polynomials with a set of monomials. The resulting equations is expressed as a product of a coefficient matrix  $\mathbf{C}$  and a monomial vector  $\mathbf{X}$ . Here we have some freedom in choosing which monomials to multiply with. We then try to find a solving basis  $\mathcal{B}$  for the problem. For a given candidate basis  $\mathcal{B}$  we have shown how to determine if  $\mathcal{B}$  constitutes a solving basis. If so then we can use numerical linear algebra to construct the action matrix and get a fast and numerically stable solution to the problem at hand. However, we do not know (i) what monomials we should multiply the original equations with and (ii) what solving basis  $\mathcal{B}$  should be used to get the simplest and most numerically stable solutions. Are there algorithmic methods for answering these questions? For a given expansion  $\mathbf{CX}$  can one determine if this allows for a solving basis? A concise theoretical understanding and practical algorithms for these problems would certainly be of great aid in the work on polynomial problems and is a highly interesting subject for future research.

Part II

Applications



## Chapter 5

# Optimal Triangulation

In this chapter we consider the problem of globally optimal triangulation from three separate views. Whereas, the two-view case has a relatively simple closed form solution, the three-view case has just the right complexity to make it an excellent target for the techniques introduced in Chapter 4. For four or more views though, optimization by solving a polynomial is still more or less infeasible.

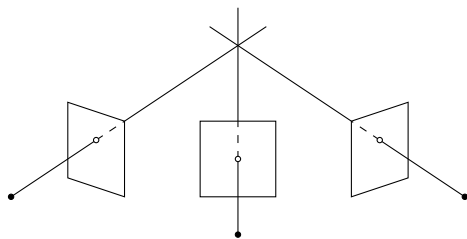


Figure 5.1: The unknown location of a point can be reconstructed using its projection in a sequence of images if the location and orientation of the cameras are known. This is usually called triangulation.

### 5.1 Introduction

Triangulation, referring to the act of reconstructing the 3D location of a point given its images in two or more known views, is an important part of numerous computer vision systems. Albeit conceptually simple, this problem is not completely solved in the general case of  $n$  views and noisy measurements.

There exist fast and relatively robust methods based on linear least squares [26]. These methods are however sub-optimal. Moreover the linear least squares formulation does not have a clear geometrical meaning, which means that in unfortunate situations, this approach can yield very poor accuracy.

The most desirable, but non-linear, approach is instead to minimize the  $L_2$  norm of the reprojection error, *i.e.* the sum of squares of the reprojection errors. The reason for this is that the  $L_2$  optimum yields the maximum likelihood estimate for the 3D point under the assumption of independent Gaussian noise on the image measurements [25]. This problem has been given a closed form

solution<sup>1</sup> by Hartley and Sturm in the case of two views [25]. However, the approach of Hartley and Sturm is not straightforward to generalize to more than two views.

In the case of  $n$  views, the standard method when high accuracy is needed is to use a two-phase strategy where an iterative scheme for non-linear least squares such as Levenberg-Marquardt (Bundle Adjustment) is initialized with a linear method [56]. This procedure is reasonably fast and in general yields excellent results. One potential drawback, however, is that the method is inherently local, *i.e.* finds local minima with no guarantee of being close to the global optimum.

An interesting alternative is to replace the  $L_2$  norm with the  $L_\infty$  norm *cf.* [31]. This way it is possible to obtain a provably optimal solution with a geometrically sound cost function in a relatively efficient way. The drawback is that the  $L_\infty$  norm is suboptimal under Gaussian noise and it is less robust to noise and outliers than the  $L_2$  norm.

The most practical existing method for  $L_2$  optimization with an optimality guarantee is to use a branch and bound approach as introduced in [1], which, however, is a computationally expensive strategy.<sup>2</sup>

In this work, we propose to solve the problem of  $L_2$  optimal triangulation from three views using a method introduced by Stewénius *et al* in [52], where the optimum was found by explicit computation of the complete set of stationary points of the likelihood function. This approach is similar to that of Hartley and Sturm [25]. However, whereas the stationary points in the two view case can be found by solving a sixth degree polynomial in one variable, the easiest known formulation of the three view case involves solving a system of three sixth degree equations in three unknowns with 47 solutions. Thus, we have to resort to more sophisticated techniques to tackle this problem.

Stewénius *et al* used algebraic geometry and Gröbner basis techniques to analyze and solve the equation system. However, as previously mentioned, Gröbner basis calculations are known to be numerically challenging and they were forced to use emulated 128 bit precision arithmetics to get a stable implementation, which rendered their solution too slow to be of any practical value.

Using the new techniques presented in this thesis, we are now able to give the Gröbner basis method a fast implementation using standard IEEE double precision. By this we also show that global optimization by calculation of stationary points is indeed a feasible approach and that Gröbner basis like techniques provide a powerful tool in this pursuit.

## 5.2 Three View Triangulation

The main motivation for triangulation from more than two views is to use the additional information to improve accuracy. In this section we briefly outline the approach we take and derive the equations to be used in the following sections.

---

<sup>1</sup>The solution is actually not *entirely* on closed form, since it involves the solution of a sixth degree polynomial, which cannot in general be solved on closed form. Therefore one has to go by *e.g.* the eigenvalues of the companion matrix, which implies an iterative process.

<sup>2</sup>Since the main part of the material of this chapter was written, a faster version of the branch and bound algorithm for  $L_2$  optimal triangulation has been published [42] that probably has comparable running time to the method presented here, even though exact running times are not available for the case of three views. However, the new branch and bound method also generalizes to  $n$  views and is therefore probably a more practical choice.

This part is essentially identical to that used in [52]. We assume a linear pin-hole camera model, *i.e.* projection in homogeneous coordinates is done according to  $\lambda_i x_i = P_i X$ , where  $P_i$  is the  $3 \times 4$  camera matrix for view  $i$ ,  $x_i$  is the image coordinates,  $\lambda_i$  is the depth and  $X$  is the 3D coordinates of the world point to be determined. In standard coordinates, this can be written as

$$x_i = \frac{1}{P_{i3}X} \begin{bmatrix} P_{i1}X \\ P_{i2}X \end{bmatrix}, \quad (5.1)$$

where *e.g.*  $P_{i3}$  refers to row 3 of camera  $i$ .

As mentioned previously, we aim at minimizing the  $L_2$  norm of the reprojection errors. Since we are free to choose coordinate system in the images, we place the three image points at the origin in their respective image coordinate systems. With this choice of coordinates, we obtain the following cost function to minimize over  $X$

$$\varphi(X) = \frac{(P_{11}X)^2 + (P_{12}X)^2}{(P_{13}X)^2} + \frac{(P_{21}X)^2 + (P_{22}X)^2}{(P_{23}X)^2} + \frac{(P_{31}X)^2 + (P_{32}X)^2}{(P_{33}X)^2}. \quad (5.2)$$

The approach we take is based on calculating the complete set of stationary points of  $\varphi(X)$ , *i.e.* solving  $\nabla\varphi(X) = 0$ . By inspection of (5.2) we see that  $\nabla\varphi(X)$  will be a sum of rational functions. The explicit derivatives can easily be calculated, but we refrain from writing them out here. Differentiating and multiplying through with the denominators produces three sixth degree polynomial equations in the three unknowns of  $X = [X_1 \ X_2 \ X_3]^T$ . To simplify the equations we also make a change of world coordinates, setting the last rows of the respective cameras to

$$P_{13} = [1 \ 0 \ 0 \ 0], \quad P_{23} = [0 \ 1 \ 0 \ 0], \quad P_{33} = [0 \ 0 \ 1 \ 0]. \quad (5.3)$$

Since we multiply with the denominator we introduce new stationary points in our equations corresponding to one of the denominators in (5.2) being equal to zero. This happens precisely when  $X$  coincides with the plane through one of the focal points parallel to the corresponding image plane. Such points have infinite/undefined value of  $\varphi(X)$  and can therefore safely be removed.

To summarize, we now have three sixth degree equations in three unknowns. The remainder of the theoretical part of the chapter will be devoted to the problem of solving these.

### 5.3 A Numerical Solution to the Three View Triangulation Problem

As discussed in Section 5.2, we optimize the  $L_2$  cost function by calculation of the stationary points. This yields three sixth degree polynomial equations in  $X = [X_1 \ X_2 \ X_3]^T$ . In addition to this, we add a fourth equation by taking the sum of our three original equations. This cancels out the leading terms, producing a fifth degree equation which will be useful in the subsequent calculations [52]. These equations generate an ideal  $I$  in  $\mathbb{C}[X]$ . We start this section out by going through the previous method of trying to compute a Gröbner basis for  $I$  and explain where this method runs into problems. This serves as a basis for employing the methods of Chapter 4 to get a fast and stable algorithm.

First, however, we need to deal with the problem where one or more of  $X_i = 0$ . When this happens, we get a parametric solution to our equations. As mentioned in Section 5.2, this corresponds to the extra stationary points introduced by multiplying up denominators and these points have infinite value of the cost function  $\varphi(X)$ . Hence, we would like to exclude solutions with any  $X_i = 0$  or equivalently  $X_1X_2X_3 = 0$ . The algebraic geometry way of doing this is to calculate the *saturation*  $\text{sat}(I, X_1X_2X_3)$  of  $I$  w.r.t  $X_1X_2X_3$ , consisting of all polynomials  $f(X)$  s.t.  $(X_1X_2X_3)^k \cdot f \in I$  for some  $k$ .

Computationally it is easier to calculate  $\text{sat}(I, X_i)$  for one variable at a time and then joining the result. This removes the same problematic parameter family of solutions, but with the side effect of producing some extra (finite) solutions with  $X_i = 0$ . These do not present any serious difficulties since they can easily be detected and filtered out.

Consider one of the variables, say  $X_1$ . The ideal  $\text{sat}(I, X_1)$  is calculated in three steps. We order the monomials according to  $X_1$  but take the monomial with the highest power of  $X_1$  to be the smallest, e.g.  $X_1X_2^2X_3 \geq X_1^2X_2^2X_3$ . With the monomials ordered this way, we perform a few steps of the Gröbner basis calculation, yielding a set of generators where the last elements can be divided by powers of  $X_1$ . We add these new equations which are “stripped” from powers of  $X_1$  to  $I$ .

More concretely, we multiply the equations by all monomials creating equations up to degree seven. After the elimination step two equations are divisible by  $X_1$  and one is divisible by  $X_1^2$ .

The saturation process is performed analogously for  $X_2$  and  $X_3$  producing the saturated ideal  $I_{\text{sat}}$ , from which we extract our solutions.

The final step is to calculate a Gröbner basis for  $I_{\text{sat}}$ , at this point generated by a set of nine fifth and sixth degree equations. To be able to do this we multiply with monomials creating 225 equations in 209 different monomials of total degree up to nine. The last step thus consists of putting the 225 by 209 matrix  $\mathbf{C}$  on reduced row echelon form.

This last part turns out to be a delicate task though due to generally very poor conditioning. In fact, the conditioning is often so poor that roundoff errors in the order of magnitude of machine epsilon (approximately  $10^{-16}$  for doubles) yield errors as large as  $10^2$  or more in the final result. This is the reason one had to resort to emulated 128 bit numerics in [52].

Using the new techniques for computing the action matrix though, we can now more or less completely avoid these conditioning problems. By extensive experimentation (see Section 4.4) we have found that using the QR method (Section 4.2.1) with an adaptive basis size (Section 4.2.3) yields the best stability/speed tradeoff, see Table 5.1.

## 5.4 Experimentas

The algorithm described in this chapter has been implemented in Matlab which suggests that further gains in speed could be made by implementing it in e.g. C. However, the main time consuming parts of the algorithm are the LU and QR factorizations and the eigenvalue decomposition of the action matrix and Matlab uses LAPACK and BLAS for these operations which contain state-of-the-art implementations of the above mentioned linear algebra operations. Care

	QR	Standard	Standard, 128 bit
Running time:	14ms	10ms	30s
Stability:	Good	Very poor	Good

Table 5.1: Overview of running time and stability characteristics for the new QR-based algorithm, for the previous method in double precision and for the previous method implemented in emulated 128 bit arithmetics. The previous method is only stable in the higher precision, which makes it very slow (a factor 300 slower). Using the QR method we get a fast and stable algorithm in standard double precision.

has been taken to make the Matlab code for the remaining operations as efficient as possible.

Experimental results for the triangulation problem have already been presented in Chapter 4, but we repeat some of them here for completeness of the chapter with the purpose of demonstrating the speed and numerical precision of the method. We have run the algorithm on both real and synthetically generated data using a 2.0 Ghz AMD Athlon X2 64 bit machine. With this setup, triangulation of one point takes approximately 13 milliseconds using the new method. This is to be contrasted with the previous implementation by Stewénus *et al* [52], which needs 30 seconds per triangulation with their setup. The branch and bound method of [1] is faster than [52] but exact running times for triangulation are not given in [1]. However, based on the performance of this algorithm on similar problems, the running time for three view triangulation is probably at least a couple of seconds using their method.

#### 5.4.1 Synthetic Data

To evaluate the intrinsic numerical stability of the solver the algorithm has been run on 100,000 randomly generated test cases. World points were drawn uniformly from the cube  $[-500, 500]^3$  and cameras were placed randomly at a distance of around 1000 from the origin with focal length of around 1000 and pointing inwards. We compare the approach presented here to that of [52] implemented in double precision here referred to as the standard method since it is based on straightforward Gröbner basis calculation. A histogram over the resulting errors in estimated 3D location is shown in Figure 5.2. As can be seen, the error is typically around a factor  $10^5$  smaller with the new method.

Since we consider triangulation by minimization of the  $L_2$  norm of the error, ideally behavior under noise should not be affected by the algorithm used. In the second experiment we assert that the algorithm behaves as expected under noise. We generate data as in the first experiment and apply Gaussian noise to the image measurements in 0.1 pixel intervals from 0 to 5 pixels. We triangulate 1000 points for each noise level. The median error in 3D location is plotted versus noise in Figure 5.3. There is a linear relation between noise and error, which confirms that the algorithm is stable also in the presence of noise.

#### 5.4.2 A Real Example

Finally, we evaluate the algorithm under real world conditions. The Oxford dinosaur [16] is a familiar image sequence of a toy dinosaur shot on a turn table.

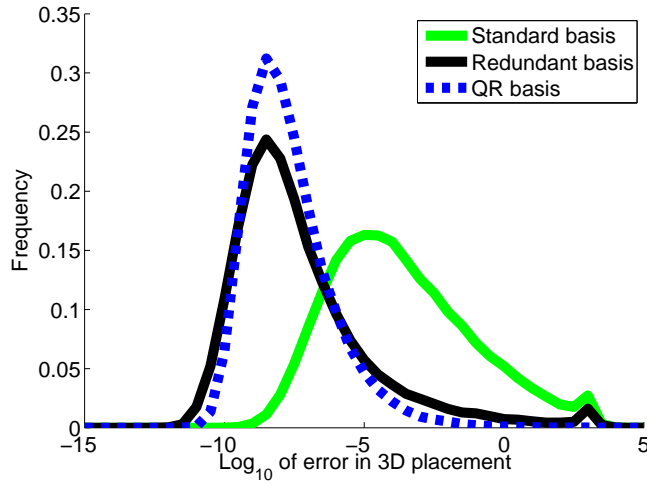


Figure 5.2: Histogram of errors for the standard, redundant basis and QR methods. The QR method improves stability in general and in particular completely removes the small set of large errors present in both the standard and redundant basis methods. Compared to the standard method, precision is improved by about a factor  $10^5$ .

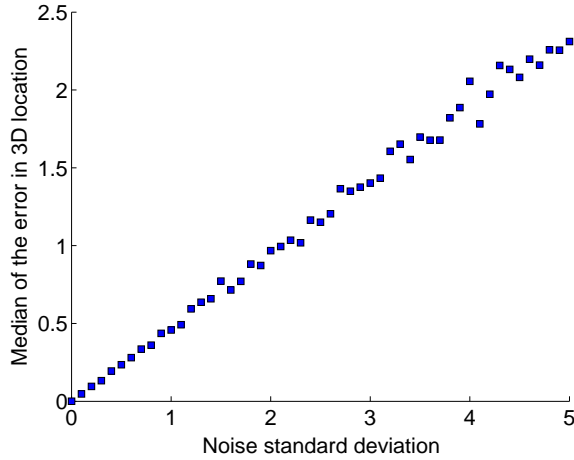


Figure 5.3: Error in 3D location of the triangulated point  $X$  as a function of image-point noise. The behavior under noise is as expected given the problem formulation.

The image sequence consists of 36 images and 4983 point tracks. For each point visible in three or more views we select the first, middle and last view and triangulate using these. This yields a total of 2683 point triplets to triangulate from. The image sequence contains some erroneous tracks which we deal with by removing any points reprojected with an error greater than two pixels in any frame. The whole sequence was processed in approximately 34 seconds and the resulting point cloud is shown in Figure 5.4.

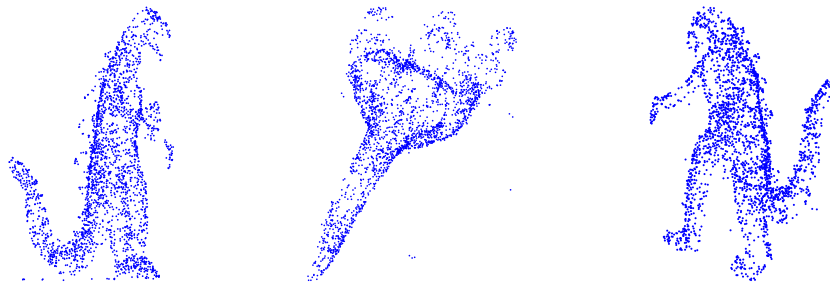


Figure 5.4: The Oxford dinosaur reconstructed from 2683 point triplets using the QR-method with variable basis size. The reconstruction was completed in approximately 34 seconds.

We have also run the same sequence using the previous method implemented in double precision, but the errors were too large to yield usable results. Note that [52] contains a successful triangulation of the dinosaur sequence, but this was done using extremely slow emulated 128 bit arithmetic yielding an estimated running time of 20h for the whole sequence.

## 5.5 Conclusions

In this chapter we have shown how a typical problem from computer vision, triangulation, can be solved for the globally optimal  $L_2$  estimate using Gröbner basis like techniques. With the new techniques for equation solving, we have taken this approach to a state where it can now have practical value in actual applications. In all fairness though, linear initialization combined with bundle adjustment will probably remain the choice for most applications since this is still significantly faster and gives excellent accuracy. However, if a guarantee of finding the provably optimal solution is desired, we provide a competitive method.

More importantly perhaps, by this example we show that global optimization by calculation of the stationary points using Gröbner basis techniques is indeed a possible way forward. This is particularly interesting since a large number of computer vision problems ultimately depend on some form of optimization.

Currently the limiting factor in many applications of Gröbner bases is numerical difficulties. Using the technique presented in this thesis, we are able to improve the numerical precision by approximately a factor  $10^5$ . We thus show that there is room for improvement on this point and there is certainly more to explore here.



## Chapter 6

# Epipolar Geometry Under Radial Distortion

In this chapter we study the problem of estimating relative camera motion between two frames in the presence of potentially heavy radial distortion. Efficient and reliable solutions to the relative motion problem serve as the core of many computer vision systems. Traditionally, one has assumed a linear camera model and at best compensated for radial distortion towards the end of the process. In this chapter it is indicated how radial distortion can be taken into account already from the outset. In particular, two minimal cases of structure from motion with radial distortion are derived and solved.

### 6.1 Introduction

Estimating camera motion and inner calibration parameters from sequences of images is a challenging computer vision problem with a broad range of applications [26]. Typically one starts with a noisy set of tentative image point correspondences. The first step then is to make decisions about inliers and outliers and get a good initial estimate to be able to deploy a more sophisticated optimization algorithm on the set of all inliers.

Two robust and widely used techniques for this purpose are RANSAC [20] and kernel voting [40], both relying on solving a large number of instances of the underlying problem, each with a small number of point correspondences. There is thus a need to develop fast and stable algorithms for solving geometric computer vision problems with a minimal number of points. Typically this amounts to solving a system of polynomial equations in several variables.

Traditionally, minimal problems have been formulated assuming a linear pin-hole camera model with different restrictions on the inner calibration parameters *etc.* However, for some cameras such fish-eye lenses this can be insufficient and one might need to handle strong radial distortions already from the outset.

Solving for the fundamental matrix under radial distortion was first studied in [3], where a *non-minimal* algorithm based on 15 point correspondences was given for a pair of uncalibrated cameras. More recently, in [36, 37], a number of different *minimal* problems with radial distortion have been studied and practical solutions have been given in some cases.

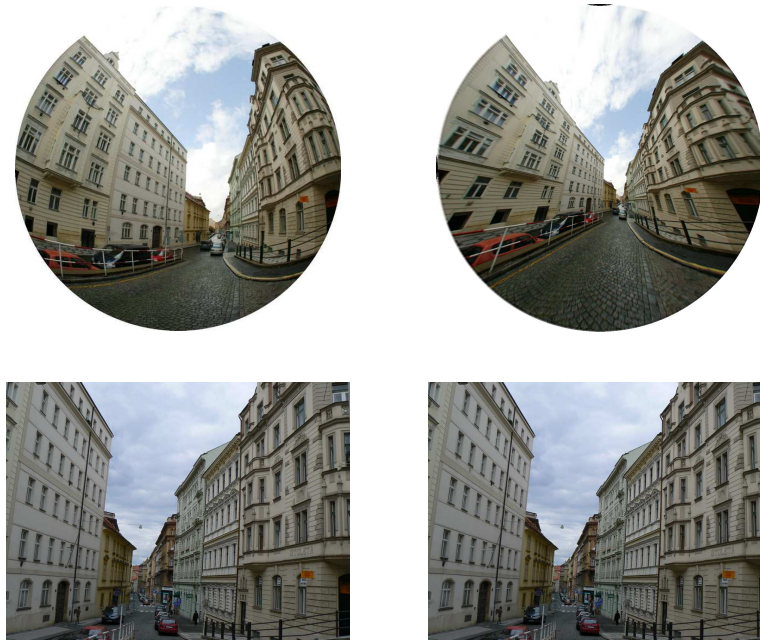


Figure 6.1: (Left) Input images with different radial distortions (Top) 66% cutout from omnidirectional image and (Bottom) image taken with standard perspective camera. (Right) Corrected images.

Leveraging on the new techniques presented in this thesis, fast and numerically stable algorithms for two minimal problems with radial distortion previously unsolved in floating point arithmetic are formulated and solved:

1. The problem of estimating a one-parameter radial distortion model and epipolar geometry from image point correspondences in two uncalibrated views with different radial distortions in each image.
2. The problem of estimating a one-parameter radial distortion model and epipolar geometry from image point correspondences in two partially calibrated views.

These two problems were previously studied in [37] and found to be numerically very challenging. In [37] the authors provide solutions to these problems computed in exact rational arithmetic only. This results in very long computational times and is not usable in practical applications. Here we show that these two problems can be efficiently solved in floating point arithmetic.

The speed and intrinsic numerical stability as well as robustness to noise of the proposed algorithms is demonstrated using both synthetic data and real images.

## 6.2 Uncalibrated Case

In this case, we study the situation with two uncalibrated cameras and two different unknown radial distortion parameters. We use the same formulation of the problem as in [37]. This formulation assumes a one-parameter division model [21] given by the formula

$$\mathbf{p}_u \sim \mathbf{p}_d / (1 + \lambda r_d^2) \quad (6.1)$$

where  $\mathbf{p}_u = (x_u, y_u, 1)^T$  and  $\mathbf{p}_d = (x_d, y_d, 1)^T$  are the corresponding undistorted, resp. distorted, image points, and  $r_d$  is the radius of  $\mathbf{p}_d$  w.r.t. the distortion center.

The minimal set of constraints needed to uniquely solve for relative motion for uncalibrated cameras with different radial distortion  $\lambda_1$  and  $\lambda_2$  is 9 point correspondences with epipolar constraints

$$\mathbf{p}_{u_i}^\top (\lambda_1) F \mathbf{p}'_{u_i} (\lambda_2) = 0, \quad i = 1, \dots, 9 \quad (6.2)$$

and the singularity of the fundamental matrix  $F$

$$\det(F) = 0. \quad (6.3)$$

Assuming  $f_{3,3} \neq 0$  we can set  $f_{3,3} = 1$  and obtain 10 equations in 10 unknowns.

By linear elimination, these 10 equations can be reduced to 4 equations in 4 unknowns (one of  $2^{nd}$  degree, two of  $3^{rd}$  degree and one of  $5^{th}$  degree). For more details see [37] where it was shown that this problem has 24 solutions.

The numerical solver is constructed starting with the four remaining equations in the four unknowns  $f_{3,1}$ ,  $f_{3,2}$ ,  $\lambda_1$  and  $\lambda_2$ . The first step is to expand the number of equations, as outlined in Section 3.1.2, by multiplying them by a handcrafted set of monomials in the four unknowns, in this case yielding 393 equations in 390 monomials. See Section 6.2.1 for details.

We now stack the coefficients of the equations in a matrix  $\mathbf{C}$ . Following this, we order the monomials and correspondingly the columns of  $\mathbf{C}$  as in (3.9). The sets  $\mathcal{E}$  and  $\mathcal{R}$  depend on which variable is used to create the action matrix. For this problem  $f_{3,1}$  was used as action variable. The classical method is thereafter to choose the linear basis  $\mathcal{B}$  of  $\mathbb{C}[\mathbf{x}]/I$  to be the 24 lowest monomials (w.r.t some monomial order). This is enough to get a solution to the problem, but we can use the methods of Chapter 4 to select a basis of linear combinations of monomials from a larger set and thereby improve numerical stability. Empirically, we have found that the linear basis can be selected from the set of all monomials up to degree four excluding the monomial  $\lambda_1^4$ . The set  $\mathcal{R}$  then consists of monomials of degree five that are reached when the monomials of degree four are multiplied with  $f_{3,1}$ . The set  $\mathcal{E}$  contains the remaining 285 monomials.

Putting the part of  $\mathbf{C}$  corresponding to  $\mathcal{E}$  and  $\mathcal{R}$  on triangular form by means of an LU decomposition now produces Equation 3.10. We can then remove all equations that include excessive monomials and still have enough information to construct the action matrix.

Finally, we use the QR method to select a linear basis for  $\mathbb{C}[\mathbf{x}]/I$  and construct the matrix  $\mathbf{m}_{f_{3,1}}$  from which the solutions are extracted.

### 6.2.1 Details on the Expansion Step for the Uncalibrated Case

We have found in experiments that to construct the necessary elements of the Gröbner basis like structure needed to construct  $\mathbf{m}_{f_{3,1}}$ , we need to generate polynomials up to total degree eight. Thus, the  $2^{nd}$  degree polynomial has to be multiplied with all monomials up to degree six and corresponding numbers for the  $3^{rd}$  and  $5^{th}$  degree polynomials.

Further investigations have shown that not exactly all monomials up to degree eight are needed, so in the implementation, the  $2^{nd}$  degree polynomial was only multiplied with monomials up to degree five and each variable not higher than four, further on was  $\lambda_1$  not multiplied with higher degree than two. For the other polynomials it was possible to limit the degree of each individual variable to one lower than the total degree.

These multiplications yield 393 equations in 390 monomials. Without the last fine tuning of the degrees, the number of equations and monomials will be larger but all extra monomials will be in the set  $\mathcal{E}$  and will make no real differences to the solver except slightly longer computation times.

## 6.3 Calibrated Case

We now turn to the setup with two calibrated cameras and one common unknown radial distortion parameter. To solve the corresponding minimal problem, we make use of the epipolar constraint for 6 point correspondences

$$\mathbf{p}_{u_i}^\top(\lambda) E \mathbf{p}'_{u_i}(\lambda) = 0, \quad i = 1, \dots, 6, \quad (6.4)$$

the singularity of the essential matrix  $E$

$$\det(E) = 0 \quad (6.5)$$

and the trace condition, which says that two singular values of the essential matrix are equal

$$2(EE^T)E - \text{trace}(EE^T)E = 0. \quad (6.6)$$

Again assuming  $e_{3,3} \neq 0$ , we can set  $e_{3,3} = 1$  and obtain 16 equations in 9 unknowns. Using similar method as in the uncalibrated case, these equations can be rewritten as 11 polynomial equations in 4 unknowns (one of  $3^{rd}$  degree, four of  $5^{th}$  degree and six of  $6^{th}$  degree). In [37] it was shown that this problem has 52 solutions.

The numerical solution of this problem largely follows that of the uncalibrated version. In the first expansion, all equations are multiplied with monomials to reach degree eight. This gives 356 equations in 378 monomials. As in the uncalibrated case it is possible to reduce the number of monomials by fine tuning the degrees we need to go to, in this case yielding 320 equations in 363 monomials.

The next step is to reorder the monomials and columns as in equation (3.9). Once again, the linear basis of  $\mathbb{C}[\mathbf{x}]/I$  can be constructed from the monomials of degree four and lower.  $\mathcal{R}$  will then consist of those monomials of degree five that are reached when the degree four monomials are multiplied with the variable  $e_{3,1}$ , which is used as action variable.

As before  $\mathbf{C}$  is transformed to triangular form by LU decomposition and after that we only consider those equations that do not include any of the monomials in  $\mathcal{E}$ . Now  $\mathbf{C}$  holds all necessary information to choose representatives in  $\mathbb{C}[\mathbf{x}]/I$  and create the action matrix with respect to multiplication by  $e_{3,1}$ .

## 6.4 Experiments

We have tested the algorithms for the uncalibrated and calibrated minimal problems on synthetic images with various levels of noise, outliers and radial distortions as well as on real images. The time consumptions for both algorithms have also been measured.

The algorithms proposed here are significantly more stable than the algorithms presented in [37] which ran in exact rational arithmetic only. Since doing the computations in exact arithmetic is extremely slow (minutes instead of milliseconds), a comparison with the floating point algorithm presented here is not meaningful and has therefore been omitted.

Both problems are solved by finding the roots of a system of polynomial equations which means that we obtain several potentially correct answers, 52 in the calibrated case and 24 in the uncalibrated case. In general we obtain more than one real root, in which case we need to select the best one, *i.e.* the root which is consistent with most measurements. To do so, we treat the real roots obtained by solving the equations for one input as real roots from different inputs and use kernel voting [40] for several inputs to select the best root among all generated roots. The kernel voting is done by a Gaussian kernel with fixed variance and the estimate of  $\lambda_1$  and  $\lambda_2$  in the uncalibrated case and  $\lambda$  in the calibrated case is found as the position of the largest peak [40, 36].

### 6.4.1 Tests on Synthetic Images

For both problems treated here, the same synthetic experiments were carried out to evaluate the quality of the solvers.

In all simulated experiments we generate synthetic data using the following procedure:

1. Generate a 3D scene consisting of 1000 points distributed randomly within a cube. Project  $M\%$  of the points on image planes of the two displaced cameras. These are matches. In both image planes, generate  $(100 - M)\%$  random points distributed uniformly in the image. These are mismatches.
2. Apply different radial distortions to the undistorted correspondences in each image and in this way generate noiseless distorted points.
3. Add Gaussian noise of standard deviation  $\sigma$  to the distorted points.

#### Uncalibrated case

In the first two experiments we study the robustness of the algorithm for the uncalibrated case to Gaussian noise added to the distorted points.

The first experiment investigates the estimation error of  $\lambda$  as a function of noise. The ground truth radial distortions parameters were  $\lambda_1 = -0.2$ ,  $\lambda_2 = -0.3$  in the first case and  $\lambda_1 = -0.01$ ,  $\lambda_2 = -0.7$  in the second case. See

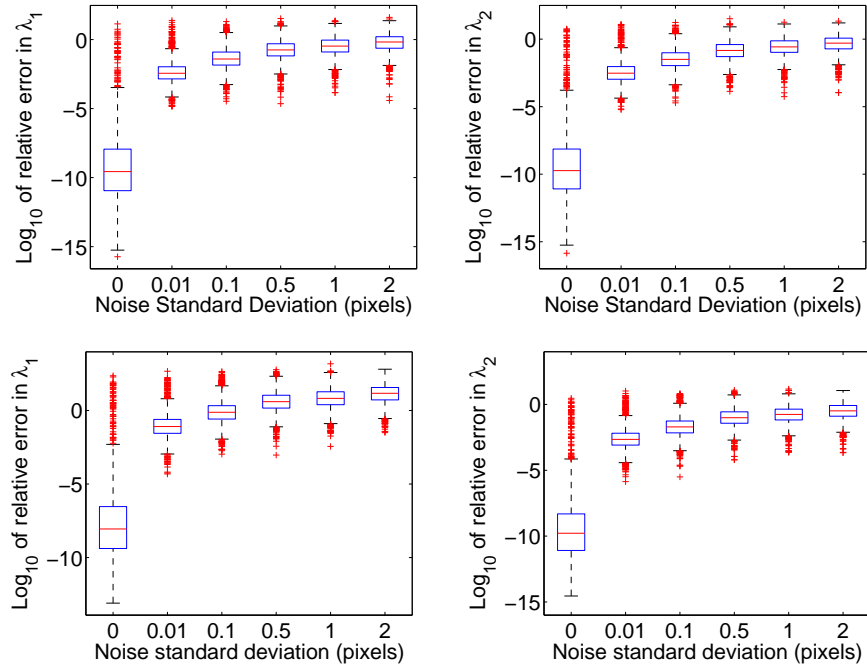


Figure 6.2: Uncalibrated case: Relative errors of (Left)  $\overline{\lambda_1}$  and (Right)  $\overline{\lambda_2}$  as a function of noise. Ground truth (Top)  $\lambda_1 = -0.2$ ,  $\lambda_2 = -0.3$  and (Bottom)  $\lambda_1 = -0.01$ ,  $\lambda_2 = -0.7$ . Blue boxes contain values from 25% to 75% quantile.

Figure 6.2. The noise varied from 0 to 2 pixels. For each noise level relative errors for 2000  $\lambda$ s (estimated as closest values to the ground truth value from all solutions) were computed. The results in Figure 6.2 for the estimated  $\overline{\lambda_1}$  (Left) and  $\overline{\lambda_2}$  (Right) are presented by the Matlab function *boxplot* which shows values 25% to 75% quantile as a blue box with red horizontal line at median. The red crosses show data beyond 1.5 times the interquartile range.

For noiseless data we obtain very accurate estimates of radial distortion parameters even for very different  $\lambda$ s. For larger noises the  $\log_{10}$  relative errors are much higher (mostly around  $10^{-1}$ ). However obtained  $\lambda$ s are still satisfactory and mostly differ from the ground truth value in the second decimal place. The main point is however not to use a minimal point set to get a good estimate, but to repeatedly draw minimal configurations from a larger set of potential matches and then use *e.g.* kernel voting to get a more reliable estimate. Finally, the result can be further enhanced using the obtained estimate as a good starting guess in a large scale bundle adjustment. The effect of kernel voting is studied in the second experiment.

In this experiment we did not select the root closest to the ground truth value for each run of the algorithm, instead we used kernel voting to select the best  $\lambda$ s among all generated roots from several runs. The ground truth radial distortion parameters were as in the previous experiment ( $\lambda_1 = -0.2$ ,  $\lambda_2 = -0.3$  in the first case and  $\lambda_1 = -0.01$ ,  $\lambda_2 = -0.7$  in the second case) and the level of noise varied from 0 to 2 pixels. Moreover, in the first case there were 10% of

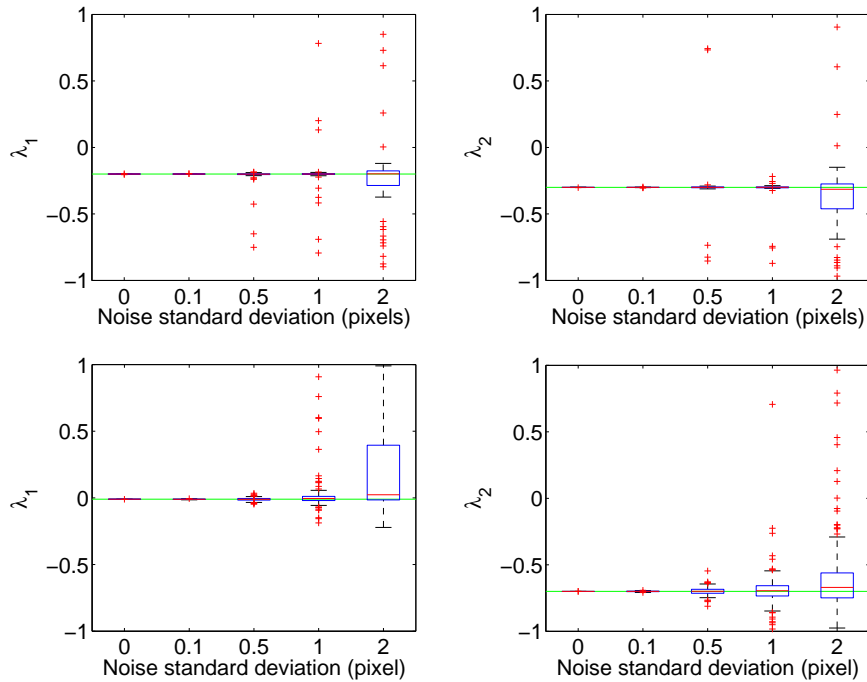


Figure 6.3: Uncalibrated case, kernel voting: Estimated (Left)  $\overline{\lambda_1}$  and (Right)  $\overline{\lambda_2}$  as a function of noise, (Top) ground truth  $\lambda_1 = -0.2$ ,  $\lambda_2 = -0.3$  (green lines), 90% of inliers and 100 samples in kernel voting and (Bottom) ground truth  $\lambda_1 = -0.01$ ,  $\lambda_2 = -0.7$ , 100% of inliers and 50 samples in kernel voting.

outliers in the image ( $M=90$ ).

The testing procedure was as follows:

1. Repeat  $K$  times (We use  $K$  from 50 to 100 though for more noisy data  $K$  from 100 to 200 gives better results).
  - (a) Randomly choose 9 point correspondences from a set of  $N$  potential correspondences (6 point correspondences for the calibrated case).
  - (b) Normalize image point coordinates to  $[-1, 1]$ .
  - (c) Find 24 roots using the presented algorithm.
  - (d) Select the real roots in the feasible interval, *i.e.*  $-1 < \lambda_1, \lambda_2 < 1$  and the corresponding  $F$ 's.
2. Use kernel voting to select the best root.

Figure 6.3 shows  $\lambda$ s computed using the algorithm for the uncalibrated case as a function of noise. In the first case with outliers Figure 6.3 (Top) 100  $\lambda$ s were estimated using kernel voting for roots computed from 100 ( $K = 100$ ) 9-tuples of correspondences randomly drawn for each noise level. In the second case Figure 6.3 (Bottom) 200  $\lambda$ s were estimated using kernel voting for roots computed from 50 ( $K = 50$ ) 9-tuples of correspondences. This means that for each noise level the algorithm ran 10,000 times in both cases. The results are again presented by the Matlab function *boxplot*.

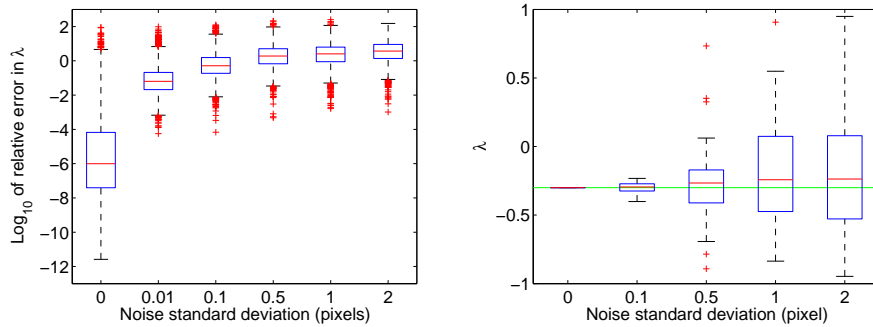


Figure 6.4: Calibrated case: (Left) relative errors of  $\bar{\lambda}$  as a function of noise, ground truth  $\lambda = -0.3$ . (Right) kernel voting: Estimated  $\bar{\lambda}$  using kernel voting for roots computed from 200 6-tuples of correspondences randomly drawn for each noise level. Ground truth  $\lambda = -0.3$  (green line).

### Calibrated case

The same synthetic experiments were carried out for the calibrated solver.

The results of the first experiment which shows relative errors of the estimated  $\bar{\lambda}$  as a function of noise are shown in Figure 6.4. The ground truth radial distortion was  $\lambda = -0.3$ . For noiseless data we again obtain very precise estimates of radial distortion parameter  $\lambda$ . For larger noise levels the  $\log_{10}$  relative errors are slightly larger than for the uncalibrated case. However, using kernel voting we can still obtain good estimates. This is shown by the second experiment.

In this experiment  $\lambda$  was estimated 50 times using kernel voting for roots computed from 200 6-tuples of correspondences randomly drawn for each noise level, Figure 6.4. The median values for  $\bar{\lambda}$  are again very close to the ground truth value  $\lambda = -0.3$  for all noise levels from 0 to 2 pixels. However the variances of this for the calibrated case are larger, especially for higher noise levels, than the variances for the uncalibrated case. This means that for good estimates of  $\lambda$  this algorithm requires more samples in the kernel voting procedure than in the uncalibrated case.

### 6.4.2 Time Consumption

To evaluate the speed of the new algorithm a reasonably optimized version of the algorithm for the uncalibrated case was implemented. The implementation was done in Matlab so rewriting the algorithm in a compiled language such as C should reduce the execution time further.

The algorithm was run 10,000 times and the time consumption was measured using the Matlab profiler. The experiments were performed on an Intel Core 2 CPU 2.13 GHz machine with 2 GB of memory. The estimated average execution time for solving one instance of the uncalibrated problem was 16 milliseconds and the corresponding time for the calibrated problem was 17 milliseconds.

These results are to be compared with the execution times given for the same problem in [37], where solutions were computed in exact rational arithmetic.



Figure 6.5: Real data, 60% cutouts from omnidirectional images. (Left) Input images with different radial distortions for camera 1 (Top) and camera 2 (Bottom). (Right) Corrected images.

There, the processing time for one problem instance was 30 s for the uncalibrated case and 1700 s for the calibrated case.

### 6.4.3 Tests on Real Images

The algorithm for uncalibrated cameras with different radial distortions has been tested on several different sets of images. In the first experiment the input images with different relatively large distortions in each image, Figure 6.5 (Left), were obtained as 60% cutouts from fish-eye images taken with two different cameras with different radial distortions. Tentative point matches were then found by the wide base-line matching algorithm [43]. They contained correct as well as incorrect matches. Distortion parameters  $\lambda_1$  and  $\lambda_2$  were estimated using the algorithm for uncalibrated cameras with different radial distortions and the kernel voting method for 100 samples. The input (Left) and corrected (Right) images are presented in Figure 6.5. Figure 6.6 shows the distribution of real roots for images from Figure 6.5, from which  $\lambda_1 = -0.301$  and  $\lambda_2 = -0.368$  were estimated as the argument of the maximum. The peaks from kernel voting are sharp and the  $\lambda$ 's are estimated accurately.

In the second experiment the algorithm was tested on images with significantly different distortions. The left image Figure 6.1 (Left), was obtained as a

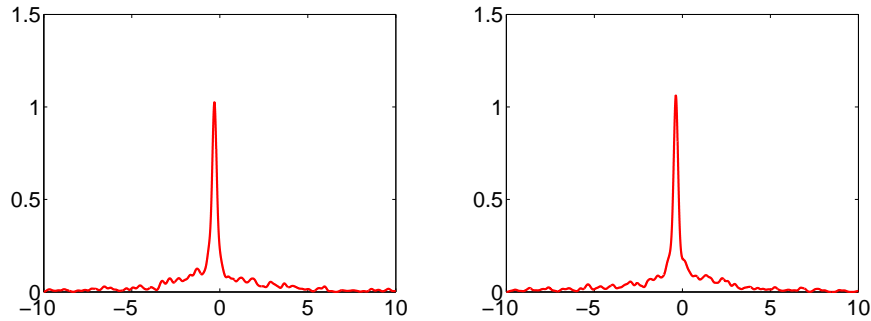


Figure 6.6: Distribution of real roots obtained by kernel voting for images in Figure 6.5. Estimated  $\lambda_1 = -0.301$  and  $\lambda_2 = -0.368$ .

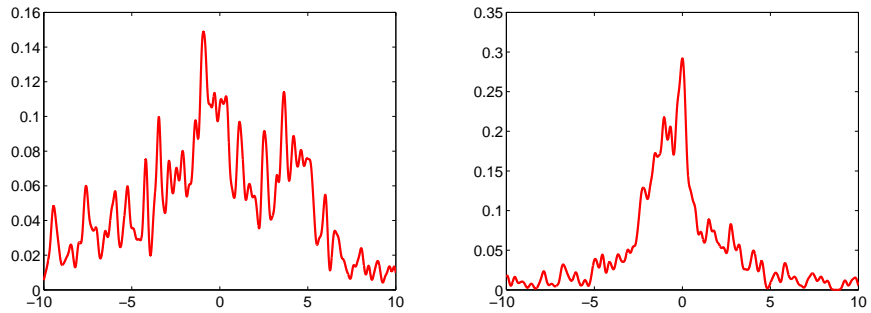


Figure 6.7: Distribution of real roots obtained by kernel voting for images in Figure 6.1. Estimated  $\lambda_1 = -0.926$  and  $\lambda_2 = 0.0025$ .

66% cutout from a fish-eye image and the right image was taken with a standard perspective camera. Since these images had a rather large difference in radial distortion, the tentative point correspondences contained a larger number of mismatches. Distortion parameters  $\lambda_1$  and  $\lambda_2$  were again estimated using the algorithm for uncalibrated cameras with different radial distortions and the kernel voting method. The input (Left) and corrected (Right) images are presented in Figure 6.1. Figure 6.7 shows the distribution of real roots for these images from which  $\lambda_1 = -0.926$  and  $\lambda_2 = 0.0025$  were estimated. As can be seen the peaks obtained by kernel voting are not so sharp but still sufficient to get good estimates of the  $\lambda$ 's even from only 100 samples.

## 6.5 Conclusions

In this chapter we have given fast and robust algorithms for two minimal problems previously unsolved in floating point arithmetic. The two problems of simultaneously solving for relative pose and radial distortion were, due to numerical problems, previously solved in exact rational arithmetic only, yielding them too time consuming to be of practical value. With the floating point algorithm presented here we have reduced the computation time from minutes to

milliseconds. Moreover, we have verified that this is done without loss of numerical precision by extensive experiments both on synthetic and real images.

In the experiments we have also demonstrated that the radial distortion estimation is reasonably robust both to outliers and noise when kernel voting is used over several runs. Finally we have shown that large differences in distortion between the two images can be handled.



## Chapter 7

# Hybrid Minimal Problems

Camera estimation tasks are usually divided into (i) absolute orientation estimation where a set of world points with known coordinates correspond to a set of image points and the task is to determine the exact pose and position of the camera and (ii) relative orientation estimation where two or more cameras view the same scene and a set of corresponding points between images are given. In this chapter we investigate the mathematics of cases that fall in between those two extremes, *i.e.* we consider a camera which captures some points with known 3D coordinates and some points which are not known but give partial information since they are seen by other known cameras. The application we have in mind is global image-based localization.

### 7.1 Introduction

Localization refers to the ability of automatically inferring the pose and the position of an observer relative a model [5]. Solving this problem using an image-based approach amounts to first establishing tentative correspondences between an input image and the model, filtering out outliers and computing the camera location and orientation. The need to understand and solve minimal setups thus arises in a manner very similar to that in Chapter 6. The model or the map of the environment can be anything from a single room in a building to a complete city. In general, one image will be used as a query image, but in principle several images can be used as input. No prior knowledge of the observer's position is assumed and therefore the problem is often referred to as global localization whereas local versions assume an approximate position. The mapping of the environment can be regarded as an off-line process since it is generally done once and for all. Such a mapping can be done using standard Structure from Motion (SfM) algorithms [26], or by some other means.

We demonstrate how a mixture of 2D and 3D features can be used simultaneously for localization. If one were to rely solely on 3D matches, one is restricting the set of possible correspondences to relatively few correspondences and a relatively rich 3D model would be required in order to be successful. On the other hand, using only 2D features requires relatively many correct correspondences to generate a single hypothesis. In addition, with existing methods such as the seven point algorithm of two views [26], one is limited to picking

all the 2D correspondences from one single image in the model. Again, one is restricting the set of correspondences to a relatively small subset. Further, the absolute scale cannot be recovered solely from 2D correspondences of one query image and one model image.

Using hybrid correspondence sets for generating hypotheses gives a number of advantages. We can make use of all possible correspondences simultaneously, even from different 2D model images. Compared to approaches using only 2D correspondences, the scale relative to the 3D map can be recovered and, more importantly, the number of correspondences is smaller which is a good property when using RANSAC. One can argue that in most cases, traditional methods would work fine. However, if one accepts possibly somewhat longer computation times, using hybrid correspondence sets (as well as traditional ones) provides a strictly greater chance of obtaining an outlier-free point set and there is hence no reason why the extra information should not be used.

The main contributions of this chapter are:

1. A complete list of minimal hybrid cases is given and for most cases we also give the number of possible solutions.
2. Algorithms for efficiently computing the solutions of two of the minimal cases are given. One of these cases was only solvable using the techniques of Chapter 4.

## 7.2 Problem Formulation

With the localization application in mind, we are interested in solving the following problem:

*Problem 17.* Under the assumption that for a query image, there are  $m$  potential correspondences to image points in views with known absolute orientation and  $n$  potential correspondences to scene points with known 3D coordinates, find the largest subset of the correspondences that admits a solution to the absolute orientation problem within a specified accuracy.

The method that we use to solve the localization problem is based on hypothesize-and-test with RANSAC [20] and local invariant features [41]. This involves solving minimal structure and motion problem with hybrid correspondence sets.

## 7.3 Minimal Hybrid Correspondence Sets

The classical absolute orientation problem (also known as camera resectioning) for calibrated cameras for three known points can be posed as finding the matrix  $P = [R \ t]$ , such that  $\lambda_i u_i = P U_i$ ,  $i = 1, 2, 3$ . Here  $R$  is a  $3 \times 3$  rotation matrix and  $t$  is a 3-element translation vector. Thus, the camera matrix encodes six degrees. Each point gives two constraints and therefore three points form a minimal case. In general there are four possible solutions [26].

We will study the absolute orientation problem for both calibrated cameras as above, for the case of unknown focal length and for the uncalibrated camera case. Furthermore we will consider both known 3D-2D correspondences  $(U_i, u_i)$

as above and 2D-2D correspondences  $(v_i, u_i)$  with features  $v_i$  in other views. Here we will assume that the camera matrices of the other views are known, so that a 2D-2D correspondence can be thought of as a 3D-2D correspondence where the unknown 3D point  $U_i$  lies on a line expressed in Plücker coordinates. Here, **the (m,n) case** denotes the case of  $m$  2D-2D correspondences and  $n$  3D-2D correspondences. Notice that each 2D-2D correspondence imposes one constraint and each 2D-3D correspondence imposes two constraints.

**Calibrated Cameras** For calibrated cameras there are six degrees of freedom, three for orientation and three for position. One way of parameterizing the camera matrix is to use a quaternion vector  $[a \ b \ c \ d]^T$  for rotation, *i.e*

$$P = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2ac + 2bd & x \\ 2ad + 2bc & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab & y \\ 2bd - 2ac & 2ab + 2cd & a^2 - b^2 - c^2 + d^2 & z \end{bmatrix}. \quad (7.1)$$

Potential minimal cases are:

*The (0,3) case.* This is the well known resectioning problem, *cf* [26] with up to four solutions in front of the camera.

*The (2,2) case.* This case is given a numerical solution here. The algorithm works equally well if the 2D-2D correspondences are to the same or to different cameras. There are up to 16 solutions.

*The (4,1) case.* The case of all four 2D-2D correspondences coming from the same model image can be solved by first projecting the 3D point in the known camera and then using the five point algorithm to solve for relative orientation (hence up to 10 solutions) and then fixing scale with the final 2D-3D correspondence. The more general problem of the 2D-2D correspondences being to different cameras is treated in this chapter.

*The (6,0) case.* This cannot be solved for absolute orientation if all points are from the same model view. However, if the correspondences come from different views, it is equivalent to the relative orientation problem for generalized cameras, *cf* [51], which has up to 64 solutions.

**Unknown Focal Length** For calibrated cameras with unknown focal length there are seven degrees of freedom, three for orientation, three for position and one for the focal length. One way of parameterizing the camera matrix is as

$$P = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2ac + 2bd & x \\ 2ad + 2bc & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab & y \\ 2f(bd - ac) & 2f(ab + cd) & f(a^2 - b^2 - c^2 + d^2) & fz \end{bmatrix}. \quad (7.2)$$

Potential minimal cases are

*The (1,3) case.* This case is solved numerically in this chapter. There are 36 solutions.

*The (3,2) case.* This case is treated here. There are 40 solutions.

*The (5,1) case.* For the case of the 5 2D-2D correspondences coming from the same model view, it can be solved using the six point algorithm to solve for relative orientation and focal length [50] and then fixing scale with the final 3D correspondence. There are then up to 15 solutions. The general case of 2D-2D correspondences to different views is treated in this chapter.

*The (7,0) case.* This cannot be solved for absolute orientation if all points are to the same view. However for the case of correspondence to different view it is an open problem.

**Uncalibrated Cameras** For the uncalibrated camera case there are 11 degrees of freedom. Each 2D-2D correspondence gives one constraint and each 2D-3D correspondence gives two constraints. Potential minimal cases are

*The (1,5) case.* This can be solved by hand-calculations as follows. Using the five 3D-2D correspondences, the camera matrix can be determined up to a one-parameter family  $P = P_1 + \nu P_2$ , where  $P_1$  and  $P_2$  are given  $3 \times 4$  matrices and  $\nu$  is an unknown scalar. The remaining 2D correspondence can be parameterized as a point on a line  $U = C + \mu D$  for some unknown parameter  $\mu$ . The projection equation gives  $\lambda u = PU = (P_1 + \nu P_2)(U_1 + \mu U_2)$ . Using resultants, it follows easily that there are two solutions for the unknowns  $\lambda, \nu, \mu$ .

*The (3,4) case.* There are eight solutions, unless all four 2D-2D correspondences are from the same model view, in which the standard seven-point-two-view algorithm can be used. There are then up to three solutions.

*The (1+2k,5-k) case with  $k = 2, 3, 4$ .* These cannot be solved for absolute orientation if all points originate from one model view. However, for the case of correspondences from different model views, there are  $2^{(1+2k)}$  solutions. The solutions procedure is analogous to the (1,5) case above and can be obtained using resultants.

**Summary** We conclude this section by summarizing all the minimal cases for hybrid 2D and 3D feature correspondences, see Table 7.1. We state an upper bound on the number of physically realizable solutions. In practice though, as we shall see later in Section 7.4.2, the number of plausible solutions is much smaller. In the next section, we give the remaining justifications to these claims and this will also lead to efficient algorithms for computing the solutions.

## 7.4 Solving Hybrid Minimal Cases with Algebraic Geometry

As we have seen in previous chapters, minimal structure and motion problems typically boil down to solving a system of polynomial equations in a number of unknowns and this is the case for all problems studied in this chapter as well. We now give the details concerning how the various hybrid problems are formulated and solved. We in turn consider calibrated, semi-calibrated and uncalibrated cameras.

### 7.4.1 Calibrated Cameras

A calibrated camera can be parameterized using quaternions as shown in (7.1). We now study the (2,2) case in more detail, *i.e.* assume that we have two correspondences between image points and scene points

$$u_1 \sim PU_1, \quad u_2 \sim PU_2.$$

2D-2D corresp.	2D-3D corresp.	number of solutions	camera setting
0	3	4	calibrated
2	2	16	calibrated
4	1	32 or 10*	calibrated
6	0	64	calibrated
1	3	36	unknown focal
3	2	40	unknown focal
5	1	112 or 15*	unknown focal
7	0	?	unknown focal
1	5	2	uncalibrated
3	4	8 or 3*	uncalibrated
$1 + 2k$	$5 - k$	$2^{1+2k}$	uncalibrated

Table 7.1: Minimal hybrid cases for structure from motion. The number of solutions indicates an upper bound of the number of physically realizable solutions. The solution numbers marked with asterisk "\*" correspond to cases where all 2D-2D correspondences originate from a single (model) view, whereas for other cases, it is implicitly assumed that the correspondence set covers multiple views. Note that one case is still an open problem (marked with "?").

Since there is a freedom in choosing coordinate systems both in the scene and in the images, these can be transformed into

$$U_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, u_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, U_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, u_2 = \begin{bmatrix} 1 \\ 0 \\ u \end{bmatrix}.$$

This gives us the following constraints

$$\begin{aligned} x = 0, \quad y = 0, \quad ad = -bc, \\ z = u(a^2 + b^2 - c^2 - d^2) - 2bd + 2ac. \end{aligned}$$

As the overall scale of the camera matrix is irrelevant, one can set  $a = 1$  and eliminate  $d$  according to  $d = -bc$ . This makes it possible to parameterize the camera matrix as

$$P = \begin{bmatrix} (1+b^2)(1-c^2) & 4bc & 2c(1-b^2) & 0 \\ 0 & (1-b^2)(1+c^2) & -2b(1+c^2) & 0 \\ -2c(1+b^2) & 2b(1-c^2) & (1-b^2)(1-c^2) & z \end{bmatrix}.$$

By setting  $a = 1$  two things happen. First the scale of the camera matrix is fixed, hence the left-hand  $3 \times 3$  sub matrix in (7.1) will only be a rotation matrix up to scale. This will not have any further impact on the problem since the measurement equations are homogeneous. The second consequence is that solutions with  $a = 0$  will not be included. Since  $a \in \mathbb{R}$  the probability for this is zero, but there might be problems if  $a$  is close to zero. However, as the synthetic experiments will show this causes no serious problems.

Assume now that we have two correspondences between image points and points that have been seen in only one other model image. This gives two points

on the viewing line  $C_i$  and  $D_i$  associated to a point  $v_i$  in the query image. If the line is represented with Plücker coordinates [26] and the camera is converted to the corresponding Plücker camera the constraints above converts to a single equation. It is furthermore easy to see that every nonzero element in the Plücker camera has a common factor  $1 + b^2$ . After removing the common factor, the constraint polynomials  $(p_1, p_2)$  are of order 2 in  $b$  and order 4 in  $c$ .

The dimension of the quotient space  $\mathbb{C}[b, c]/I$  is 16 with  $I = \{p_1, p_2\}$  which can be checked with computer algebra. By multiplying the polynomials with  $\{1, b, c, bc\}$  we obtain 8 equations in 24 monomials. It is then possible to express 8 of the monomials in terms of the remaining 16 monomials

$$\{bc^4, b^3c^2, c^4, bc^3, b^2c^2, b^3c, c^3, bc^2, b^2c, b^3, c^2, bc, b^2, c, b, 1\}$$

which then form a basis for the quotient space  $\mathbb{C}[b, c]/I$ . From this it is straightforward to construct the  $16 \times 16$  action matrix  $\mathbf{m}_c$  for the linear mapping  $\mathbb{C}[b, c]/I \ni p(b, c) \mapsto cp(b, c) \in \mathbb{C}[b, c]/I$ . From the eigenvalue decomposition of the matrix  $\mathbf{m}_c$  the 16 solutions are obtained. Since this problem is of relatively low degree with only two variables, the eliminations are well conditioned as they are and there is no pressing need to apply any stabilizing techniques.

Gröbner basis calculations with a computer algebra system show that there are 32 solutions for the (4,1) case, but we have not implemented a numerical solver for this case.

## 7.4.2 Experimental Results for the (2,2) Case

The purpose of this section is to evaluate the stability of the algorithm for solving the (2,2) case. To this end we use synthetically generated data in the form of randomly generated cameras and points. This allows us to measure the typical errors and the typical number of plausible solutions, over a large range of cases.

The point features are drawn uniformly from the cube  $\pm 500$  units from the origin in each direction. The cameras (two known and one unknown) are generated at approximately 1000 units from the origin pointing roughly in the direction of the center of the point cloud.

The algorithm has been run on 10,000 randomly generated cases as described above. To evaluate the accuracy of the solution we take the minimal error (over the plausible solutions) of the standard matrix 2-norm  $\|P' - P\|$  of the difference between the estimated camera  $P'$  and the true camera  $P$ . The cameras were normalized by setting the last element to one. The result is illustrated in Figure 7.1. As can be seen, the errors typically stay as low as  $10^{-15}$  to  $10^{-10}$ , but occasionally larger errors occur. However, since the solver is used as a subroutine in a RANSAC engine, which relies on solving a large number of different instances, these very rare cases with poor accuracy are not a serious problem.

As shown in Section 7.4 the (2,2) calibrated case in general has 16 solutions. Since obviously only one of these solutions is the correct one it is interesting to investigate how many plausible solutions are typically obtained. With plausible solutions we mean real valued camera matrices which yield positive depths for all four problem points. In Figure 7.1 a histogram which shows the typical number of plausible solutions is given. As can be seen the most common situation is

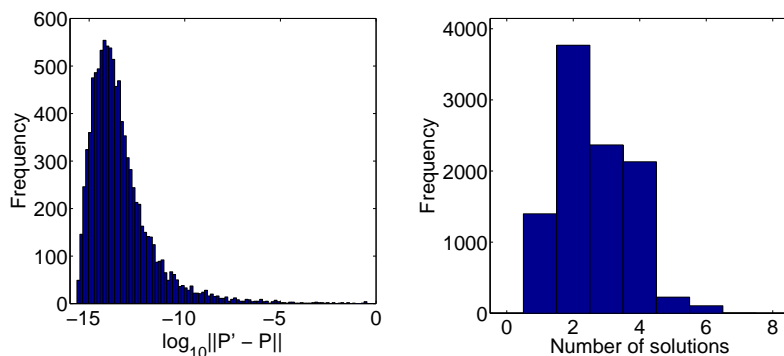


Figure 7.1: Statistics from the evaluation of the solver for the (2,2) case for calibrated cameras. The solver was run on 10,000 randomly generated cases. Left: Histogram over the error in matrix norm between the estimated camera  $P'$  and the true camera  $P$ . The error is plotted on a logarithmic scale. Right: Histogram over the number of real valued solutions yielding positive depths.

one to four plausible solutions. In one of the 10,000 cases, the algorithm was unable to find a real solution with positive depths for all points. This is probably due to numerical problems when the points and/or cameras are unfortunately positioned (two or more real solutions irrespective of the sign of the depths were found in all cases). In three of the cases seven solutions were found and in one case eight plausible solutions were found. The average number of plausible solutions was 2.6 and the average number of real solutions was 6.4. In some of the cases all 16 solutions were real.

### 7.4.3 Unknown Focal Length

For the case of unknown focal length we have one additional unknown and we thus need one extra constraint. There are several interesting minimal cases: (1,3), (3,2) and (5,1). However for the last case (assuming that all the five points were in correspondence with the same view) one could solve the relative orientation problem using the six point algorithm [50] and then fix the scale using the known 3D correspondence.

Using (7.2) as parameterization for the camera matrix and assuming that two of the 3D point correspondences are with

$$U_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, U_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, u_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

it is possible to eliminate  $y = 0$  and  $x = zf = g(b, c, d, f)$ . We fix the scale by setting  $a = 1$ . For both the (1,3) case and the (3,2) case we get polynomial constraints in the five remaining unknowns  $(b, c, d, z, f)$ . Calculations with computer algebra suggest that there are 36 solutions for the (1,3) case, 40 solutions to the (3,2) case and 112 in the (5,1) case.

A numerical algorithm for the (1,3) case has been obtained as follows. The above parameterization gives four equations in four unknowns. The unknowns

are the three quaternion parameters and the focal length. The equation derived from the line correspondence is of degree 6 and those obtained from the 3D points are of degree 3. The coefficient matrix  $\mathbf{C}$  is then constructed by expanding all equations up to degree 10. This means that the equation derived from the line is multiplied with all monomials up to degree 4, but no single variable in the monomials is of higher degree than 2. In the same manner the point correspondence equations are multiplied with monomials up to degree 7 but no single variable of degree more than 5. The described expansion gives 980 equations in 873 monomials. All of these equations were necessary to get a working solution to the problem.

Whereas the (2,2) case was reasonably well conditioned in itself, the (1,3) case is significantly more complicated as can be seen by the number of equations that need to be generated and we were not even able to construct a numerical solver using previous methods. As it turned out, truncation had to be used (*i.e.* a redundant basis for  $\mathbb{C}[\mathbf{x}]/I$ ) to avoid a rank deficient elimination step.

Proceeding as in Chapter 4 we partition and reorder the monomials into excessive monomials ( $\mathcal{E}$ ), monomials to reduce ( $\mathcal{R}$ ) and permissible basis monomials ( $\mathcal{P}$ ). In this problem  $\mathbf{C}_{\mathcal{P}}$  corresponds to all monomials up to degree 4 except  $f^4$  where  $f$  is the focal length, which gives 69 columns in  $\mathbf{C}_{\mathcal{P}}$ . The part  $\mathbf{C}_{\mathcal{R}}$  corresponds to the 5:th degree monomials that appear when the monomials in  $\mathcal{P}$  are multiplied with the first of the unknown quaternion parameters.

#### 7.4.4 Experimental Results for the (1,3) Case

The synthetic examples for the (1,3) problem were generated in the same manner as for the (2,2) case. Here, this gives one unknown camera with three point correspondences and one line correspondence. The experiment was run 10,000 times.

Figure 7.2 gives the distribution of relative errors in the estimated focal length. It can be seen that both the SVD method and the faster QR method give useful results. We emphasize that we were not able to construct a solver with the standard method and hence no error distribution for that method is available.

In Figure 7.3 the distribution of basis sizes is shown for the QR method with variable basis size. For the SVD method the basis size was identical to the QR method in over 97% of the cases and never differed by more than one element.

## 7.5 Conclusions

In this chapter we have shown new ways to use both 2D and 3D correspondences to solve the localization problem. Several minimal configurations have been classified and the numbers of solutions have been derived.

For the case of calibrated cameras with two 2D and two 3D correspondences and for the case of a camera with unknown focal length with one 2D and three 3D correspondences experiments have been performed. The synthetic experiments show that by using Gröbner basis methods, it is possible to get numerically stable calculations for these problems. The latter of the two cases turned out to be not even solvable using previous methods.

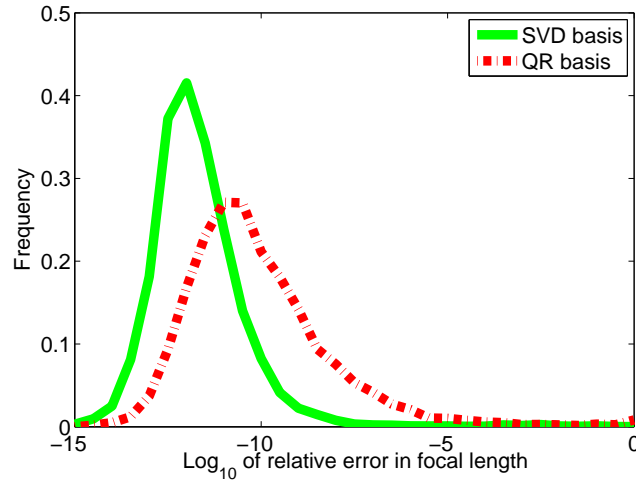


Figure 7.2: Error histogram for the (1,3) case with unknown focal length. The standard method is omitted since we did not manage to construct a standard solver due to numerical problems.

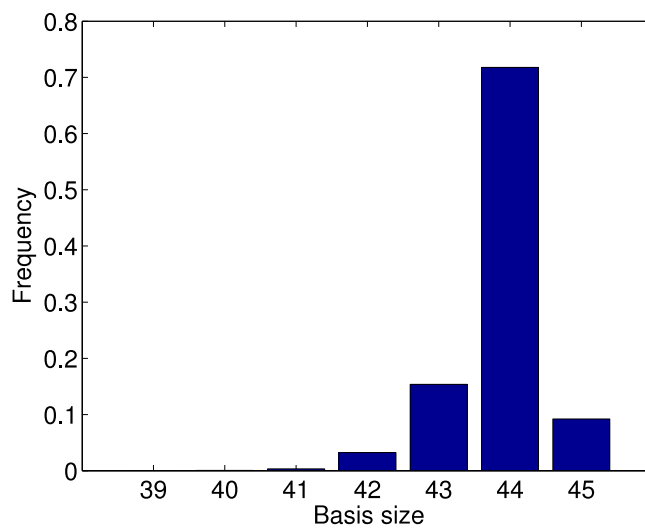


Figure 7.3: The distribution of basis sizes for the (1,3) problem with unknown focal length solved with the QR method with variable basis size. The number of solutions are 36 and since we always add three dimensions to the truncated ideal the minimal possible basis size is 39.

The logical way to extend this work is to test the remaining cases with semi-calibrated (unknown focal length) and uncalibrated cameras. Another natural line of future work is to investigate how the proposed methods perform in a complete structure and motion system.

## Chapter 8

# Conclusions

In a sense, the title of the thesis is a bit misleading. The words “fast and stable” seem to imply that we now have the tools to easily and efficiently solve most polynomial equations. This is far from true. Whereas we have come a long way as demonstrated by many examples in this thesis, many problems still remain way out of reach. However, in the cases we have encountered so far, numerical stability has no longer been the limiting factor. Instead what sets the limit for what we can solve numerically is the sheer size of the matrices occurring in the computations, leading to infeasible time and memory requirements. We can typically deal with systems of up to 50 or sometimes even 100 solutions, but above that our methods are simply insufficient. Alternatively, these problems are perhaps inherently so difficult that no really efficient methods to solve them exist.

There are however many interesting questions directly connected to the methods presented in this thesis that still have not been answered. The central theme of this work has been to generalize the action matrix method and exploit as many previously overlooked opportunities to improve speed and stability as possible. The most important discovery here is arguably the large freedom in how a basis can be selected from the set of all monomials  $\mathcal{M}$  occurring in an expanded set of equations. This is also where most topics still to be explored can be found. For instance, given an expanded set of equations, one would like to know if it is at all possible to construct a solving basis for this set of equations and in that case how it should be chosen. A solid theoretical understanding of this question and efficient and reliable algorithms for answering this for particular cases would be immensely helpful in applications. Furthermore, except for manual testing, we have no real guidance in how to construct the expanded set of equations. Currently, this is largely an empirical process done by hand. What degrees should we go to? Should we go to the same degrees for all equations? For all variables? Are there any bounds on what degrees we will need to go to? These questions are most likely very difficult to answer and have been studied for quite a long time in the algebraic geometry community.

This thesis discusses both stability and speed, but looking at the main contributions and the experiments it is evident that numerical stability has been the main focus. Since huge amounts of data is typically paired with real time requirements in computer vision applications, speed is however always of high priority. An interesting topic which has not been much explored yet in com-

puter vision applications of polynomial solvers is real root extraction. It is not uncommon in a case with say 50 solutions that only a handful of these are real. It seems that an algorithm which computes only these could be much faster. A promising possibility here is to compute a total degree Gröbner basis and then convert it to a lexicographical Gröbner basis using the FGLM algorithm [17]. This way one obtains a one-variable polynomial for which the real roots can be bracketed very efficiently using sturm sequences [28]. This would then have to be done once for each variable.

To summarize, we have introduced a range of techniques which have enlarged the class of problems that can now be handled successfully. However, approaching a particular problem by formulating it as a system of polynomial equations still comes with a degree of uncertainty. It is difficult to tell a priori what the outcome will be in terms of number of solutions, speed and stability. Under the right circumstances, solving a polynomial equation is by far the best method, especially in terms of speed. In other cases the polynomial system is simply too complex to yield anything valuable. Due to this, so far the main application in computer vision of these techniques has been to solve minimal problems of structure from motion. Only time will tell whether the strategy of formulating a given problem as a polynomial equation system will have a broader use.

# Bibliography

- [1] S. Agarwal, M. K. Chandraker, F. Kahl, D. J. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. In *Proc. 9th European Conf. on Computer Vision, Graz, Austria*, pages 592–605, 2006.
- [2] A. Almadi, A. Dhingra, and D. Kohli. A gröbner-sylvester hybrid method for closed-form displacement analysis of mechanisms. *Journal of Mechanical Design*, 122(4):431 – 438, 12 2000.
- [3] J. Barreto and K. Daniilidis. Fundamental matrix for cameras with radial distortion. In *IEEE International Conference on Computer Vision*, Beijing, China, 2005.
- [4] D. Bayer and M. Stillman. Macaulay. [www.math.columbia.edu/~bayer/Macaulay/](http://www.math.columbia.edu/~bayer/Macaulay/), 1994. An open source computer algebra software.
- [5] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996.
- [6] M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. In *Asian Conference on Computer Vision*, 2007.
- [7] M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of gröbner basis polynomial equation solvers. In *Proc. 11th Int. Conf. on Computer Vision, Rio de Janeiro, Brazil*, Rio de Janeiro, Brazil, 2007.
- [8] M. Byröd, K. Josephson, and K. Åström. A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations. Submitted, 2008.
- [9] M. Byröd, K. Josephson, and K. Åström. Optimal three view triangulation by polynomial equation solving. Submitted to *IPSN Transactions on Computer Vision and Applications*, 2008.
- [10] M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. Accepted for publication at the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [11] E. Cattani, D. A. Cox, G. Chèze, A. Dickenstein, M. Elkadi, I. Z. Emiris, A. Galligo, A. Kehrein, M. Kreuzer, and B. Mourrain. *Solving Polynomial Equations: Foundations, Algorithms, and Applications (Algorithms*

- and Computation in Mathematics*). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
- [12] M. Chasles. Question 296. *Nouv. Ann. Math.*, 14(50), 1855.
- [13] D. Cox, J. Little, and D. O’Shea. *Using Algebraic Geometry*. Springer Verlag, 1998.
- [14] D. Cox, J. Little, and D. O’Shea. *Ideals, Varieties, and Algorithms*. Springer, 2007.
- [15] M. Demazure. Sur deux problemes de reconstruction. Technical Report 882, INRIA, Rocquencourt, France, 1988.
- [16] Visual geometry group, university of oxford. <http://www.robots.ox.ac.uk/~vgg>.
- [17] J. Faugère, P. Gianni, and T. Lazard, D. och Mora. Efficient computation of zero-dimensional gröbner-bases by change of ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
- [18] J.-C. Faugère. A new efficient algorithm for computing gröbner bases ( $f_4$ ). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
- [19] J.-C. Faugère and A. Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In *CRYPTO 2003*, pages 44–60, 2003.
- [20] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–95, 1981.
- [21] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proceedings of Computer Vision and Pattern Recognition Conference (CPVR)*, pages 125–132, 2001.
- [22] C. Geyer and H. Stewénius. A nine-point algorithm for estimating paracatadioptric fundamental matrices. In *Proc. Conf. Computer Vision and Pattern Recognition, Minneapolis, USA*, June 2007.
- [23] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.
- [24] R. Hartley and F. Schaffalitzky.  $L_\infty$  minimization in geometric reconstruction problems. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, pages 504–509, Washington DC, USA, 2004.
- [25] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, 1997.
- [26] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.
- [27] R. Holt, R. Huang, and A. Netravali. Algebraic methods for image processing and computer vision. *IEEE Transactions on Image Processing*, 5:976–986, 1996.

- [28] D. G. Hook and P. R. McAree. Using sturm sequences to bracket real roots of polynomial equations. *Graphics gems*, pages 416–422, 1990.
- [29] K. Josephson, M. Byröd, F. Kahl, and K. Åström. Image-based localization using hybrid feature correspondences. In *The second international ISPRS workshop BenCOS 2007, Towards Benchmarking Automated Calibration, Orientation, and Surface Reconstruction from Images*, 2007.
- [30] F. Kahl. Multiple view geometry and the  $l_\infty$ -norm. In *ICCV*, pages 1002–1009, 2005.
- [31] F. Kahl. Multiple view geometry and the  $L_\infty$ -norm. In *International Conference on Computer Vision*, pages 1002–1009, Beijing, China, 2005.
- [32] F. Kahl and D. Henrion. Globally optimal estimates for geometric reconstruction problems. In *Proc. 10th Int. Conf. on Computer Vision, Beijing, China*, pages 978–985, 2005.
- [33] I. Karasalo. A criterion for truncation of the  $QR$ -decomposition algorithm for the singular linear least squares problem. *BIT Numerical Mathematics*, 14(2):156–166, June 1974.
- [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. Byröd, T. Pajdla, K. Josephson, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. Submitted to *Computer Vision and Image Understanding*, 2008.
- [36] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.
- [37] Z. Kukelova and T. Pajdla. Two minimal problems for cameras with radial distortion. In *Proceedings of The Seventh Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*, 2007.
- [38] Lapack - linear algebra package. <http://www.netlib.org/lapack>.
- [39] D. Lazard. Resolution des systemes d'equations algebriques. *Theor. Comput. Sci.*, 15:77–110, 1981.
- [40] H. Li and R. Hartley. A non-iterative method for lens distortion correction from point matches. In *Workshop on Omnidirectional Vision*, Beijing China, October 2005.
- [41] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 2004.
- [42] Fangfang Lu and Richard Hartley. A fast optimal algorithm for  $2$  triangulation. In *ACCV*, pages 279–288, 2007.
- [43] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image Vision Computing*, 22(10):761–767, 2004.

- [44] D. Nistér. An efficient solution to the five-point relative pose problem. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 195–202. IEEE Computer Society Press, 2003.
- [45] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.
- [46] J. Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *Photogrammetric Record*, 15(88):589–599, October 1996.
- [47] R. Pless. Using many cameras as one. In *Proc. Conf. Computer Vision and Pattern Recognition, Madison, USA*, 2003.
- [48] H. Stewénus. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, April 2005.
- [49] H. Stewénus, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.
- [50] H. Stewénus, F. Kahl, D. Nistér, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *Proc. Conf. Computer Vision and Pattern Recognition, San Diego, USA*, 2005.
- [51] H. Stewénus, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision*, Beijing China, October 2005.
- [52] H. Stewénus, F. Schaffalitzky, and D. Nistér. How hard is three-view triangulation really? In *Proc. Int. Conf. on Computer Vision*, pages 686–693, Beijing, China, 2005.
- [53] E. H. Thompson. A rational algebraic formulation of the problem of relative orientation. *Photogrammetric Record*, 14(3):152–159, 1959.
- [54] P. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15(8):591–605, 1997.
- [55] P. Torr and A. Zisserman. Robust computation and parametrization of multiple view relations. In *Proc. 6th Int. Conf. on Computer Vision, Mumbai, India*, pages 727–732, 1998.
- [56] W. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: A modern synthesis. In *Vision Algorithms: Theory and Practice*, LNCS. Springer Verlag, 2000.
- [57] J. Verschelde. Phcpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25(2):251–276, 1999.