Mathematical Methods for Image Based Localization

Josephson, Klas

2010

Mathematical Methods for Image Based Localization

Klas Josephson

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/

Doctoral Theses in Mathematical Sciences 2010:2
ISSN 1404-0034

ISBN 978-91-628-8062-0
LUTFMA-1040-2010

Printed in Sweden by Media-Tryck, Lund 2010

Abstract

The underlying question in localization is *where am I?* In this thesis a purely image based approach is proposed to solve this problem. In order to create a complete image based system, there are many subproblems that have to be addressed. The localization problem can also be solved in other ways, for example, with a GPS. Two advantages with using images compared to GPS are that no open sky is needed and that a higher precision is possible to achieve.

The thesis consists of an introductory chapter followed by six papers. In the first paper, enhancements of Gröbner basis techniques to solve systems of polynomial equations are presented. The new strategies improve the numeric stability with several orders of magnitudes, compared to previous state of the art. This framework is then applied in the next three papers to solve several geometrical pose problems relevant for localization. The main difference between the papers is the level of knowledge of the inner calibration of the cameras. The calibration knowledge ranges from completely calibrated cameras to uncalibrated cameras with unknown radial distortion. The fifth paper of the thesis also treats the pose problem, but the method differs from the previous papers. In this paper a method is presented that guarantees a globally optimal solution at the price of computational complexity. To achieve this, the pose problem is reformulated and solved via a minimal vertex cover. The final paper is devoted to large-scale localization. Methods from image retrieval are utilized, and extended, to be able to perform city-scale localization. Moreover geometry is directly incorporated in the retrieval stage.

Sammanfattning

Var är bilden tagen? Det är en fråga som vi ofta ställer oss och som denna avhandling innehåller matematiska metoder för att låta en dator besvara. Om vi lyckas svara på frågan kan metoderna för att hitta svaret användas till vitt skilda saker. Det går att använda för att låta robotar navigera med hjälp av kameror men också för att sortera privata bilder i digitala fotoalbum efter var de är tagna.

För att lösa lokaliseringsproblemet har flera delproblem studerats. I ett av delproblemen har en tredimensionell modell av Malmö byggts upp utifrån 95 000 bilder tagna utmed stadens gator. Med hjälp av modellen går det för en ny bild, tagen någonstans i Malmö att avgöra var fotografen befann sig. Det görs genom att titta på många små områden i den nya bilden. De områdena beskrivs sedan med en uppsättning tal så att liknande områden går att hitta i den tredimensionella modellen. Om tillräckligt mycket är känt om kameran som tagit bilden går det att räkna ut vilken vinkel det är mellan riktningarna till två punkter som avbildats i bilden, sett från kameran. Tack vare den tredimensionella modellen av staden är det sedan möjligt att räkna ut var i Malmö den vinkeln kan uppstå mellan punkterna. Med hjälp av det går det att få fram en lista på platser där bilden kan vara tagen.

Det som hittills har beskrivits kan fungera som ett första steg i en lokaliseringsprocess. För att ta reda på mera om kamerans position, så som riktning, eller för att få en högre precision i resultatet kan man utgå från resultatet av den första delen för att hitta positioner att undersöka noggrannare. Även de fortsatta algoritmerna för att söka efter position har utvecklats i avhandlingen. När antalet tänkbara platser begränsats efter det första steget är det möjligt att utföra mera resurskrävande beräkningar som en dator inte skulle klara av för en hel stad på rimlig tid. I arbetet har det utvecklats två typer av algoritmer för detta problem, dels en som är optimal, alltså att det är säkert att det inte finns en bättre lösning, dels metoder som går snabbare men som inte garanterat ger ett optimalt resultat.

Det kan låta konstigt att inte nöja sig med en metod som ger ett optimalt svar, men dessa algoritmer tar ofta för lång tid att använda, och då får vi hoppas på att kunna ta en genväg för att hitta en nästan lika bra lösning. Det visar sig också att det i de flesta fallen blir fullt tillfredsställande lösningar även när den snabba vägen används.

Det arbete som gjorts på de snabbare metoderna i avhandlingen handlar till stor del om beräkningar kopplade till hur kameror avbildar omgivningen. För olika matematiska modeller av en kamera, som är beroende av hur mycket som är känt om kamerans konstruktion och även vilken brännvidd som användes vid fototillfället, blir det olika geometriska problem att lösa. Vid användning av de snabba algoritmerna utnyttjas ett minimalt antal punkter i bilden, som har kopplats samman med punkter i den tredimensionella modellen, för att problemet ska kunna lösas. Sedan löses problemet för denna minimala uppsättning punkter. Lösningen testas hur väl den stämmer överens med de övriga punkterna som kopplats samman mellan bilden och modellen. Många tidigare olösta sådana problem, där ett minimalt antal punkter är kopplade mellan kameran och modellen, presenteras det lösningar på i avhandlingen. Lösningarna blir då en viktig kugge i ett fullständigt lokaliseringssystem.

Preface

This thesis considers the problem of image based localization. The objective is to find at what position and in which direction an image was taken relative a world model. To achieve this several subproblems are studied. These problems range from minimal cases for camera position estimation in a small model, to city-scale localization. Also techniques for solving systems of polynomial equations are presented. These methods are the foundation for many of the problems solved in the thesis.

The work has been funded by the Swedish Research Council through grant no. 2004-4579 'Image-Based Localisation and Recognition of Scenes' and by the Swedish Foundation for Strategic Research (SSF) through the program Future Research Leaders.

The thesis consists of the following six papers:

I M. Byröd, K. Josephson, K. Åström, Fast and Stable Polynomial Equation Solving and Its Application to Computer Vision, published in *Int. Journal of Computer Vision*, 84(3):237–256, 2009.

II K. Josephson, M. Byröd, F. Kahl, K. Åström, Image Based Localization Using Hybrid Features, submitted to *Journal of Mathematical Imaging and Vision*, 2009.

III Z. Kúkelová, M, Byröd, K. Josephson, T. Pajdla, K. Åström, Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion, published in *Computer Vision and Image Understanding*, 114(2):234–244, 2010.

IV K. Josephson, M. Byröd, Pose Estimation with Radial Distortion and Unknown Focal Length, published at *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* Miami, FL, USA, 2009.

V O. Enqvist, K. Josephson, F. Kahl, Optimal Correspondences, Geometry and the Vertex Cover Problem, submitted to *Int. Journal of Computer Vision*, 2010.

VI K. Josephson, L. Svärm, O. Enqvist, F. Kahl, City-Scale Localization Using Geometrical Visual Words, submitted to *European Conference on Computer Vision*, 2010.

During the work of this thesis, the following papers have also been written of which some in parts overlap with the ones in the thesis.

- K. Josephson, M, Byröd, F. Kahl, K. Åström, Image Based Localization Using Hybrid Features Correspondences, *Proc. ISPRS workshop BenCOS at CVPR*, Minneapolis, MN, USA, 2007.
- M. Byröd, K. Josephson, K. Åström, Improving Numerical Accuracy of Gröbner Basis Polynomial Equation Solvers, *Proc. Int. Conference on Computer Vision*, Rio de Janeiro, Brazil, 2007.
- M, Byröd, K. Josephson, K. Åström, Fast Optimal Three View Triangulation, *Proc. Asian Conference on Computer Vision*, Tokyo, Japan, 2007.
- K. Josephson, F. Kahl, Triangulation of Points, Lines and Conics, published in *Journal of Mathematical Imaging and Vision*, 32(2):215–225 2008.
- M, Byröd, Z. Kúkelová, K. Josephson, T. Pajdla, K. Åström, Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion, *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition,*, Anchorage, AK, USA, 2008.
- M, Byröd, K. Josephson, K. Åström, A Column-Pivoting Based Strategy for Monomial Ordering in Numerical Gröbner Basis Calculations, *Proc. European Conference on Computer Vision*, Marseille, France, 2008.
- H. Aanaes, K. Josephson, F. Anton and J.A. Bærentzen, and F. Kahl, Camera Resectioning from a Box, *Proc. Scandinavian Conference on Image Analysis*, Oslo, Norway, 2009
- O. Enqvist, K. Josephson, F. Kahl, Optimal Correspondences from Pairwise Constraints, *Proc. Int. Conference on Computer Vision*, Kyoto, Japan, 2009.

Acknowledgements

First of all, I would like to thank my supervisor Fredrik Kahl for his inspiration, good research ideas, always being available for discussions and for his careful examination of the manuscript of this thesis as well as other manuscripts. Also Magnus Oskarsson has read and examined the manuscript and by that improved the quality considerably. I also would like to thank Kalle Åström for introducing me to the art of solving polynomial equations by Gröbner basis techniques, which is one of the main topics of this thesis, and for many discussions on that topic as well as other topics. My gratitude also goes to the Mathematical Imaging Group at the Centre for Mathematical Sciences and especially to Martin Byröd and Olof Enqvist. These two have been my main collaborators, Martin mostly in the first half of my time as PhD student and Olof after that. Linus Svärm, the remaining co-author from the group needs a mention for the efforts he put in to our collaboration during the last months. I also want to thank all those sharing the same coffee room for the joyful breaks in my work. Anki Ottosson needs a special mention for all help with all kinds of administrative difficulties.

Finally, I would also like to thank family and friends for all those things that are not work.

Contents

Introduction **13**
1 Background 13
1.1 Geometry 13
1.2 Overview 14
2 Geometry in Computer Vision 15
2.1 The Pinhole Camera 15
2.2 Two View Geometry 19
2.3 The Triangulation Problem 21
3 Algebraic Geometry 23
3.1 The Action Matrix 24
3.2 Gröbner Basis 25
4 The Correspondence Problem 26
4.1 RANSAC 26
4.2 Optimal Methods 29
5 Graph Theory and the Vertex Cover Problem 30
5.1 NP-Completeness 31
5.2 Factor-2 Approximation 31
5.3 Greedy Algorithm for Vertex Cover 32
6 Image Search 33
6.1 Solving Localization with Image Search 34
7 Summary of the Papers 35

Paper I **43**

Paper II **87**

Paper III **115**

Paper IV **143**

Paper V **163**

Paper VI **197**

Introduction

1 Background

The aim of this thesis is to develop new methods for image based localization, the problem of finding out where an image is taken. This is a problem that researchers have been working on for a long time and many localization systems have been constructed over time *e.g.* [36, 1]. Common for these two are that they only solve the problem in two dimensions. In this thesis both two and three dimensional localization are treated. GPS is a competing technique, but GPS has several drawbacks. These include that GPS needs an open sky to work, and that it gets very expensive for an accuracy below a few meters.

One branch of localization is called global localization, and that branch has been in focus for this thesis. The global localization problem, or the kidnapped robot problem as it is called in robotics, is the problem when there is no pre-knowledge of the camera position. Early systems that solved this problem used lasers to find the position and navigate a robot [36]. But in recent years more interest has been turned to the use of images to do the localization and navigation. One major reason for this is that digital cameras have become a consumer product which is cheap compared to lasers that have to be custom made for the application.

Another application for global localization is in personal photo collections. In the last few years cameras have started to be equipped with GPS, and now also compass, to be able to encode the position and the direction into each image. This has also led to that it is possible to navigate by the image positions in digital photo albums. But since the GPS technique just has found its path in to the cameras there exist millions of images that do not have a geo-position. To be able to tag these images with a position automatically, the global localization problem needs to be solved.

1.1 Geometry

All papers in this thesis are in the field of geometrical computer vision. The foundation of this field was long before the computers existed, and probably begun in 1841 with a paper by Grunert [15]. The paper treats the problem of calculating where a camera was

posed given three correspondences between the image and a model of the world. The next relevant paper was presented in 1855 by Chasles [3]. In his paper, Chasles looks at the problem of finding the geometry between two cameras if seven point correspondences are known between two images. Another important paper in the field was written by Kruppa in 1913 [25]. In this paper he almost correctly solves the problem of relative pose between two cameras when the inner calibration of the cameras is known. A corrected solution of the problem was presented by Thomson in 1959 [47]. After that the field really took off and the amount of publications has constantly increased rapidly.

One of the fundamental problems in geometrical computer vision is, given one or many images of the same object, to infer something about the structure of the imaged object and/or the camera motion between the images. This problem can be stated in several different ways depending on what is known about the different entities. In some problems the placement of two or more cameras is known and the problem is to calculate the three dimensional structure. This is the so-called *triangulation problem*. In other cases the three dimensional structure of the object is known, but nothing is known about the camera that took the images. The problem of finding the location of the camera is then known as the *camera pose problem*. Another situation is when neither the three dimensional structure nor the camera positions are known. This is called the *structure and motion problem*. A special case of the structure and motion problem consists of calculating the motion between two images. This is known as the relative pose problem. All these problems can also be varied depending on the knowledge about the inner parameters of the cameras used, such as, the focal length.

Of course there are some fundamental limitations in the problem of structure and motion. Depending on the a priori knowledge of the camera there are different levels of these ambiguities. For example, if the full inner calibration of a pin-hole camera is known, that is, the camera can be seen as a tool for measuring angles, then there will only be an ambiguity in size. In other words, it is impossible to know if both the object and the movements of the camera are small or if both are large.

1.2 Overview

The papers of this thesis address different aspects of the global localization problem. One of the fundamental steps, in the construction of a localization system, is to build a model wherein the search for position is done. In Paper III this problem is addressed, where problems of relative movement between pairs of images are treated. The relative positions of cameras need to be known in order to be able to triangulate points and by that construct a three dimensional model of the search area.

The constructed three dimensional models are then used in different kinds of localization in Papers II, IV and V. The difference between the methods used in these papers is mainly how much of the inner calibration of the cameras, that is known. They also differ in demands of computational power. In Paper V, optimality is guaranteed for the

solution, with the price that more computations need to be done.

The methods presented in Paper II, IV and V only work when the unknown position is within a few blocks of a city. To be able to handle large scale problems, such as city-scale localization, a pre-processing step is needed. In Paper VI such an initial step is presented. In this paper methods from image retrieval are used to reduce the search space from a complete city to just a few small possible areas, where the image can have been taken.

Papers II, III and IV of the thesis, all have a step where a system of polynomial equations needs to be solved. To do that a method based on Gröbner basis, from the field of algebraic geometry is used. The Gröbner basis method is known to be a fast way to solve systems of polynomial equations, but it has been plagued by numerical problems. In Paper I, these numerical issues are addressed, and methods to significantly reduce these problems are presented. The first paper is, hence, a base for several of the following papers in the thesis.

Next in this introduction, the most fundamental concepts of geometrical computer vision and algebraic geometry will be explained. In the introduction an overview of important concepts in computer vision will be given, such as how a camera is modeled, the geometry between two images and some aspects of the triangulation problem. In Section 3 the concepts of ideals and varieties in the field of algebraic geometry are explained. This is followed by the basics on Gröbner bases, and an introduction to the important concept of the action matrix. This is what will be used in Paper I and it will be an important base in the three subsequent papers.

In Section 4, methods used to solve the correspondence problem are presented. This includes RANSAC, a common algorithm to remove false matches, and some theory on four classical geometrical camera resection problems. There is also explanation on how the correspondence problem can be solved optimally. These techniques result in a graph problem called vertex cover, so the basics of this problem and how to solve it, are given in Section 5. The last section of this introduction, Section 6, treats how image search can be performed, and on how this is used in the thesis to solve the global localization problem.

2 Geometry in Computer Vision

2.1 The Pinhole Camera

One of the foundations of geometrical computer vision is the pinhole camera model. The modeled pinhole camera consists of two components, the focal point and the image plane. Figure 1 shows a schematic figure of a pinhole camera, where $\mathbf{C}$ is the focal point and π the image plane. Further, $\mathbf{X}_1$ and $\mathbf{X}_2$ are world points and $\mathbf{x}_1$ and $\mathbf{x}_2$ are projected image points.

The Pinhole camera can be modeled mathematically with the so called camera equation,

$$\lambda \mathbf{x} = P\mathbf{X}. \tag{2.1}$$

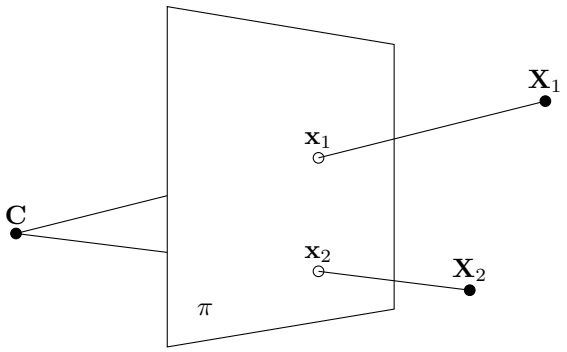

Figure 1: The geometry of a pinhole camera. $\mathbf{X}_i$ are points in the three dimensional space projected to the image points, $\mathbf{x}_i$ on the image plane π. $\mathbf{C}$ is the focal point, also referred to as the camera center.

In this equation λ holds the distance between the focal point and the world point and both the image and the world point are in homogeneous/extended coordinates [19]. P is the camera matrix of size 3×4 and holds both the intrinsic and extrinsic parameters of the camera. To understand why (2.1) models a pinhole camera put the focal point in the origin and the image plane as $z = f$ where f is the focal length, that is, the distance between the image plane and the focal point. In this case the setup is as in Figure 2.

As can be seen in Figure 2 a point $\mathbf{X} = (0, Y, Z)^T$ is mapped in the camera to the point $(0, fY/Z, f)^T$. Since it is possible to rotate this setup to include X, with values different from zero, the mapping becomes,

$$(X, Y, Z)^T \mapsto (\frac{fX}{Z}, \frac{fY}{Z})^T. \tag{2.2}$$

With homogenous coordinates this mapping can be expressed with matrix multiplication according to

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{2.3}$$

The principal point is the orthogonal projection of the focal point on the image plane. During this derivation it has been assumed that the pinhole camera has its principal point at the origin of the image plane. This can not be taken for granted so the mapping in (2.3)

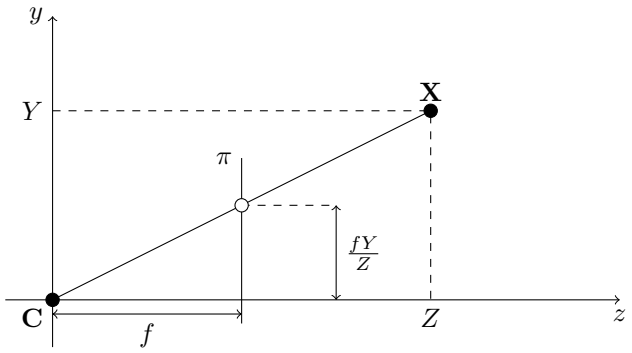

Figure 2: A pinhole camera with the image plane, π, placed at $z = f$ where f is the focal length of the camera. The focal point $\mathbf{C}$ is placed in the origin and the world point $\mathbf{X}$ is projected to the image plane.

can be adjusted to account for different principal points by putting

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{2.4}$$

where (p_x, p_y) is the principal point. The camera matrix can now be written as $\lambda\mathbf{x} = K[\mathrm{I} \mid \mathbf{0}]\mathbf{X}$, where K is

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}. \tag{2.5}$$

There are two more calibration parameters, that are associated with the design of the CCD, the image registration sensor of a digital camera. The first of those is the aspect ratio of the pixels, γ. If the pixels are square than γ equals 1, which is true for almost all CCD's. The second parameter is the skew, s. The skew parameter will be zero if the x and y axes are perpendicular on the CCD, which for almost all cameras will be true. These parameters are included in the calibration matrix as follows,

$$K = \begin{bmatrix} f & s & p_x \\ & \gamma f & p_y \\ & & 1 \end{bmatrix}. \tag{2.6}$$

This is the full calibration matrix that is used to model a pinhole camera. The only things that are still needed is to move the camera away from the origin and to incorporate full

freedom in rotation. This is done by defining the general camera matrix as,

$$P = K[R \mid \mathbf{t}], \tag{2.7}$$

where R is an orthogonal matrix and $\mathbf{t}$ a translation vector. This is the camera matrix for an arbitrary pinhole camera. With this camera (2.1) represents a projection through a pinhole camera in homogenous coordinates. For a more exhaustive derivation of the camera equation see [19].

2.1.1 Radial Distortion

The pinhole camera model, presented in the previous section, is the model of a camera used throughout this thesis. In Papers III and IV, is this camera model extended to also model radial distortion, present in camera lenses. The distortion is usually larger in small lenses, such as those in mobile phones and compact cameras, but it is also present in lenses used in SLR cameras. The importance of modeling the radial distortion in some problems was shown in [11]. In that paper examples of two three dimensional reconstructions are shown, one modeling radial distortion and one not doing that, and only the one taking radial distortion in consideration manage to get a tractable result.

A common model for radial distortion is to assume that the distortion center coincides with the image center, then the distortion can be written,

$$\mathbf{x}_d = \mathcal{L}(\|\mathbf{x}_u\|)\mathbf{x}_u, \tag{2.8}$$

where $\mathbf{x}_d$ is the measured image points, $\mathbf{x}_u$ the ideal undistorted coordinates from the camera equation (2.1), and $\mathcal{L}$ the distortion factor. The function $\mathcal{L}(r)$ is usually a polynomial, $\mathcal{L}(r) = 1+\kappa_1 r+\kappa_2 r^2+\kappa_3 r^3+\cdots$, which can be seen as the Taylor expansion of an arbitrary smooth function $\mathcal{L}(r)$ with $\mathcal{L}(0) = 1$.

This model of radial distortion, called the plumbline model, is the most commonly used. One problem with the distortion model in (2.8) is that it usually, when applied, not gives tractable mathematics, since we often want to express $\mathbf{x}_u$ in terms of $\mathbf{x}_d$. One way to avoid that problem is to use the similar, but different, division model for the radial distortion,

$$\mathbf{x}_u = \frac{\mathbf{x}_d}{1 + \mu\|\mathbf{x}_d\|^2}. \tag{2.9}$$

This model was shown to give an equally good approximation to the radial distortion, if only κ_2 was used, as (2.8) in [11].

2.1.2 Different Levels of Camera Calibration

Depending on how much that is known of the inner calibration, different methods are used to solve the geometrical problems. If all the parameters in the matrix K, see (2.6), are known the camera is said to be calibrated. On the other hand, if none of the parameters

are know the camera is called uncalibrated. But it can also be intermediate levels of calibration. The most frequent case of these semi-calibrated cases is that everything except the focal length f is known about the inner calibration. This often models a digital camera well since the principal point is usually centered to the middle of the image. In some cases this is not the case, especially when the image has been cropped afterwards. In those cases the principal point moves away from the center of the image, and it has to be estimated.

2.2 Two View Geometry

Geometrical computer vision often includes several cameras, or several images with different views by the same camera. In two-view geometry, the most important property is the epipolar geometry. The epipolar geometry is the intrinsic geometry between two views. One main motivation for the study of epipolar geometry is the search for correspondences in triangulation. The reason for this is that the epipolar geometry defines a line to search along in the second image to find a corresponding point, given a point in the first image. But the epipolar geometry is also used the other way around. The epipolar geometry can be calculated through several correspondences between two images which then can be used to find the camera matrices in the relative pose problem.

Given two corresponding points $\mathbf{x}$ and $\mathbf{x}'$ in two images, then these two points have a common point $\mathbf{X}$ in the three dimensional space. With the information from one of the images the possible locations of $\mathbf{X}$ are reduced to a line *i.e.* the point is known but not the depth. This line is projected down to a line in the second image and $\mathbf{x}'$ should be placed on this line, at least in absence of noise. The line which the corresponding point can be located on is called the epipolar line.

The geometric entities of the epipolar geometry are shown in Figure 3. The epipoles, $\mathbf{e}$ and $\mathbf{e}'$, are the intersection of the image planes with the baseline joining the camera centers. Another way to characterize the epipoles is that they are the images of the focal points of the other. The second entity is the epipolar plane, which is a plane containing the baseline and the point of interest in the three dimensional space. The last entity is the epipolar line, which is the intersection of an epipolar plane with the image plane. If the epipolar geometry is known between two cameras, a point in one image defines an epipolar line in the other image. Furthermore the epipolar line always intersects the epipole.

2.2.1 The Fundamental Matrix

To algebraically describe the epipolar geometry between two cameras the fundamental matrix is used. The fundamental matrix holds the information of how a point in one of the images generates the corresponding epipolar line in the other image. The 3×3 fundamental matrix F will map points to lines in homogeneous coordinates according

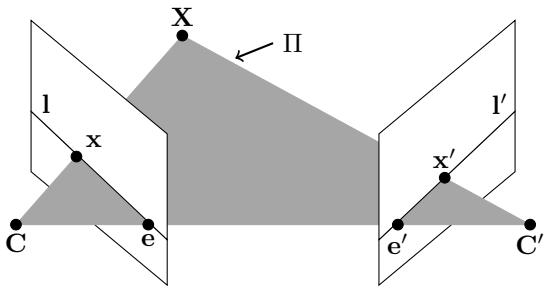

Figure 3: A geometry with two cameras and a point $\mathbf{X}$ projected down to $\mathbf{x}$ and $\mathbf{x}'$, respectively, in the cameras. The baseline between the two camera centers, $\mathbf{C}$ and $\mathbf{C}'$, intersect the image planes in the epipoles, $\mathbf{e}$ and $\mathbf{e}'$. The lines $\mathbf{l}$ and $\mathbf{l}'$ in the images, given by the plane Π, are called epipolar lines. The plane Π is defined by the camera centers and the point $\mathbf{X}$.

to,

$$\mathbf{l}' = F\mathbf{x}. \tag{2.10}$$

Due to the properties of the epipolar geometry described in the last section the corresponding point $\mathbf{x}'$ to $\mathbf{x}$ has to be contained in the epipolar line $\mathbf{l}'$, hence

$$\mathbf{x}'^T\mathbf{l}' = \mathbf{x}'^T F\mathbf{x} = 0, \tag{2.11}$$

has to be fulfilled for all corresponding points. The fundamental matrix has several important properties. If F is the fundamental matrix of a camera pair (P, P') then F^T is the fundamental matrix for the pair (P', P). This follows directly from (2.11). Furthermore the fundamental matrix is only given up to scale and has rank 2. The reason why F has rank 2 can be understood by the fact that the fundamental matrix maps points into lines and all these lines intersect in the epipole. Hence it is enough to give a point on a circle around the epipole, to describe the epipolar geometry.

The fundamental matrix has seven degrees of freedom. The nine elements of the fundamental matrix encode these seven degrees of freedom since it is homogenous and $\det(F) = 0$ due to the fact that it does not has full rank.

2.2.2 The Essential Matrix

If the cameras are calibrated the degrees of freedom for the fundamental matrix get reduced, and the name also changes to the essential matrix. Without calibration the fundamental matrix had seven degrees of freedom but the essential matrix only has five. This can be realized since if the cameras are calibrated the first camera can be set $P = [I \mid \mathbf{0}]$ and the second to $P = [R \mid \mathbf{t}]$. Both the translation, $\mathbf{t}$, and the rotation, R, have three

degrees of freedom, and we also have the remaining freedom to scale the visible system which leaves five degrees of freedom.

The essential matrix is defined in the same way as the fundamental matrix,

$$\hat{\mathbf{x}}'^T E\hat{\mathbf{x}} = 0, \tag{2.12}$$

where $\hat{\mathbf{x}}'$ and $\hat{\mathbf{x}}$ are the coordinates when the effects of the calibration is removed, so-called normalized coordinates. From this definition and the relation $\hat{\mathbf{x}} = K^{-1}\mathbf{x}$ it follows that $\mathbf{x}'^T K'^{-T} E K^{-1}\mathbf{x} = 0$. This equation gives the connection between the essential and fundamental matrices to be as follows,

$$E = K'^T F K. \tag{2.13}$$

The reduced number of degrees of freedom gives additional constraints on the essential matrix compared to the fundamental matrix. These constraints are that the essential matrix is a matrix where two of the singular values values are equal and the last is zero [19].

In [37] Nistér uses the property,

$$2EE^T E - \mathrm{tr}(EE^T)E = 0, \tag{2.14}$$

introduced by Demazure [7] and later also used by [8], of the essential matrix. This property is derived from the fact that two singular values are equal and is used to find the essential matrix given the minimum set of five correspondences between two calibrated cameras. This constraint has later been used with different constraints on the calibration matrix to solve several minimal cases in geometrical computer vision [27, 45]. This method is also used on several occasions in this thesis.

2.3 The Triangulation Problem

A classical problem in geometrical computer vision is the triangulation problem for points. The formulation of this problem is:

Problem 2.1 (Triangulation)**.** *Given two or more images and corresponding points in those, find which point in the world that is the source given that the cameras are known.*

Figure 4 holds a sketch of the problem of triangulation, where the task of triangulation is to find the point $\mathbf{X}$ that best fulfills the image points.

The triangulation problem is well studied and hence a lot of literature can be found, see [19, 32] and the references therein. In the absence of noise this problem is trivial and can be solved with two views by constructing the matrix,

$$A = \begin{bmatrix} x\mathbf{p}_3 - \mathbf{p}_1 \\ y\mathbf{p}_3 - \mathbf{p}_2 \\ x'\mathbf{p}_3' - \mathbf{p}_1' \\ y'\mathbf{p}_3' - \mathbf{p}_2' \end{bmatrix}, \tag{2.15}$$

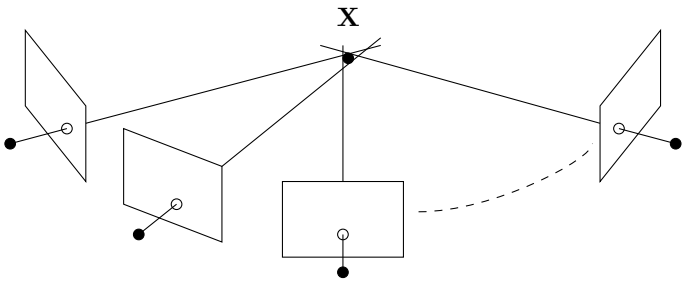

Figure 4: Sketch of multiple view triangulation. If there was no noise, all lines would intersect in a single point, but due to noise this is not true. The problem of triangulation is to find the point $\mathbf{X}$ that is the best approximation to such a point given some type of measure.

where $\mathbf{p}_i$, $\mathbf{p}'_i$ is the i:th row of the camera matrices P and P', respectively, of the two views. This matrix will not have full rank and the null space will be the sought after point, in homogeneous coordinates.

If there is noise in the images the problem becomes more difficult. First of all an optimization criterion has to be chosen. The most common choice, and the statistically optimal given gaussian noise in the image data, is the reprojection errors in L_2-norm. In recent years also the L_∞-norm has been given more interest, see *e.g.* [17, 22].

If the L_2-norm is chosen the most widely used method to triangulate a point is to use local optimization methods based on gradient decent. After an initialization using a linear method, so called bundle adjustment is applied [19]. This method is used even though it is well known that the error function will not be convex in general, and hence local minima can exists.

To overcome the problem of local minima, global optimal methods are interesting. Such methods have been given for two and three views. The solution of the two view case was given by Hartley and Sturm in 1997 [18]. The solution is given by calculating the stationary points of the cost function. This leads to solving a polynomial of degree six. The three view case was solved in 2005 by Stewénius *et al.* [46]. Also this problem is solved by calculating the stationary points of the cost function, but in this problem, the number of those points is 47. To solve that problem Gröbner basis methods are used. The paper of Stewénius *et al.* was one of the main inspirations for the first paper of this thesis due to the numerical difficulties they had. The problem of three view triangulation is used as a benchmarking problem in Paper I.

3 Algebraic Geometry

Due to the structure of the camera equation (2.1) minimal problems in geometrical computer vision often lead to solving systems of polynomial equations. This is also true if equation (2.14) is studied. The occurrence of polynomial systems get even more obvious for the camera equation if the camera is calibrated. In this case only the rotation and translation have to be determined. Representing the rotation with a quaternion $\mathbf{q} = (a, b, c, d)^T$ and the translation with a vector $\mathbf{t} = (x, y, z)^T$ the calibrated camera matrix can be described as follows:

$$P = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2ac+2bd & x \\ 2ad+2bc & a^2-b^2+c^2-d^2 & 2cd-2ab & y \\ 2bd-2ac & 2ab+2cd & a^2-b^2-c^2+d^2 & z \end{pmatrix}. \quad (3.1)$$

Under these prerequisites it is natural that many computer vision problems end up in solving a system of m polynomial equations in s variables, $\mathbf{x} = (x_1, \ldots, x_s)$, formalized as,

$$\begin{aligned} f_1(\mathbf{x}) &= 0, \\ &\vdots \\ f_m(\mathbf{x}) &= 0. \end{aligned} \quad (3.2)$$

To solve systems like this many methods exist, see [2] and references therein. In this thesis methods from algebraic geometry are used to solve these problems. See [5, 6] for an introduction to the field.

One of the benefits of using methods from algebraic geometry in computer vision is that these can be solved by fast algorithms under certain conditions. In these cases a problem specific solver can be constructed that computes a solution in a few milliseconds on a standard PC. This is a very attractive feature since these solvers often are used in the core of a hypothesis and test algorithm and hence have to be executed many times.

The two most important objects in algebraic geometry are the variety and the ideal.

Definition 3.1 (Variety)**.** *The variety V is the set of points $\mathbf{x}$ where all equations in a system of polynomial equations vanish.*

The variety does not need to be a finite set but in this thesis only varieties consisting of a finite set of points are considered.

Definition 3.2 (Ideal)**.** *The ideal I generated of a set of polynomials equation $f_i(\mathbf{x}) = 0$ is, ideal $I = \sum_i h_i(\mathbf{x}) f_i(\mathbf{x})$, where $h_i \in \mathbb{C}[\mathbf{x}]$.*

In this definition $\mathbb{C}[\mathbf{x}]$ denotes the set of all polynomials over the complex numbers.

The reason to study the ideal I to a set of equation as in (3.2) is that a point $\mathbf{x}$ is a zero of (3.2) iff it is a zero to the ideal. Now the ideal is used to define equivalence of two polynomials. This is done by saying that two polynomials f and g are equivalent modulo

I iff $f - g \in I$, denoted by $f \sim g$. With this definition a quotient space $\mathbb{C}[\mathbf{x}]/I$ is achieved of all equivalent classes modulo I.

It is also possible to construct equivalence classes based on the variety V. Here two polynomials are said to be equivalent if they are equal on all $\mathbf{x} \in V$. It is obvious that if two polynomials are equivalent modulo I then they are also equal on V. If furthermore the ideal is a radical [6] then the converse is also true and the two structures are isomorphic. The condition on I to be a radical is that, if $f^m \in I$ for some $m \geq 1$ then $f \in I$.

The most important property of the quotient space, $\mathbb{C}[\mathbf{x}]/I$, is that the dimension is equal to the number of solutions of (3.2).

3.1 The Action Matrix

The knowledge of ideals and varieties can be used to solve systems of polynomial equations. To understand this let us start of by looking at the problem of finding the roots of a polynomial of degree three. If the polynomial is

$$q(x) = x^3 + a_2x^2 + a_1x + a_0, \tag{3.3}$$

then the roots to $q(x)$ can be found through the so-called companion matrix,

$$\begin{pmatrix} -a_2 & 1 & 0 \\ -a_1 & 0 & 1 \\ -a_0 & 0 & 0 \end{pmatrix}. \tag{3.4}$$

The eigenvalues of the companion matrix are equal to the zeros of (3.3). This is easily seen since the characteristic polynomial of (3.4) is equal to (3.3). This is well known and for example used by the Matlab command `roots` as it is a numerically stable way to find the roots.

The method of the companion matrix can be extended to the multivariate case [5, 28]. Consider again the system of polynomial equations in (3.2), and the corresponding ideal I and variety V. Since the dimension of the quotient space is equal to the number of solutions to the system, the quotient space forms a linear vector space with the same dimensionality as the number of solutions. Take $p \in \mathbb{C}[\mathbf{x}]$ and consider the operation $T_p : f(\mathbf{x}) \mapsto p(\mathbf{x})f(\mathbf{x})$. This operation is now linear and given some basis in $\mathbb{C}[\mathbf{x}]/I$ it can be represented with a matrix $\mathbf{m}_p$. This matrix is called the action matrix and is a generalization of the companion matrix.

The action matrix $\mathbf{m}_p$ has some nice properties. The eigenvalues of $\mathbf{m}_p$ are $p(\mathbf{x})$ evaluated at the points of V. Furthermore, the eigenvectors of $\mathbf{m}_p^T$ are equal to the vector of the basis elements of $\mathbb{C}[\mathbf{x}]/I$ evaluated on V. To understand this consider an arbitrary polynomial $f(\mathbf{x})$. It can be represented as $f(\mathbf{x}) = c^T\mathbf{b}(\mathbf{x})$, where c is a vector with coefficients and $\mathbf{b}$ a vector of monomials forming a basis of the quotient space. Further,

let $[\cdot]$ denote the reduction modulo I, then the following holds for any c,

$$[p \cdot c^T \mathbf{b}] = [(\mathbf{m}_p c)^T \mathbf{b}] = [c^T \mathbf{m}_p^T \mathbf{b}]. \tag{3.5}$$

Since it holds for any c it holds especially for $[p\mathbf{b}] = [\mathbf{m}_p^T \mathbf{b}]$. If this is evaluated where $\mathbf{x} \in V$ the brackets can be left out and the result becomes,

$$p(\mathbf{x})\mathbf{b}(\mathbf{x}) = \mathbf{m}_p^T \mathbf{b}(\mathbf{x}), \quad \text{for } \mathbf{x} \in V. \tag{3.6}$$

This is recognized as an eigenvalue problem for $\mathbf{m}_p^T$. Hence the eigenvalues of $\mathbf{m}_p^T$ are equal to $p(\mathbf{x})$ evaluated at the points of V and the eigenvectors are the basis elements of $\mathbf{b}$ evaluated at the zeros.

3.2 Gröbner Basis

The difficulties in solving systems of polynomial equations are now reduced to constructing the action matrix since the eigenvalue problem is well studied and several algorithms to solve the problem exist [13]. One way to find the action matrix is to select a linear basis $\mathcal{B}$ of $\mathbb{C}[\mathbf{x}]/I$ and for each element of $\mathcal{B}$ calculate $[p \cdot b_i]$ where $b_i \in \mathcal{B}$. To do these calculations it is necessary to be able to represent each equivalence class of $\mathbb{C}[\mathbf{x}]/I$ by a well defined representative.

One way to find these representatives is to use so called Gröbner bases [6]. The key idea is to use multivariate polynomial division. A Gröbner basis exists for each ideal and yields a well defined remainder for every polynomial. Well defined, means that for all $f_1, f_2 \in [f]$ the reduction with the Gröbner basis $\overline{f_1}^G = \overline{f_2}^G$. There is also an algorithm called Buchberger's algorithm [6] that is guaranteed to find this basis. This method unfortunately only works with exact arithmetics. The Gröbner basis is also dependent on a monomial order, which is needed to define a consistent division of multivariate polynomials.

The reason why Buchberger's algorithm only works with exact arithmetics is that in each step a check is done if a remainder is zero. With floating point arithmetics it is impossible to check this exactly since the calculations always are plagued by rounding off errors. There has been work to overcome this problem by Faugère [9]. The main idea of his work is to do many reductions in one step using well studied methods of numerical linear algebra. This is done by putting the system in (3.2) on the form,

$$C\mathbf{X} = 0, \tag{3.7}$$

where $\mathbf{X} = [\mathbf{x}^{\alpha_1}, \ldots, \mathbf{x}^{\alpha_n}]^T$ is a vector holding all occurring monomials and C is a coefficient matrix. With this notion all operations may be done on the coefficient matrix C and the numerics can be enhanced with different methods from numerical linear algebra [13].

This method of working with Gröbner basis methods has been adopted by the computer vision community and used to write special purpose solvers to several problems such as relative pose with two calibrated cameras [44], autocalibration with radial distortion [26], relative pose with unknown but common focal length [45], optimal three view triangulation [46], calculation of the fundamental matrix for a catadioptric camera [12] and more. Even though the method of Faugère improves the numerical stability it is not always enough. This becomes obvious in the problem of optimal three view triangulation where the authors were forced to use 128 bits mantissa in the floating point arithmetics to be able to get accurate results, which resulted in an extremely slow solver. The first paper of this thesis presents several methods to overcome this problem.

4 The Correspondence Problem

The correspondence problem between images is, given two or more images of the same scene, find a set of points which can be identified as images of the same points in different images. The most commonly used methods in the computer vision community to solve this problem is to use interest points. This means that points are located with some interest point detector, such as the Maximally Stable Extremal Regions (MSER) detector [31]. This method looks for regions that are consistent under large changes of intensity. These regions are then used as interest points, where the points are the centers of the regions. For each such point a descriptor is calculated that makes it possible to locate the point in a model or another image. The most used descriptor is the SIFT descriptor [30]. The SIFT descriptor is rotation invariant and moderately scale invariant. The SIFT descriptor looks at gradients in the region around an interest point and constructs a histogram over directions of the gradients. This information is stored in 128 element feature vectors that describes the point and are used for matching. For more information and comparison of interest point detectors and descriptors see [33, 34].

The correspondence problem can more generally be seen as the problem of finding matching points between two point sets, with a transformation mapping one of the point set to the other. Also this correspondence problem will be addressed in the thesis.

4.1 RANSAC

When the matching between a model and a query image has been done there will be a large number of incorrect matches, *outliers*, in a vast majority of the cases. One method to find and remove the outliers is to use the RANSAC method (RANdom SAmple Consensus). RANSAC was introduced by Fischler and Bolles [10]. The method takes a small random subset, usually minimal, of all correspondences and calculates a possible solution. The solution is then tested on all correspondences and the number of correspondences that agree with the solution is counted as inliers. This procedure is performed a large number of times and the inliers to the solution with most inliers are then considered to

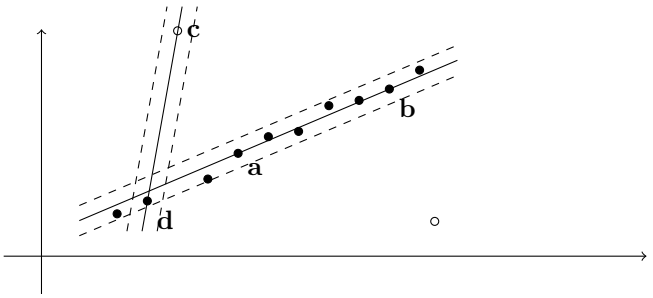

Figure 5: Estimation of a line using RANSAC: The open points are outliers and the solid points inliers. A random minimal set of two points is chosen and an exact fitting is made to this minimal set. One random set is $(\mathbf{a}, \mathbf{b})$ and the other $(\mathbf{c}, \mathbf{d})$. In both cases a margin is added (dashed lines) and the number of points that falls within the margin are counted as inliers. In this case, the $(\mathbf{a}, \mathbf{b})$ set gives ten inliers, whereas $(\mathbf{c}, \mathbf{d})$ only results in two. Hence the inlier set of $(\mathbf{a}, \mathbf{b})$ is chosen.

be true inliers. On this larger set it is possible to apply other methods in order to calculate a better solution, typically involving local non-linear optimization. In Figure 5 an example of line fitting to data with outliers using RANSAC is shown.

4.1.1 Minimal Cases

In RANSAC minimal cases are the standard method to find the solutions for a randomized set of correspondences. In Figure 5 the minimal set is two points to give a test line. In the pose problem this, of course, gets more complicated. Four of the most classical and most widely used minimal problems will now be described. These are the resection problem, finding the position of a camera from 2D projections of a 3D model, and the relative pose problem, finding the transformation between two cameras. Both these problems have been solved both for calibrated and uncalibrated cameras.

Uncalibrated Resection. The problem of camera resection is,

Problem 4.1. *Given a model with known feature points and their corresponding positions in three dimensions, find where a query image was taken and how the camera was orientated.*

If the inner calibration is unknown the camera matrix has eleven degrees of freedom, the twelve matrix elements minus one due to the unknown scale. Every point correspondence will give three equations by the projection formula (2.1), but also one new unknown, λ. So every new point correspondence gives two new constraints. This gives that six points are needed to calculate both the camera position and the inner calibration.

To solve the problem rewrite (2.1) as,

$$\lambda_j \mathbf{x}_j - P\mathbf{X}_j = 0. \tag{4.1}$$

Then collect all points correspondences into a common matrix,

$$A\hat{\mathbf{P}} = 0, \tag{4.2}$$

where $\hat{\mathbf{P}}$ holds all camera parameters and the point depths, λ_j, and A contains all measurements. The solution can be found by estimating the null space of A using *e.g.* singular value decomposition.

Calibrated Resection. If the camera is calibrated, *i.e.* K in equation (2.6) is known, it is enough with three known point correspondences, see [15, 16]. On the other hand, this does not give a single solution, instead it can give up to four possible solutions. To solve the problem a polynomial equation of degree four has to be solved. One way of solving this problem is based on the observation that the angle between any pair of image points, seen from the camera centre, must be the same as the angle between the corresponding world points. By that three different pairs of points can be used to construct second orders constraints expressed in the unknown point depths. These three quadratic polynomial equations can be combined to an eight degree polynomial equation using resultants [6]. In this polynomial the unknown parameter only appears in even powers, and since the unknown represents the depth, only positive values are valid. Hence up to four geometrical plausible solutions can exist.

Uncalibrated Relative Pose. Another set of minimal problems are those where the relative pose is calculated between two cameras with known image correspondences. Since the global coordinate system can be chosen in this case, one of the cameras can be placed in the origin and with rotation defined by the identity matrix. The problem of relative pose can then be formulated as follows:

Problem 4.2 (Relative Pose)**.** *Given two images, and a number of point correspondences, find the translation and rotation between the cameras used.*

If uncalibrated cameras are used, the problem was solved by Chasles [3] in 1855. This problem requires at least seven corresponding points and can give up to three possible solutions.

To solve it look at (2.11). In this equation every pair of corresponding points between the cameras gives one linear constraint on the fundamental matrix. By using singular value decomposition these seven equations gives a two dimensional null space, that includes the fundamental matrix. Due to the fact that the fundamental matrix only is determined up to scale F can be written,

$$F = \mu F_1 + F_2. \tag{4.3}$$

The last condition of F that will be used is that it has rank two, which implies that $\det(F) = 0$. Since F is a 3×3 matrix this gives an equation of degree three in μ, and hence three solution to the problem of relative motion. The second camera can then be retrieved from the fundamental matrix [19].

Calibrated Relative Pose. This problem was as mentioned before almost solved by Kruppa [25] in 1913 and corrected by Thomson in 1959 [47]. A fast and practical useable solution to the problem was given by Nistér [37] in 2004. To solve the problem five correspondences are needed. The first step of the solution mimics the uncalibrated case by using (2.12), and from that calculating a base for the null space using singular value decomposition. With the essential matrix written as,

$$E = xE_1 + yE_2 + zE_3 + wE_4, \tag{4.4}$$

where w can be fixed to one due to the scale ambiguity, the next step is to use the trace constraint (2.14), together with $\det(E) = 0$. This results in ten polynomial equations of degree three. To solve this system of polynomial equations Nistér [37] proposed to eliminate all but one variable, and by that get a ten degree polynomial to solve. Later it has been proposed [44] to use Gröbner basis techniques to solve the problem, and by that get an improved numerical stability.

4.2 Optimal Methods

RANSAC, and variants of it,[1] is widely used in the computer vision community. In most cases RANSAC works well but it has one fundamental limitation. Since it is a hypothesis and test algorithm only statistical estimates of the result can be given. In recent years an increased attention has been focused on optimal methods that overcome these problems. One branch of this research has focused on the L_∞-norm [23]. With use of the L_∞-norm instead of the more regularly used L_2-norm the optimization problems often shift from being non-convex to quasi-convex. The big advantage of quasi-convex problems compared to non-convex problem is that it can only exist one local minimum, and hence it is the global minimum. In the non-convex case, optimization algorithms can get stuck in a local minimum, and by that not find the global minimum.

One drawback of the L_∞ methods is that the result adjusts to the worst measurement, and hence is not robust to outliers. This, which at first might look like a severe problem, can actually be used as an advantage. In [42] Sim *et al.* prove that the set of measurements with the greatest residual must contain an outlier, if any present, for a large class of L_∞ problems. Thus, the simple method of just throwing the points in this set will lead to a removal of outliers. This method will not work in general for the L_2-norm.

These ideas have been used in several papers to remove outliers in geometrical correspondence problems. Li used it for triangulation in [29], and similar methods were used

[1] Such as: MEKSAC, PROSAC, NAPSAC, LO-RANSAC, and many more.

in [40]. In the later more complicated geometrical problems were solved, and instead of giving the algorithm a number of outliers to remove, a predefined threshold was used to define outliers.

In this thesis the L_∞-norm is also used to remove outliers. The problem to solve is:

Problem 4.3. *Given a tolerance ϵ and a set of hypothetical correspondences C, find the transformation T that maximizes $|I|$, where $I \subset C$ is the maximal set of correspondences consistent with T.*

A consistent set is then defined as:

Definition 4.1. *Given an error tolerance ϵ, we say that a set of correspondences C is consistent with a transformation, T, if any pair $(i, j) \in C$ satisfies $d(T(x_i), y_j) \leq \epsilon$ and if C is one-to-one.*

In Paper V of this thesis 3D-3D registration is studied where the allowed transformation, T, is a similarity transform and the error residual is the geometric distance. In that paper is also 2D-3D registration treated, where T is a rigid transformation followed by a perspective mapping. The error residual in the latter case is the angular reprojection error.

5 Graph Theory and the Vertex Cover Problem

A undirected graph G is defined as:

Definition 5.1 (Undirected Graph)**.** *Let V be a finite nonempty set, and let $E \subseteq V \times V$ where no concern is taken to the order. The pair $G = (V, E)$ is called an undirected graph, where V is the set of vertices and E is the set of edges.*

In this thesis methods from graph theory are used to solve the problem in the previous section. We will look at pairs of correspondences, to decide if those can be consistent under the same transform. If two correspondences are inconsistent with each other they cannot belong to the same solution. This can be represented by a graph, where each vertex is a hypothetical correspondence and edges are added between inconsistent pairs of correspondences.

With this formulation the problem of finding the largest possible set of inliers transforms to remove all edges from the graph by removing as few vertices as possible. This problem is known as the vertex cover problem and in its decision form it is one of Karp's 21 NP-complete problems [24] presented in 1972. To define the vertex cover problem more precisely we first define a vertex cover.

Definition 5.2 (Vertex Cover)**.** *A vertex cover for an undirected graph, is a subset S of its vertices such that every edge of the graph has at least one endpoint in S.*

With this definition of a vertex cover, the vertex cover problem is formulated as follows:

Problem 5.1. *The vertex cover problem is the problem of finding the smallest vertex cover for a given graph.*

One example of a minimal vertex cover is the set $\{b, e\}$ in the graph in Figure 6.

5.1 NP-Completeness

As said in the previous section the vertex cover problem is an NP-complete problem in its decision form.

Definition 5.3. *A problem having the following two properties is NP-complete:*

- *Any given solution to the problem can be verified in polynomial time,*
- *If the problem can be solved in polynomial time, then so can every problem in NP.*

If only the first property of the NP-complete problems holds the problem is in the class of NP (nondeterministic polynomial time) problems. The NP problems are decision problems where the solution can be verified in polynomial time.

One further, and important, characteristic of the NP-complete problems is that there exist no known solution that gives an answer in polynomial time. But there neither exist any proof that there do not exist such an algorithm. The question if there exist a polynomial bounded algorithm is known as the $P = NP$ problem, and is one of the Clay Institute seven millennium problems.[2] This problem was formulated by Cook [4] in 1971. The general assumption among researcher is that $P \neq NP$.

Since it does not exists any known polynomial time algorithm to solve the NP-complete problems, they are often considered hard to solve. This is in the meaning that they require lots of computational effort to solve, if the problem is large.

Many combinatorial optimization problems belong to the class of NP-complete problems. Probably the most known of these is the traveling salesman problem. This problem seeks the order to visits cities for a salesman in such a way that he visit all cities one time and returns to where he began, and he shall do this in a way such that the path traveled is as short as possible. This is also one of Karp's 21 problems, there known as the Undirected Hamilton Circuit problem.

5.2 Factor-2 Approximation

Since the vertex cover problem is NP-hard, approximation methods are needed. One such approximation is a factor-k approximation, that for a minimization problem guarantees to give a solution no worse then k times the optimal value. For the vertex cover problem

[2] http://www.claymath.org/millennium/

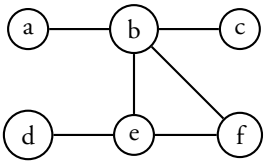

Figure 6: Example graph for vertex cover.

there is an easy to use, linear time, factor-2 approximation. The approximation algorithm works by randomly picking an edge in the graph and then putting the two vertices that the graph connects in the covering. For example, if in Figure 6, the edge between a and b is chosen these two vertices are added to the solution. After that the edge between d and e is randomly chosen, this gives us a covering consisting of $\{a, b, d, e\}$. This is exactly twice as big as the minimal vertex cover, $\{b, e\}$. Note that if the edge between b and e would have been randomly chosen, the approximation method would have given the optimal covering.

5.3 Greedy Algorithm for Vertex Cover

It is not only interesting to find a bound on the optimal solution, it is also interesting to find a good approximation to the best solution, even if it does not have a theoretical bound on how close to the optimal solution it will be. The method that will be proposed, and used in Papers V and VI, is well fitted to the problems that occur in localization. The vertex cover problems that will appear in the papers will all be used to remove outliers. Since the outliers have completely random geometrical positions the intuition is that they will be inconsistent with almost all other correspondences. This will have the consequence that the outlier correspondences will have the most number of edges connected to the node representing the correspondence. Founded in this intuition we propose Algorithm 1 to find a good approximation to the vertex cover problem. This algorithm will give a good approximation to the minimal vertex cover problem for the problems that occurs in this thesis.

Algorithm 1 Greedy Vertex Cover Approximation

1: Keep track of the number of edges to each vertex.
2: Iterate until no edges remain.
3: Select the vertex with most number of edges.
4: Remove selected vertex.
5: Remove affected edges and reduce the edge-count of all affected vertices.

6 Image Search

The last paper of the thesis is about large scale image based localization. This paper will use, and extend, techniques that have been developed in the image retrieval research; the problem of finding similar images to a query image. The most commonly used method in image retrieval is based on the concept of bag-of-words. This method tries to mimic the methods used in text retrieval [43].

The bag-of-words retrieval systems generally employ a number of standard steps, which are similar to those of text retrieval. These are:

1. Visual words are extracted from the documents (to search among).
2. Each document is represented by a vector with components given by the frequency of occurrence of the visual words the document contains.
3. The visual words are weighted depending on how common they are.
4. All the data is represented in an inverted file structure to achieve a fast search routine.
5. The query image is transformed to a vector of visual words and the images with the most similar feature vectors in the database are returned.
6. The top results of the previous step are re-ranked using more computation demanding algorithms, than in the previous step.

The first four of these steps can be made in advance which makes it possible to perform the retrieval step in around a second even for very large databases.

One problem though when the methods from text retrieval are adjusted to work for images is that there exists no obvious structure to substitute the words for. One way to overcome this problem is to use visual words. A visual word is usually derived from a SIFT feature. The SIFT features are mapped to visual words by a vocabulary. The vocabulary is constructed from a large set of training images where SIFT features are extracted and then clustered into different words. The number of clusters used, to achieve good search results in this method, is around one million words [38].

There also exist other methods of image search. One of the more popular among those is to use the GIST descriptor [39]. Instead of using several features to describe an image, this method uses a single feature to describe an image. After that only a nearest neighbor search is necessary to find similar images.

The last step in the image retrieval scheme presented above, the re-ranking, often uses geometric verification to find the best matches. One way to do this is to estimate a homography between the query image and the images in the database. The image that gets the highest number of inliers in the homography estimation is then ranked in the top. These type of calculations are relatively time consuming and in the latest years researchers

have started to look at how to include the geometry already in the earlier steps of the image retrieval procedure. One approach, presented in [21], is to encode weak geometric consistency into the inverted files. The weak geometric consistency favors that the scale and rotation of the feature points have been transformed consistently between the search image and the query image. Another way to include geometry was presented by Wu *et al*. in [48]. They bundled SIFT features together to enforce close placed features in the query image to also appear next to each other in the database images. The method uses MSER interest points to give an area where all SIFT features should be bundled together. By doing this, and also encode the geometry within the MSER regions, they improve image retrieval results significantly.

6.1 Solving Localization with Image Search

Image search methods have been used in localization in several articles. In [35] the GIST descriptor is used to find the most visually similar image with a known position. A similar approach is used in [41] but instead of the GIST descriptor the bag-of-words framework is used to perform the image retrieval.

A new approach to the localization problem, that uses an image retrieval step, was presented by Irschara *et al*. in [20]. Instead of looking at similar images with known position they created a three dimensional model of the area where the localization was performed. From the three dimensional model they were able to create synthetic views of the search area, so instead of searching for visually similar images they searched for matches within the set of synthetic views.

In Paper VI, a method is presented that is inspired by Irschara's method, and by the methods that include geometry in an early step of the search algorithm. The method creates synthetical views that represent an area of a city. In Paper VI such an area is eight times eight meters. For each such square is it calculated which points that are visible, but opposed to the paper by Irschara *et al*. also the angles between pairs of feature points are stored and used in the search step.

A re-rank method is also presented in the paper. This re-ranking uses pairwise consistencies in a similar way as in Paper V. So again the vertex cover problem needs to be solved, and the greedy method in Algorithm 1 is used. The presented re-ranking is fast and boosts the performance of the localization algorithm.

7 Summary of the Papers

PAPER I — Fast and Stable Polynomial Equation Solving and Its Application to Computer Vision

This paper addresses the problem of numerical instability when systems of polynomial equations are solved with Gröbner basis methods. This is a common problem; *e.g.* in [46] Stewénius *et al.* were forced to use emulated floating point numbers with higher accuracy than usual, which resulted in a very slow algorithm. Another example of numerical instability is the paper [27] where Kúkelová *et al.* were not able to construct a floating point solver due to numerical problems.

To overcome the numerical problems two new methods are presented, and a combination of these. The first method truncates the ideal and makes the basis over-redundant. This gives additional false solutions but we show that all true solution still appear. The other method used is a change of monomials used for the basis in the quotient space. This is also extended so that polynomials can be used in the basis instead of monomials. The combination of these methods is also used with an adaptive step deciding to what extent the truncation should be done.

The methods are tested on several recently reported problems involving a step where a system of polynomial equations was solved. On all these problems a significant increase in numerical stability is shown with a small extra cost in computational complexity.

Author contribution: This is joint work by myself, Martin Byröd and Kalle Åström. Martin acted as first author but I carried out most of the experiments and participated during the development of the new methods.

PAPER II — Image Based Localization Using Hybrid Features

In this paper the camera pose problem is considered. Several new minimal cases are studied with different degrees of camera calibration. The minimal cases are based on what we call hybrid features. By hybrid features we mean that both correspondence to other images and correspondences to known world points are used simultaneously. For most of these minimal cases upper bounds on the number of solutions are given. Most of these bounds are found by Gröbner basis methods using Macaulay 2 [14]. Some of the new minimal cases are also solved numerically by Gröbner basis techniques and in some cases the improved strategies of Paper I were necessary to achieve a solution.

Author contribution: This is a joint work by myself, Martin Byröd and our supervisors Fredrik Kahl and Kalle Åström. As first author I developed most of the theory and carried out the largest part of the experiments with aid primarily by Martin.

PAPER III — Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion

This paper is an extension to the paper by Kúkelová and Pajdla [27]. In their paper they presented the theory of two minimal problems of relative pose with radial distortion. The two problems are (i) simultaneous estimation of essential matrix and a common radial distortion parameter for two partially calibrated views and six image point correspondences and (ii) estimation of fundamental matrix and two different radial distortion parameters for two uncalibrated views and nine image point correspondences.

The method presented by Kúkelová and Pajdla only worked with exact arithmetics, and hence was not useable in practice. In this paper methods to achieve a practically applicable solver are presented that use regular floating point arithmetics. This required the use of methods from Paper I, but also other improvements from the original paper were necessary.

Author contribution: This is a joint work by myself, Martin Byröd, and Kalle Åström, as well as Zuzana Kúkelová and Tomas Pajdla from the Czech Technical University in Prague. I participated actively in the development of the new methods and implemented the new practical useable solvers. I also carried out parts of the experiments.

PAPER IV — Pose Estimation with Radial Distortion and Unknown Focal Length

This paper treats the problem of camera pose estimation. For calibrated cameras this can be solved with three known correspondences between an image and a three dimensional model. In this paper we instead use four correspondences. The extra correspondence is used to also be able to estimate the unknown focal length of the camera and a radial distortion. The paper shows that by making reasonable assumptions on the remaining inner camera parameters, it is possible to solve the camera pose estimation problem for uncalibrated cameras as well. The solution to the problem uses Gröbner basis methods developed in Paper I. To be able to use these methods the parametrization of the problem has to be chosen carefully, so the most important part of the paper is on how to parametrize the problem. The experiments of this paper show that this can be a practically useable method to estimate the position of the camera.

Author contribution: This is a joint work with Martin Byröd. I acted as first author and developed most of the theory and conducted most of the experiments.

PAPER V — Optimal Correspondences, Geometry and the Vertex Cover Problem

This paper presents a globally optimal method for two geometrical problems, the 3D-3D registration problem and the camera pose estimation of a calibrated camera. The methods can find the transformation that generates the maximum number of inliers for

a pre-defined error threshold. To achieve this, pairwise constraints are used to transform the problem into a graph problem. In the end the solution is found by solving the vertex cover problem on the graph. The vertex cover problem is NP-hard but we show that for the kind of graphs occurring for these geometrical problems, it can be efficiently solved.

Author contribution: This is a joint work by myself, Olof Enqvist, and our supervisor Fredrik Kahl. My main contribution to this paper is in the experimental section. I created a large part of the experiments and implementations of the methods, especially for the 3D-3D registration problem. I also actively participated in the development of the methods presented.

PAPER VI — City-Scale Localization Using Geometrical Visual Words

This paper treats the problem of finding the location of where an image was taken in a city. To make this possible a dataset of 95,000 images of Malmö is used to build a three dimensional point model of the city. With this model every point can be associated with an area where it is visible. This is used to create synthetical views for every eight by eight meters grid of the complete city. In this model, pair of features are used together with the angle between them to define a geometrical visual word. The use of visual words makes it possible to apply methods developed in the images retrieval research, to quickly find possible grid points to look in for the positions. The paper also presents a method to re-rank the top results of the first search step. This re-ranking procedure uses similar geometrical conditions as in the previous paper to set up a vertex cover problem, and by that finding consistent sets of correspondences.

Author contribution: This is a joint work by myself, Linus Svärm, Olof Enqvist and our supervisor Fredrik Kahl. I acted as first author and is responsible for most of the work. Linus Svärm added a great effort to the implementation and participated in many discussions of the methods. Also Olof Enqvist was active in many discussions and helped with the implementation of small but essential parts.

Bibliography

[1] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *National Conference on Artificial Intelligence (AAAI'98)*, Madison, Wisconsin, USA, 1998.

[2] 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.

[3] M. Chasles. Question 296. *Nouv. Ann. Math.*, 14(50), 1855.

[4] S. A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM.

[5] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer Verlag, 1998.

[6] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 2007.

[7] M. Demazure. Sur deux problemes de reconstruction. Technical Report 882, INRIA, Rocquencourt, France, 1988.

[8] O. Faugeras and S. Maybanks. Motion from point matches: multiplicity of solutions. *Int. Journal Computer vision*, 1990.

[9] 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.

[10] 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.

[11] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proc. of Computer Vision and Pattern Recognition Conference*, pages 125–132, 2001.

[12] C. Geyer and H. Stewénius. A nine-point algorithm for estimating para-catadioptric fundamental matrices. In *Proc. Conf. Computer Vision and Pattern Recognition, Minneapolis, USA*, June 2007.

[13] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[14] D. Grayson and M. Stillman. Macaulay 2. Available at http://www.math.uiuc.edu/Macaulay2/, 1993-2002. An open source computer algebra software.

[15] J. A. Grunert. Das pothenot'sche problem, in erweiterter gestalt; nebst bemerkungen über seine anwendung in der geodäsie. *Archiv der Mathematik und Physik*, 1:238–248, 1841.

[16] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions for the three point perspective pose estimation problem. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 592–598, 1991.

[17] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, pages 504–509, Washington DC, USA, 2004.

[18] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, 1997.

[19] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.

[20] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 2599–2606, 2009.

[21] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317, oct 2008.

[22] F. Kahl. Multiple view geometry and the L_∞-norm. In *International Conference on Computer Vision*, pages 1002–1009, Beijing, China, 2005.

[23] F. Kahl and R. Hartley. Multiple view geometry under the l-infinity norm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1603–1617, 2008.

[24] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations (Symposium Proceedings)*, 1972.

[25] 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.

[26] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *In Proc. Conf. Computer Vision and Pattern Recognition*, 2007.

[27] 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.

[28] D. Lazard. Resolution des systemes d'equations algebriques. *Theor. Comput. Sci.*, 15:77–110, 1981.

[29] H. Li. A practical algorithm for l_∞ triangulation with outliers. In *Proc. Conf. Computer Vision and Pattern Recognition*, Minneapolis, USA, 2007.

[30] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[31] 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.

[32] C. McGlone, E. Mikhail, and J. Bethel, editors. *Manual of Photogrammetry, 5th Edition*. 2004.

[33] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proc. Conf. Computer Vision and Pattern Recognition*, 2003.

[34] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.

[35] A. Murillo Arnal and J. Kosecka. Experiments in place recognition using gist panoramas. In *Proceedings of The Ninth Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*, 2009.

[36] P. Newman, J. J. Leonard, J. D. Tardos, and J. Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Proc. Int. Conf. on Robotics and Automation, Washington D.C., USA*, pages 1802–1809, 2002.

[37] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(6):756–770, June 2004.

[38] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, June 2006.

[39] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. In *Visual Perception, Progress in Brain Research*, 2006.

[40] C. Olsson, O. Enqvist, and F. Kahl. A polynomial-time bound for matching and registration with outliers. In *Proc. Conf. on Computer Vision and Pattern Recognition*, 2008.

[41] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proc. Conf. Computer Vision and Pattern Recognition, Minneapolis, USA*, pages 1–7, 2007.

[42] K. Sim and R. Hartley. Removing outliers using the L_∞-norm. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 485–492, New York City, USA, 2006.

[43] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, 2003.

[44] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

[45] H. Stewénius, 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.

[46] H. Stewénius, 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.

[47] E. H. Thompson. A rational algebraic formulation of the problem of relative orientation. *Photogrammetric Record*, 14(3):152–159, 1959.

[48] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *Proc. Conf. Computer Vision and Pattern Recognition, Miami, Florida, USA*, pages 25–32, 2009.

PAPER I

Published in *International Journal of Computer Vision*, 2008.

Fast and Stable Polynomial Equation Solving and Its Application to Computer Vision

Martin Byröd, Klas Josephson, Kalle Åström

Abstract

This paper presents several new results on techniques for solving systems of polynomial equations in computer vision. Gröbner basis techniques for equation solving have been applied successfully to several geometric computer vision problems. However, in many cases these methods are plagued by numerical problems. In this paper we derive a generalization of the Gröbner basis method for polynomial equation solving, which improves overall numerical stability. We show how the action matrix can be computed in the general setting of an arbitrary linear basis for $\mathbb{C}[\mathbf{x}]/I$. In particular, two improvements on the stability of the computations are made by studying how the linear basis for $\mathbb{C}[\mathbf{x}]/I$ should be selected. The first of these strategies utilizes QR factorization with column pivoting and the second is based on singular value decomposition (SVD). Moreover, it is shown how to improve stability further by an adaptive scheme for truncation of the Gröbner basis. These new techniques are studied on some of the latest reported uses of Gröbner basis methods in computer vision and we demonstrate dramatically improved numerical stability making it possible to solve a larger class of problems than previously possible.

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, *e.g*. [11, 29, 42]. Solutions to minimal structure and motion problems can often be used in RANSAC algorithms to find inliers in noisy data [17, 43, 44]. For such applications one needs to efficiently solve a large number of minimal structure and motion problems in order to find the best set of inliers. There is thus a need for fast and numerically stable algorithms for solving particular systems of polynomial equations.

Recent applications of polynomial techniques in computer vision include solving for fundamental and essential matrices and radial distortion [9], panoramic stitching [4] and pose with unknown focal length [5].

Another area of recent interest is global optimization used *e.g*. for 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_∞-norm methods [21, 26] and methods using linear matrix inequalities (LMIs) [27]. An alternative way to find the global optimum is to calculate stationary points directly, usually by solving some polynomial equation system [22, 41]. So far, this has been an approach of limited applicability since calculation of stationary points is numerically difficult for larger problems. By using the methods presented in this paper it is 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 [41]. We show that this problem can now be solved efficiently with an algorithm implemented in standard IEEE double precision.

Traditionally, researchers have hand-coded elimination schemes in order to solve systems of polynomial equations. Recently, however, new techniques based on algebraic geometry and numerical linear algebra have been used to find all solutions, *cf*. [37]. The 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 solution to the problem can be written down as a sequence of pre determined elimination steps using numerical linear algebra.

Currently, the limiting factor in using these methods for larger and more difficult cases is numerical problems. For example in [41] it was necessary to use emulated 128 bit numerics to make the system work, which made the implementation very slow. This paper 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 paper we pinpoint 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 in this paper 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 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. Note that this basis is not a true basis in the sense that it is linearly dependent.

Secondly, we show how a change of basis in the quotient space $\mathbb{C}[\mathbf{x}]/I$ can be used to improve the numerical precision of the Gröbner basis computations. This approach can be seen as an attempt at finding 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 a 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 [2], 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 somewhat 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.

The paper is organized as follows. After a brief discussion of related techniques for polynomial equation solving in Section 1.1, we give an overview of the classical theory of algebraic geometry underlying the ideas presented in this paper in Section 2. Thereafter, in Section 3, we present the theoretical underpinnings of the new numerical techniques introduced here. The main contributions in terms of numerical techniques are given in Sections 4, 5 and 6. In Section 7 we evaluate the speed and numerical stability of the proposed techniques on a range of previously solved and unsolved geometric computer vision problems and finally we give some concluding remarks.

1.1 Related Work

The area of polynomial equation solving is currently very active. See *e.g*. [10] 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 relies 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 [24].

In several variables a first method is to use resultants [13], 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. In some cases, a related technique known as the hidden variable method can be used to eliminate variables, see *e.g*. [33]. 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 [13]. This approach is however often numerically unstable.

A radically different approach is provided by homotopy continuation methods [45]. 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 time 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 large numbers of solutions (up to about a hundred) 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*. [37] even though Gröbner basis methods were mentioned in [23].

The work presented in this paper is based on preliminary results presented in [6, 7, 8] and essentially develops the action matrix method further to resolve numerical issues arising in the construction of the action matrix. Using the methods presented here, it is now possible to solve a larger class of problems than previously possible.

2 Review of 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 $\mathbf{x} = (x_1, \ldots, x_s)$, determine the complete set of solutions to

$$\begin{aligned} f_1(\mathbf{x}) &= 0 \\ &\vdots \\ f_m(\mathbf{x}) &= 0. \end{aligned} \tag{1}$$

We denote by V the zero set of (1). In general V need not be finite, but in this paper 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 [13] and [12] 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* $I = \{g \in \mathbb{C}[\mathbf{x}] : g = \Sigma_i h_i(\mathbf{x}) f_i(\mathbf{x})\}$, 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 (1). A point $\mathbf{x}$ is a zero of (1) iff it is a zero of I. Being even more general, we could ask for the complete set of polynomials vanishing 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. Further, we let $[\cdot]$ denote the natural projection $\mathbb{C}[\mathbf{x}] \to \mathbb{C}[\mathbf{x}]/I$, *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. 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 $r = |V|$.

2.1 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

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

the companion matrix is

$$\begin{bmatrix} -a_2 & 1 & 0 \\ -a_1 & 0 & 1 \\ -a_0 & 0 & 0 \end{bmatrix}. \tag{3}$$

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.

With some care, this technique can be extended to the multivariate case as well, which was first done by Lazard in 1981 [32]. 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 [13]. For some $p \in \mathbb{C}[\mathbf{x}]$ consider now the operation $T_p : f(\mathbf{x}) \rightarrow p(\mathbf{x})f(\mathbf{x})$. 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. 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

$$[p \cdot c^T\mathbf{b}] = [(\mathbf{m}_p c)^T\mathbf{b}] = [c^T\mathbf{m}_p^T\mathbf{b}]. \tag{4}$$

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

$$p(\bar{\mathbf{x}})\mathbf{b}(\bar{\mathbf{x}}) = \mathbf{m}_p^T\mathbf{b}(\bar{\mathbf{x}}), \tag{5}$$

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.

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 done by selecting a 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 [13]. 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 1. The following two equations describe the intersection of a line and a circle

$$\begin{aligned} x^2 + y^2 - 1 &= 0 \\ x - y &= 0. \end{aligned} \tag{6}$$

A Gröbner basis for this system is

$$\begin{aligned} y^2 - \tfrac{1}{2} &= 0 \\ x - y &= 0, \end{aligned} \tag{7}$$

from which we trivially see that the solutions are $\frac{1}{\sqrt{2}}(1,1)$ and $\frac{1}{\sqrt{2}}(-1,-1)$. 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

$$\mathbf{m}_x = \begin{bmatrix} 0 & 1 \\ \frac{1}{2} & 0 \end{bmatrix}, \tag{8}$$

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

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* and the latter as an *algebraic basis*.

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 [13]. 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 [16] is to write the system of equations on matrix form

$$\mathbf{CX} = 0, \tag{9}$$

where $\mathbf{X} = \begin{bmatrix} \mathbf{x}^{\boldsymbol{\alpha}_1} & \dots & \mathbf{x}^{\boldsymbol{\alpha}_n} \end{bmatrix}^T$ 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 (*e.g*. relative orientation for omnidirectional cameras [18], fundamental matrix estimation with radial distortion [30], optimal three view triangulation [41], etc.) and use knowledge of what the structure of the Gröbner basis should be to design a special purpose Gröbner basis solver [37]. 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 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 [41, 31]. 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 paper 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 core of 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 a general $f \in \mathbb{C}[\mathbf{x}]$. 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.

3 A New Approach to the Action Matrix Method

In this section 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. It should however be noted that a fundamental limitation still remains. As is the case with all other floating point methods, we do not present an algorithm which is guaranteed to work for all situations. To some extent, the method relies on heuristics and will only work if certain conditions can be fulfilled. As we show later, it does however expand the domain of application compared to previous methods. We start with a few examples that we will use to clarify these ideas.

Example 2. In the five point relative orientation problem for calibrated cameras, *cf*. [29, 14, 34, 38], 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 (9) and using a total degree ordering on the monomials we get a coefficient matrix $\mathbf{C}$ of size 10×20 and a monomial vector $\mathbf{X} = [\mathbf{x}^{\boldsymbol{\alpha}_1} \dots \mathbf{x}^{\boldsymbol{\alpha}_n}]^T$ with 20 monomials. It turns out that the first 10×10 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

$$\mathbf{X}_1 = -\mathbf{C}_1^{-1}\mathbf{C}_2\mathbf{X}_2. \tag{10}$$

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 (10). □

In this example the linear basis $\mathbf{X}_2$ was 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 again 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 $\overline{f}$ and (ii) it suffices to calculate $\overline{f}$ on the elements $x \cdot \mathbf{x}^{\alpha_i}$, *i.e.* we do not need to be able to compute $\overline{f}$ for all $f \in \mathbb{C}[\mathbf{x}]$. These statements and their implications will be made more precise further on.

Example 3. Consider the equations

$$\begin{aligned} f_1 &= xy + x - y - 1 = 0 \\ f_2 &= xy - x + y - 1 = 0, \end{aligned} \tag{11}$$

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 $T_y[1] = [y]$, $T_y[x] = [xy] = [x - y + 1]$ and $T_y[y] = [y^2] = [xy] = [x - y + 1]$ 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. □

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 $T_y(\mathcal{B})$ of the set $\mathcal{B}$ under T_y 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 3. 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 Solving Bases

We start off with a set of polynomial equations as in (1) and a (point) set of zeros $V(f_1, \ldots, f_m)$ and make the following definition.

Definition 1. 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

$$p(\mathbf{x})b_i(\mathbf{x}) = \Sigma_j m_{ij} b_j(\mathbf{x}) \tag{12}$$

for some (not necessarily unique) coefficients m_{ij} and $\mathbf{x} \in V$, then we call $\mathcal{B}$ a *solving basis* for (1) *w.r.t* p. □

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

Theorem 1. *Given a solving basis $\mathcal{B}$ for (1)* w.r.t *p, the evaluation of p on V is an eigenvalue of the matrix $\mathbf{m}_p$. Moreover, the vector $\mathbf{b} = (b_1, \ldots, b_r)^T$ evaluated on V is an eigenvector of $\mathbf{m}_p$.*

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

$$p(\mathbf{x})\mathbf{b}(\mathbf{x}) = \begin{bmatrix} pb_1 \\ \vdots \\ pb_r \end{bmatrix} = \begin{bmatrix} \Sigma_j m_{1j} b_j \\ \vdots \\ \Sigma_j m_{rj} b_j \end{bmatrix} = \mathbf{m}_p \mathbf{b}(\mathbf{x}) \tag{13}$$

for $\mathbf{x} \in V$. □

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

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

Theorem 2. *Let $\mathcal{B}$ be a complete solving basis for (1) with r elements and $\mathbf{m}_p$ as above and assume that for all eigenvalues λ_i we have $\lambda_i \neq \lambda_j$ for $i \neq j$. Then the complete set of solutions to (1) can be obtained from the set of eigenvectors $\{v_i\}$ of $\mathbf{m}_p$.*

Proof. Due to Theorem 1, we know that the vector of monomials $\mathbf{b}(\mathbf{x})$ evaluated on a point in the set of zeros V is an eigenvector of $\mathbf{m}_p$. The number of eigenvectors and eigenvalues of $\mathbf{m}_p$ is finite. This means that if we compute all eigenvectors $\{w_i\}$ of $\mathbf{m}_p$, then the set of vectors $\{\mathbf{b}(\bar{\mathbf{x}}) : \bar{\mathbf{x}} \in V\}$ must be a subset of $\{w_i\}$. We now consider $\mathbf{b}(\mathbf{x})$ as a mapping $\mathbf{b} : \mathbb{R}^s \mapsto \mathbb{R}^r$ and look at the inverse image $\mathbf{b}^{-1}(v_i)$. By the requirements, this is finite for all i and applying $\mathbf{b}^{-1}$ to all w_i thus yields a finite set of points which must contain V. Evaluation in (1) allows us to filter out the points of this set which are not in V (and hence not solutions to (1)). □

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 (1) as illustrated in the following example.

Example 4. Consider the polynomial system

$$\begin{aligned} y^2 - 2 &= 0 \\ x^2 - 1 &= 0 \end{aligned} \tag{14}$$

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

$$\mathbf{m}_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{15}$$

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 $\mathcal{B} = \{xy, x, y, 1\}$. □

At a first glance, Theorem 2 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 a set of polynomials. However, it is often practical to work with a basis of monomials. We get a particularly convenient situation if the coordinate variables x_i are included in $\mathcal{B}$ as seen in the following straight forward result:

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

Proof. Since the monomials $\{1, x_1, \ldots, x_s\}$ occur in $\mathcal{B}$, they enter in the vector $\mathbf{b}(x)$ and hence the mapping in Definition 2 is injective with a trivial inverse. □

This fact suggests that we should always try to include the coordinate variables in $\mathcal{B}$ to make for easy extraction of the final solutions. In practice, this is nearly always easy to do. And even if for some reason a few variables have to be left out, 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 2.

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 3. *If the ideal I generated by (1) is radical, then a complete solving basis $\mathcal{B}$* w.r.t *to p for (1) 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 $B = \big[\mathbf{b}(x_1), \ldots, \mathbf{b}(x_r)\big]$. 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 1, 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. □

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 transposed action matrix.

However, relaxing these restrictions we get something which is not necessarily a basis for $\mathbb{C}[\mathbf{x}]/I$ in the usual sense, but still serves our needs in terms of equation solving. More specifically, 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 2 still provides the right solutions. This fact is exploited in Section 4.

(ii) Typically, the arithmetics 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 5.1 and 5.2.

Finally, (i) and (ii) are combined in Section 5.3.

3.2 Solving Basis Computations using Numerical Linear Algebra

We now describe the most straight forward 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 4, we could multiply by *e.g.* $\{x, y, 1\}$, yielding $xy^2 - 2x, x^3 - x, y^3 - 2y, x^2y - y, y^2 - 2, x^2 - 1$. The general question of which and how many additional equations to generate is a tough one and there exist no watertight answers. This has to do with the fact that computing a Gröbner basis is an NP hard problem in general. The rule of thumb is to start out with a small set of equations and then sequentially adding more equations until the computations go through. However, some additional insight into the difficulty of the problem can be obtained by studying it symbolically using a computer algebra system, *e.g.* Macaulay2 [3].

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 $\mathcal{M} = \mathcal{E} \bigcup \mathcal{R} \bigcup \mathcal{B}$. The set $\mathcal{R} = x_k\mathcal{B} \setminus \mathcal{B}$ is the set of monomials that need to be expressed in terms of $\mathcal{B}$ to satisfy the definition of a solving basis and $\mathcal{E} = \mathcal{M} \setminus (\mathcal{R} \bigcup \mathcal{B})$ is the set of remaining (excessive) monomials. Each column in the coefficient matrix represents a monomial, so we reorder the columns and write

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_\mathcal{E} & \mathbf{C}_R & \mathbf{C}_B \end{bmatrix}, \tag{16}$$

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

$$\begin{bmatrix} \mathbf{U}_{\mathcal{E}1} & \mathbf{C}_{\mathcal{R}1} & \mathbf{C}_{\mathcal{B}1} \\ \mathbf{0} & \mathbf{C}_{\mathcal{R}2} & \mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_\mathcal{E} \\ \mathbf{X}_\mathcal{R} \\ \mathbf{X}_\mathcal{B} \end{bmatrix} = 0. \tag{17}$$

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

$$\begin{bmatrix} \mathbf{I} & \mathbf{C}_{\mathcal{R}2}^{-1}\mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_\mathcal{R} \\ \mathbf{X}_\mathcal{B} \end{bmatrix} = 0, \tag{18}$$

or equivalently

$$\mathbf{X}_\mathcal{R} = -\mathbf{C}_{\mathcal{R}2}^{-1}\mathbf{C}_{\mathcal{B}2}\mathbf{X}_\mathcal{B}, \tag{19}$$

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 (12). 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 (1) via eigendecomposition of $\mathbf{m}_{x_k}$. The above

method is summarized in Algorithm 1 and given the results of Section 3.1 we now have the following:

Result 1. *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 2 and Corollary 1 then guarantees the statement. □

So far, we have given general conditions for when a candidate basis $\mathcal{B}$ can be a solving basis, but we have not said anything about how a candidate basis can be chosen. A set $\mathcal{B}$ which is guaranteed to be a solving basis can be found by fixating a monomial order and then computing a Gröbner basis symbolically using a computer algebra system. One can then collect all monomials which are not divisible by any leading monomial in the Gröbner basis which yields a solving basis. However, any set which includes these monomials will also be a solving basis and in applications it turns out that a somewhat larger set $\mathcal{B}$ is often beneficial for numerical stability. A strategy which works well in practice is to start with the lowest order monomials and then sequentially add more monomials until enough have been added to make it a solving basis and to ensure numerical stability.

Example 5. Consider the equations from Example 1. Multiplying the second equation by x and y yields the enlarged system

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{bmatrix} = 0, \tag{20}$$

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 $\mathbf{C}_{\mathcal{R}2} = \left[\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ and $\mathbf{C}_{\mathcal{B}2} = \left[\begin{smallmatrix} 0 & -1 \\ -1 & 0 \end{smallmatrix}\right]$ and inserting into (19) we obtain

$$\begin{bmatrix} xy \\ x \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{2} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y \\ 1 \end{bmatrix}, \tag{21}$$

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

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

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

Require: List of equations $F = \{f_1, \ldots, f_m\}$, set of basis monomials $\mathcal{B}$ containing the coordinate variables $x_1, \ldots x_s$, 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** $\mathbf{x}^{\boldsymbol{\alpha}_j} \in L_i$ **do**
4: $F_{\text{ext}} \leftarrow F_{\text{ext}} \bigcup \{\mathbf{x}^{\boldsymbol{\alpha}_j} \cdot f_i\}$
5: **end for**
6: **end for**
7: Construct coefficient matrix $\mathbf{C}$ from F_{ext}.
8: $\mathcal{M} \leftarrow$ The set of all monomials occurring in F_{ext}.
9: $\mathcal{R} \leftarrow x_k \cdot \mathcal{B} \setminus \mathcal{B}$
10: $\mathcal{E} \leftarrow \mathcal{M} \setminus (\mathcal{R} \bigcup \mathcal{B})$
11: Reorder and partition $\mathbf{C}$: $\tilde{\mathbf{C}} = [\mathbf{C}_{\mathcal{E}}\ \mathbf{C}_R\ \mathbf{C}_B]$.
12: LU-factorize to obtain $\mathbf{C}_{\mathcal{R}2}$ and $\mathbf{C}_{\mathcal{B}2}$ as in (17).
13: Use (19) to express $x_k \cdot \mathbf{x}^{\boldsymbol{\alpha}_i}$ 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.

$\mathbf{m}_{x_k}$ for several k and then forming a random linear combination $\mathbf{m}_{a_1x_1+\cdots+a_sx_s} = a_1\mathbf{m}_{x_1} + \cdots + a_s\mathbf{m}_{x_s}$, 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 (19). 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 [41] to get a working algorithm.

In Sections 4 and 5 we will investigate techniques to circumvent this problem and produce well conditioned $\mathbf{C}_{\mathcal{R}2}$, thus drastically improving numerical stability. But first we will look at how to construct the action matrix given a solving basis for a polynomial system.

3.3 Constructing the Action Matrix

Given a solving basis $\mathcal{B}$ w.r.t. x_k and the corresponding vector $\mathbf{X}_{\mathcal{B}}$ for the polynomial system we wish to solve, it is easy to construct the corresponding action matrix. In short, this section fleshes out Step 13 of Algorithm 1. After performing Algorithm 1, the solving

basis property of $\mathcal{B}$ together with (19) guarantees $x_k\mathcal{B} \setminus \mathcal{B} = \mathcal{R}$ and the following:

$$\mathbf{X}_\mathcal{R} = A\mathbf{X}_\mathcal{B}, \tag{22}$$

where A is a matrix encoding this relation. To construct $\mathbf{m}_{x_k}$, we start with $\mathbf{m}_{x_k} = \mathbf{0}$ and sequentially go through all $b_i \in \mathcal{B}$. For each of them we perform one of two operations

1. If $x_k b_i = b_j \in \mathcal{B}$, then we simply set position j of column i of $\mathbf{m}_{x_k}$ to 1.
2. Else we have $x_k b_i = r_j \in \mathcal{R}$ and we need to use (22) to express r_j in terms of $\mathcal{B}$. This implies inserting $A^T_{j\cdot}$ (row j of A transposed) as column i in $\mathbf{m}_{x_k}$.

Hence, given A, constructing the action matrix is a computationally cheap operation.

4 Using Redundant Solving Bases - The Truncation Method

As mentioned in Section 3, the sub matrix $\mathbf{C}_{\mathcal{R}2}$ which appears in Equation 17 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 2, 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 see that $\mathcal{B}$ has to be a subset of the following set, which we denote the *permissible* monomials, $\mathcal{P} = \{b \in \mathcal{M} : pb \in \mathcal{M}\}$. The permissible monomials $\mathcal{P}$ is the set 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 [41]. 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 [41], 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,

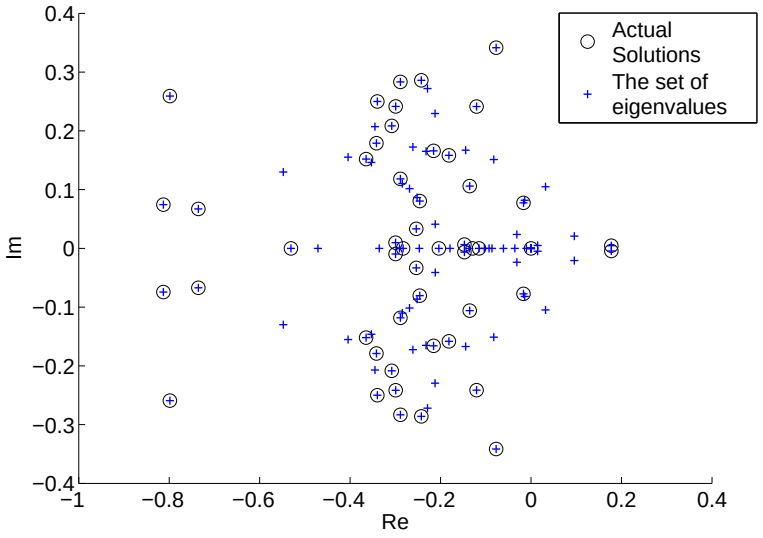

Figure 1: Eigenvalues of the action matrix using the redundant basis method and actual solutions to the polynomials system plotted in the complex number plane. The former are a strict superset of the latter.

the complete permissible set 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 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.

5 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 (1) 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 $\mathcal{P}$.

5.1 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 (17), 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

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1} \\ \mathbf{0} & \mathbf{C}_{\mathcal{P}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \tag{23}$$

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 Π of the columns of $\mathbf{C}_{\mathcal{P}2}$,

$$\mathbf{C}_{\mathcal{P}2}\Pi = \begin{bmatrix} c_{\pi(1)} & \dots & c_{\pi(p)} \end{bmatrix} \tag{24}$$

and then proceed using standard elimination. The goal must thus be to make this choice so as to minimize the condition number $\kappa(\begin{bmatrix} c_{\pi(1)} & \dots & c_{\pi(p-r)} \end{bmatrix})$ 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 [19]. With this algorithm, $\mathbf{C}_{\mathcal{P}2}$ is factorized as

$$\mathbf{C}_{\mathcal{P}2}\Pi = \mathbf{Q}\mathbf{U}, \tag{25}$$

where $\mathbf{Q}$ is orthogonal and $\mathbf{U}$ is upper triangular. By solving for $\mathbf{C}_{\mathcal{P}2}$ in (25) and substituting into (23) followed by multiplication from the left with $\left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^T \end{smallmatrix}\right]$ and from the right with $[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \Pi \end{smallmatrix}]$, we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1}\Pi \\ \mathbf{0} & \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \Pi^T\mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \tag{26}$$

We observe that $\mathbf{U}$ is in general not square write $\mathbf{U} = \begin{bmatrix}\mathbf{U}_{\mathcal{P}'2} & \mathbf{C}_{\mathcal{B}2}\end{bmatrix}$, where $\mathbf{U}_{\mathcal{P}'2}$ is square upper triangular. We also write $\mathbf{C}_{\mathcal{P}1}\Pi = \begin{bmatrix}\mathbf{C}_{\mathcal{P}'1} & \mathbf{C}_{\mathcal{B}1}\end{bmatrix}$ and $\Pi^T\mathbf{X}_{\mathcal{P}1} = \begin{bmatrix}\mathbf{X}_{\mathcal{P}'1} & \mathbf{X}_{\mathcal{B}}\end{bmatrix}^T$ yielding

$$\begin{bmatrix}\mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}'1} & \mathbf{C}_{\mathcal{B}1} \\ \mathbf{0} & \mathbf{U}_{\mathcal{P}'2} & \mathbf{C}_{\mathcal{B}2}\end{bmatrix}\begin{bmatrix}\mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \\ \mathbf{X}_{\mathcal{B}}\end{bmatrix} = 0. \tag{27}$$

Finally

$$\begin{bmatrix}\mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'}\end{bmatrix} = -\begin{bmatrix}\mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}'1} \\ \mathbf{0} & \mathbf{U}_{\mathcal{P}'2}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{C}_{\mathcal{B}1} \\ \mathbf{C}_{\mathcal{B}2}\end{bmatrix}\mathbf{X}_{\mathcal{B}} \tag{28}$$

is analogous to (19) and amounts to solving r upper triangular equation systems which can be efficiently done by back substitution. Given this relation we can now apply the recipe of Section 3.3 to construct the action matrix.

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 [2], 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. See [19] for more about QR factorization and column pivoting.

5.2 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 5.1 we start out by selecting an as large as possible (redundant) solving basis and perform preliminary matrix operations forming (23), 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

$$\mathbf{C}_{\mathcal{P}2} = \mathbf{U}\Sigma\mathbf{V}^T, \tag{29}$$

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

Now multiplying from the left with $\left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}^T \end{smallmatrix}\right]$ and from the right with $\left[\begin{smallmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{V} \end{smallmatrix}\right]$ in (23), we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}_1}\mathbf{V} \\ \mathbf{0} & \Sigma \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{V}^T\mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \tag{30}$$

The matrix $\mathbf{V}$ induces a change of basis in the space spanned by $\mathcal{P}$ and we write $\tilde{\mathbf{X}}_{\mathcal{P}} = \mathbf{V}^T\mathbf{X}_{\mathcal{P}} = \left[\begin{smallmatrix} \mathbf{X}_{\mathcal{P}'} & \mathbf{X}_{\mathcal{B}} \end{smallmatrix}\right]^T$, where $\mathcal{P}'$ and $\mathcal{B}$ are now sets of polynomials. Using this notation we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{0} & \mathbf{C}_{\mathcal{B}} \\ \mathbf{0} & \Sigma' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0, \tag{31}$$

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

$$\begin{cases} \mathbf{X}_{\mathcal{P}'} &= \mathbf{0} \\ \mathbf{X}_{\mathcal{R}} &= -\mathbf{U}_{\mathcal{R}}^{-1}\mathbf{C}_{\mathcal{B}}\mathbf{X}_{\mathcal{B}} \end{cases} \tag{32}$$

on V, which allows us to express any elements in $\mathrm{span}(\mathcal{M})$ 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 $\tilde{\mathcal{P}} = \mathcal{P}' \bigcup \mathcal{B}$ we assign a vector $v_k = \left[\begin{smallmatrix} 0 & \dots & 1 & \dots & 0 \end{smallmatrix}\right]^T \in \mathbb{R}^{|\tilde{\mathcal{P}}|}$, 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 $R : \mathrm{span}(\mathcal{M}) \to \mathrm{span}(\tilde{\mathcal{B}})$, which using (32) associates an element of $\mathrm{span}(\mathcal{M})$ with an element in $\mathrm{span}(\tilde{\mathcal{B}})$. We now represent R by a $|\mathcal{B}| \times |\mathcal{M}|$ matrix

$$\mathbf{R} = \begin{bmatrix} -\mathbf{C}_{\mathcal{B}}^T\mathbf{U}_{\mathcal{R}}^{-T} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{33}$$

acting on the space spanned by the vectors u_k.

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

$$(\mathbf{M}_p)_{ij} = I(x^{\alpha_i} = p \cdot x^{\alpha_j}), \tag{34}$$

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 a$\mathbf{V}$ we thus get

$$\tilde{\mathbf{M}}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}^T \end{bmatrix} \mathbf{M}_p\mathbf{V}. \tag{35}$$

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

$$\tilde{\mathbf{m}}_p = \mathbf{R}\tilde{\mathbf{M}}_p\mathbf{L}, \tag{36}$$

where $\mathbf{L} = \left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right]$ 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.

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 $\mathbf{m}_p^T$. To do this, we need to construct the change of basis matrix $\mathbf{V}_q$ in the quotient space. Using $\mathbf{R}$ and $\mathbf{L}$, we get $\mathbf{V}_q^{-1} = \mathbf{\Pi}\mathbf{V}^T\mathbf{L}$, where $\mathbf{\Pi}$ projects from $\mathbb{R}^{|\mathcal{P}'|}$ to $\mathbb{R}^{|\mathcal{B}|}$ using (32). From this we get $v = \mathbf{V}_q^{-T}\tilde{v}$ in our original basis.

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.

5.3 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 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*. [28].

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 $\mathrm{rank}(\mathbf{U}) = r$, then $|u_{rr}| > 0$ and $u_{r+1,r+1} = \cdots = u_{nn} = 0$. 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 τ 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[1]. 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 [2] 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.

[1]Performance is not very sensitive to the choice of τ and values in the range 10^6 to 10^{10} yield similar results.

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.

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 $i = \mathrm{rank}(\mathbf{C}_{\mathcal{P}2})$ is exactly the condition number of $\mathbf{C}_{\mathcal{P}2}$.

6 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 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

$$\mathbf{m}_{x_j}\mathbf{V} = \mathbf{V}\mathbf{D}_{x_j}. \tag{37}$$

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. The observations in this section also suggest a slightly easier way of filtering out false solutions obtained using the method in Section 4. If the coordinate variables x_i are present in the basis, they correspond to both eigenvalues *and* elements in the eigenvectors. Any discrepancy here implies a false solution which can immediately be discarded.

7 Experiments

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 this is an effect of the problem formulation and *not* of the particular equation solving technique.

In Section 7.1 all the main methods (the standard method, essentially what is outlined in Section 3.2, the truncation method, and the SVD and QR methods) are tested on optimal three view triangulation first studied by Stewénius *et al*. in [41]. They had to use emulated 128 bit arithmetics to get usable results, whereas with the techniques in this paper, we solve the equations in standard IEEE double precision. Furthermore, the improved methods are tested on: the problems of localization with hybrid features [25], relative pose with unknown but common focal length [39] and relative pose for generalized cameras [40]. Significant improvements in stability are shown in all cases. In the localization example we failed completely to solve the equations using previous methods and hence this case omits a comparison with previous methods.

7.1 Optimal Three View Triangulation

The triangulation problem is formulated as finding the world point that minimizes the sum of squares of the reprojection errors. This means that we are minimizing the likelihood function, thus obtaining a statistically optimal estimate given Gaussian noise. A solution to this problem was presented by Stewénius *et al*. in [41]. They solved the problem by computing the stationary points of the likelihood function which amounts to solving a system of polynomial equations. The calculations in [41] were conducted using emulated 128 bit arithmetics yielding very long computation times and in the conclusions the authors write that one goal of further work is to improve the numerical stability to be able to use standard IEEE double-precision (52 bit mantissa) and thereby increase the speed significantly. With the techniques presented in this paper it is shown that it is now possible to take the step to double-precision arithmetics.

To construct the solver for this example some changes in the algorithm of [41] were done to make better use of the changes of basis according to Section 5. The initial three equations are still the same as well as the first step of partial saturation (w.r.t. x). However, instead of proceeding to perform another step of partial saturation on the new ideal, we saturate (w.r.t. y and z respectively) from the initial three equations and join the three different partially saturated ideals. Finally, we discard the initial three equations and obtain totally nine equations.

This method does not give the same ideal as the one in [41] were $\mathrm{sat}(I, xyz)$ was used. The method in this paper produces an ideal of degree 61 instead of 47 as obtained by Stewénius *et al*. The difference is 11 solutions located at the origin and 3 solutions where one of the variables is zeros, this can be checked with Macaulay 2 [20]. The 11 solutions at the origin can be ignored in the calculations and the other three can easily be filtered out in a later stage.

To build the solver we use the nine equation from the saturated ideal (3 of degree 5 and 6 of degree 6) and multiply with x, y and z up to degree 9. This gives 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 and the code has been made available for download[2].

7.1.1 Numerical Experiments

The first experiment investigates what improvement can be achieved by simply avoiding the problematic matrix elimination using the techniques of Section 4. 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 154×154 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 2. As can be seen, this relatively straightforward technique already yields a large enough improvement in numerical stability to give the solver practical value.

Looking closely at Figure 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 5.1 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 3.

In Figure 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 ~ 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 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 coordinate magnitudes are around 1000 this means that we have a relative error $\approx 10^{-16}\kappa = \epsilon_{mach}\kappa$, where ϵ_{mach} is the machine precision. This observation justifies our strategy of minimizing the condition number.

[2] `http://www.maths.lth.se/vision/downloads`

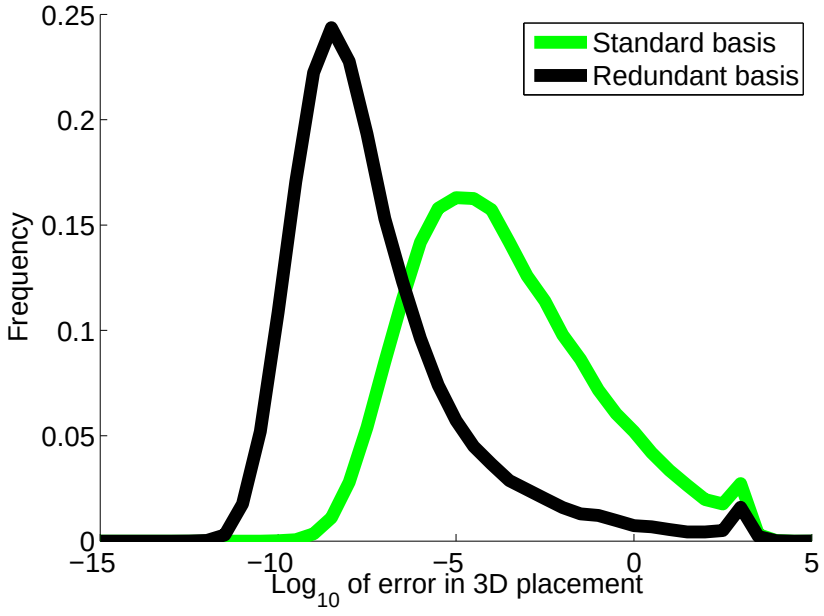

Figure 2: Histogram of errors over 100,000 points. The improvement in stability using the redundant basis renders the algorithm feasible in standard IEEE double precision.

As mentioned in Section 6, 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 50×50 that have to be solved. The impact of the switch from eigenvectors to eigenvalues can be seen in Figure 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.

However, we can use the trick of Section 6 to get nearly the same accuracy using only a single eigenvalue decomposition. Figure 7 shows the results of this method. The main advantage of using the eigenvalues is that we push down the number of large errors.

Finally we study the combination of basis selection and early stopping yielding a redundant Gröbner basis for the three view triangulation problem. The basis size was determined adaptively as described in Section 5.3 with a threshold $\tau = 10^8$. Table 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

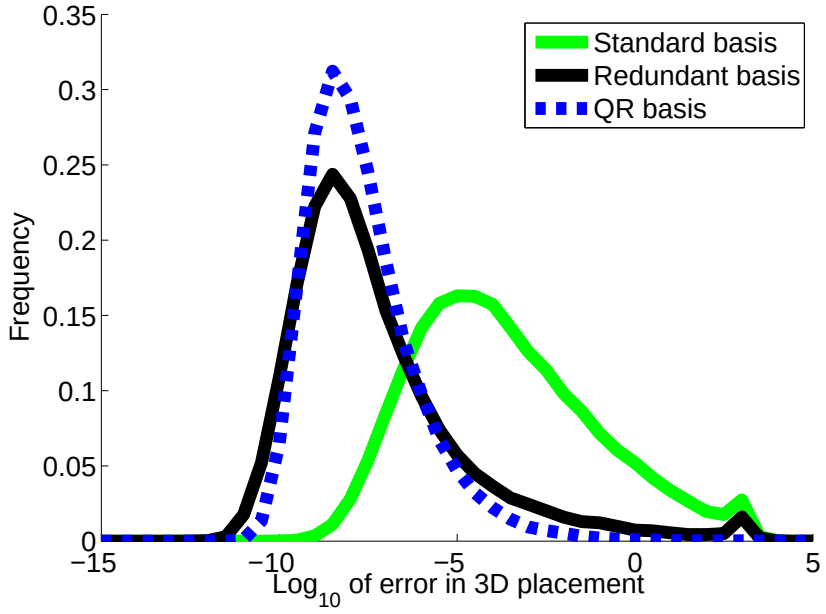

Figure 3: Histogram of errors for the standard, redundant basis and QR methods. The QR method improves stability in general and in particular completely removes the small set of large errors present in both the standard and redundant basis methods.

can be seen in Table 2 the number of large errors are reduced for the QR method. This is probably due to the fact that truncation is carried out only when the matrices are close to being singular. This effect is not present for the SVD method.

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 6) and the use of a variable size basis (Section 5.3). Table 2 shows how many times the error gets larger the some given levels for several solvers. 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 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.

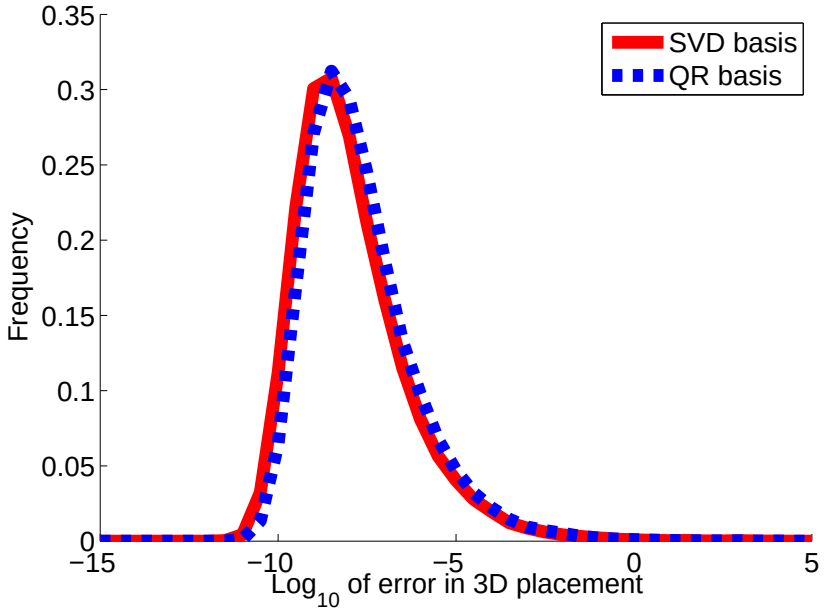

Figure 4: Comparison between the SVD and QR methods. The SVD method improves somewhat over the QR method at the cost of the computationally more demanding SVD factorization.

7.1.2 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. Our results show that the QR method with adaptive truncation is approximately four times faster than the

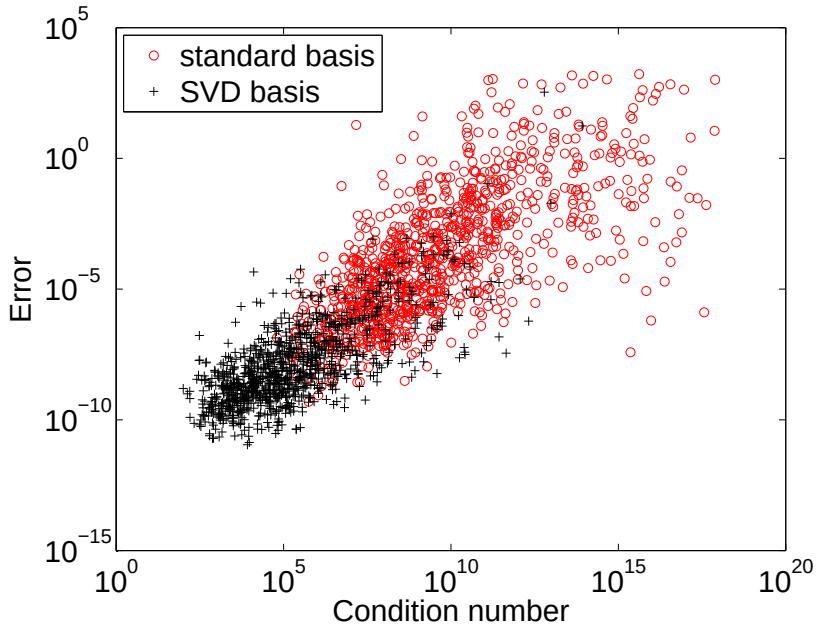

Figure 5: Error versus condition number for the part of the matrix which is inverted in the solution procedure.

SVD method but 40% 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.

7.1.3 Triangulation of Real Data

Finally, the algorithm is evaluated under real world conditions. The Oxford dinosaur [15] is a familiar image sequence of a toy dinosaur shot on a turn table. The image sequence consists of 36 images and 4983 point tracks. For each point visible in three or more

	50	51	52	53	54	≥ 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 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.

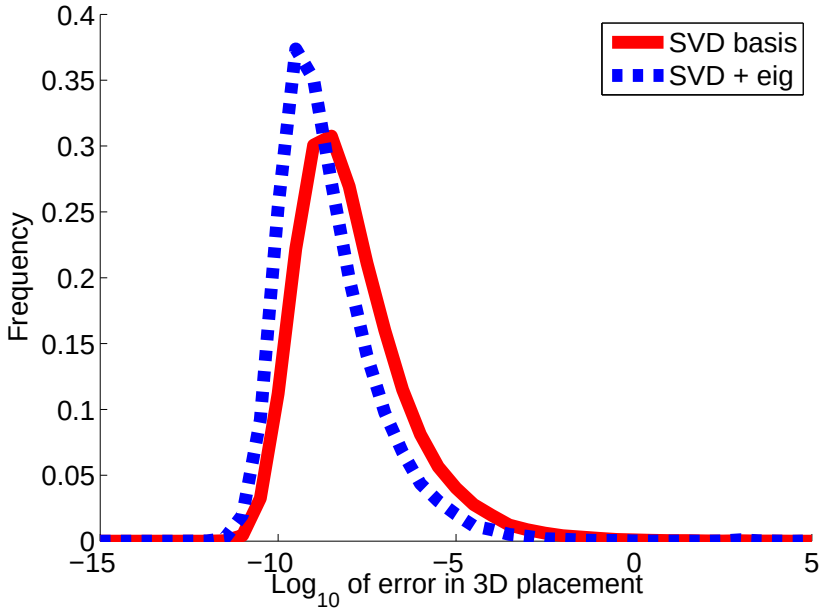

Figure 6: Error histograms showing the difference in precision between the eigenvalue and eigenvector methods.

views we select the first, middle and last view and triangulate using these. This yields a total of 2683 point triplets to triangulate. The image sequence contains some erroneous tracks that 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 45 seconds in a Matlab implementation on an Intel Core 2 2.13 GHz CPU with 2 GB of memory by the QR method with variable basis size. The resulting point cloud is shown in Figure 8.

We have also run the same sequence using the standard method, but the errors were to large to yield usable results (typically larger errors than the dimensions of the dinosaur itself).

7.2 Localization with Hybrid Features

In this experiment, we study the problem of finding the pose of a calibrated camera with unknown focal length. One minimal setup for this problem is three point-correspondences with known world points and one correspondence to a world line. The last feature is equivalent to having a point correspondence with another known calibrated camera. These types of mixed features are called hybrid features and were introduced in [25], where the authors propose a parameterization of the problem but no solution

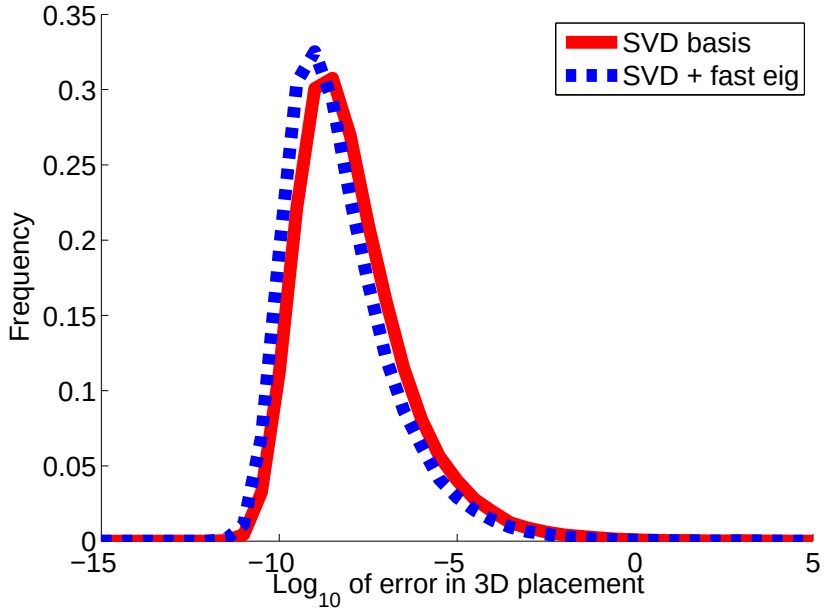

Figure 7: This graph shows the increase in performance when the fast eigenvalue method is used instead of the eigenvector method.

was given apart from showing that the problem has 36 solutions.

The parameterization in [25] gives four equations in four unknowns. The unknowns are 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.

The next step is to reorder the monomials as in (16). 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.

For this problem, we were not able to obtain a standard numerical solver. The reason for this was that even going to significantly higher degrees than mentioned above, we did

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 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 but the SVD method with eigenvalues is the best in general.

not obtain a numerical invertible $\mathbf{C}$ matrix. In fact, with an exact linear basis (same number of basis elements as solutions), even the QR and SVD methods failed and truncation had to be used.

In this example we found that increasing the linear basis of $\mathbb{C}[\mathbf{x}]/I$ by a few elements more than produced by the adaptive criterion was beneficial for the stability. In this experiment, an increase by three basis elements was used.

The synthetic experiments for this problem were generated by randomly drawing four points from a cube with side length 1000 centered at the origin and two cameras with a distance of approximately 1000 to the origin. One of these cameras was treated as unknown and one was used to get the camera to camera point correspondence. This gives one unknown camera with three point correspondences and one line correspondence. The experiment was run 10,000 times.

In Figure 9 the distribution of basis sizes is shown for the QR method. 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.

Figure 10 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.

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 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.

Method	Time per call / ms	Relative time
SVD	41.69	1
TRUNC	38.11	0.91
QR:var+fast	10.94	0.26
STD	8.03	0.19

Table 4: Time consumption in the solver part for four different methods. The time is an average over 1000 function calls.

7.3 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 [38]. 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 [39].

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

$$F = F_0 + F_1 l_1 + F_2 l_2. \tag{38}$$

Then putting $f^{-2} = p$ one obtains nine equations from the constraint on the essential matrix [35]

$$2EE^tE - \mathrm{tr}(EE^t)E = 0. \tag{39}$$

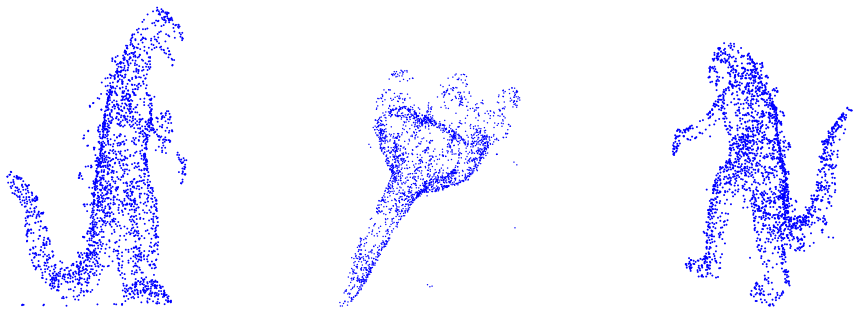

Figure 8: The Oxford dinosaur reconstructed from 2683 point triplets using the QR method with variable basis size. The reconstruction was completed in approximately 45 seconds by a Matlab implementation on an Intel Core 2 2.13 GHz CPU with 2 GB of memory.

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 11. 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.

7.4 Relative Pose for Generalized Camera

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*. [36]. 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 [40] 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 [40]. The problem has

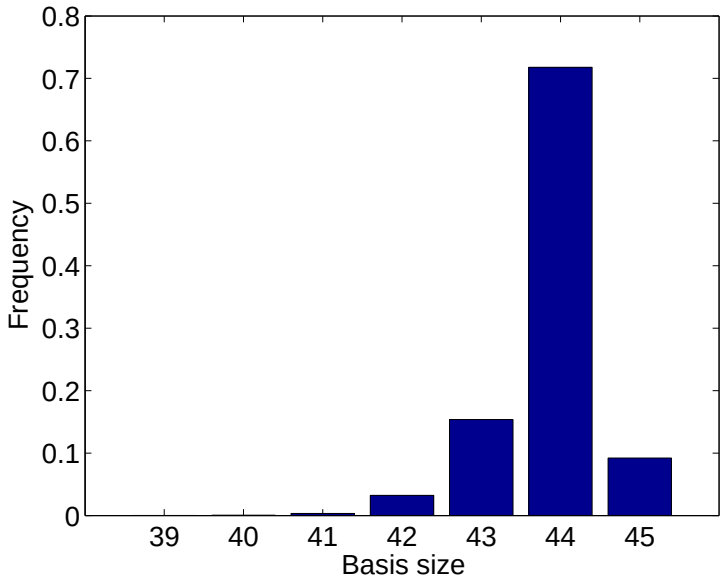

Figure 9: The basis size for localization with hybrid features. The number of solutions are 36 and since we always add three dimensions to the truncated ideal the minimal possible basis size is 39.

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 result from this experiment is shown in Figure 12. As can be seen, a good choice of basis yields drastically improved numerical precision over the standard method.

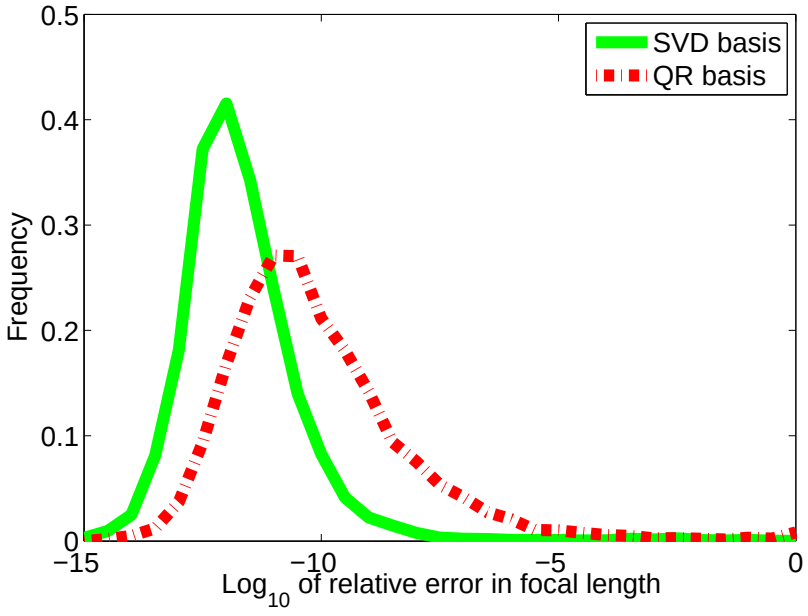

Figure 10: The error for localization with hybrid features. The standard method is omitted since we did not manage to construct a standard solver due to numerical problems.

8 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 technique for solving polynomial equations that we use in this paper 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 numerical linear algebra is used 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

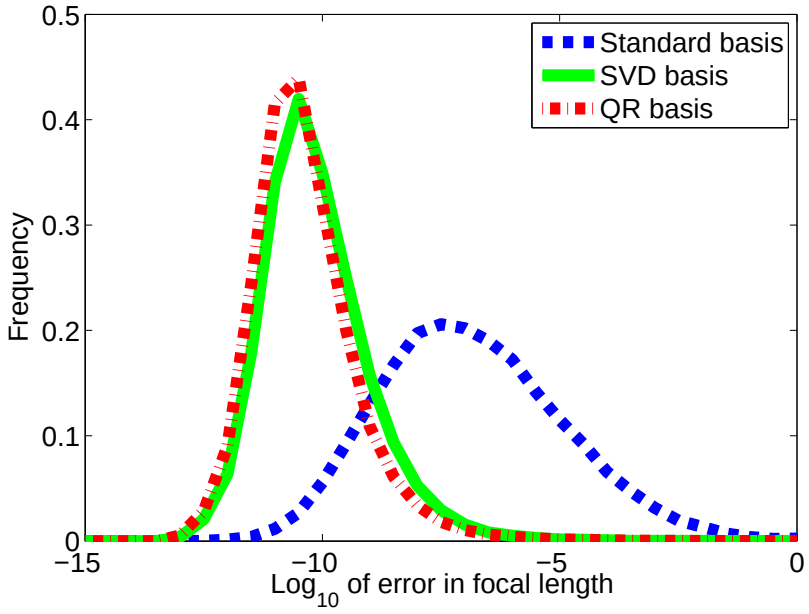

Figure 11: The error in focal length for relative pose with two semi calibrated cameras with unknown but common focal length.

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.

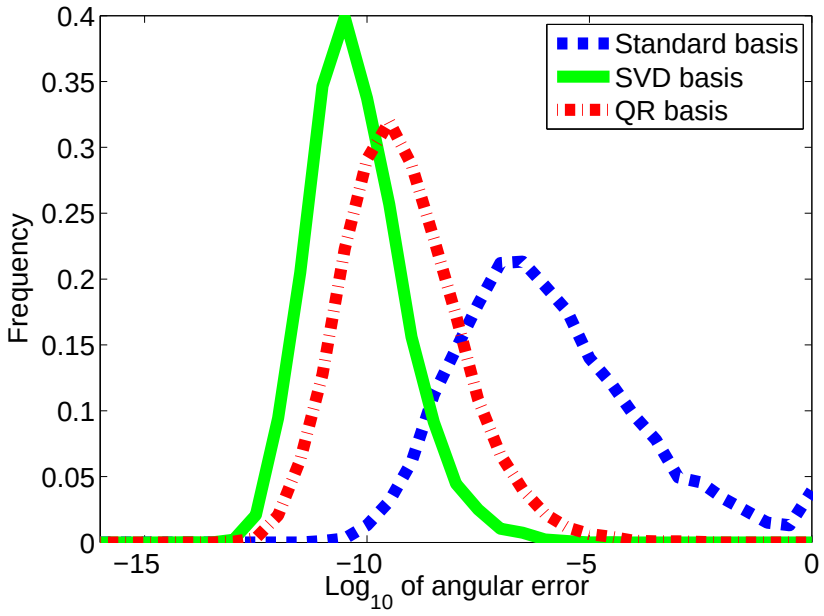

Figure 12: The angular error in degrees for relative pose with generalized cameras.

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] E. Anderson and et al. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[3] D. Bayer and M. Stillman. Macaulay. http://www.math.columbia.edu/∼bayer/Macaulay/, 1994.

[4] M. Brown, R. Hartley, and D. Nister. Minimal solutions for panoramic stitching. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR07)*, Minneapolis, June 2007.

[5] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.

[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. In *The 10th European Conference on Computer Vision*, 2008.

[9] M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.

[10] 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.

[11] M. Chasles. Question 296. *Nouv. Ann. Math.*, 14(50), 1855.

[12] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer Verlag, 1998.

[13] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 2007.

[14] M. Demazure. Sur deux problemes de reconstruction. Technical Report 882, INRIA, 1988.

[15] Visual geometry group, university of oxford. http://www.robots.ox.ac.uk/∼vgg.

[16] 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.

[17] 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.

[18] C. Geyer and H. Stewénius. A nine-point algorithm for estimating para-catadioptric fundamental matrices. In *Proc. Conf. Computer Vision and Pattern Recognition, Minneapolis, USA*, June 2007.

[19] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[20] D. Grayson and M. Stillman. Macaulay 2. Available at http://www.math.uiuc.edu/Macaulay2/, 1993-2002. An open source computer algebra software.

[21] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, pages 504–509, Washington DC, USA, 2004.

[22] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, 1997.

[23] R. Holt, R. Huang, and A. Netravali. Algebraic methods for image processing and computer vision. *IEEE Transactions on Image Processing*, 5:976–986, 1996.

[24] D. G. Hook and P. R. McAree. Using sturm sequences to bracket real roots of polynomial equations. *Graphics gems*, pages 416–422, 1990.

[25] 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.

[26] F. Kahl. Multiple view geometry and the l_∞-norm. In *ICCV*, pages 1002–1009, 2005.

[27] F. Kahl and D. Henrion. Globally optimal estimates for geometric reconstruction problems. In *Proc. 10th Int. Conf. on Computer Vision, Bejing, China*, pages 978–985, 2005.

[28] 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.

[29] 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.

[30] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *In Proc. Conf. Computer Vision and Pattern Recognition*, 2007.

[31] 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.

[32] D. Lazard. Resolution des systemes d'equations algebriques. *Theor. Comput. Sci.*, 15:77–110, 1981.

[33] H. Li. A simple solution to the two-view focal-length algorithm. In *Proc. 9th European Conf. on Computer Vision, Graz, Austria*, 2006.

[34] 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.

[35] J. Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *Photogrammetric Record*, 15(88):589–599, Oct. 1996.

[36] R. Pless. Using many cameras as one. In *Proc. Conf. Computer Vision and Pattern Recognition, Madison, USA*, 2003.

[37] H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, Apr. 2005.

[38] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

[39] H. Stewénius, 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.

[40] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision*, Beijing China, Oct. 2005.

[41] H. Stewénius, 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.

[42] E. H. Thompson. A rational algebraic formulation of the problem of relative orientation. *Photogrammetric Record*, 14(3):152–159, 1959.

[43] P. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15(8):591–605, 1997.

[44] 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.

[45] J. Verschelde. Phcpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25(2):251–276, 1999.

PAPER II

Submitted to *Journal of Mathematical Imaging and Vision*.

Image Based Localization Using Hybrid Features

Klas Josephson, Martin Byröd, Fredrik Kahl, Kalle Åström

Abstract

Where am I and what am I seeing? This is a classical vision problem and this paper presents a solution based on a combination of 2D and 3D features. Given a model of a scene, the objective is to find the relative camera location of a new input image. Unlike traditional hypothesize-and-test methods that try to estimate the unknown camera position based on 3D model features only, or alternatively, based on 2D model features only, we show that using a mixture of such features, that is, a hybrid correspondence set, may improve performance. We use minimal cases of structure-from-motion for hypothesis generation in a RANSAC engine. For this purpose, several new and useful minimal cases are derived for calibrated, semi-calibrated and uncalibrated settings. Based on algebraic geometry methods, we show how these minimal hybrid cases can be solved efficiently and in numerically sound way. The whole approach has been validated on both synthetic and real data, and we demonstrate improvements compared to the standard methodology. The software for solving hybrid geometry problems will be made publicly available.

1 Introduction

Localization refers to the ability of automatically inferring the pose and the position of an observer relative to a model, cf. [3]. We propose to solve the problem using an image-based approach. 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 [13, 19, 25], or by some other means.

The key idea of this paper is to use a mixture of 2D and 3D features simultaneously for localization. A 3D feature in this paper refers to a point with known location in the room with associated features. A 2D feature is a feature point detected in one view with known camera matrix, this gives a feature with the 3D position known up to a line in 3D. 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 [13], 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 combinations of 2D and 3D features, what we call 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 in a minimal hybrid set which is a good property when using RANSAC. One can argue that in most cases, traditional methods would work fine. However, we demonstrate that hybrid correspondence sets are indeed useful and there is simply no reason why this information should not be used as it leads to improvements.

The three main contributions of this paper are:
1. A complete list of minimal hybrid cases is given. We also list the number of possible solutions for each problem.
2. Algorithms for efficiently computing the solutions of the minimal problems using Gröbner bases are given for some cases. These are also tested on synthetical and real data.
3. We demonstrate how hybrid feature correspondences (combinations of 2D and 3D features) can be used for improved image-based localization.

1.1 Related Work

Localization and scene recognition are key components of any autonomous system. In robotics, (global) localization is also known as the *kidnapped robot problem*. Successful solutions have generally been achieved with laser, sonar or stereo vision range sensors and built maps for controlled robots moving in 2D, e.g., [21]. Another example is the Deutches Museum Bonn tour-guide robot RHINO [4] where laser sensors are used. Another competing technique (at least, for some applications) is GPS. However, the accuracy is typically only in the order of 10-20 meters and no direction information is obtained.

Image-based localization using special landmarks is a common approach, e.g., [2], but this severely limits the flexibility and the applicability of the method. Similar to

our approach, distinctive visual features were utilized in [23] to overcome this limitation. They also showed that RANSAC is an effective way of generating hypotheses. However, only 3D model features were used and this requires a rich 3D model to work well.

For large-scale models, an image search technique is required to speed up the process. This can be seen as a pre-processing step which produces a small number of hypothetical part-models that need further verification. Possible pre-processing schemes are developed in [22].

The wealth of research in the SfM field is, of course, related to the present work, in particular, the work concerned with RANSAC [31] and wide baseline matching [33, 20]. The same approach as proposed in this paper can be used to solve the wide baseline matching problem to build up 3D models [25].

Understanding of the geometry and the number of solutions of minimal structure and motion problems has a long history. For the uncalibrated case, the minimal problem of seven points in two views (which has three solutions) was studied and solved already in 1855, cf. [7]. The corresponding calibrated case was in principle solved in 1913 [15]. 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 [13, 32].

2 Problem Formulation

We are interested in solving the following problem:

Problem 1. 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 will use to solve the localization problem is based on hypothesize-and-test with RANSAC [10] and local invariant features [18]. This involves solving minimal structure and motion problem with hybrid correspondence sets. The underlying idea is as follows. Since exhaustive search for the largest, consistent subset among all possible correspondence sets is intractible, we will restrict our attention to minimal subsets. A minimal subset is, by definition, the smallest possible subset of correspondences such that a hypothetical solution can be generated. Given a hypothetical solution, one can test how many of the correspondences are consistent with this solution. Still, the number of minimal subsets abound, so only random subsets will be used for generating hypothetical solutions.

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 $\lambda_i u_i = PU_i$, $i = 1, 2, 3$. Here R is a 3×3 rotation matrix and t is a translation vector with 3-elements. Thus, the camera matrix encodes six degrees of freedom of unknown parameters. Each point gives two constrains and therefore three points form a minimal case. In general there are four possible solutions [11, 12, 13].

We will study the absolute orientation problem both for calibrated cameras as above, for the case of unknown focal length and for the uncalibrated camera case. As explained in the introduction we will also consider both 2D-2D correspondences (v_i, u_i) with features v_i in images of the model and known 3D-2D correspondences (U_i, u_i) as above. Here we will assume that the camera matrices of the model images 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 in space. In this paper **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 3D-2D correspondence imposes two constraints on the camera pose problem.

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 quaternions vector (a, b, c, d) for rotation, i.e.

$$P = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2ac+2bd & x \\ 2ad+2bc & a^2-b^2+c^2-d^2 & 2cd-2ab & y \\ 2bd-2ac & 2ab+2cd & a^2-b^2-c^2+d^2 & z \end{pmatrix}. \quad (1)$$

Potential minimal cases are:

The (0,3) case. This is the well-known resection problem, cf. [11, 12, 13] with up to four solutions in front of the camera.

The (2,2) case. This case is given a numerical solution in this paper. 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. There are two cases here. In the first case all 2D-2D correspondences are to the same view. In this first case the problem 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), cf. [15]. The scale is then fixed using the 2D-3D correspondence. The second case is when the 2D-2D correspondences are to at least two different views. This is studied in this paper and there are up to 32 solutions for this case. No numerical algorithm is presented.

The (6,0) case. This cannot be solved for absolute orientation if all points are from the same model view. In this case only relative orientation can be found and only five

correspondences are needed as discussed above. However, if the correspondences come from different views, it is in fact equivalent with the relative orientation problem for generalized cameras, cf. [29], 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

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

where f denotes the (inverted) focal length. Potential minimal cases are:

The (1,3) case. This case is given a numerical solution in this paper. There are up to 36 solutions.

The (3,2) case. This is studied in this paper and we have found the number of solutions to be up to 40 in the general case. No numerical algorithm is presented.

The (5,1) case. There are two cases here. In the first case all 2D-2D correspondences are to the same view. In this first case the problem can be solved by first projecting the 3D point in the known camera and then using the six point algorithm to solve for relative orientation and focal length [28]. The scale is then fixed using the 2D-3D correspondence. There are up to 15 solutions. The second case is when the 2D-2D correspondences are to at least two different views. This is studied in this paper and with the aid of Macaulay2, we have found that there are potentially up to 112 solutions. No numerical algorithm is presented.

The (7,0) case. This cannot be solved for absolute orientation if all points are to the same view. It can be solved for relative orientation using 6 points as discussed above. However for the case of correspondences to different views it should be solvable, but to our knowledge it is still an open problem, and we have not been able to obtain a solution with our approach.

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 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 3×4 matrices and ν 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 μ. The projection equation gives $\lambda u \times u = u \times PU = u \times (P_1 + \nu P_2)(U_1 + \mu U_2) = 0$. Thus we obtain two, second degree constraints in ν, μ. Using resultants, it follows that there are two solutions for the unknowns ν, μ.

The (3,4) case. There are two cases here. In the first case all 2D-2D correspondences are from the same view. In this first case the problem can be solved by first projecting the four 3D points in the known view and then using the seven point algorithm to solve for relative orientation [7]. There are then 3 solutions. The projective coordinate system is then fixed using the 2D-3D correspondences. The second case is when the 2D-2D correspondences are to at least two different views. In this case the solutions procedure is analogous to the previous case and can be obtained using resultants, this method shows that there can be up to eight solutions. No numerical algorithm is presented.

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. Relative orientation is still possible using at least seven points. The remaining case is when the 2D-2D correspondences are to at least two different views. Once again the methods of the (1,5) case can be used and there are up to $2^{(1+2k)}$ solutions. No numerical algorithm is presented.

Summary. We conclude this section by summarizing all the minimal cases for hybrid 2D and 3D feature correspondences, see Table 1. We state an upper bound on the number of physically realizable solutions. In general, as we shall see later in Section 6, the number of plausible solutions is much smaller. In the next section, we clarify the calculations for some of these cases. This will also lead to efficient algorithms for computing the solutions. Algorithms in matlab for solving the (2,2) and (1,3) cases, that later are evaluated in this paper, will be available for download on `http://www.maths.lth.se/vision/downloads/`.

4 Solving Minimal Cases with Algebraic Geometry

Minimal structure and motion problems typically boil down to solving a system of polynomial equations in a number of unknowns and in this section we describe how these types of polynomial problems can be solved with a combination of algebraic geometry and numerical linear algebra. We start by describing how to make symbolic calculations and after that the numerical method to find the solutions is presented.

4.1 Symbolic Calculations

For a specific application problem the structure of the polynomial system is fixed. Thus the number of solutions to a structure and motion problem typically depends only on the type of problem at hand. Common examples are relative orientation for calibrated cameras, five points in two views, which has ten solutions [27] and fundamental matrix estimation, seven points in two views, which has three solutions.

Solving systems of polynomial equations is a challenging numerical task and there is no general method of universal applicability as for linear equation systems. A first step towards solving a certain class of polynomial systems is to determine the number of

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 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 "?"). In the table italic is used to denote cases which are analogous to well-known structure and motion problems and bold face is used for cases that are solved in this paper.

solutions. There are several techniques for doing this. The theory of mixed volumes [8] can be used to prove the number of solutions for a set of polynomial equations assuming general coefficients of the polynomial. The software package `phc` [34] is useful both for calculating mixed volume and for finding solutions with homotopy methods. Another method is to calculate the so called Gröbner basis of the polynomial system which then easily yields the number of solutions to the problem. Gröbner bases are a concept within algebraic geometry, which is the general theory of multivariate polynomials over any field. See, for example, [8, 9] for a good introduction to the field. For problems that are not synthetic (where the coefficients are represented as floating point approximations) finding the Gröbner basis is numerically difficult due to accumulating round-off errors. One technique implemented in the computer algebra system Macaulay2 [1] is to work with integer coefficients and then projecting the equations from $\mathbb{C}[\mathbf{x}]$ to $\mathbb{Z}_p[\mathbf{x}]$, where p represents a large prime number typical 31.991, and computing the Gröbner basis there.

In [14] it is shown that an upper bound on the number of solutions to a problem can be found by solving a single instance of the problem. That is, if you find the solution to a problem for one instance that gives an upper bound on the number of solutions in the general case. There can still exist degenerate configurations with a higher number

of solution but the probability to end up in such a solution is very small for random coefficients. Based on this theoretical result, it is enough to work with integer coefficient in $\mathbb{Z}_p[\mathbf{x}]$ to determine the number of solutions.

The use of Gröbner basis calculations is not new in the computer vision community. It has been used for the minimal cases both in [27] and [29].

4.2 Numerical Calculations

As mentioned above, solving a system of polynomial equations is often algorithmically difficult and there exist no efficient general purpose methods. Instead, specialized solvers are developed for particular cases. Recently, progress has been made using Gröbner basis techniques [26, 5]. These solvers are known to suffer from numerical problems [30, 16] in some cases, but fortunately progress has been made on this point [6, 5]. Here, we make use of these advances to obtain efficient and robust solvers for the equations derived in the following sections. We choose not to give a self contained treatment of the polynomial techniques in this paper and this section should be read in conjunction with e.g., [26, 5] for the method to be fully repeatable.

The goal of the numerical calculations is to find the set of solutions to a system $p_1(\mathbf{x}) = \cdots = p_m(\mathbf{x}) = 0$ of m polynomial equations in s variables. The polynomials $p_1, \ldots, p_m$ generate an *ideal* I in $\mathbb{C}[\mathbf{x}]$, the ring of multivariate polynomials in $\mathbf{x} = (x_1, \ldots, x_s)$ over the complex numbers. The Gröbner basis method for polynomial solving is based on computing the eigenvalues of multiplication mappings in the quotient space $\mathbb{C}[\mathbf{x}]/I$ in analogy with the companion matrix method for one variable polynomials. One can view $\mathbb{C}[\mathbf{x}]/I$ as a vector space over the monomials that generate it. Then multiplication with a variable, x_i modulo I becomes a linear mapping in this space. Since the space is finite dimensional we can represent the mapping with a matrix $\mathbf{m}_{x_i}$. The eigenvalues of this matrix equal the values of x_i evaluated at the zeros of the system. The key steps in the procedure are to (i) expand the system of equations by multiplying with a problem dependent set of monomials and then write this on matrix form as a product between a coefficient matrix $\mathbf{C}$ and a vector of monomials $\mathbf{X}$, $\mathbf{CX} = 0$, and thereafter (ii) to put the system on reduced row echelon form using, for example, Gauss Jordan elimination which allows one to extract the necessary multiplication mappings.

5 Solving the Hybrid Minimal Cases

We will now go into the details of how the systems of equations are derived and how numerical Gröbner basis solvers can be constructed for the more interesting of the hybrid minimal problems studied in this paper.

5.1 Calibrated Cameras

A calibrated camera can be parameterized using quaternions as shown in (1). Assume now that we have two correspondences between image points and scene points

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

Since there is a freedom in choosing coordinate systems both in the scene and in the images, we can use this to simplify our expressions. In particular we let the origin in the coordinate frame coincide with U_1 and we let U_2 coincide with the point one measurement unit down the x axis. Together with similar choices in the image plane this yields

$$U_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \; u_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \; U_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \; u_2 = \begin{pmatrix} 1 \\ 0 \\ u \end{pmatrix},$$

which gives us the following constraints

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

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

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

By putting $a = 1$ two things happen. First the scale of the camera matrix is fixed, hence the left-hand 3×3 sub-matrix in (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. This may look like a serious problem but we argue that it is not for several reasons. If the solution is close to a point where $a = 0$, but not exactly, the solution will work since the fixed value on a will be compensated by large values of the other parameters of the camera. If the worst case happens that the solution has exactly $a = 0$, then the solver will not work, but since we propose to use the solver in a RANSAC engine where noise is present not all solutions for a minimal random set will have $a = 0$ exactly. Hence a slightly perturbed set of points is likely to exist that will give a solution with enough inliers to use in a final, local optimization step. It is also possible to derive a solution with the assumption $a = 0$, but we choose to not do that.

Now two correspondences remain between an image point in the query image and a point that has only been seen in one of the model images. Since the camera centre

is known for every image in the model each point in a model image can be associated with the line connecting the camera centre and the image point. This gives two image points associated to two lines l_1 and l_2 in 3D space. If the line is represented with Plücker coordinates [13] and the camera is converted to the corresponding Plücker camera, then each of the two constraints above converts to a single equation through

$$u_i^T P_L l_i = 0. \tag{3}$$

The two polynomial equations generated above are called p_1 and p_2. In the equation P_L is the Plücker camera that is constructed from the usual camera matrix [13]. Through inspection it is easy to see that the nonzero elements of P_L has the common factor $1+b^2$. After removing the common factor the two polynomial constraints (p_1, p_2) from (3) are of order 2 in b and order 4 in c.

By calculation in $\mathbb{Z}_p[\mathbf{x}]$ using Macaulay 2 [1] we find that for the general case the number of solutions to this problem is 16.

To construct the numerical solver we obtain 8 equations in 24 monomials by multiplying the polynomials with $\{1, b, c, bc\}$. 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 16×16 action matrix $\mathbf{m}_c$ for the linear mapping $\mathbb{C}[b, c]/I \ni p(c) \mapsto cp(c) \in \mathbb{C}[b, c]/I$. From the eigenvalue decomposition of the matrix $\mathbf{m}_c$ the 16 (some possibly complex) solutions are obtained.

To find the number of the solutions to the (4,1) case similar calculations are used. In this case only the first pair U_1, u_1 can be chosen as above. The other correspondences are to 2D features and are hence handled with Plücker lines as described above. The resulting four equations in four remaining unknowns, with the extra constraint that the $z \neq 0$, are then used to find the dimension of the quotient space. The constraint $z \neq 0$ forces the focal point of the camera to not be placed in the origin where we already have chosen to place one of the 3D points. Symbolic calculations with random instances of the problem in $\mathbb{Z}_p[\mathbf{x}]$ show that the (4,1) problem has 32 solutions.

5.2 Unknown Focal Length

For the case of unknown focal length we have one additional unknown. Thus we 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 [28] and then fix the scale using the known 3D correspondence.

Using (2) as parameterization for the camera matrix and assuming that two of the 3D point correspondences have coordinates

$$U_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, U_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, u_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \tag{4}$$

it is possible to eliminate $y = 0$ and $x = zf = g(b, c, d, f)$. We fix the scale by setting $a = 1$ with the same motivation as in the calibrated case. For both the $(1, 3)$ case and the $(3, 2)$ case we get polynomial constraints in the four remaining unknowns (b, c, d, f). Symbolic calculations with computer algebra software [1] in $\mathbb{Z}_p[\mathbf{x}]$, using the same methods as described in the calibrated case, show that there are 36 solutions for the $(1, 3)$ case, 40 solutions to the $(3, 2)$ case. In the $(5, 1)$ the same parameterization is used and the symbolic calculations then show that there can be up to 112 solutions in the case.

Constructing a Solver for the $(1, 3)$ Case

Using the parameterization above, one 3D correspondence and one 2D correspondence remain. The 2D correspondence is handled as in the $(2, 2)$ case by constructing the Plücker camera. This results in one equation. The other three required equations are generated from the last 3D point and from U_2. If U_2 is projected through the camera, two equations appear. By eliminating the depth λ one equation remains. The last two equations are generated through the last 3D point correspondence (U_3, u_3). Let $u_3 = (x_1, x_2, 1)^T$ and then construct the matrix,

$$[u_3]_\times = \begin{pmatrix} 0 & -1 & x_2 \\ -1 & 0 & x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}, \tag{5}$$

from which three equations can be generated by putting,

$$[u_3]_\times P U_3 = 0. \tag{6}$$

These three equations are linearly dependent and only the two first are kept.

The four constructed equations are multivariate polynomials where the equation constructed by the Plücker line will have degree 4 in (b, c, d) and degree 2 in f. The maximal total degree of the polynomials in this equation is 6. For the other three equations derived from 3D-2D correspondences the total degree is 3 and the maximal for each variable is 2 for (b, c, d) and 1 for f.

The next step in constructing the solver is to expand the coefficient matrix $\mathbf{C}$. At this stage the matrix has size 4×82. The expansion is made by adding the equations when the polynomial constructed by the Plücker line is multiplied with all monomials

up to degree 4 but with no single variable with a degree higher than 2. The other three equations are expanded with all monomials up to degree 7 but with no single variable with a degree higher than 5. After this expansion the coefficient matrix $\mathbf{C}$ is of size 980×873.

After $\mathbf{C}$ is expanded, we follow the method and notation from [5] and divide the monomials into the three sets $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{B}$. The basis $\mathcal{B}$ is selected to be all monomials of degree 4 and lower except for f^4. The set $\mathcal{R}$ contains the monomials which are not in $\mathcal{B}$ but are produced when the monomials of $\mathcal{B}$ are multiplied with b. The rest of the monomials are collected in the set $\mathcal{E}$. This partition gives 69 monomials in $\mathcal{B}$, 34 in $\mathcal{R}$ and the remaining 770 in $\mathcal{E}$.

We use LU-decomposition to eliminate monomials in the set $\mathcal{E}$, which gives a subset of equations, now with only 103 monomials. The (1,3) problem is numerically significantly more challenging than the (2,2) problem and to get a solution a combination of techniques for improving numerical stability introduced in [5] are used. The techniques are essentially to (i) use a matrix factorization algorithm to select a numerically well conditioned basis set $\mathcal{B}$ and (ii) to add some extra elements to $\mathcal{B}$ making it linearly dependent, but with the advantage that the added elements do not need to be expressed in terms of the basis since they are now a part of the basis.

6 Synthetic Experiments

6.1 The (2,2)-Solver

The purpose of this section is to evaluate the stability of the algorithm for solving the $(2, 2)$ case introduced in Section 4. 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 ± 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 tested on 10,000 randomly generated cases as described above. In the calculations, double precision arithmetics was used . In order to evaluate the accuracy of the solutions we take the square root of the sum of the for reprojection errors. The result is illustrated in Figure 1. As can be seen, the error typically stays as low as 10^{-15} to 10^{-10}, but occasionally much 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 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 that yield positive depths for all four problem points. In Figure 1 a

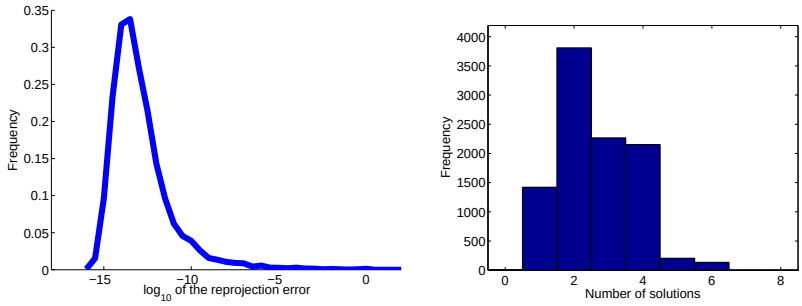

Figure 1: Statistics from the evaluation of the solver for the (2,2) case for calibrated cameras. The solver was run on 10,000 randomly generated cases. Left: Histogram over the error in matrix norm between the estimated camera P' and the true camera P when double precision arithmetics was used. The error is plotted on a logarithmic scale. Right: Histogram over the number of real valued solutions yielding positive depths.

histogram which shows the typical number of plausible solutions is given. As can be seen the most common situation is 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 in unfortunate positions (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.

The stability of the solver was also tested in the presence of noise on the image points. The noise is Gaussian distributed with standard deviation as given in the table and the error is measured as the mean of the reprojection errors. The result is shown in Table 2. Notice that both the median error and the $80th$ percentile error depends roughly linearly on the noise. This shows that the solver is robust to noise.

6.2 The (1,3)-Solver

The synthetic experiments for the (1,3) problem were done in the same manner as in the (2,2) case. Four points were generated by randomly drawing from a cube with side length 1000 centered at the origin and two cameras with a distance of approximately 1000 to the origin. One of these cameras was treated as unknown and one was used to get the camera to camera point correspondence. This gives one unknown camera with three point correspondences and one line correspondence. The experiment was repeated 10,000 times. As in the (2,2) case double precision arithmetics was used.

The result is shown in Figure 2. As can be seen, the error is not as low as in the (2,2)

Noise level	median	80 percentile
0	$3.9 \cdot 10^{-12}$	$4.0 \cdot 10^{-11}$
0.5	0.60	1.81
1.0	1.20	3.70
1.5	1.81	5.6
2.0	2.44	7.38
2.5	2.94	9.08
3.0	3.45	10.28

Table 2: Errors in the precence of noise on the measured points. The noise is the standard deviation in pixels and the errors are the mean reprojection for the four points in pixels. For each noise level, 10000 instances have been evaluated.

Noise level	median error	80 precentile
0	$3.46 \cdot 10^{-11}$	$1.51 \cdot 10^{-9}$
0.5	$1.06 \cdot 10^{-2}$	$4.87 \cdot 10^{-2}$
1.0	$2.10 \cdot 10^{-2}$	$8.64 \cdot 10^{-2}$
1.5	$2.78 \cdot 10^{-2}$	$1.41 \cdot 10^{-1}$
2.0	$4.18 \cdot 10^{-2}$	$1.80 \cdot 10^{-1}$
2.5	$5.81 \cdot 10^{-2}$	$3.14 \cdot 10^{-1}$
3.0	$6.60 \cdot 10^{-2}$	$3.40 \cdot 10^{-1}$

Table 3: Relative error in focal length for the (1,3)-solver. The noise value is the standard deviation of Gaussian noise in pixels on a 1000×1000 image. Every noise level has been ran 1000 times.

case but the numerical precision is still good enough and in most cases the error stays below 10^{-5}. The same figure also holds a histogram over the number of solutions that are real and with positive focal length. The histogram shows that even though there can be up to 36 real solutions in theory, the actual number of plausible solutions is usually below ten.

The stability in presence of noise of the solver was also tested and the results are shown in Table 3. The measured error is the relative error in focal length and the noise values is the standard deviation of a Gaussian distributed noise given approximately in pixels on a 1000×1000 image. The results show that the median value of the solutions are stable even under a large amount of noise. This is enough if the solver is used as the core of a RANSAC engine.

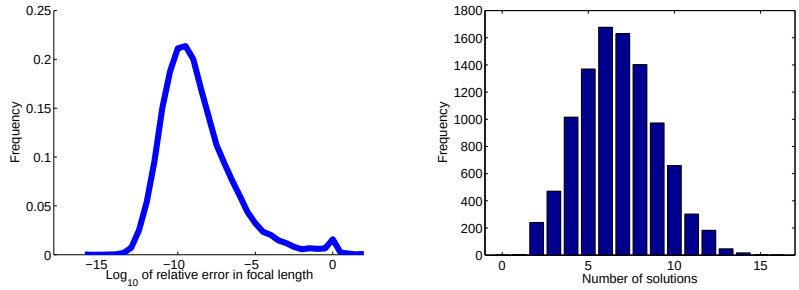

Figure 2: Statistics from the evaluation of the solver for the (1,3) case for calibrated cameras with unknown focal length. The solver was run on 10,000 randomly generated cases and double precision was used. Left: Histogram over the error in focal length between the estimated camera and the true camera. The error is plotted on a logarithmic scale. Right: Histogram over the number of real valued solutions with positive focal length.

7 Localization System Experiments

In this section we evaluate the proposed methods under real world conditions by integrating them in a complete localization system. For the data sets used in this section, we began by constructing a model using the methodology described in [25]. In that work, a 3D model consisting of cameras and triangulated 3D points is built. Here, we additionally keep all detected SIFT features from each image to increase the probability of being able to correctly localize a new input image. To this end we use the new hybrid minimal solvers to handle combinations of triangulated 3D points and pure image features in the RANSAC stage.

For the experiments data sets were used. One data set with pictures of a shopping street in Lund and the other was the Zürich building database [24]. For the experiments on the shopping street two models were constructed of the same area but with slightly different properties. The first model, called Model 1 in the following, was built from a sequence of only seven images. These images are taken along a street with approximately fifty meters baseline between the most separated views. The sparsity of this model makes it very challenging for a localization system. The second model, called Model 2, was constructed using thirty images but covers a much shorter distance. The maximal baseline between two images is approximately 10 meters. Examples of both images used to construct the model and test images are shown in Figure 3. The test images were taken approximately two months after the model images and with a different camera. Some statistics on the two models are given in Table 4.

Figure 3: To the left is one of the images used to construct the models and to the right is one of the test images. The test images were taken about two months after the model images. Different cameras were used when the model and test images were taken.

	Model 1	Model 2
Number of cameras	7	30
Number of SIFT features	18675	65385
Number of 3D points	593	3199
Number of 2D points	17097	55925

Table 4: Some data about the models. The SIFT features are the complete set of SIFT features from all images in the model, the 3D points are all triangulated point and the 2D points are all SIFT features which do not correspond to a triangulated 3D point.

The Zürich building database consists of 201 objects where each object has five training images taken with two different cameras. There are also 115 testing images of the objects in the training set but there is no location ground truth for these images. For neither the training images nor the test images the calibration is known. To construct the model, our assumptions were that there was no skew, aspect ratio equal to one, the principal point in the middle of the image and that the focal length was equal to 1000 pixels. The assumption on the focal length is not correct but to get a good estimate of the focal length the value was optimized during the bundle adjustment.

Using the method mentioned above, we managed to construct a 3D model for 173 of the examples in the Zürich database. Since the topic of this paper is not model construction we did not do any more work to construct models for the remaining buildings. This makes it impossible to correctly localize some of the test examples. In the constructed models 85% have less than 500 triangulated 3D points and about ten-thousand 2D points.

7.1 Localization Performance

Two of the new methods given in this paper are tested: the (2,2) solver for calibrated cameras and the (1,3) solver for calibrated cameras with unknown focal length. In the (2,2) case the number of inliers is measured after the RANSAC step has been carried out and is compared with the number of inliers when the three point solver is used in the same RANSAC loop. All located cameras were also manually examined, since we do not have any ground truth data, to decide whether the localization was correct.

In the (1,3) case more experiments are done. Due to the fact that the focal length is calculated this value can be verified. This is done by first calculating the calibration of the camera with more data than during the localization step. By then normalizing the coordinates we know that the focal length should be one since the inner calibration is removed.

The comparison for this method is made with uncalibrated cameras. To do that the principal point of the camera is assumed to be in the middle of the image, the skew is fixed to zero and the aspect ratio to one. These assumptions on the camera matrix are true for almost all digital cameras constructed today, even though the principal point may differ somewhat in location. Since the (1,3) solver is tested in uncalibrated images the solver is compared with a linear solver of six three dimensional points. This is not a minimal case, but it is probably the most commonly used method on uncalibrated images today.

The localization step is carried out as follows for all methods used. First the SIFT descriptors are calculated for the query image. After that, all these SIFT features are compared with those in the model to find the closest match. To make this more robust a distance ratio between the best and the second best match is used. The threshold for the ratio is fixed to 0.7 and the match is only accepted if the matching score for the first match is 0.7 better than the second match. For each established point track, we only match to the best point in the track.

This will give a set of matches between the query image and the model. Some of these matches are to 3D points and some correspond to 2D features without triangulated 3D points. These matches will also contain a significant number of outliers. To handle the outliers RANSAC is used with the proposed minimal solvers. When the number of inliers is counted during the RANSAC iterations the number of 2D correspondences is divided by 10. This is done to compensate for two sources, first the fact that the number of 2D points in the model is about ten times as large as the number of 3D points and secondly since the 3D points are first triangulated using the 2D points. The thresholds for the reprojection error in the RANSAC algorithm is fixed to 0.6% of the image size as in [25]. As a final step of the localization, bundle adjustment is performed on the camera position using the 3D inliers.

Figure 4: The reprojection of visible 3D points when the (2,2) solver (left) and the three point solver (right) are used. One out of four model points in Model 2 is reprojected and both solvers have resulted in a correct position.

	Model 1		Model 2		Models 1 & 2	
	Inliers 2D	Inliers 3D	Inliers 2D	Inliers 3D	Inliers 2D	Inliers 3D
(2,2) solver	9.83	4.60	7.63	9.41	8.68	7.11
(0,3) solver		4.72		9.88		7.42

Table 5: The mean number of inliers after 500 RANSAC iterations for the (2,2) solver and the three point solver. Model 1 is a sparse model and Model 2 much denser.

Test of the (2,2) Solver on the Shopping Street Dataset

The (2,2) solver is tested and compared with the three point solver [12]. Both these methods assumes calibrated cameras. Figure 4 shows an example of a test image which has been correctly positioned by both methods.

To evaluate the solver, all images that had three or more correspondences to 3D points are used. On these images, 500 RANSAC iterations were performed and the number of inliers was counted. For the (2,2) solver, both the numbers of 3D and 2D inliers are given. The result of this experiment is shown in Table 5. The three point solver gives a slightly higher number of 3D inliers. The fact that an incorrectly placed image usually gives the (2,2) solver two inliers but gives the three point solver three inliers is probably the reason that the three point solver has a slightly higher number of inliers.

Furthermore, the number of correctly located images was counted. For the (2,2) solver, the number of correctly placed cameras was 55 out of 65 images in Model 1 and the corresponding figure for the three point solver was 45. The total number of images, are as before those test images that had at least three 3D correspondences to the model. The images with fewer correspondences are not counted. In Model 2 the figures are closer. There were 54 images correctly placed with the (2,2) solver and 53 images with the three

	Inliers 2D	Inliers 3D
(2,2) solver	151.9	15.0
(0,3) solver		17.7

Table 6: The mean number of inliers after 500 RANSAC iterations for the (2,2) solver and the three point solver on the Zürich building database.

point method. The total number of images was 71. The reason why the performance differs more in the first model is probably due to the sparsity of that model. When the model is sparse the importance of also using the 2D correspondences increases.

Test of the (2,2) Solver on the Zürich Buildings Database

The same tests were run on the Zürich building database. As in the experiments on the shopping street the number of correctly placed images were counted. Since the calibration of the camera is unknown for these images the (1,3) method was first run and bundle adjustment was applied to the output of the (1,3) method. The calibration of the estimated camera from the bundle adjustment was then used as the calibration for the (2,2) and the three point solvers. The outcome of the experiment was that the (2,2) solver was able to place the camera approximately correct in 44 of 46 images where the (1,3) solver had given a reasonable result on the focal length and the 3 point solver gave good results in 42 of the cases. The reason why the (0,3) solver gives worse results than the (2,2) even though there should be at least three correct 3D points from the (1,3) solver is probably the fact that the (2,2) solver is helped by the 2D inliers when the size of the consensus set is counted in the RANSAC engine. Table 6 shows the number of inliers for the $(2, 2)$ and $(0, 3)$ solver respectively.

Test of the (1,3) Solver on the Shopping Street Data Set

If we assume nominal values for the principal point, skew and aspect ratio, then the (1,3) solver can be used on uncalibrated images. Making these assumptions, we compare the result with the six point linear solver for the uncalibrated case. The results of an image positioned by the (1,3) solver and the six point solver are shown in Figure 5.

As in the previous case, we used all images that had more than six 3D correspondences and hence could be used in both algorithms. On these images the 500 RANSAC iterations were performed and the number of inliers was counted. For the (1,3) solver both the numbers of 3D and 2D inliers are given. The result of this experiment is presented in Table 7. The results are very similar to those of the previous case.

The number of correctly located images was also counted, as in the previous case. For the (1,3) solver the number of correctly placed cameras was 22 in Model 1, out of 32 images. The six point method on the other hand only managed to locate 4 of these 32

Figure 5: The reprojection of visible 3D points when the (1,3) solver (left) and the six point solver (right) is used. In this example Model 1 is used and all points in front of the camera are reprojected. The (1,3) solver has a correct position but the six point solver has not.

	Model 1		Model 2		Models 1 & 2	
	Inliers 2D	Inliers 3D	Inliers 2D	Inliers 3D	Inliers 2D	Inliers 3D
(1,3) solver	14.7	6.52	9.84	11.36	11.42	9.78
(0,6) solver		7.37		12.20		10.63

Table 7: The number of inliers after 500 RANSAC iterations for the (1,3) solver and the six point solver. Model 1 is a sparse model and Model 2 much denser.

images correctly. In the second model, the (1,3) method succeeded in the localization in 41 out of 62 trials. On the same data, the six point solver managed to place 14 cameras correctly.

Test of the (1,3) Solver on the Zürich Buildings Database

When the number of correctly positioned cameras were counted the (1,3) solver managed to place 46 out of 86 images correctly compared with the six point solver that managed to place 32 of the images correctly. Once again the number of inliers were compared with the $(0, 6)$ method. The result of the experiment is shown in Table 8.

Determining the Focal Length

The (1,3) solver also estimates the focal length and this can be used to evaluate the solver since we can find the calibration of the camera using other data. The experiment is carried out as follows. The localization step is performed as usual but the calibration dependency is removed in the beginning by normalizing the image coordinates. The result should then be that the (1,3) solver returns a focal length equal to one. To test this every image

	Inliers 2D	Inliers 3D
(1,3) solver	131.3	26.0
(0,6) solver		33.4

Table 8: The mean number of inliers after 500 RANSAC iterations for the (1,3) solver and the six point solver on the Zürich building database.

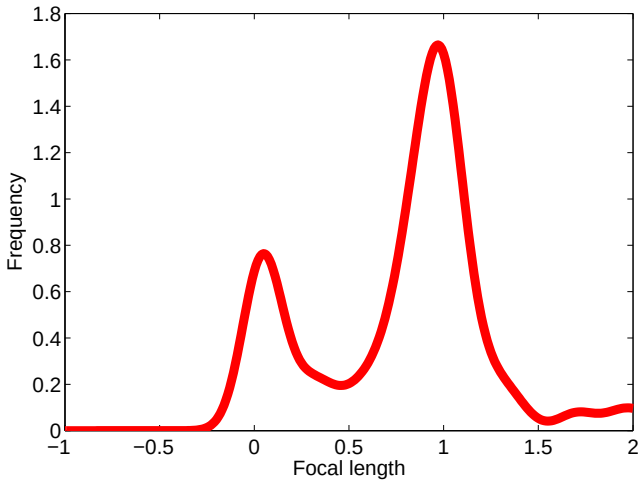

Figure 6: The result of the kernel voting for the focal length. The true focal length in this experiment is one. In the kernel voting only estimations with more the minimal number, which is three, of inliers are used in the voting process.

with enough of correspondences was localized. All images that got at least one extra inlier in 3D were then used in kernel voting [17] to estimate the focal length. The constraint that one extra inlier should appear is used to remove a large part of the miss placed images. The result of the kernel voting is shown in Figure 6. As can be seen in the figure, the main peak is localized close to one.

8 Conclusions

In this paper we have presented new minimal cases for the resection problem. These use a mixture of correspondences to known 3D points and correspondences to points that have only been found in one image in the model. In all except one of these minimal cases we have given an upper bound on the possible number of solutions with Gröbner basis

techniques. In two of the cases we have also presented and evaluated numerical solvers. The first of these cases is the (2,2) problem that finds the pose for a calibrated camera. The solution with Gröbner basis techniques leads to a very fast and numerically stable algorithm. We also present a solver for the (1,3) case for cameras with unknown focal length. This problem is significantly more complicated than the (2,2) problem but we can still present a numerically sound and computationally efficient algorithm with Gröbner basis methods. Both these methods are tested in a complete localization system and they are shown to improve on the standard methodology of today. The experiments show also that the (1,3) solver for pose and unknown focal length can be used on uncalibrated cameras under some reasonable assumptions. With these assumptions the solver shows promising results. The improvements are most significant on sparse models.

In the paper all methods are used separately. A convenient way to extend this work would be to use all methods simultaneously and make an automatic choice of which method to use depending on the data present. By that our new methods can help to improve the robustness of any localization system using hybrid cases.

Bibliography

[1] D. Bayer and M. Stillman. Macaulay. http://www.math.columbia.edu/∼bayer/Macaulay/, 1994.

[2] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Trans. on Robotics and Automation*, 13(2):251–262, 1997.

[3] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA, 1996.

[4] W. Burgard, A. Cremers, D. Fox, D. Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. The interactive museum tour-guide robot. In *National Conference on Artificial Intelligence (AAAI'98)*, Madison, Wisconsin, USA, 1998.

[5] M. Byröd, K. Josephson, and K. Åström. Fast and stable polynomial equation solving and its application to computer vision. *Int. Journal of Computer Vision*, 84(3):237–255, 2009.

[6] M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.

[7] M. Chasles. Question 296. *Nouv. Ann. Math.*, 14(50), 1855.

[8] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer Verlag, 1998.

[9] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 2007.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[11] J. A. Grunert. Das pothenot'sche problem, in erweiterter gestalt; nebst bemerkungen über seine anwendung in der geodäsie. *Archiv der Mathematik und Physik*, 1:238–248, 1841.

[12] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions for the three point perspective pose estimation problem. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 592–598, 1991.

[13] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.

[14] R. Holt, R. Huang, and A. Netravali. Algebraic methods for image processing and computer vision. *IEEE Transactions on Image Processing*, 5:976–986, 1996.

[15] 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.

[16] 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.

[17] H. Li and R. Hartley. A non-iterative method for lens distortion correction from point matches. In *Workshop on Omnidirectional Vision*, Beijing China, Oct. 2005.

[18] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal Computer Vision*, 60(2):91–110, 2004.

[19] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

[20] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conf.*, pages 384–393, Cardiff, UK, September 2002.

[21] P. Newman, J. Leonard, J. Neira, and J.Tardos. Explore and return: Experimental validation of real time concurrent mapping and localization. In *Int. Conf. Robotics and Automation*, pages 1802–1809, 2002.

[22] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, June 2006.

[23] S. Se, D. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375, 2005.

[24] H. Shao, T. Svoboda, and L. V. Gool. Zubud – zurich buildings database for image based recognition. Technical Report 260, Computer Vision Lab, Swiss Federal Institute of Technology, 2003.

[25] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2007.

[26] H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, Apr. 2005.

[27] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

[28] H. Stewenius, D. Nister, F. Kahl, and F. Schaffilitzky. A minimal solution for relative pose with unknown focal length. *Image and Vision Computing*, 26(7):871–877, 2008.

[29] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision*, Beijing China, Oct. 2005.

[30] H. Stewénius, 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.

[31] P. Torr and A. Zisserman. Robust computation and parametrization of multiple view relations. In *Int. Conf. Computer Vision*, pages 727–732, Mumbai, India, 1998.

[32] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. In *Proc. 7th Int. Conf. on Computer Vision, Kerkyra, Greece*, pages 278–284. IEEE Computer Society Press, 1999.

[33] T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. *Int. Journal Computer Vision*, 59(1):61–85, 2004.

[34] J. Verschelde. Algorithm 795: Phcpack: a general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.*, 25(2):251–276, 1999.

PAPER III

Published in *Computer Vision and Image Understanding*.

Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion

Zuzana Kúkelová, Martin Byröd, Klas Josephson,
Tomas Pajdla, Kalle Åström

Abstract

A number of minimal problems of structure from motion for cameras with radial distortion have recently been studied and solved in some cases. These problems are known to be numerically very challenging and in several cases there were no practical algorithm yielding solutions in floating point arithmetic. We make some crucial observations concerning the floating point implementation of Gröbner basis computations and use these new insights to formulate fast and stable algorithms for two minimal problems with radial distortion previously solved in exact rational arithmetic only: (i) simultaneous estimation of essential matrix and a common radial distortion parameter for two partially calibrated views and six image point correspondences and (ii) estimation of fundamental matrix and two different radial distortion parameters for two uncalibrated views and nine image point correspondences. We demonstrate on simulated and real experiments that these two problems can be efficiently solved in floating point arithmetic. For comparison we have also invented a new non-minimal algorithm for estimating fundamental matrix and two different radial distortion parameters for two uncalibrated views and twelve image point correspondences based on a generalized eigenvalue problem.

1 Introduction

Estimating camera motion and internal calibration parameters from sequences of images is a challenging computer vision problem with a broad range of applications [11]. One typically starts with a noisy set of tentative image point correspondences. The first step then is to make decisions about correct and incorrect matches and get a good initial estimate to be able to deploy a more sophisticated optimization algorithm on the set of all correct matches.

Two robust and widely used techniques for this purpose are RANSAC [8] and kernel voting [14], 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 vision problems with a minimal number of points. Typically this amounts to solving a system of polynomial equations in several variables. These problems are known to be numerically very challenging and in several cases there exist no practical algorithm yielding solutions in floating point arithmetic.

Traditionally, minimal problems have been formulated assuming a linear pin-hole camera model with different restrictions on the internal 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.

The particularly interesting solution to the simultaneous estimation of the fundamental matrix and single radial distortion parameter, based on the division model, has been introduced by Fitzgibbon [9]. His formulation leads to solving a system of algebraic equations. However, Fitzgibbon did not use the algebraic constraints on the fundamental matrix. Thanks to neglecting the constraints, he worked with a very special system of algebraic equations which can be solved numerically by using a quadratic eigenvalue solver. Micusik and Pajdla [16, 17] also neglected the constraints when formulating the estimation of a paracatadioptric camera model from image matches as a quartic eigenvalue problem.

Li and Hartley [14] treated the original Fitzgibbon's problem as a system of algebraic equations and used the hidden variable technique [4] to solve them. The resulting technique solves exactly the same problem as [9] but in a different way.

Solving for the fundamental matrix under different radial distortions was first studied in [2], where a *non-minimal* algorithm based on 15 point correspondences was given for a pair of uncalibrated cameras.

More recently, in [12, 13], a number of different *minimal* problems with radial distortion have been studied and practical solutions were given in some cases.

The state-of-the-art method for solving polynomial equations is based on calculations with Gröbner bases [18] and has many applications in computer vision, but also in other fields such as cryptology [7] and robotics [1]. In [21, 3] Gröbner bases were used to derive a fast algorithm for globally optimal three view triangulation under the L_2-norm.

In this paper, we further develop the techniques of numerical Gröbner basis computations. In particular we (i) note the importance of obtaining a single elimination step in the Gröbner basis computation, (ii) give guidelines for how this can be achieved and (iii) give a new simplified formulation of the Gröbner basis computation procedure based on LU factorization, which reduces the computational burden of the elimination step.

Leveraging on these new insights, we formulate fast and numerically stable algorithms for two minimal problems with radial distortion previously unsolved in floating point arithmetic: (i) simultaneous estimation of essential matrix and a common radial distortion parameter for two partially calibrated views and six image point correspondences and

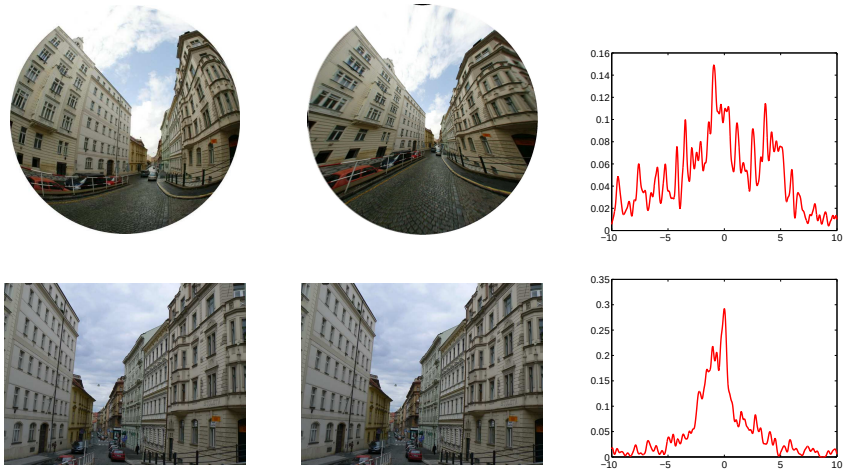

Figure 1: (Left) Input images with different radial distortions (Top) 66% cutout from omnidirectional image and (Bottom) image taken with a standard perspective camera with very mild distortion. (Center) Corrected images. (Right) Distribution of real roots obtained by kernel voting. Estimated $\lambda_1 = -0.925625$ and $\lambda_2 = 0.002500$.

(ii) estimation of fundamental matrix and two different radial distortion parameters for two uncalibrated views and nine image point correspondences.

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

We compare our minimal algorithms with the existing Fitzgibbon's non-minimal algorithm [9] for estimating fundamental matrix F and a single radial distortion from 9 point correspondences based on QEP, the Gröbner basis minimal algorithm [12] for a single radial distortion and 8 point correspondences, which uses $det(F) = 0$, a linear 12 point algorithm for estimating F and two different radial distortions and finally with the new non-minimal algorithm for two different radial distortions based on the generalized eigenvalue problem proposed in this paper.

2 Review of Gröbner Basis Techniques for Polynomial Equation Solving

Solving systems of polynomial equations is a challenging problem in many respects and there exist no practical numerically stable algorithms for the general case. Instead, special purpose algorithms need to be developed for specific applications. The state-of-the-art

tool for doing this is calculations with Gröbner bases [4].

Our general goal is to find the complete set of solutions to a system

$$f_1(\mathbf{x}) = 0, \ldots, f_m(\mathbf{x}) = 0, \tag{1}$$

of m polynomial equations in n variables $\mathbf{x} = (x_1, \ldots, x_n)$. The polynomials $f_1, \ldots, f_m$ generate an *ideal* I in $\mathbb{C}[\mathbf{x}]$, the ring of multivariate polynomials in $\mathbf{x}$ over the field of complex numbers defined as the set

$$I = \{g(\mathbf{x}) : g(\mathbf{x}) = \Sigma_k h_k(\mathbf{x}) f_k(\mathbf{x})\}, \tag{2}$$

where the $h_k(\mathbf{x})$ are any polynomials.

The Gröbner basis method for equation solving essentially builds on a generalization of polynomial division to the multivariate case. A concept arising in multivariate polynomial division which does not exist in the univariate case is division by a *set* of polynomials. See [4] for details. Division by an ideal as given by (2) can then be defined as division by the set of generators f_k.

The starting point now is to consider the space of all possible remainders under division by I. This space is denoted $\mathbb{C}[\mathbf{x}]/I$ and referred to as the *quotient space*. It can be seen as a generalization of the modulo rings $\mathbb{Z}_n$ to polynomials. A famous result from algebraic geometry now states that if the set of equations (1) has a finite set of zeros, then $\mathbb{C}[\mathbf{x}]/I$ will be a finite-dimensional linear space with dimension equal to the number of zeros of (1) [4].

With the space $\mathbb{C}[\mathbf{x}]/I$ in hand an elegant trick now yields the solutions to (1). Consider multiplication by one of the variables x_k. This generates a linear mapping from $\mathbb{C}[\mathbf{x}]/I$ to itself and since we are in a finite-dimensional space, by selecting an appropriate linear basis, this mapping can be represented as a matrix $\mathbf{m}_{x_k}$. This matrix is known as the *action matrix* and the eigenvalues of $\mathbf{m}_{x_k}$ are exactly the values of x_k on the zeros of (1) [4]. Furthermore, the eigenvectors of $\mathbf{m}_{x_k}^T$ correspond to the vector of monomials evaluated at the zeros of (1).

The crucial step in the process is to compute the remainder arithmetic of $\mathbb{C}[\mathbf{x}]/I$. Multivariate polynomial division by I is complicated by the fact that it is not well defined for most choices of generators. Consider the operator $\mathbf{P} : \mathbb{C}[\mathbf{x}] \to \mathbb{C}[\mathbf{x}]/I$ representing division by I for some choice of generators. For $\mathbf{P}$ to be well defined we require that $\mathbf{P}(f_1(\mathbf{x}) + f_2(\mathbf{x})) = \mathbf{P}f_1(\mathbf{x}) + \mathbf{P}f_2(\mathbf{x})$ for all $f_1(\mathbf{x}), f_2(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$.

Fortunately there exist a canonical choice of generators for which $\mathbf{P}$ is well defined. This set of generators of I is known as the Gröbner basis of I and allows direct construction of the action matrix, see [3] for details. Calculating the Gröbner basis of I is therefore our main concern. In general, this is accomplished by Buchberger's algorithm which works well in exact arithmetic. However, in floating point arithmetic it very easily becomes unstable. There exist some attempts to remedy this [5, 6], but for more difficult cases the only reliable approach (so far) is to study a particular class of equations (*e.g.*

relative orientation for calibrated cameras [19], optimal three view triangulation [21], etc.) and use knowledge of what the structure of the Gröbner basis should be to design a special purpose Gröbner basis solver. This method has been developed by Stewenius and others in a number of papers [18, 12, 20]. In the following section we outline how this is done and provide new insights enabling us to solve the two problems with radial distortion treated in this paper.

3 A Matrix Version of Buchberger's Algorithm

The reason why Buchberger's algorithm breaks down in floating point arithmetic is that eliminations of monomials are performed successively and this causes round-off errors to accumulate to the point where it is completely impossible to tell whether a certain coefficient should be zero or not. The idea introduced by Faugere [5] is to write the list of equations in a matrix form

$$\mathbf{C}\begin{bmatrix}\mathbf{x}^{\alpha_1}\\ \vdots \\ \mathbf{x}^{\alpha_n}\end{bmatrix} = 0, \tag{3}$$

where $\begin{bmatrix}\mathbf{x}^{\alpha_1} & \cdots & \mathbf{x}^{\alpha_n}\end{bmatrix}^T$ is a vector of monomials with the notation $\mathbf{x}^{\alpha_k} = x_1^{\alpha_{k1}} \cdots x_p^{\alpha_{kp}}$. 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.

However, as mentioned above, the real power of this approach is brought out by combining it with knowledge about a specific problem obtained in advance with a computer algebra system such as Macaulay2 [10]. One can then get information about exactly which monomials occur in Buchberger's algorithm and the dimension of $\mathbb{C}[\mathbf{x}]/I$.

3.1 Obtaining a Single Elimination Step

With knowledge of the particular problem at hand, the ideal situation is to obtain a single big elimination step. The reason for this is that each elimination step can be ill conditioned and with errors accumulating the situation soon becomes hopeless. With a single elimination step we get maximal control over row pivoting etc. Moreover, the basis selection method introduced in [3] can further improve stability, but is only applicable when a single elimination step is possible.

In Buchberger's Algorithm, two polynomials are picked and the least common multiple of their leading terms is eliminated by multiplying them with the right monomials and then subtracting them. This is done a large number of times until convergence. We mimick this process but aim at completely separating multiplication by monomials and elimination. The steps are

1. Multiply the original set of equations with a large number of monomials yielding an expanded set of equations.
2. Stack the coefficients of these equations in an expanded coefficient matrix $\mathbf{C}_{\text{exp}}$.
3. If enough new equations were generated in the previous step, row operations on $\mathbf{C}_{\text{exp}}$ yield the elements of the Gröbner basis we need.

An important observation made independently in [3] and [12] is that not all elements of the Gröbner basis are needed. Let $\mathcal{B}$ denote a selection of basis monomials for $\mathbb{C}[\mathbf{x}]/I$. Then to construct the action matrix $\mathbf{m}_{x_k}$ we only need to calculate the elements of the ideal I with leading monomials in the set $(x_k \cdot \mathcal{B}) \setminus \mathcal{B}$.

Let $\mathcal{M}$ denote the complete set of monomials and let $\mathcal{R} = (x_k \cdot \mathcal{B}) \setminus \mathcal{B}$ denote the set of monomials that need to be reduced to $\mathbb{C}[\mathbf{x}]/I$. Finally, let $\mathcal{E}$ ($\mathcal{E}$ for excessive) denote the remaining monomials. We then have a partitioning of the monomials as $\mathcal{M} = \mathcal{E} \bigcup \mathcal{R} \bigcup \mathcal{B}$.

Now, reorder the columns of $\mathbf{C}_{\text{exp}}$ and the vector of monomials $\mathbf{X}$ to reflect this

$$\begin{bmatrix}\mathbf{C}_{\mathcal{E}} & \mathbf{C}_{\mathcal{R}} & \mathbf{C}_{\mathcal{B}}\end{bmatrix} \begin{bmatrix}\mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}}\end{bmatrix} = 0. \tag{4}$$

The $\mathcal{E}$-monomials are not in the basis and do not need to be reduced so we eliminate them performing an LU factorization of $\mathbf{C}_{\text{exp}}$ yielding the following schematic result:

$$\begin{bmatrix}\mathbf{U}_{\mathcal{E}_1} & \mathbf{C}_{\mathcal{R}_1} & \mathbf{C}_{\mathcal{B}_1} \\ 0 & \mathbf{U}_{\mathcal{R}_2} & \mathbf{C}_{\mathcal{B}_2}\end{bmatrix} \begin{bmatrix}\mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}}\end{bmatrix} = 0, \tag{5}$$

where $\mathbf{U}_{\mathcal{E}_1}$ and $\mathbf{U}_{\mathcal{R}_2}$ are upper triangular. We can now discard the top rows of the coefficient matrix producing

$$\begin{bmatrix}\mathbf{U}_{\mathcal{R}_2} & \mathbf{C}_{\mathcal{B}_2}\end{bmatrix} \begin{bmatrix}\mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}}\end{bmatrix} = 0. \tag{6}$$

From this we see that if the submatrix $\mathbf{U}_{\mathcal{R}_2}$ is of full rank we get precisely the polynomials from the ideal I we need by forming

$$\begin{bmatrix}\mathbf{I} & \mathbf{U}_{\mathcal{R}_2}^{-1}\mathbf{C}_{\mathcal{B}_2}\end{bmatrix} \begin{bmatrix}\mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}}\end{bmatrix} = 0, \tag{7}$$

or equivalently

$$\boldsymbol{X}_{\mathcal{R}} = -\mathbf{U}_{\mathcal{R}_2}^{-1}\mathbf{C}_{\mathcal{B}_2}\boldsymbol{X}_{\mathcal{B}}, \tag{8}$$

which means that the $\mathcal{R}$-monomials can now be expressed uniquely in terms of the $\mathcal{B}$-monomials. This is precisely what we need to compute the action matrix $\mathbf{m}_{x_k}$ in $\mathbb{C}[\mathbf{x}]/I$. In other words, the property of $\mathbf{U}_{\mathcal{R}_2}$ as being of full rank is sufficient to get the part of the remainder arithmetic of $\mathbb{C}[\mathbf{x}]/I$ that we need to compute $\mathbf{m}_{x_k}$.

4 Application to Minimal Problems with Radial Distortion

Based on the techniques described in the previous section, we are now able to provide fast and stable algorithms for two previously untractable minimal problems with radial distortion:

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 [13] and found to be numerically very challenging. In [13] the authors provided 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.

4.1 Uncalibrated Case

In our solution we use the same formulation of the problem as in [13]. This formulation assumes a one-parameter division model [9] given by the formula

$$\mathbf{p}_u \sim \mathbf{p}_d/(1+\lambda r_d^2) \tag{9}$$

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.

It is known that to get solutions to this minimal problem for uncalibrated cameras with different radial distortions λ_1 and λ_2 in each image, we have to use the epipolar constraint for 9 point correspondences

$$\mathbf{p}_{u_i}^{\top}(\lambda_1)\,\mathtt{F}\,\mathbf{p}'_{u_i}(\lambda_2) = 0, \quad i = 1,\ldots,9 \tag{10}$$

and the singularity of the fundamental matrix $\mathtt{F}$

$$\det(\mathtt{F}) = 0. \tag{11}$$

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

Eliminating variables

The epipolar constraint gives 9 equations with 16 monomials ($f_{3,1}\lambda_1, f_{3,2}\lambda_1, f_{1,3}\lambda_2, f_{2,3}\lambda_2, \lambda_1\lambda_2, f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2, 1$) and 10 variables ($f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2$).

Among them, we have again four variables which appear in one monomial only ($f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}$) and four variables which appear in two monomials ($f_{1,3}, f_{2,3}, f_{3,1}, f_{3,2}$). Since we have 9 equations from the epipolar constraint we can use these equations to eliminate 6 variables, four variables which appear in one monomial only and two of the variables which appear in two monomials. In this solution we have selected $f_{1,3}$ and $f_{2,3}$.

We reorder the monomials contained in the 9 equations and put the monomials containing $f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}$ and $f_{2,3}$ at the beginning. The reordered monomial vector becomes $\mathbf{X} = [f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}\lambda_2, f_{1,3}, f_{2,3}\lambda_2, f_{2,3}, f_{3,1}\lambda_1, f_{3,2}\lambda_1, \lambda_1\lambda_2, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2, 1]^T$.

We rewrite the 9 equations from the epipolar constraint on matrix form $\mathbf{C}\,\mathbf{X} = 0$, where $\mathbf{C}$ is the coefficient matrix. After performing Gauss-Jordan (G-J) elimination on the matrix $\mathbf{C}$, we obtain 9 equations on the form

$$f_i = LT(f_i) + g_i(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) = 0, \tag{12}$$

where $LT(f_i) = f_{1,1}, f_{1,2}, f_{2,1}, f_{2,2}, f_{1,3}\lambda_2, f_{1,3}, f_{2,3}\lambda_2, f_{2,3}$ resp. $f_{3,1}\lambda_1$ for $i = 1, 2, 3, 4, 5, 6, 7, 8$ resp. 9 and $g_i(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)$ are 2^{nd}order polynomials in four variables $f_{3,1}, f_{3,2}, \lambda_1, \lambda_2$. This means that we can express the 6 variables, $f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}$ as functions of the other four variables $f_{3,1}, f_{3,2}, \lambda_1, \lambda_2$.

$$\begin{aligned}
f_{1,1} &= -g_1(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) \\
f_{1,2} &= -g_2(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) \\
f_{1,3} &= -g_6(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) \\
f_{2,1} &= -g_3(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) \\
f_{2,2} &= -g_4(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) \\
f_{2,3} &= -g_8(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2),
\end{aligned} \tag{13}$$

Substituting these expressions into the other three equations from the epipolar constraint and also into the singularity constraint for F gives 4 polynomial equations in 4 unknowns (one of 2^{nd} degree, two of 3^{rd} degree and one of 5^{th} degree)

$$\begin{aligned}
\lambda_2(-g_6(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)) + g_5(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) = 0 \\
\lambda_2(-g_8(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)) + g_7(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) = 0 \\
f_{3,1}\lambda_1 + g_9(f_{3,1}, f_{3,2}, \lambda_1, \lambda_2) = 0 \\
\det \begin{pmatrix} -g_1 & -g_2 & -g_6 \\ -g_3 & -g_4 & -g_8 \\ f_{3,1} & f_{3,2} & 1 \end{pmatrix} = 0.
\end{aligned} \tag{14}$$

This problem has 24 solutions in general [13].

The Solver

The numerical solver is constructed starting with the four remaining equations (14) in the four unknowns $f_{3,1}$, $f_{3,2}$, λ_1 and λ_2. The first step is to expand the number of equations, as outlined in Section 3, by multiplying them by a handcrafted set of monomials in the four unknowns yielding 393 equations in 390 monomials. See Section 4.1 for details.

The coefficients of the equations are then stacked in a matrix $\mathbf{C}$ as in (3). Following this, the monomials are ordered as in (4). 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 the "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 as mentioned in Section 3 we can use the method introduced in [3] 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 λ_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 $f_{3,1}$. $\mathcal{E}$ is the remaining set of 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 (5). We can then remove all equations that include excessive monomials and still have enough information to construct the action matrix.

Finally, we make the choice of representatives for $\mathbb{C}[\mathbf{x}]/I$ by the method in [3] and do the last elimination to get the part of the Gröbner basis we need to construct the action matrix.

Details on the Expansion Step for the Uncalibrated Case

We have found experimentally that to construct the necessary elements of the Gröbner basis, we need to generate polynomials up to a total degree of eight. Thus, the 2^{nd} degree polynomial has to be multiplied with all monomials up to degree six and monomials with the corresponding degrees for the 3^{rd} and 5^{th} degree polynomials.

Further investigations has 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. Moreover λ_1 was not multiplied with degrees higher 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 difference to the solver except slightly longer computation times.

4.2 Calibrated Case

To solve the minimal problem for calibrated cameras, we make use of the epipolar constraint for 6 point correspondences

$$\mathbf{p}_{u_i}^{\top}(\lambda)\,\mathtt{E}\,\mathbf{p}'_{u_i}(\lambda) = 0, \quad i = 1, \ldots, 6, \tag{15}$$

the singularity of the essential matrix $\mathtt{E}$

$$\det(\mathtt{E}) = 0 \tag{16}$$

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

$$2\left(\mathtt{E}\mathtt{E}^{T}\right)\mathtt{E} - trace(\mathtt{E}\mathtt{E}^{T})\mathtt{E} = 0. \tag{17}$$

Again assuming $e_{3,3} \neq 0$, we can set $e_{3,3} = 1$ and obtain 16 equations in 9 unknowns.

Eliminating variables

The epipolar constraint gives 6 equations in 15 monomials $(\lambda e_{1,3}, \lambda e_{2,3}, \lambda e_{3,1}, \lambda e_{3,2}, \lambda^2, e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}, e_{2,3}, e_{3,1}, e_{3,2}, \lambda, 1)$ and 9 variables $(e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}, e_{2,3}, e_{3,1}, e_{3,2}, \lambda)$.

Using similar a elimination method as in the uncalibrated case we eliminate 5 of these 9 variables. All these variables can be eliminated simultaneously.

We have four variables which appear in one monomial only $(e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2})$ and four variables which appear in two monomials $(e_{1,3}, e_{2,3}, e_{3,1}, e_{3,2})$. Since we have six equations of which each contains all 15 monomials we can eliminate five of these nine variables. We select the first four variables $e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}$ that appear in one monomial only (and can be straightforwardly eliminated) and the fifth variable as $e_{1,3}$ which appears in two monomials.

We reorder the monomials contained in the 6 equations putting monomials containing $e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}$ and $e_{1,3}$ at the beginning. The reordered monomial vector will be $\mathbf{X} = [e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}, e_{1,3}\lambda, e_{1,3}, e_{2,3}\lambda, e_{3,1}\lambda, e_{3,2}\lambda, \lambda^2, e_{2,3}, e_{3,1}, e_{3,2}, \lambda, 1]^T$.

We rewrite 6 equations from the epipolar constraint on matrix form $\mathbf{C}\mathbf{X} = 0$. After performing Gauss-Jordan (G-J) elimination on the matrix $\mathbf{C}$, we obtain 6 equations of the form

$$f_i = LT(f_i) + g_i(e_{2,3}, e_{3,1}, e_{3,2}, \lambda) = 0, \tag{18}$$

where $LT(f_i) = e_{1,1}, e_{1,2}, e_{2,1}, e_{2,2}, e_{1,3}\lambda$, resp. $e_{1,3}$ for $i = 1, 2, 3, 4, 5$ resp. 6 and $g_i(e_{2,3}, e_{3,1}, e_{3,2}, \lambda)$ are 2^{nd}order polynomials in 4 variables $e_{2,3}, e_{3,1}, e_{3,2}, \lambda$. So, the five variables $e_{1,1}, e_{1,2}, e_{1,3}, e_{2,1}, e_{2,2}$ can be expressed as functions of the other four variables $e_{2,3}, e_{3,1}, e_{3,2}, \lambda$.

$$e_{1,1} = -g_1(e_{2,3}, e_{3,1}, e_{3,2}, \lambda)$$

$$\begin{aligned} e_{1,2} &= -g_2(e_{2,3}, e_{3,1}, e_{3,2}, \lambda) \\ e_{1,3} &= -g_6(e_{2,3}, e_{3,1}, e_{3,2}, \lambda) \\ e_{2,1} &= -g_3(e_{2,3}, e_{3,1}, e_{3,2}, \lambda) \\ e_{2,2} &= -g_4(e_{2,3}, e_{3,1}, e_{3,2}, \lambda). \end{aligned} \tag{19}$$

We substitute these expressions into the remaining equation from the epipolar constraint and into the singularity and trace constraints for $\mathtt{E}$. In this way we obtain 11 polynomial equations in 4 unknowns (one of degree 3, four of degree 5 and six of degree 6):

One equation from the epipolar constraint

$$\lambda(-g_6(e_{2,3}, e_{3,1}, e_{3,2}, \lambda)) + g_5(e_{2,3}, e_{3,1}, e_{3,2}, \lambda) = 0, \tag{20}$$

one equation from the singularity constraint

$$\det(\mathtt{E}) = 0, \tag{21}$$

and 9 equations from the trace constraint

$$2\left(\mathtt{E}\mathtt{E}^T\right)\mathtt{E} - trace(\mathtt{E}\mathtt{E}^T)\mathtt{E} = 0, \tag{22}$$

with

$$\mathtt{E} = \begin{pmatrix} -g_1 & -g_2 & -g_6 \\ -g_3 & -g_4 & e_{2,3} \\ e_{3,1} & e_{3,2} & 1 \end{pmatrix}. \tag{23}$$

In [13] it was shown that this problem has 52 solutions.

The Solver

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 use, in this case yielding 320 equations in 363 monomials.

The next step is to reorder the monomials as in equation (4). 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 the "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 $\mathbb{C}[\mathbf{x}]/I$ by the method of [3] and create the action matrix with respect to multiplication by $e_{3,1}$.

5 Non-minimal solution to the uncalibrated case with different distortions.

For comparison we have also created a new non-minimal algorithm for estimating fundamental matrix and two different radial distortion parameters for two uncalibrated views and twelve image point correspondences based on the generalized eigenvalue problem. This algorithm is similar to the well-known algorithm for estimating F and a single distortion parameter from nine point correspondences proposed by Fitzgibbon [9] which was formulated as a quadratic eigenvalue problem.

We also formulate the problem with different distortions as a generalized eigenvalue problem. We use equations from the epipolar constraint for 12 point correspondences

$$\mathbf{p}_{u_i}^{\top}(\lambda_1)\,\mathtt{F}\,\mathbf{p}'_{u_i}(\lambda_2) = 0, \quad i = 1,\ldots,12. \tag{24}$$

Assuming $f_{3,3} \neq 0$ we can set $f_{3,3} = 1$ and obtain 12 equations with 16 monomials $(f_{3,1}\lambda_1, f_{3,2}\lambda_1, f_{1,3}\lambda_2, f_{2,3}\lambda_2, \lambda_1\lambda_2, f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2, 1)$ and 10 variables $(f_{1,1}, f_{1,2}, f_{1,3}, f_{2,1}, f_{2,2}, f_{2,3}, f_{3,1}, f_{3,2}, \lambda_1, \lambda_2)$.

Using the standard notation for the division model

$$\mathbf{p}_u[\lambda] \sim \left(x_d, y_d, 1 + \lambda r_d^2\right]^T, \tag{25}$$

we can rewrite the equations from the epipolar constraint as

$$(D_1 + \lambda_2 D_2)\,\mathbf{v} = \mathbf{0}, \tag{26}$$

where $D_1 \equiv [x_{d_i}x'_{d_i}\ x_{d_i}y'_{d_i}\ x_{d_i}y_{d_i}x'_{d_i}\ y_{d_i}y'_{d_i}\ y_{d_i}\ x'_{d_i}\ y'_{d_i}\ r_{d_i}^2x'_{d_i}\ r_{d_i}^2y'_{d_i}\ r_{d_i}^2\ 1]$ and $D_2 \equiv [0\ 0\ x_{d_i}r_{d_i}'^2\ 0\ 0\ y_{d_i}r_{d_i}'^2\ 0\ 0\ 0\ 0\ r_{d_i}^2r_{d_i}'^2\ r_{d_i}'^2]$, $i = 1,\ldots 12$, are 12×12 matrices containing only known distorted coordinates and $\mathbf{v}$ is the 12×1 vector of unknown monomials $\mathbf{v} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{3,1}\lambda_1, f_{3,2}\lambda_1, \lambda_1, 1]$.

The formulation (26) is a generalized eigenvalue problem which can be easily solved using standard efficient algorithms. For example MATLAB provides the function polyeig.

Because D3 has eight zero columns, this generalized eigenvalue problem leads to eight "infinite" eigenvalues. Thus, there are at most four finite real solutions to this problem.

6 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. For comparison we have also tested non-minimal algorithms on synthetic images. The time of computation has been measured for both minimal algorithms.

The minimal algorithms proposed in this paper are significantly more stable than the algorithms presented in [13] which ran in exact rational arithmetic only. Since doing the

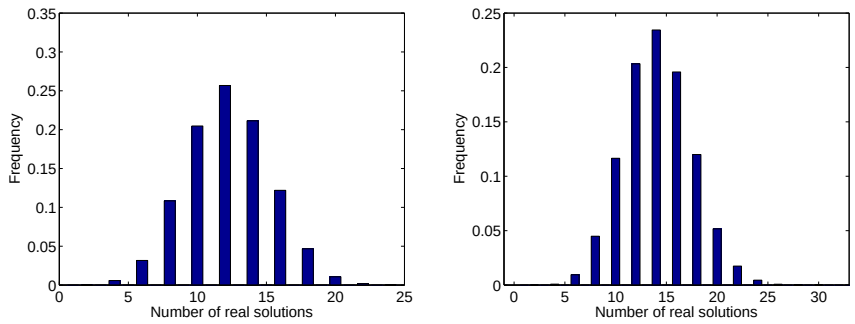

Figure 2: (Left) The number of real solutions for the uncalibrated case. (Right) The number of real solutions for the calibrated case.

computations in exact arithmetic is extremely slow (minutes instead of milliseconds), a comparison with the floating point algorithm presented in this paper is not meaningful and has therefore been omitted.

The problems presented in this paper 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, 24 in the uncalibrated minimal case and 4 in the uncalibrated non-minimal case. In general we obtain more than one real root (Figure 2), 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 [14] for several inputs to select the best root among all generated roots. The kernel voting is done using a Gaussian kernel with fixed variance and the estimates of λ_1 and λ_2 in the uncalibrated case and λ in the calibrated case are found as the positions of the largest peaks [14, 12].

6.1 Tests on Synthetic Images

For all problems treated in this paper, the same synthetic experiments were carried out to evaluate the quality of the solvers.

In all our simulated experiments we generate our 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. Altogether, they become undistorted correspondences, true as well as false matches.

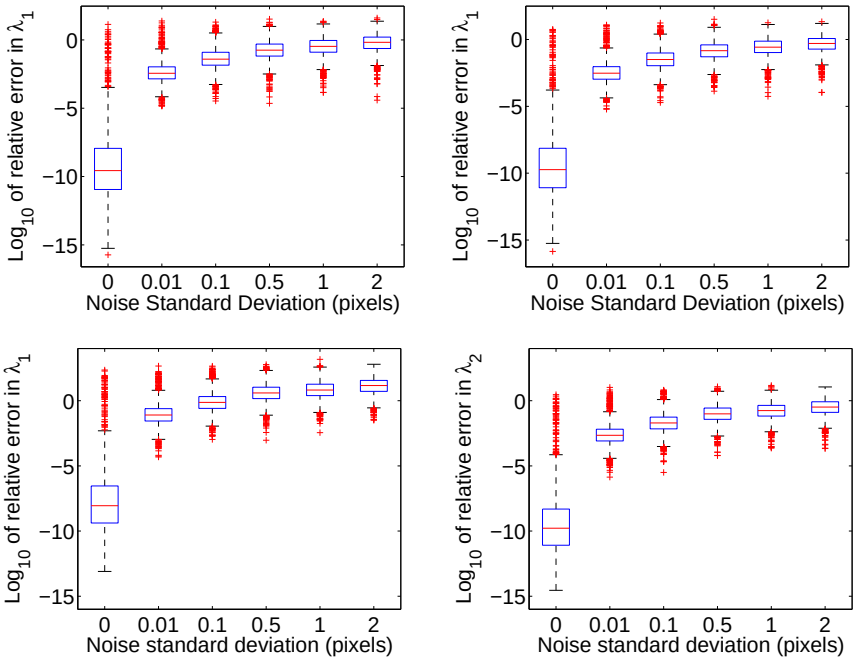

Figure 3: Uncalibrated case: Relative errors of (Left) $\overline{\lambda_1}$ and (Right) $\overline{\lambda_2}$ as a function of noise. Ground truth (Top) $\lambda_1 = -0.2$, $\lambda_2 = -0.3$ and (Bottom) $\lambda_1 = -0.01$, $\lambda_2 = -0.7$. Blue boxes contain values from 25% to 75% quantile.

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 σ to the distorted points.

Uncalibrated case

In the first two experiments we study the robustness of our minimal as well as non-minimal algorithm for the uncalibrated case to Gaussian noise added to the distorted points.

The first experiment investigates the estimation error of λ as a function of noise. Results for the minimal algorithm are presented in Figure 3 and for the non-minimal algorithm in the Figure 4. The ground truth radial distortions parameters were $\lambda_1 = -0.2$, $\lambda_2 = -0.3$ Figure 3 and Figure 4 (Top) in the first case and $\lambda_1 = -0.01$, $\lambda_2 = -0.7$ Figure 3 and Figure 4 (Bottom) in the second case. The noise varied from 0

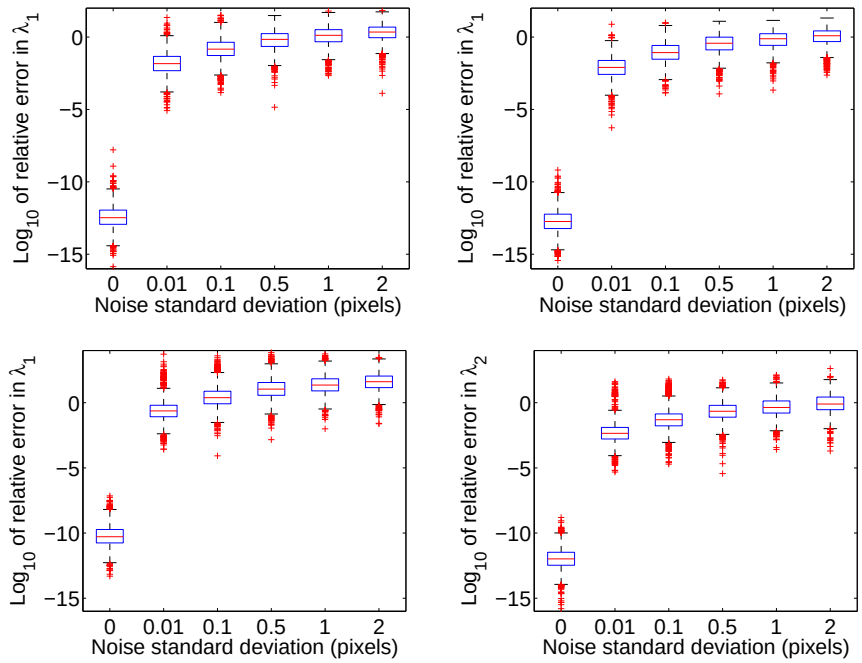

Figure 4: Non-minimal "12 point" algorithm for uncalibrated case: Relative errors of (Left) $\overline{\lambda_1}$ and (Right) $\overline{\lambda_2}$ as a function of noise. Ground truth (Top) $\lambda_1 = -0.2$, $\lambda_2 = -0.3$ and (Bottom) $\lambda_1 = -0.01$, $\lambda_2 = -0.7$. Blue boxes contain values from 25% to 75% quantile.

to 2 pixels. For each noise level relative errors for 2000 λ's (estimated as the closest values to the ground truth value from all solutions) were computed. The results in Figure 3 and Figure 4 for the estimated $\overline{\lambda_1}$ (Left) and $\overline{\lambda_2}$ (Right) are presented by the Matlab function *boxplot* which shows values of the 25% to 75% quantiles as a blue box with red horizontal line at median. The red crosses show data beyond 1.5 times the interquartile range.

Both algorithms give similar results. For noiseless data we obtain very accurate estimates of radial distortion parameters even for very different λ's. For larger noises the $\log_{10}$ relative errors are much higher (mostly around 10^{-1}). However obtained λ's are still satisfactory and mostly differ from the ground truth value in the second decimal place. The main point though is not to use a one set of points to get a good estimate, but to repeatedly draw 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

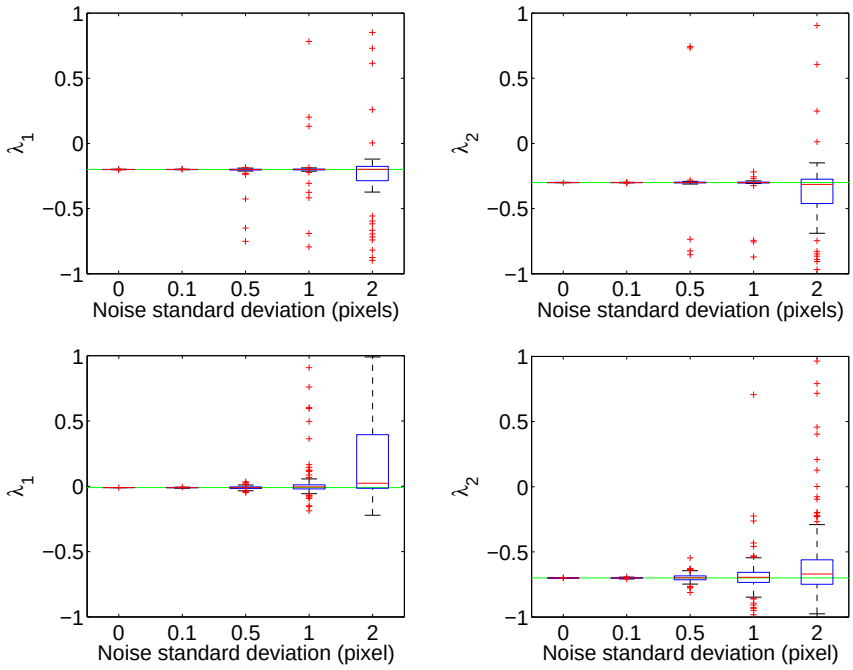

Figure 5: Uncalibrated case, kernel voting: Estimated (Left) $\overline{\lambda_1}$ and (Right) $\overline{\lambda_2}$ as a function of noise, (Top) ground truth $\lambda_1 = -0.2$, $\lambda_2 = -0.3$ (green lines), 90% of inliers and 100 samples in kernel voting and (Bottom) ground truth $\lambda_1 = -0.01$, $\lambda_2 = -0.7$, 100% of inliers and 50 samples in kernel voting.

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 λ'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 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 corre-

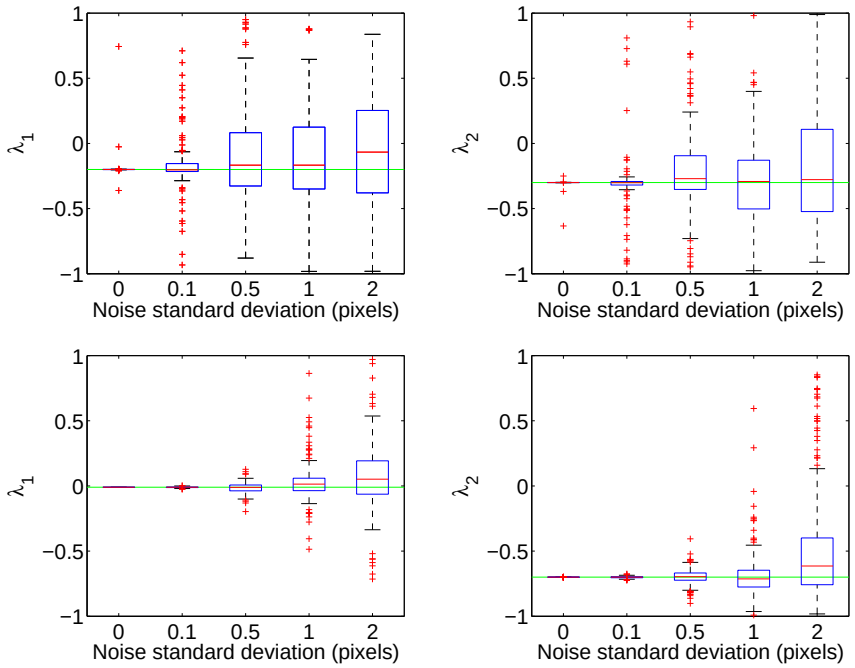

Figure 6: Non-minimal "12 point" algorithm for uncalibrated case, kernel voting: Estimated (Left) $\overline{\lambda_1}$ and (Right) $\overline{\lambda_2}$ as a function of noise, (Top) ground truth $\lambda_1 = -0.2$, $\lambda_2 = -0.3$ (green lines), 90% of inliers and 100 samples in kernel voting and (Bottom) ground truth $\lambda_1 = -0.01$, $\lambda_2 = -0.7$, 100% of inliers and 50 samples in kernel voting.

spondences (6 point correspondences for the calibrated case).

(b) Normalize image point coordinates to $[-1, 1]$.

(c) Find 24 roots using our algorithm (4 roots for the non-minimal algorithm and 52 for the calibrated case).

(d) Select the real roots in the feasible interval, *i.e.* $-1 < \lambda_1, \lambda_2 < 1$ and the corresponding $\mathbf{F}$'s.

2. Use kernel voting to select the best root.

Figure 5 shows λ's computed using our minimal algorithm for the uncalibrated case as a function of noise and Figure 6 shows λ's computed using our non-minimal algorithm. In the first case with outliers Figure 5 and Figure 6 (Top) 100 λ'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 5 and Figure 6 (Bottom)

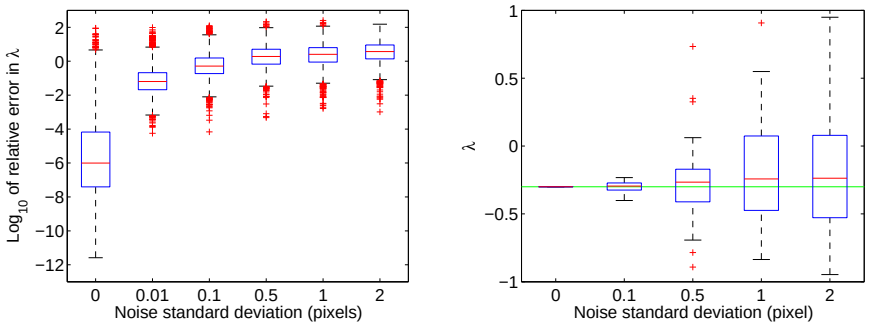

Figure 7: Calibrated case: (Left) relative errors of $\overline{\lambda}$ as a function of noise, ground truth $\lambda = -0.3$. (Right) kernel voting: Estimated $\overline{\lambda}$ using kernel voting for roots computed from 200 6-tuples of correspondences randomly drawn for each noise level. Ground truth $\lambda = -0.3$ (green line).

200 λ's were estimated using kernel voting for roots computed from 50 ($K = 50$) 9-tuples of correspondences. This means that for each noise level our algorithm ran 10,000 times in both cases.

The results are again presented by the Matlab function *boxplot*.

For the minimal algorithm the median values for $\overline{\lambda_1}$ and $\overline{\lambda_2}$ are very close to the ground truth value for all noise levels from 0 to 2 pixels and also for very different radial distortion parameters Figure 5 (Bottom) and 10% of outliers Figure 5 (Top).

The median values for the non-minimal "12 point" algorithm are also close to the ground truth values for all noise levels and also for very different radial distortion parameters Figure 6 (Bottom) and 10% of outliers Figure 6 (Top). However, the variances of this "12 point" algorithm are considerably larger, especially for higher noise levels, than the variances of the minimal algorithm Figure 5. It is significant especially for data with outliers Figure 6 (Top). This is because for 12 points we have higher probability of choosing contaminated sample (sample containing outliers) than for 9 points. The minimal algorithm thus produces higher number of good estimates for the fixed number of samples. This is good both for RANSAC as well as for kernel voting.

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 $\overline{\lambda}$ as a function of noise are shown in Figure 7. The ground truth radial distortion was $\lambda = -0.3$. For noiseless data we again obtain very precise estimates of radial distortion parameter λ. For larger noise levels the $\log_{10}$ relative errors are slightly larger than for the

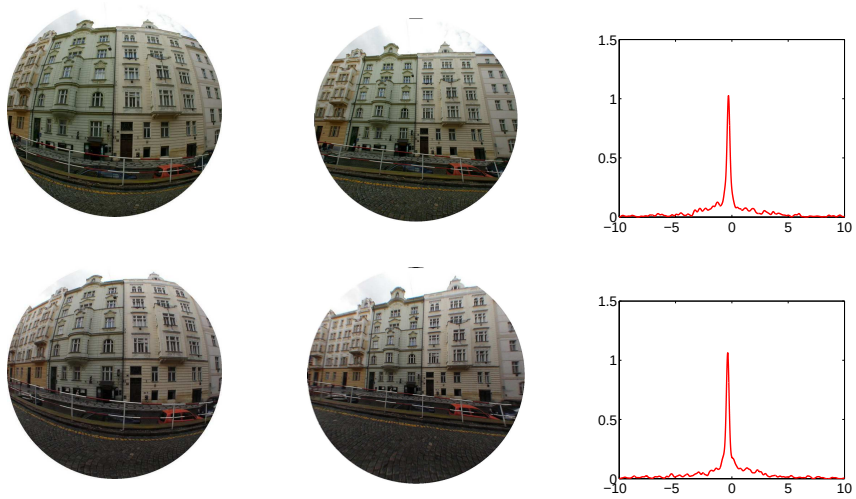

Figure 8: Real data, 60% cutouts from omnidirectional images. (Left) Input images with different radial distortions for camera 1 (Top) and camera 2 (Bottom). (Center) Corrected images. (Right) Distribution of real roots obtained by kernel voting. Estimated $\lambda_1 = -0.301250$ and $\lambda_2 = -0.368125$.

uncalibrated case. However, using kernel voting we can still obtain good estimates. This is shown by our second experiment.

In this experiment λ was estimated 50 times using kernel voting for roots computed from 200 6-tuples of correspondences randomly drawn for each noise level, Figure 7. The median values for $\overline{\lambda}$ 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 λ this algorithm requires more samples in the kernel voting procedure than in the uncalibrated case.

RANSAC experiment

In the last experiment we compare our algorithms with other existing algorithms within the RANSAC paradigm by showing the number of correct matches recovered as a function of the number of samples made from a set of tentative matches contaminated by mismatches. The number of samples used for our experiment was 10, 100 and 1000. We compare the following algorithms

1. Fitzgibbon's non-minimal algorithm [9] for estimating fundamental matrix F and single radial distortion from nine point correspondences based on QEP;

2. Gröbner based minimal algorithm [12] for estimating F and single radial distortion from eight point correspondences, which uses $det(F) = 0$;

3. Our new minimal algorithm for estimating E and single radial distortion for calibrated cameras from 6 point correspondences, which uses $det(E) = 0$ and $2EE^TE - trace(EE^T)E = 0$;

4. Linear algorithm for estimating F and two different radial distortions from 16 point correspondences;

5. Our new non-minimal algorithm for estimating F and two different radial distortions from 12 point correspondences based on generalized eigenvalue problem;

6. Our new minimal algorithm for estimating F and two different radial distortions from nine point correspondences, which uses $det(F) = 0$.

Two sets of images I1, I2 were generated using the procedure described in Subsection 6.1, I1 for $\lambda_1 = \lambda_2$ and I2 for $\lambda_1 \neq \lambda_2$. Gaussian noise with standard deviation 1 pixel was added to the image coordinates.

The number of correct matches ($M\%$ of correct matches) among 1000 tentative matches was fixed and the image points were corrupted by $(100 - M)\%$ of random mismatches. In our case M varies from 100% to 80%.

Table 1 shows results of the first 3 algorithms (i-iii) for single radial distortion $\lambda = -0.3$ and fixed threshold $\tau = 1$ pixel on the distance of an image point to epipolar curves. The results were obtain as a mean values from 100 runs of RANSAC for 10,100 and 1000 samples. Table 2 shows results of the last 3 algorithms (iv-vi) for different radial distortions $\lambda_1 = -0.2$ and $\lambda_2 = -0.3$ and fixed threshold $\tau = 1$ pixel.

The first row in both tables shows the number of matches $K1$ within τ threshold and the second row shows the number of correct matches $K2$ among them.

The results show that the algorithms sampling fewer points faster hit a non-contaminated sample than the algorithms sampling more points. Therefore, our minimal algorithms give better results than non-minimal algorithms. This is significant especially for higher number of mismatches.

6.2 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

RANSAC experiment for $\lambda_1 = \lambda_2$										
		no mismatches			10% mismatches			20% mismatches		
Algorithm		10	100	1000	10	100	1000	10	100	1000
Fitzgobbon 9pt	$K1$	631.44	826.75	873.35	479.17	671.42	772.4	302.17	532.04	632.1
	$K2$	631.44	826.75	873.35	479.08	671.332	772.36	301.5	531.9	632.06
Kukelova 9pt	$K1$	695.62	835.38	835.38	520.84	685.67	773.36	322.36	574.65	660.5
	$K2$	695.62	835.38	835.38	520.73	685.6	773.28	321.78	574.54	660.36
Our minimal 6pt	$K1$	787.5	898.55	930.88	662.24	780.85	838.94	517.98	675.49	719.2
	$K2$	787.5	898.55	930.88	662.07	780.6	838.78	517.53	675.22	718.96

Table 1: The comparison of our "single distortion" algorithm for calibrated cameras with other existing algorithms within the RANSAC paradigm by showing the number of matches $K1$ within τ threshold and the number of correct matches $K2$ among them as a function of the number of samples made from a set of tentative matches contaminated by mismatches. The number of samples was $10, 100$ and 1000, noise level was 1 pixel and threshold $\tau = 1$ pixel. Number of mismatches varied from 0% to 20%.

RANSAC experiment for $\lambda_1 \neq \lambda_2$										
		no mismatches			10% mismatches			20% mismatches		
Algorithm		10	100	1000	10	100	1000	10	100	1000
Linear 16pt	$K1$	316.34	681.22	748.03	232.55	415.7	485.1	86.48	322.35	328.06
	$K2$	316.34	681.22	748.03	232.55	415.69	485.08	86.459	322.31	328.04
Our non-minimal 12pt	$K1$	418.18	697.93	763.51	242.4	546.27	546.27	148.55	414.5	528.02
	$K2$	418.18	697.93	763.51	242.33	546.22	575.56	148.51	414.44	527.85
Our minimal 9pt	$K1$	625.03	801.98	845.76	513.03	671.77	732.56	342.23	538.85	654.86
	$K2$	625.03	801.98	845.76	512.71	671.67	732.54	341.31	538.4	654.65

Table 2: The comparison of our minimal "different distortion" algorithm and our non-minimal "different distortion" algorithm with the linear 16 point algorithm within the RANSAC paradigm by showing the number of matches $K1$ within τ threshold and the number of correct matches $K2$ among them as a function of the number of samples made from a set of tentative matches contaminated by mismatches. The number of samples was $10, 100$ and 1000, noise level was 1 pixel and threshold $\tau = 1$ pixel. Number of mismatches varied from 0% to 20%.

instance of the uncalibrated problem was 16 milliseconds. The corresponding time for the calibrated problem was 17 milliseconds. The time consuming parts of the algorithms are the initial LU-factorization and the eigenvalue decomposition and these are of comparable sizes.

These results are to be compared with the execution times given for the same problem in [13], where solutions were computed in exact rational arithmetic. There, the processing time for one problem instance was 30 seconds for the uncalibrated case and 1700 seconds for the calibrated case.

6.3 Tests on Real Images

We have tested our minimal algorithm for uncalibrated cameras with different radial distortions on several different sets of images. In the first experiment the input images with different relatively large distortions in each image, Figure 8 (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 [15]. They contained correct as well as incorrect matches. Distortion parameters λ_1 and λ_2 were estimated using our algorithm for uncalibrated cameras with different radial distortions and the kernel voting method for 100 samples. The input (Left) and corrected (Center) images are presented in Figure 8. Figure 8 (Right) shows the distribution of real roots for these images, from which $\lambda_1 = -0.301250$ and $\lambda_2 = -0.368125$ were estimated as the argument of the maximum. The peaks from kernel voting are sharp and the λ's are estimated accurately.

In the second experiment we tested our algorithm on images with significantly different distortions. The left image Figure 1 (Left), was obtained as a 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 λ_1 and λ_2 were again estimated using our algorithm for uncalibrated cameras with different radial distortions and the kernel voting method. The input (Left) and corrected (Center) images are presented in Figure 1. Figure 1 (Right) shows the distribution of real roots for these images from which $\lambda_1 = -0.925625$ and $\lambda_2 = 0.002500$ 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 λ's even from only 100 samples.

7 Conclusions

In this paper 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 to time consuming to be of practical value.

With the floating point algorithm presented in this paper 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.

We have also proposed a non-minimal algorithm for estimating F and two different radial distortions from 12 point correspondences based on a generalized eigenvalue formulation.

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

Acknowledgment

This work has been supported by grants EU FP6-IST-027787 DIRAC and MSM684-0770038 DMCM III and the Swedish Research Council through grant no. 2005-3230 Geometry of multi-camera systems, grant no. 2004-4579 Image- Based Localisation and Recognition of Scenes

Bibliography

[1] 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.

[2] J. Barreto and K. Daniilidis. Fundamental matrix for cameras with radial distortion. In *IEEE International Conference on Computer Vision*, Beijing, China, 2005.

[3] M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of gröbner basis polynomial equation solver. In *International Conference on Computer Vision*, 2007.

[4] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer Verlag, 2007.

[5] J.-C. Faugère. A new efficient algorithm for computing grobner bases (f4). *Journal of pure and applied algebra*, 139:61–88, 6 1999.

[6] J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *ISSAC '02: Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, New York, NY, USA, 2002. ACM Press.

[7] J.-C. Faugère and A. Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In *CRYPTO 2003*, pages 44–60, 2003.

[8] 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.

[9] 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.

[10] D. Grayson and M. Stillman. Macaulay 2. Available at http://www.math.uiuc.edu/Macaulay2/, 1993-2002. An open source computer algebra software.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[12] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *Proceedings of Computer Vision and Pattern Recognition Conference (CPVR)*. IEEE Press, 2007.

[13] 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.

[14] H. Li and R. Hartley. A non-iterative method for correcting lens distortion from nine-point correspondences. In *Proceedings of OmniVision'05, ICCV-workshop*, 2005.

[15] 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.

[16] B. Micusik and T. Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In *Proceedings of Computer Vision and Pattern Recognition Conference (CPVR)*, pages 485–490, 2003.

[17] B. Micusik and T. Pajdla. Structure from motion with wide circular field of view cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28, 2006.

[18] H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, Apr. 2005.

[19] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

[20] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision*, Beijing China, Oct. 2005.

[21] H. Stewénius, F. Schaffalitzky, and D. Nistér. How hard is three-view triangulation really? In *Proc. 10th Int. Conf. on Computer Vision*, pages 686–693, Beijing, China, 2005.

PAPER IV

Published at *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Miami, FL, USA*, 2009

Pose Estimation with Radial Distortion and Unknown Focal Length

Klas Josephson, Martin Byröd

Abstract

This paper presents a solution to the problem of pose estimation in the presence of heavy radial distortion and a potentially large number of outliers. The main contribution is an algorithm that solves for radial distortion, focal length and camera pose using a minimal set of four point correspondences between 3D world points and image points. We use a RANSAC loop to find a set of inliers and an initial estimate for bundle adjustment. Unlike previous approaches where one starts out by assuming a linear projection model, our minimal solver allows us to handle large radial distortions already at the RANSAC stage. We demonstrate that with the inclusion of radial distortion in an early stage of the process, a broader variety of cameras can be handled than was previously possible. In the experiments, no calibration whatsoever is applied to the camera. Instead we assume square pixels, zero skew and centered principal point. Although these assumptions are not strictly true, we show that good results are still obtained and by that conclude that the proposed method is applicable to uncalibrated photographs.

1 Introduction

The ability to find the position and the direction in which a camera points is an old and challenging problem in computer vision. If an image based approach is chosen, as in this paper, the common way to solve the problem is to find correspondences between an image taken with a camera with unknown position and a three dimensional model. This method has for example been used in Photo tourism [25]. In this paper we choose to follow the same outline of the algorithm but add one extra component to the model, radial distortion. The enhancement with radial distortion makes it possible to use photos taken with fisheye lenses and other heavily distorted images, see Figure 1 for an example.

The oldest papers on localization are from the time before the research field of computer vision existed. Already in 1841 Grunert [14] showed that there can be up to four real solutions to the problem of localization if inner calibration of the camera is known

Figure 1: Left: An image taken with a fisheye lens. Right: The same image rectified when kernel voting is used to determine the radial distortion

and there are three correspondences between the images and known three dimensional points. For an easier description of the problem and how to solve it, [15] is recommended.

If the inner calibration is unknown it is necessary to have six correspondences between the image and the 3D model. In that case a linear method to find the camera position exists [16]. This method usually gives poor results since digital cameras have square pixels and the principal point close to the center of the image. By not imposing these assumptions to the camera model, too many degrees of freedoms are used which makes the model unnecessarily unstable. These assumptions can however be incorporated and the problem is then to find the pose along with an unknown focal length. In 1995 Abidi and Chandra [1] presented a solution to this problem that worked on planar scenes. Four years later Triggs [28] gave a solution to the same problem that worked well on non-planar scenes. In the same paper he also presented a solution to the same problem but without any assumptions on the principal point of the camera. In 2008 the latest paper [4] on this problem was presented. In this paper Bujnak *et al*. presents a solution that works on both planar and non-planar data. In that solution Gröbner basis methods were used to solve the system of polynomial equations that arises in their solution. Gröbner bases were also mentioned in the paper by Triggs and to the authors knowledge this is the first paper in the computer vision community that uses Gröbner basis methods to solve a system of polynomial equations. This is also the method that will be used in this paper to solve the systems of polynomial equations arising in the problem.

The problem of pose with unknown focal length is not a true minimal case with four points, hence no exact solution can be found. In [4] the fact that the problem is over constrained is resolved by ignoring one equation in an early step of the solver and then using the last equation to verify which of the multiple solutions to use. An alternative method to find the focal length was presented by Josephson *et al*. [18]. In that

paper a correspondence to another image replaced one of the correspondences to a three dimensional point. That method can also be used for the four points problem if one of the points is substituted by an arbitrary line through that point. In this work we instead choose to include radial distortion into the model. This adds one degree of freedom and hence the four points problem becomes minimal.

Minimal problems such as those described above and the one presented in this paper are in computer vision usually used as the key component in a RANSAC engine [10]. RANSAC is the most commonly used method to estimate camera pose from an image, also used in this paper, works as follows. Start by finding correspondences between the image and a model, this is usually done by finding interest points in the images and calculate descriptors of these, see [2, 23, 24]. In this paper SIFT is used. After that the RANSAC engine is used to find consistent correspondences, so called inliers. The inliers are in the end used in a local optimization initiated by the camera model given by the RANSAC engine.

The contribution of this paper is to use radial distortion already in the RANSAC step in the problem of absolute pose. Radial distortion was introduced to the computer vision community by Devernay and Faugeras [9] in 1995. But it was used long before that in the photogrammetry literature, *e.g*. [3]. In both these paper the so-called "plumb line model" is used. This model is probably the most used model, *e.g*. Photo tourism uses this model. However this model is not well suited for minimal problems. Instead we use the division model introduced by Fitzgibbon in 2001 [11]. In that paper he shows that this model is equally powerful as the plumb line model. Due to its simpler form, the division model has been used on several minimal problems [6, 17, 20].

Although the main focus of this paper is the pose estimation problem, the solver of the minimal solution can also be used to estimate the focal length and the radial distortion of a camera lens and by that un-distort an image. A method to find the radial distortion is to use kernel voting. In [22] Li and Hartley used it to find the radial distortion and in [21] Li used it to find the focal length.

2 The Camera Model

The basis of the camera model used in this paper is the standard pinhole camera model [16] where the projection equation is written,

$$\lambda \mathbf{x} = P\mathbf{X}. \tag{1}$$

Here, P is the so-called camera matrix of size 3×4. The camera matrix can be factorized as,

$$P = K[R \mid \mathbf{t}]. \tag{2}$$

In this factorization R is a rotation matrix and holds the information in which direction the camera is pointing and $\mathbf{t}$ gives information of camera position. K is the calibration

matrix of the camera and compensates for the intrinsic setup of the camera. The K matrix can be written

$$K = \begin{bmatrix} f & s & p_x \\ 0 & \gamma f & p_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

In this matrix f represents the focal length of the camera. Further on represents s the skew, for most digital cameras this is zero and the aspect ratio of the pixels described by γ is very close to one. The principal point of these cameras given by (p_x, p_y) is also close to the center of the images. In the rest of the paper a principal point in the center of an image and square pixels with zero skew are assumed and it will be showed that in practice, these assumptions yield good results even though they are not strictly true.

In this paper the pinhole camera model is extended with radial distortion. The radial distortion is modeled by the division model introduced by Fitzgibbon [11]. The reason to choose this model is that it gives easier calculations than the plumb line model. The model transforms the distorted coordinates given by the pinhole camera model according to,

$$\mathbf{p}_u = \mathbf{p}_d/(1 + \mu r_d^2). \tag{4}$$

Here μ is the distortion parameter and $\mathbf{p}_u = (x_u, y_u)$ and $\mathbf{p}_d = (x_d, y_d)$ are the undistorted and distorted positions, respectively. In this paper, the distortion center is fixed to coincide with the principal point of the camera and we set $r_d = \|\mathbf{p}_d\|$. To get a consistent radial distortion independent of image size all images coordinates are initially scaled with a factor of

$$scale = \frac{2}{\max(width, height) - 1}, \tag{5}$$

which maps all image coordinates to be between minus one and one.

3 Pose with Radial Distortion

The problem of solving for radial distortion, focal lengths and pose has eight degrees of freedom; one distortion parameter, focal length, three translation parameters and three rotation angles. To simplify the calculations the inverted focal length is used, and by that the calibration matrix will be,

$$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/f \end{bmatrix}. \tag{6}$$

This can be done since the camera matrix only is given up to scale. In the following $1/f$ will be substituted by w to simplify the notation.

The rotation is parameterized with quaternions. This gives the following rotation matrix,

$$R = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2ac+2bd \\ 2ad+2bc & a^2-b^2+c^2-d^2 & 2cd-2ab \\ 2bd-2ac & 2ab+2cd & a^2-b^2-c^2+d^2 \end{bmatrix}. \quad (7)$$

Last is the translation given by a vector $\mathbf{t} = \begin{bmatrix} x & y & z \end{bmatrix}^T$. Composing these, the camera matrix P will be given according to equation (2). To include the radial distortion in this model the projection will be modeled by,

$$\lambda \begin{bmatrix} x_1 \\ x_2 \\ 1+\mu(x_1^2+x_2^2) \end{bmatrix} = P\mathbf{X}. \quad (8)$$

At this stage the number of unknowns is nine. But since the camera matrix is only defined up to scale, the number of unknowns can be reduced by one by setting the quaternion parameter a equal to one. This will result in that the rotation matrix also will include a scale factor and that the scale of the camera matrix will be fixed. By putting $a = 1$ the possibility of a to vanish is also eliminated. This might look like a problem but since a is a real number the probability of it to be zero is zero. We have also verified in the experiments that this does not cause any problems.

The number of unknowns is now down to eight. Every correspondence between an image point and a world point will now give rise to three equations and one additional unknown. Hence four correspondences are necessary to solve the problem. This is a true minimal case were all equations are necessary and in the next two sections it is explained how to solve this problem.

4 Solving the Minimal Setup

To solve the equations generated by (8) the equation are first simplified by using the freedom in choice of coordinate system. In the three dimensional space any similarity transform can be applied. This freedom is used to put the first point in origin and the second in $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$. In the image, only rotation and scaling is allowed since the focal length is unknown. Due to this, the first point is moved to $\begin{bmatrix} 1 & 0 \end{bmatrix}$. To summarize the following point positions will hold for every problem setup,

$$\mathbf{X}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{X}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}. \quad (9)$$

This choice of coordinate system leads to several simplifications of the problem. First we can express the translation coordinate x in measured image points and the quaternion parameters as follows,

$$x = g_1(a, b, c, d) = \frac{u_1}{u_2}(2ad + 2bc) - (a^2 + b^2 - c^2 - d^2). \tag{10}$$

u_1 and u_2 are here the x and y coordinates of the second image point. The second simplification is that y can be set to zero. The last simplification from the choice of the coordinate system is that the product between the inverted focal length and z can be expressed in the quaternion parameters and the distortion parameter according to,

$$zw = g_2(a, b, c, d, \lambda) = x(1 + \mu), \tag{11}$$

were x is from equation (10).

The next step is to include the last two point correspondences and the last information from the second point $\mathbf{x}_2$. This is done by eliminating λ in equation (8). The elimination is done by multiplying $P\mathbf{X}$ with the following matrix from the left,

$$B = \begin{bmatrix} 0 & -x_3 & x_2 \\ -x_3 & 0 & x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}, \tag{12}$$

where $x_3 = 1 + \mu(x_1^2 + x_2^2)$. This is a rank 2 matrix so not all rows need to be used from the equation $BP\mathbf{X} = 0$. For the second image point only the second row of B is used and for the other two the first and the last row are used. This results in five equations in the five unknowns b, c, d, w and μ. With use of Gröbner basis methods this system of polynomial equations will be solved.

5 Gröbner Basis Solver

To solve the system of polynomial equation Gröbner basis methods are used. Gröbner basis methods have successfully been used to solve several systems of polynomial equations derived from computer vision problem in recent years, *e.g.* [4, 5, 12, 19, 27]. The advantages of using Gröbner basis methods is that if the structure of the system is the same for a large number of problem some calculations can be done symbolically in advance, which leads to an efficient method to solve the systems of polynomial equations.

Only the major concepts of Gröbner basis methods will be described in this paper. For basic background theory we recommend [8] and [7] by Cox *et al*. For details for the use in computer vision see [26] for example.

The first step in constructing a Gröbner basis solver is to find out the number of solutions of the system. This can be done once and will hold for all geometrical setups of the same minimal problem. The method to find the number of solutions is to use a

symbolic program *e.g*. Macaulay 2 [13]. The problem of this paper turns out to have 24 solutions with the given formulation. However, it is quadratic in the focal length so it will never give more than 12 geometrically plausible solutions.

The second step is to expand the initial set of equations. This is done by multiplying the initial equations with a set of monomials. This results in more linearly independent equations with the same solution set and by that it is possible to construct the Gröbner basis. In the problem at hand, the two original equations of lowest degree, these resulting from multiplication with the last row of B in equation (12), are multiplied with μ and w. After that all the nine equations, at this stage, are multiplied with all monomials up to degree four in the unknowns. The result of this expansion is 1134 equations and 720 different monomials. This can be written as,

$$C_{\text{exp}}\mathbf{X}_{\text{exp}} = 0, \tag{13}$$

where C_{exp} is a 1134×720 matrix holding all coefficients and $\mathbf{X}_{\text{exp}}$ is a 720 elements long column vector with all occurring monomials. This equation corresponds to equation (4) in [5] and the rest of the solver will follow that paper. Those details will not be given here. Another way to construct the solver is to use the automatic solver generator by Kukelova *et al*. [19]. We chose the first alternative since it enhances the numerics.

The usual step at this stage is to sort the monomials in a monomial order and then find the Gröbner basis by a Gauss-Jordan elimination. Instead the method from [5] is followed to enhance the numerics. This means that QR-factorization with column pivoting is used together with adaptive truncation of the ideal. The truncation threshold used is 10^{-8}.

To construct the action matrix describing multiplication following [5], the permissible monomials and the action variable needs to be given. In this paper we choose all monomials up to degree three to be in the permissible set and b to be the action variable. The number of permissible monomials with the given choice is 56.

With this the action matrix can be constructed and the eigenvectors of the transposed action matrix will hold all solutions to the system, see [5] for details on how to construct the action matrix with the method chosen in this paper.

Matlab code for the solver used in this paper is available online at
`http://www.maths.lth.se/vision/downloads`.

6 Experiments on Synthetical Data

In this section we study some basic properties of our new algorithm on synthetic data. We start off with a straightforward test on noise free data to check stability and the distribution of plausible solutions. In this experiment, random scenes were generated by drawing four points uniformly from a cube with side length 1000 centered at the origin. A camera was then placed at a distance of 1000 from the origin pointing approximately

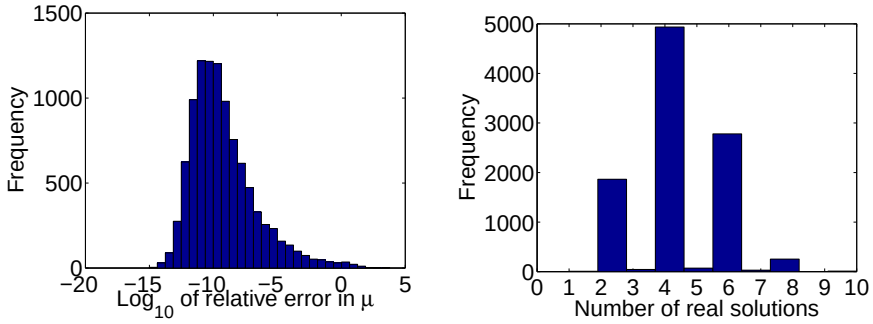

Figure 2: Left: Histogram of errors over 10000 runs on noise free data. Right: Histogram of the number of solutions with real positive focal length found on the same data.

at the center. The camera was calibrated except for the focal length that was set to around 1000. Radial distortion was then added to the projected points and the distortion parameter was uniformly drawn from the interval $[-0.5, 0]$. Our new minimal solver was run on 10000 such instances and Figure 2 displays the results of this experiment. The numerical error stays very low for almost all cases. A very small number of examples show larger errors, but these do not pose any serious problem since the intended application is RANSAC where lots of instances are solved and only the best one is kept. As previously mentioned, the largest possible number of plausible solutions (real positive focal length) is 12. However, the largest observed number of plausible solutions for the 10000 random instances was 10 and in all but a few exceptions we got 6 solutions or fewer.

To verify that the solver does give accurate results and not just adapts to noise we made an experiment where we measured the relative error in focal length as a function of noise. The setup was the same as in the previous experiment and the standard deviation of the noise was varied between (the equivalent of) zero and three pixels on a 1000×1000 pixel image. For each noise level, 1000 problem instances were tested. The results are given in Table 1 and show that our method is robust to noise. Even with as large errors as three pixels, the median error in focal length is less than seven percent.

The time consumed of the solver was also measured. On an Intel Core 2 with clock rate of 2.13 GHz the average time for a call over 1000 tests was 60 ms when a Matlab implementation was used.

The next synthetic experiment was designed to investigate how important it is to include radial distortion in the minimal solver. To do that, a setup with 80 inliers and 120 outliers was constructed. Radial distortion was then added to all image points. Three different levels of radial distortion were used, 0, -0.07 and -0.2. Zero distortion was included to test our algorithm compared to a method that assumes no radial distortion. A distortion of -0.07 was used since the normal lens later used in the real experiments

Noise	Median	75th percentile
0.0	$1.5 \cdot 10^{-11}$	$5.1 \cdot 10^{-10}$
0.5	$1.4 \cdot 10^{-2}$	$4.1 \cdot 10^{-2}$
1.0	$2.3 \cdot 10^{-2}$	$6.8 \cdot 10^{-2}$
2.0	$5.2 \cdot 10^{-2}$	$1.5 \cdot 10^{-1}$
3.0	$6.7 \cdot 10^{-2}$	$1.5 \cdot 10^{-1}$

Table 1: The relative error of the focal length for different levels of noise. The noise is given in pixels.

roughly has this distortion. This lens is shipped with a consumer level SLR camera. The last value, -0.2, corresponds to the distortion of the fisheye lens later used in the experiments. Noise corresponding to one pixel in a 1000×1000 image was then added to each image point. On this data a RANSAC step was applied and the number of inliers was counted. In the RANSAC loop a point was considered to be an inlier if the reprojection error was less than 0.01 times the mean value of all coordinates of all points given that the origin is in the center of the image. One hundred individual scenes were used for each distortion level. All distortion levels were tested both on the proposed method and on the method of Bujnak *et al*. [4]. The algorithm of Bujnak *et al*. solves for pose and focal length using four points. The results of this experiment are shown in Figure 3 with increasing radial distortion from left to the right. Our method is plotted with a solid blue line and Bujnak's with dashed red. The results show as expected that if radial distortion is zero it is slightly better not to estimate it. The two other plots show that the use of radial distortion gives a large boost in performance. Note especially the large difference even with a small distortion of a standard SLR camera lens. These results will also be confirmed in the experiments with real data.

7 Experiments on Real Data

The real world experiments were done in a leave one out manner. This was done by first creating a model of a scene using the Photo tourism bundler [25]. To build the model 93 images from a shopping street were used covering around one hundred meters. Example of one of those images is shown to the right in Figure 4. In all these images a regular lens was used. For 29 of these images a second image was taken from the exact same position (a tripod was used to fixate the position) with a fisheye lens. See Figure 4 (left) for an example. Then one image at a time (of those images with a corresponding fisheye image) were removed from the model. The pose of the removed image was estimated using the proposed method both for the fisheye image and the regular image. The positions were then compared with the positions estimated by Photo tourism. Note that Photo tourism does not give an exact solution and the authors do not know the precision, but it will still

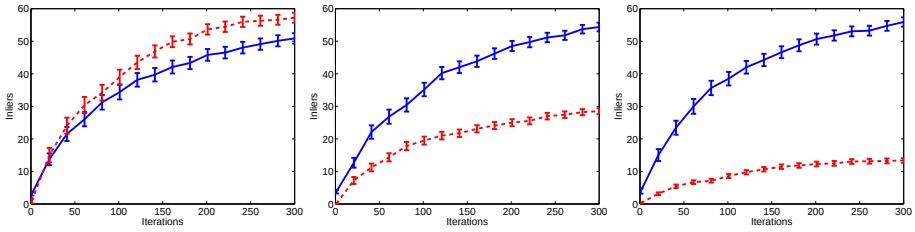

Figure 3: Number of inliers given the number of RANSAC iterations for an example with 80 inliers and 120 outliers. Noise was set to correspond to one pixel in a 1000×1000 pixels image. The distortion parameter, μ, was fixed to, from left to right, 0, -0.07, -0.2 and one hundred examples were performed for each level of distortion. The blue solid line is the method of this paper and the dashed red line is the method proposed by Bujnak *et al*.

be used as ground truth in this paper. The results of this experiment was also compared with the method by Bujnak *et al*.

The pose estimation is done with the following method. First SIFT is applied to the image for which the pose should be estimated. The next step is to find potential correspondences in the image. This is done by nearest neighbor. A point is considered a correspondence if the distance to the closest point times 0.9 is not smaller than the distance to the second closest point. When potential correspondences are found, a RANSAC step is performed to find true correspondences. In the end local optimization is performed.

The first evaluation on the real data experiments was to find the number of inliers given the number of RANSAC iterations. The threshold for a point to be considered an inlier is the same as in the corresponding synthetic experiment. In Figure 5 the result is shown. To the left is the result when the fisheye lens is used and to the right is the result for the regular lens. The graphs show an average over one hundred trials for the images shown in Figure 4. The result is typical for most of the images. It is obvious that the use of radial distortion boosts the performance significantly. In some of the tests the method without distortion almost fails to get more inliers than the minimal set. This shows that the use of radial distortion already in the RANSAC step is an important way to increase the performance of the pose estimation. The result is similar to those in the synthetic experiments.

The next experiment to evaluate the proposed method is to compare the pose estimated by our new method with the position given by Photo tourism. To do this the inliers, position, focal length and radial distortion given by the RANSAC engine are used in a local optimization. The optimization is done for all the unknown parameters. The result is compared with the result when Bujnak's method is used. For that method the same local optimization is performed with the radial distortion initiated with $\mu = 0$. The

Figure 4: Test images used for the experiment whose results are shown in Figure 5. The images were taken at the exact same position.

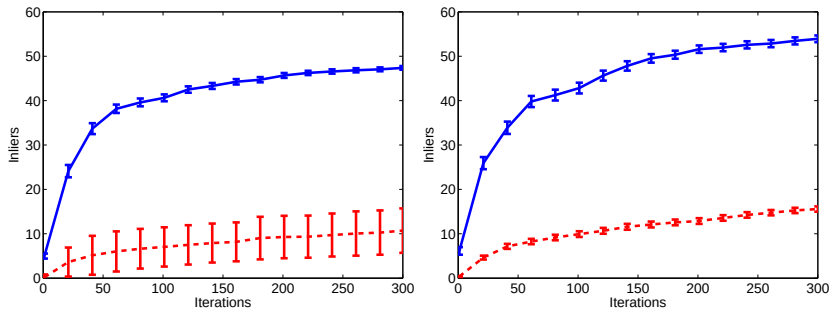

Figure 5: The number of inliers given the number of RANSAC iterations. To the left, a fisheye lens was used and to the right a regular lens was used. The blue solid line is for the method proposed in this paper and the dashed red line is for the method which does not include distortion.

scale of the model in this experiment is adjusted so that the errors roughly correspond to meters in camera position. Each of the 29 camera positions used in the experiment is estimated one hundred times so the pose estimation has been performed 2900 times. In Figure 6 the result of this experiment is shown. To the left is the result when the fisheye lens is used and to the right is the result for the regular lens.

The precision of Photo tourism that is used for the error measurements is unknown to the authors. Due to that the result for the smallest errors are hard to interpret. We estimate that on this data set, Photo tourism achieves roughly an accuracy of one to a couple of meters. Thus error measurements below that are not reliable. Nevertheless, one can see clearly that our new minimal algorithm gives much more accurate results compared to the previous method which does not take distortion into account in the RANSAC engine.

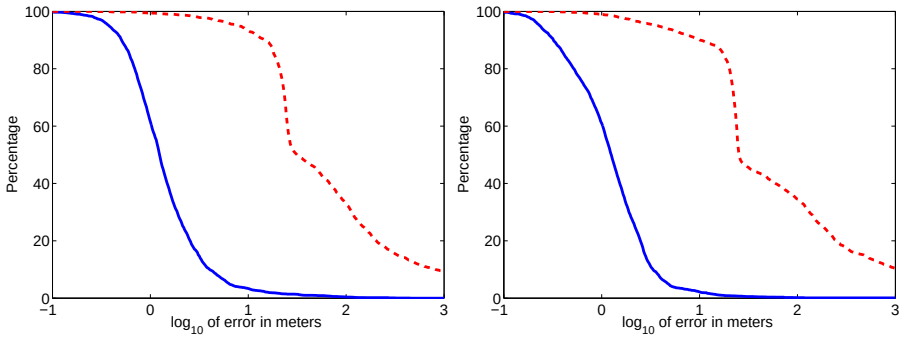

Figure 6: The percentage of images with an estimated position further away then a given distance to the position given by Photo tourism. The error is roughly given in meters. Notice the logarithmic scale. The blue solid line is for the proposed method and the red dashed represents method without distortion. The left plot is for the fisheye lens and the right is for a regular lens.

The results of the pose estimation for the proposed method on a fisheye lens was also compared with the result when the regular lens was used. In Figure 7 the result is shown. In the figure, the blue solid line shows the result with the fisheye lens and the red dashed line shows the result with the regular lens. The plot shows that the amount of radial distortion gives almost no impact on the result.

The last experiment is a kernel voting experiment where the distorted image in Figure 1 (left) was used. The image was localized 500 times with the minimal solver and the results of the estimations of the radial distortion were used in a kernel voting scheme to find the radial distortion. The results of the kernel voting is shown in Figure 8. The peak of the curve is at $\mu = -0.20$ and that value was used to remove the distortion on the original fisheye image. The undistorted image is shown in Figure 1 (right). Notice how the curved lines in the original image have been straightened in the undistorted image. This shows that the estimated radial distortion is reasonably accurate.

8 Conclusions

In this paper a method to estimate the position, rotation, focal length and radial distortion from a minimal set of correspondences to a 3D model is presented. This is the first algorithm presented to do this estimation. The parameterization used in this paper gives a system of polynomial equations. This system is solved with Gröbner basis methods. This gives a fast and numerical stable method that can be used in a RANSAC loop.

Previous methods have not assumed radial distortion in the RANSAC engine and in this paper it is shown that the benefits of using radial distortion in the core of the

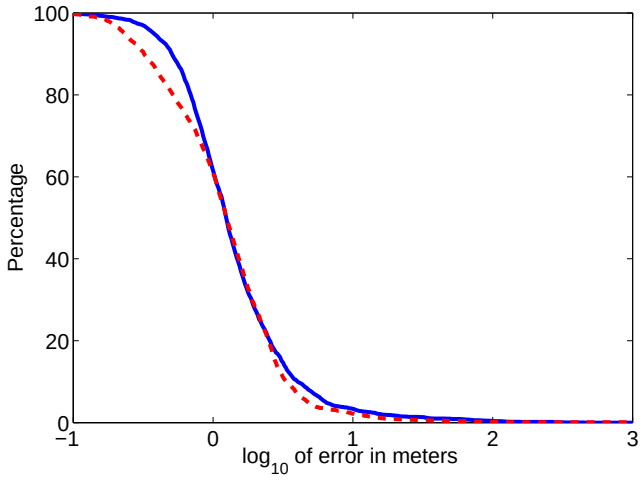

Figure 7: Error in meters for distorted image and non-distorted image using the proposed algorithm on a logarithmic scale. The blue line represents the distorted images and the red show the result for images taken with a regular lens.

RANSAC engine is significant. This is shown both on synthetical and real data when a fisheye lens is used. That the improvements are large with the fisheye lens comes as no surprise due to the heavy radial distortion in this case. More surprising are the large improvements for a regular lens of an SLR camera. The reason for this is that there is some radial distortion even in those kind of lenses and evidently, that distortion can have a large effect on the estimated position.

The experiments in the paper also show that the amount of radial distortion has little impact on the result. Hence can the new method be used both when no radial distortion is present and on images with heavy radial distortion.

Acknowledgments

This work has been funded by the European Research Council (GlobalVision grant no. 209480), the Swedish Research Council (grant no. 2007-6476 and 2008-5393) and the Swedish Foundation for Strategic Research (SSF) through the programme Future Research Leaders and the two projects ENGROSS and Wearable Visual Systems.

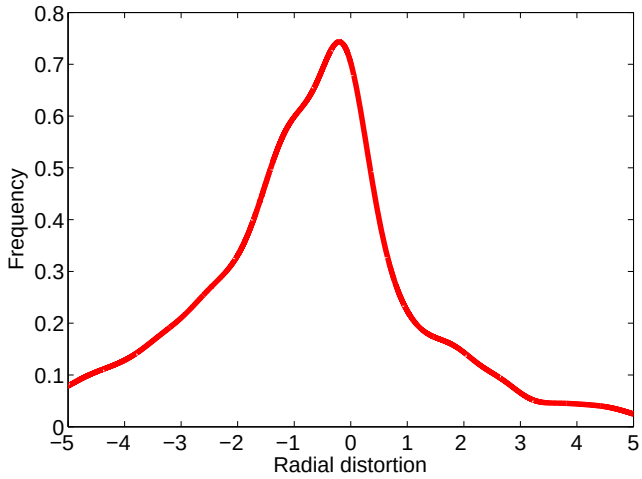

Figure 8: Result of the kernel voting for radial distortion. The standard deviation of the Gaussian kernel was fixed to $1/3$ and the peak of the curve is at $\mu = -0.20$.

Bibliography

[1] M. Abidi and T. Chandra. A new efficient and direct solution for pose estimation using quadrangular targets: Algorithm and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):534–538, 1995.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *9th European Conference on Computer Vision*, Graz Austria, 2006.

[3] D. C. Brown. Close-range camera calibration. *Photometric Engineering*, 37:855–866, 1971.

[4] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.

[5] M. Byröd, K. Josephson, and K. Åström. A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations. In *The 10th European Conference on Computer Vision*, 2008.

[6] M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.

[7] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer Verlag, 1998.

[8] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 2007.

[9] F. Devernay and O. D. Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. *Investigative and Trial Image Processing*, 2567:62–72, 1995.

[10] 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.

[11] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proc. of Computer Vision and Pattern Recognition Conference*, pages 125–132, 2001.

[12] C. Geyer and H. Stewénius. A nine-point algorithm for estimating para-catadioptric fundamental matrices. In *Proc. Conf. Computer Vision and Pattern Recognition, Minneapolis, USA*, June 2007.

[13] D. Grayson and M. Stillman. Macaulay 2. Available at http://www.math.uiuc.edu/Macaulay2/, 1993-2002. An open source computer algebra software.

[14] J. A. Grunert. Das pothenot'sche problem, in erweiterter gestalt; nebst bemerkungen über seine anwendung in der geodäsie. *Archiv der Mathematik und Physik*, 1:238–248, 1841.

[15] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions for the three point perspective pose estimation problem. In *Proc. Conf. Computer Vision and Pattern Recognition*, pages 592–598, 1991.

[16] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.

[17] H. Jin. A three-point minimal solution for panoramic stitching with lens distortion. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[18] 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.

[19] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *Proc. 10th European Conf. on Computer Vision, Marseille, France*, 2008.

[20] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *In Proc. Conf. Computer Vision and Pattern Recognition*, 2007.

[21] H. Li. A simple solution to the two-view focal-length algorithm. In *Proc. 9th European Conf. on Computer Vision, Graz, Austria*, 2006.

[22] H. Li and R. Hartley. A non-iterative method for lens distortion correction from point matches. In *Workshop on Omnidirectional Vision*, Beijing China, Oct. 2005.

[23] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[24] 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.

[25] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2007.

[26] H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, 2005.

[27] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.

[28] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. In *Proc. 7th Int. Conf. on Computer Vision, Kerkyra, Greece*, pages 278–284. IEEE Computer Society Press, 1999.

PAPER V

Submitted to *International Journal of Computer Vision*, 2010

Optimal Correspondences, Geometry and the Vertex Cover Problem

Olof Enqvist, Klas Josephson, Fredrik Kahl

Abstract

Correspondence problems are at the core of many applications in computer vision. Automatically matching features from appearance only is difficult and prone to errors. Geometric consistency is a powerful constraint to identify correct correspondences. Prior work is mainly based on heuristic methods such as RANSAC or EM-like algorithms to solve this task and hence there is a risk of getting trapped in a poor, local solution. We remedy this by developing new theory and algorithms that are guaranteed to find the best solution.

In this paper, geometric correspondence problems are addressed as optimization tasks, trying to find the transformation that maximizes the number of inliers. It is shown how constraints on pairs of correspondences induces a graph problem, which is solved in order to find both the optimal correspondences and the underlying transformation. The ideas are general enough for a wide range of application problems and are implemented for the basic problems of camera pose estimation and 3D-3D registration under rigid transformations and similarities. To further facilitate research in this area, an open source implemention of the algorithms is made publicly available.

1 Introduction

Establishing point-to-point correspondences between images or point sets is a common problem in both computer vision and photogrammetry, for example, in object recognition, 3D reconstructions, image-based localization or medical image alignment. Naturally, due to its importance, many different solutions have been proposed, for example, [5, 9, 16, 20, 22, 31, 43]. Essential for many methods is some type of appearance-based feature matching like SIFT or SURF [32, 4]. However matching local features based on appearance only is difficult and errors are frequent. Therefore it is generally necessary to use geometry to remove incorrect matches.

We focus on two basic geometric problems. The first is 2D-3D registration, also known as calibrated camera pose. Given one set of 2D image points and one set of 3D

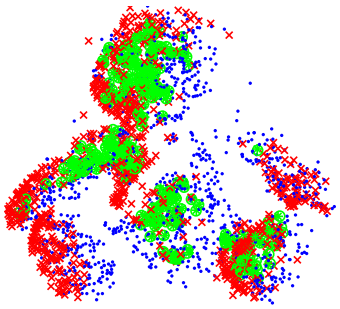

Figure 1: 3D-3D registration of two stereo reconstructions. Green circles and stars are matched 3D points in a maximal inlier set. Red x-marks and blue dots are unmatched points in the two models.

points, the task is to estimate a perspective camera mapping, projecting 3D points to image points. What makes the problem challenging is the difficulty of estimating the correspondences, that is, which points should be matched between the sets. Hence we need to estimate the transformation while simultaneously determining which correspondences are correct. The second problem we consider is 3D-3D registration under rigid transformations and similarity transformations. As in the 2D-3D case, we need to estimate both the correspondences and the transformation. See Figure 1 for an alignment result based on our method.

Given correct correspondences, it is straightforward to estimate the transformation, see [20]. For 3D-3D registration, it can even be done in closed form. On the other hand, given an estimate of the transformation, it is easy to determine likely correspondences. So, a natural idea is to use an alternating minimization procedure, and there are many such algorithms in the literature (e.g., [16]), the most famous one being Iterative Closest Point (ICP) [5]. However, such approaches require a good initial estimate of the transformation and still there is no guarantee of getting a reasonable solution, especially when there are lots of outliers.

The aim of this paper is to find a more reliable and if possible faster method to deal with incorrect matchings. Our goal is to find the largest set of consistent correspondences. To our knowledge, this is the first approach to be able to optimally compute *both* the correspondences and the transformation with a provable guarantee of optimality with proper error modelling. The key idea is to consider point correspondences and check whether pairs of such correspondences are consistent or not. This leads us to the vertex cover problem of graph theory. The bad news is that this problem is known to be NP-hard. The good news is that we are still able to solve instances of the problem for quite large data sets using a combination of branch-and-bound and approximation algorithms.

The outline of the paper is as follows. In the next section, we review related work and compare to our approach. In Section 3, a mathematical problem formulation is given together with an introduction to the vertex cover problem and the use of pairwise constraints. We first apply the framework to a simplified example, namely the estimation of planar rotations, in Section 4. In the following section, some application-adapted algorithms for solving the vertex cover problem are developed. Then, in Section 6, a slightly more advanced problem of estimating 3D rotations is analyzed. Next, in Section 7, the framework is generalized to incorporate more general transformations. In Sections 8 and 9, our two main applications are analyzed using the complete framework. Finally, in Section 10, experimental results are reported and followed by conclusions.

2 Related Work

Graph matching. The idea of using pairwise constraints in combination with vertex cover for finding mutually consistent correspondences is not new in the vision literature. The earliest work we have found is [6] where it is used for 2D part location. Pairwise constraints are also discussed in [17]. In [41], an association graph is built for 2D matching which results in a maximum clique problem. Similarly, the stereo correspondence problem was formulated as a maximum clique problem in [21]. This is an alternative formulation of the same graph problem as ours. In [29], the registration problem was formulated in a graph setting, but solved using a non-optimal spectral technique. Other graph matching formulations include [9, 40, 43] where the objective is to match correspondences such that pairwise distances within the two point sets are similar. This leads to a purely combinatorial problem equivalent to the quadratic assignment problem. A similar approach is pursued in [33]. One advantage (besides being purely combinatorial) is that such approaches can be used for solving both rigid and non-rigid registration problems. However, the underlying transformation mapping one point set to the other is ignored.

3D-3D registration. For 3D-3D registration, the most popular class of methods are EM-type algorithms using alternating optimization, such as ICP [5] and SoftAssign [16]. Other non-optimal approaches include the Hough transform, geometric hashing and hypothesize-and-test algorithms like RANSAC [15]. Recently, in [31], the problem was solved globally using branch and bound in rotation space. The method requires that all points are matched and that the translation component is given, which are severe limitations so this is effectively not solving the complete problem. We have compared our algorithm to several of these competitors in the experimental section.

In [7], a branch-and-bound algorithm over rigid transformations in the 2D plane is proposed. This works well as the transformations have only three degrees of freedom, but the approach becomes computationally infeasible for rigid transformations in 3D space (which have six degress of freedom).

2D-3D registration. Camera pose estimation, or 2D-3D registration, is a well studied problem in both computer vision and photogrammetry [2, 20] and one of the earliest references on the topic dates back to 1841 [18]. There are several heuristic methods with no guarantee of optimality such as [12, 24, 35] where RANSAC is perhaps the most popular one [15]. In multiple view geometry, the camera pose problem is often solved with DLT [20], but the method optimizes an algebraic cost function and cannot handle outliers among the correspondences. Another class of methods is based on subdividing transformation space and aiming for global solutions such as [25] using an affine camera model. The problem is important on its own as a core problem within the field of multiple view geometry, and moreover, it appears as a subproblem for many other vision applications, like motion segmentation [35], object recognition [10, 23, 25], and more generally model matching and fitting problems, see [7, 10, 12, 23, 25, 26, 35]. Yet, previous approaches for solving the camera pose problem have not been able to solve the problem in the presence of outliers with a guarantee of global optimality.

The first globally optimal algorithm for this problem using a geometric error norm was presented in [37]. They also investigate the problem of local minima for the pose problem and show that this is indeed a real problem for small numbers of correspondences. For larger numbers of correspondence pairs or small noise levels, the risk of getting trapped in a local minimum is small. This is our experience as well. In their work the L_2 norm of the reprojection errors is used, but the algorithm converges rather slowly. In [19], the authors choose to work with the L_∞ norm instead and present a more efficient algorithm that finds the optimum by searching in the space of rotations and solving a series of second order cone programs. The reported execution times for both these algorithms are in the order of several minutes, while our algorithm performs the same task within a few seconds. Perhaps more important though is that our algorithm is capable of discarding outliers. If some correspondences are incorrect, then fitting a solution to all data might give a very bad result.

Outliers. A method for detecting outliers for L_∞ solutions was given in [38] but it only applies to quasiconvex problems. The uncalibrated pose problem - often refered to as camera resectioning - is quasiconvex, but the calibrated pose problem is not. Further, the strategy in [38] for removing outliers is rather crude - all measurements that are in the support set are discarded. Hence, inlier correspondences may also be removed. Possible solutions to this problem was given in [30, 36] for outliers, but they are computationally expensive and also restricted to quasiconvex problems. Another well-known approach for estimating camera pose in cases where it is hard to find correct correspondences is to apply RANSAC-type algorithms [15]. Such algorithms leave no guarantees of optimality.

Our work is also related to the rich body of literature on matching problems, see [7, 10, 11, 12, 23, 25, 26, 33, 34, 35]. Many of these algorithms are quite sophisticated and have been an inspiration to our work. However, some do not guarantee any kind of optimality [11, 12, 35], while others use simplified camera models like affine approx-

imations [7, 23, 25, 26, 34]. Other drawbacks of these methods include simplified cost functions not based on reprojection errors.

Unlike many of the above mentioned methods, our framework makes it possible to generate multiple correspondence hypotheses for a single point, but in the final solution, it is guaranteed that no conflicting hypotheses are included. Another advantage is that it is possible to use extra information from the feature extractor, for example, the orientation.

3 Preliminaries

3.1 Formulation

The different geometric problems discussed in this article have many similarities. Given two point sets, we seek a transformation mapping one set to the other, as well as the point-to-point correspondences inferred by this transformation. We can assume that we have a set of hypothetical correspondences. Often, these have been generated by some appearance-based point matching algorithm such as SIFT, but we can also choose to consider all possible correspondences. Whatever the case, it is likely that a large portion of these correspondences are wrong. What this means will of course depend on the problem we are considering. The following definition will be helpful.

Definition 1. *Given an error tolerance ϵ, we say that a set of correspondences C is consistent with a transformation, T, if any pair $(i, j) \in C$ satisfies $d(T(x_i), y_j) \leq \epsilon$ and if C is one-to-one.*

Requiring C to be one-to-one is natural in most applications, but in no way required for the presented approach.

Having many hypothetical correspondences pure chance can lead to small consistent sets, but large consistent sets are extremely unlikely to appear by chance and are thus reflecting the geometry of the problem. Thus it is natural to seek large consistent sets.

Problem 1. *Given a tolerance ϵ and a set of hypothetical correspondences C, find the transformation T that maximizes $|I|$, where $I \subset C$ is the maximal set of correspondences consistent with T.*

To be more concrete, for 3D-3D registration, the transformation T is a similarity transformation and the error residual is the geometric distance. For 2D-3D registration, T is a rigid transformation followed by a perspective mapping and the error residual is the angular reprojection error, that is, the angular difference between the measured image point and the reprojected 3D point.

Note that the popular RANSAC method [15] is one way of solving Problem 1, but even with exhaustive testing of all minimal subsets, the method will not necessarily obtain the optimal solution.

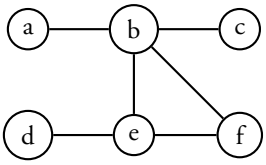

Figure 2: Example graph for vertex cover.

3.2 Pairwise Constraints and Vertex Cover

We approach Problem 1 by considering pairs of correspondences. For each problem we show how to detect geometric inconsistency of such a pair. Two geometrically inconsistent correspondences cannot both belong to the optimal solution and hence we look for large sets of pairwise consistent correspondences.

Assume that pairwise inconsistencies have been established. Consider a graph with all hypothetical correspondences as vertices and edges connecting inconsistent ones. Clearly a consistent subset according to Definition 1 cannot include any edges. Thus the maximal subset of pairwise consistent correspondences should be a good candidate for the optimal solution. Finding this set is equivalent to removing as few vertices as possible while *covering* all edges. This is known as the vertex cover problem and it is one of Karp's 21 NP-complete problems [27] presented in 1972.

Definition 2. *A* vertex cover *for an undirected graph is a subset S of its vertices such that every edge of the graph has at least one endpoint in S.*

Problem 2. *The* vertex cover problem *is the problem of finding the smallest vertex cover for a given graph.*

Factor-2 approximation. For any minimization problem, a factor-k approximation is an algorithm that gives no worse than k times the optimal value. For the vertex cover problem there is a well-known linear-time factor-2 approximation [42]. It is obtained by repeatedly picking a random edge in the graph and then placing both its endpoints in the covering. Take for example the graph in Figure 2. We start by picking the edge between vertices a and b, add these vertices to our solution set and remove them from the graph. In the next step we pick the edge between d and e and remove these two vertices. We have found a vertex cover $\{a, b, d, e\}$. This happens to be exactly twice as big as the minimal vertex cover $\{b, e\}$. Note that starting with the edge between b and e would have yielded the optimal solution. See also Algorithm 1.

Alternative formulations. Alternatively, the same graph problem could be formulated as finding the biggest subset containing no edges. This formulation is known as Maximum Independent Set and is used in for example [8]. We get yet another formulation by

Algorithm 1 Factor-2 approximation

Initialize $S = \emptyset$.
Repeat until no edges remain:
- Pick a random edge e from the graph.
- Add the two vertices connected to e to S.
- Remove the two vertices and all connecting edges from the graph.
When no edges remain S is a vertex cover for the graph.

having edges between consistent correspondences instead. Finding a consistent set then amounts to finding the Maximum Clique in the graph.

4 Example: Planar Rotations

To introduce the approach of pairwise constraints we start with a discussion of the rather elementary problem of estimating the planar rotation between sets of two-dimensional unit vectors. The problem is presented merely as an example, but it does have practical interest when working with one-dimensional cameras [1].

Given two sets of two-dimensional unit vectors $\{x_i\}$ and $\{y_i\}$, and a set of hypothetical correspondences $C = \{(m_i, n_i)\}$, we seek a rotation R that solves Problem 1. To achieve this we examine pairwise angles in the following fashion.

Assume $(m_1, n_1) \in C$ and $(m_2, n_2) \in C$. Let $\alpha(x, y)$ denote the angle - with sign - between 2D vectors x and y. Consider $\alpha(x_{m_1}, x_{m_2})$ and $\alpha(y_{n_1}, y_{n_2})$. For both correspondences to be part of the optimal solution we must have

$$\alpha(x_{m_1}, x_{m_2}) - 2\epsilon < \alpha(y_{n_1}, y_{n_2}) < \alpha(x_{m_1}, x_{m_2}) + 2\epsilon. \tag{1}$$

It turns out that these constraints are actually sufficient.

Theorem 1. *Let ϵ be a positive constant $< \pi/3$ and let C be a set of correspondences such that* (1) *is satisfied for every pair of correspondences in C. Then there exists a rotation R such that*

$$|\alpha(Rx_m, y_n)| < \epsilon \tag{2}$$

for any $(m, n) \in C$.

Proof. We can parameterize R as a 2D unit vector. In order for such a unit vector to exist, condition (2) restricts the vector to an interval on the unit circle with length 2ϵ. Let I_k be the interval connected to the kth correspondence. We claim that the intersection of any pair of such intervals is non-empty. Consider

$$\alpha(y_m, Rx_m) = \alpha(y_m, y_n) + \alpha(y_n, Rx_n) + \alpha(Rx_n, Rx_m). \tag{3}$$

Using (1) and the fact that $\alpha(Rx_n, Rx_m) = -\alpha(x_m, x_n)$, it follows

$$\alpha(y_m, Rx_m) = \alpha(y_n, Rx_n) + e, \tag{4}$$

for some $|e| < 2\epsilon$. Since within I_n we can choose $\alpha(y_n, Rx_n)$ arbitrary in $[-\epsilon, \epsilon]$ there is some $R \in I_n \bigcap I_m$. To complete the proof, we need the following lemma from [14].

Lemma 1. *Consider a set of intervals I_k on the unit circle such that $|I_k| < 2\pi/3$ for all k. If the intersection $I_j \bigcap I_k$ is non-empty for any pair (j, k) , then $\bigcap_k I_k$ is non-empty as well.*

Proof. Pick any interval I_k and let m be the centre of this interval. Note that the distance between I_k and the point $m + \pi$ is larger than $2\pi/3$ and thus $m + \pi$ lies in no interval. Thus we can cut the unit circle at $m + \pi$ and map it to the real line $\mathbb{R}$. The theorem now follows from Helly's theorem [13]. □

Since $\epsilon < \pi/3$, it follows that $|I_k| < 2\pi/3$. This completes the proof. □

Theorem 1 implies that we can solve the the original problem by finding the largest set of correspondences such that each pair satisfies (1). As discussed in Section 3.2 this problem can be formulated as finding the minimal vertex cover for a graph.

5 Basic Algorithms for Vertex Cover

This section presents an approach to the vertex cover problem suitable for registration problems. Some of the advantages will not be apparent until later. To get started, we need an upper bound on the size of the minimal vertex cover. Of course, the factor-2 approximation of Algorithm 1 can be used, but empirically, we have found that in most cases Algorithm 2 produces better bounds. Having a bound on the minimal vertex cover, we can start improving this bound. Our strategy is to remove outliers, or in graph terminology, to remove vertices from the graph that must be part of the minimal vertex cover. As we shall see later, in many situations we will not need to solve the complete vertex problem, but it will suffice with a good bound.

Algorithm 2 Finding a vertex cover

Initialize $M = \emptyset$.
Repeat until no edges remain:
- Find the vertex v with most edges. Add v to M.
- Remove all edges connected to v.
When no edges remain M is a vertex cover for the graph.

Algorithm 3 Fast approximation

Initialize $U = 0$.
For each point x_i in the first point set, if any vertex in the graph
represents a correspondence (i, j), then $U = U + 1$.
When all x_i's have been checked, then U is an upper
bound for the number of inliers.

Algorithm 4 Removing a vertex

1. Assume that $v \notin S^*$. Then all vertices
   having an edge to v must lie in S^*.
2. Remove these and update the graph.
   The solution to the reduced problem is the
   smallest vertex cover that does not include v.
3. Use Algorithm 3 to compute a fast approximation for the reduced problem.
   If this is $> N$, reject the hypothesis that $v \notin S^*$ and
   remove v from the graph.
4. Use Algorithm 1 to compute a factor-2 approximation for the reduced problem.
   If this is $> N$, reject the hypothesis that $v \notin S^*$ and
   remove v from the graph.

In addition to the factor-2 approximation of Algorithm 1, an even faster approximation, but more crude, will be useful. Each point in the first point set can be matched to at most one point in the other set. This yields an upper bound for the number of inliers, see Algorithm 3.

Now, let N be an upper bound for the minimal vertex cover, denoted S^*. To prove that a certain vertex v is an element of S^*, we use Algorithm 4.

Branch and bound. Using branch and bound one can get guaranteed optimal solutions even to NP-hard problems such as vertex cover. Both the fast approximation and the factor-2 approximation can be used as bounding functions. See Algorithm 5.

To get an algorithm which is fast for simple problems while still able to handle difficult ones we divide it into two steps. The first step removes obvious outliers while the second branch-and-bound step is guaranteed to solve any instance. Details are given in Algorithm 6

Multiple hypotheses. One advantage of our approach to correspondence problems is that multiple hypotheses can easily be incorporated, that is, several hypothetical correspondences concerning the same point. To avoid getting a solution where one point is

Algorithm 5 Branch and bound

Initialize the queue with the original graph.
Iterate until convergence.
1. Pick a node from the queue.
2. Compute a fast approximation for the node problem.
   If this is $> N$, then discard the node.
3. Compute a factor-2 approximation for the reduced problem.
   If this is $> N$, then discard the node.

Otherwise
4. Use Algorithm 2 on the reduced problem to improve N.
5. Pick a vertex v and split the problem in the following way.
   (i) v belongs to the minimal vertex cover so we can remove all edges to v.
   (ii) v does not belong to the minimal vertex cover which implies that all correspondences connected to v does.
6. Add these two nodes to the end of the queue.

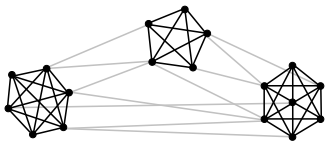

Figure 3: Example graph with multiple hypotheses. Correspondences matching the same point form a clique in the larger graph.

matched to several points, we simply add edges between any vertices (correspondences) matching the same point. The set of vertices matching a certain point will thus form a clique in the graph (Figure 3).

6 Example: 3D Rotations

In this section we will consider the slightly harder problem of estimating a three-dimensional rotation. This problem has its own application when stitching images together to panoramas, but it will also be important when we discuss 2D-3D registration. Like in the planar case, we look at angular errors and use constraints based on pairwise angles. For correct correspondences it must hold that

$$\angle(y_i, y_j) - 2\epsilon \leq \angle(x_i, x_j) \leq \angle(y_i, y_j) + 2\epsilon. \tag{5}$$

Algorithm 6 Minimal vertex cover

Let N be an upper bound for the minimal vertex cover.
Iterate until convergence:
1. Pick a vertex v from the graph.
2. Use Algorithm 4 to prove that v lies in the minimal vertex cover.
   If this works
   - Remove v and update the graph.
   Otherwise
   - Find a vertex cover S with $v \notin S$.
   - If $|S| < N$, update N.

If correspondences remain that are not in the minimal vertex cover:
- Branch on the vertex having most edges.

The most important difference to planar rotations is that even when pairwise constraints are satisfied there may not be a 3D rotation consistent with all correspondences. This forces us to make two alterations to our algorithm.

6.1 Splitting Correspondences

If some outliers cannot be removed with the approach outlined in Section 5, we can split correspondences to refine our search and eventually remove all outliers. This section will present a method to perform such a refinement. First we look at the noise-free case.

Theorem 2. *Consider two sets of three-dimensional unit vectors ordered such that x_i corresponds to y_i. If for all i, j the scalar products satisfy*

$$\langle x_i, x_j \rangle = \langle y_i, y_j \rangle, \tag{6}$$

then there exists a rotation or a reflection or a combination of both R such that

$$Rx_i = y_i \text{ for all } i. \tag{7}$$

Proof. Assume that $\{x_1, x_2, x_3\}$ is a basis for $\mathbb{R}^3$. The special case when no such triplet exists can be handled similarly. Let R be a linear mapping such that

$$Rx_i = y_i \quad \text{for } i = 1, 2, 3. \tag{8}$$

Then for $j = 1, 2, 3$

$$x_i^T x_j = y_i^T y_j = (Rx_i)^T Rx_j = x_i^T (R^T R) x_j \quad \text{for } i = 1, 2, 3 \tag{9}$$

so $R^T R x_j = x_j$. Since this holds for $j = 1, 2, 3$ and $\{x_1, x_2, x_3\}$ is a basis, we can conclude that $R^T R = I$ so R is an orthogonal mapping. But an orthogonal mapping on $\mathbb{R}^3$ is either a rotation, a reflection or a combination of the two (see [3]). Moreover

$$(Rx_i - y_i)^T y_k = (Rx_i)^T (Rx_k) - y_i^T y_k = x_i^T x_k - y_i^T y_k = 0 \tag{10}$$

for $k = 1, 2, 3$. Since $\{y_1, y_2, y_3\}$ is a basis this implies $Rx_i = y_i$ so (7) holds. □

This shows that in the noise-free case our constraints are sufficient and we are guaranteed to find any outliers. Thus our problems depend on the uncertainty introduced by the error tolerance ϵ in Problem 1. We look at the following criterion for consistency.

Lemma 2. *A set of correspondences C is consistent with a transformation R if and only if for each i there exists $\tilde{x}_i$ such that $\angle(\tilde{x}_i, x_i) \leq \epsilon$ and*

$$R\tilde{x}_i = y_i \text{ for all } (i, j) \in C. \tag{11}$$

Proof.

$$\angle(Rx_i, y_i) \leq \angle(Rx_i, R\tilde{x}_i) + \angle R\tilde{x}_i, y_i \leq \epsilon + 0 \tag{12}$$

which proves the if part. For the only if assume there is an R such that

$$\angle(Rx_i, y_i) \leq \epsilon \text{ for all } i. \tag{13}$$

Then $\tilde{x}_i = R^T y_i$ satisfies

$$\angle(x_i, \tilde{x}_i) = \angle(x_i, R^T y_i) = \angle(Rx_i, y_i) \leq \epsilon. \tag{14}$$

□

If we can pinpoint the $\tilde{x}_i$'s, we will get R as well. Initially we have $\angle(\tilde{x}_i, x_i) < \epsilon$. This constrains $\tilde{x}_i$ to a small circle S around x_i, see Figure 4. The size of this circle will be refered to as the correspondence uncertainty. The assumption is that $\tilde{x}_i$ lies in this circle and that $R\tilde{x}_i = y_i$. We can refine our search by dividing this circle into smaller areas S_k, see Figure 4. This generates new hypotheses of type $\tilde{x}_i \in S_k$ and $R\tilde{x}_i = y_i$. We can handle these new hypotheses just like the original correspondences. Simply inscribe S_k in a circle centered at x_i^k with radius δ_k. Using this form the constraint in (5) can be sharpened,

$$\angle(y_i, y_j) - \epsilon - \delta_k \leq \angle(x_i^k, x_j) \leq \angle(y_i, y_j) + \epsilon + \delta_k. \tag{15}$$

Algorithm 7 shows how to incorporate this approach into the overall algorithm. The splitting can be performed in any fashion as long as the diameter of the set is decreased. On the unit sphere it is most convenient to use spherical triangles, because a spherical triangle can be easily divided into smaller spherical triangles, see Figure 4.

It seems intuitive that this method will get us closer and closer to the exact case of Theorem 2. That this is indeed the case is shown by the following theorem, showing that Algorithm 7 converges in a finite number of iterations.

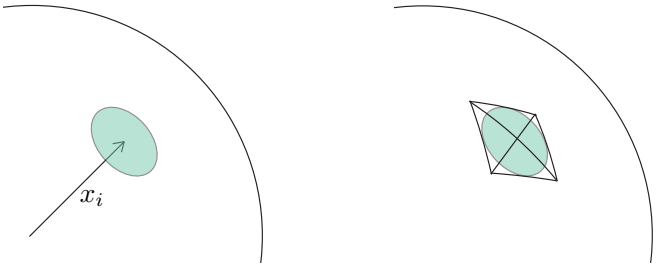

Figure 4: The images show a part of the unit sphere. Originally $\tilde{x}_i$ is constrained to the green circle around x_i shown on the left. In the refinement step this circle is divided as shown on the right, generating new, more precise hypotheses. Working with spherical triangles simplifies subsequent division steps.

Algorithm 7 Splitting correspondences

While our lower bound is strictly smaller than the upper bound:
1. Pick a correspondence and divide it as described in the text.
2. Update all constraints concerning this correspondence.
3. Compute new lower and upper bounds on the solution.

Theorem 3. *Consider two sets of three-dimensional unit vectors ordered such that x_i corresponds to y_i. In a finite number of iterations, Algorithm 7 will determine if there is a rotation or a reflection or a combination of both R such that*

$$\angle(Rx_i, y_i) \leq \epsilon. \tag{16}$$

Proof. Consider the function

$$h(\tilde{x}_1, \ldots, \tilde{x}_N) = \max_{i,j} |\angle(\tilde{x}_i, \tilde{x}_j) - \angle(y_i, y_j)| \quad \text{on } \angle(\tilde{x}_i, x_i) \leq \epsilon \tag{17}$$

This is a continuous function, defined on a compact set. Thus it takes a minimum value $\delta \geq 0$ in this set. If there is an R satisfying (16) then according to Lemma 2, $\delta = 0$. Otherwise, when the uncertainty of Algorithm 7 is less than $\delta/2$, the inconsistency will be detected. □

6.2 Verifying Consistency

For a planar rotation, we obtained a bound on the optimal solution simply by finding a small vertex cover, but in the case of three-dimensional rotations it is not certain that a

Algorithm 8 Verifying consistency

1. Given a set of correspondences with no pairwise inconsistencies.
2. Use an approximate method to compute a transformation.
3. Apply the transformation and count the number of correspondences which are consistent according to Definition 1

vertex cover corresponds to a solution to the original problem. We need to verify that the correspondences are not only pairwise consistent. To do so, we estimate a rotation using an approximate method and count the number of inliers to this rotation. Algorithm 8 can be used for all problems discussed here and with different approximate solvers. For 3D rotations, Horn's method using unit quaternions [22] is a natural choice. As we remove more and more outliers and as the uncertainty of our hypotheses approach zero, the approximate solutions will tend to the correct one.

7 Parameter Search and Vertex Cover

For many problems a pair of correspondences is not sufficient to establish inconsistency. We get a straightforward example by adding an unknown scaling factor to the rigid registration problem discussed previously. Given just two point-to-point correspondences, there is always a similarity transformation mapping the points perfectly to each other. To approach this type of problems, we will combine the vertex cover techniques with a search algorithm over part of the parameter space. For the example of similarity registration, the parameter space is one-dimensional due to the scaling factor. In this section, we will show how to solve this type of problems. An overview of our approach is given in Algorithm 9.

We assume that we start with a bounded set in parameter space. For similarity registration this would be an interval of possible scaling factors and for camera pose a set of possible camera positions. We also assume that we have some way of establishing consistency of a pair of correspondences given the extra restriction that we are in a specific part of the parameter space. This gives us a way to discard large parts of the parameter space at once. The cost of course is an uncertainty in evaluating the constraints. This uncertainty will be referred to as an approximation uncertainty.

For a given box in the parameter space, it is not necessary that we solve the complete vertex cover problem. Since we are performing branch and bound, we only require a bound on the number of inliers. Algorithm 6 does indeed keep track of this bound. The success of the algorithm is dependent on discarding whole boxes at an early stage without having to subdivide them many times. Another crucial factor is that if we are able to discard individual outliers early, because then we will get smaller vertex cover problems when we subdivide the box.

Algorithm 9 Parameter Search and Vertex Cover

Initialize the queue with a bounded set in parameter space.
Iterate until desired precision is reached:

1. Pick a box from the queue.
2. Use Algorithm 3 to discard the box.
3. Use factor-2 approximation to discard the box.
4. Use Algorithm 4 to remove outliers
5. If the rate of outliers is over 0.3.
   - Use branch and bound to discard the box.
6. If the box cannot be discarded:
   - Divide the box and update the queue.
   - Use Algorithm 2 to find a small vertex cover.
   - Use Algorithm 8 to update the bound on the optimum.
7. Remove the box from the queue.

7.1 Splitting Correspondences

To get guaranteed convergence in Algorithm 9 we might need to include the ideas from Section 6.1. At each iteration we can chose between keeping the current correspondences or splitting them as described in Section 6.1. In chosing between the two we can compare the correspondence uncertainty (see Section 6.1) and the approximation uncertainty caused by the size of the boxes in the parameter search. If the correspondence uncertainty is considerably larger than the approximation uncertainty, we split correspondences.

8 Application: 2D-3D Registration

8.1 Problem Formulation

Given an image from a calibrated camera and a 3D model of the scene, we want to estimate the position t, and orientation R, of the camera.

Since the camera is calibrated we can choose to represent the image as a sphere (rather than an image plane) and detected feature points in the image as points on the sphere or unit vectors, x_i. Points in the 3D model will be denoted by X_j. We also have a set of hypothetical correspondences C. For clarity, we restate Problem 1 for 2D-3D registration:

Problem 3. *For a prescribed threshold $\epsilon > 0$, find the rotation R and translation t that maximizes $|I|$ where $I \subset C$ is the largest consistent set of correspondences such that*

$$\angle\left(x_i\,,\, R(X_j - t)\right) < \epsilon \quad \textit{for any } (i,j) \in I. \tag{18}$$

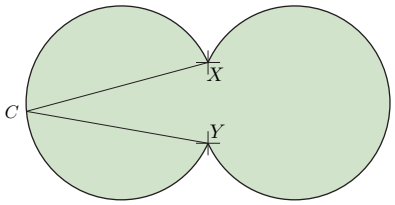

Figure 5: The points t such that $XtY = \alpha$.

Like before, $\angle(u, v)$ denotes the angle between vectors u and v. To fit this problem into the framework of Section 7, we need some way to determine pairwise inconsistency.

8.2 Pumpkin Constraints

Assume for a moment that our camera is ideal and noise-free. Given two 3D points, X and Y, and corresponding image vectors x and y, it follows that

$$\angle\big(X - t, Y - t\big) = \angle\big(x, y\big) = \alpha. \tag{19}$$

Here X and Y are part of the 3D model and the angle α on the right hand side can be calculated from the measured image coordinates. We seek a camera position t that satisfies this constraint.

First consider a plane through X and Y. We know from elementary geometry that all points on a circular arc through X and Y form the same angle XtY. Thus the points t such that XtY equals α form two circular arcs in the plane as shown in Figure 5. Moreover, if for another t the angle XtY is larger than α, then t lies in the set enclosed by the two arcs.

In space the points t for which $XtY = \alpha$ form a surface, obtained by rotating the circular arcs around the line through X and Y, see Figure 6. For angles smaller then $\pi/2$ this surface is pumpkin-shaped and non-convex. Like in the planar case inside this surface form larger angles and points outside form smaller.

Consider the optimal camera pose (R, t) in the sense of Problem 3. Let X and Y be two points that satisfy (18). The triangle inequality for angles yields,

$$\alpha - 2\epsilon \leq \angle\big(X - t, Y - t\big) \leq \alpha + 2\epsilon, \tag{20}$$

These constraints have the important characteristic that they do not involve the orientation of the camera. Thus we can seek the optimal camera centre by a branch and bound search over a subset of $\mathbb{R}^3$. Assuming that we have some bounds on the camera centre, t, we can use the approach from Section 7.

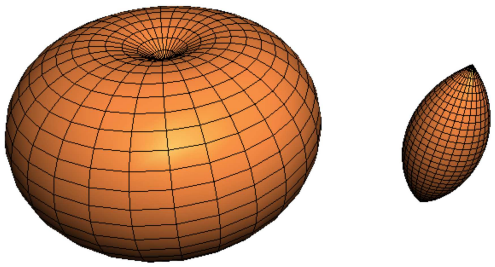

Figure 6: Sets $XtY = \alpha$ for $\alpha < \pi/2$ (*left*) and $\alpha > \pi/2$ (*right*).

8.2.1 Checking the Constraints

Consider two world points X and Y and their corresponding image points x and y. Given a set S parameter space how can we determine if (20) holds for any $t \in S$.

We assume that S is a sphere. Other sets can always be handled by inscribing them in a sphere. First we note that if either X or Y lies in the S, then any angle XtY can be obtained, so (20) is satisfied. In the general case, we can use the following geometric observation.

Lemma 3. *The maximum and minimum for XtY are attained in the plane defined by X, Y and the sphere centre S_c.*

Proof. Let S_{2D} be the intersection of the mentioned plane and the sphere S. Now, it is clear that rotating t around the axis given by X and Y will not change the angle XtY. Thus any point not in the mentioned plane can be rotated into this plane without changing the problem. But any $t \in S$ will be rotated into the S_{2D}, which completes the proof. □

This means that we only need to consider the plane defined by X, Y and the sphere centre S_c. But in this plane the pumpkin constraint is simply two circular arcs as shown in Figure 6. Our problem is thus reduced to determining the intersection of the circle S_{2d} and these circular arcs.

8.3 Sufficiency

In order for the refinement strategy discussed in Section 6.1 to be applicable, we need our constraints to be sufficient as the correspondences uncertainty tends to zero.

Theorem 4. *Consider 3D points X_i and image unit vectors x_i ordered such that X_i corresponds to x_i. If for some camera centre t and all i, j*

$$\angle(X_i - t, X_j - t) = \angle(x_i, x_j), \tag{21}$$

then there exists a rotation or a reflection (or a combination of both) R such that

$$\angle(R(X_i - t), x_i) = 0 \tag{22}$$

for all i.

Proof. Let $y_i = (X_i - t)/|X_i - t|$. Then, (21) implies $\angle(y_i, y_j) = \angle(x_i, x_j)$ and since both x_i's and y_i's are unit vectors $\langle x_i, x_j \rangle = \langle y_i, y_j \rangle$. This means that Theorem 2 is applicable and completes the proof. □

Remark 1. If some 3D point $X = t$, then the angles in this proof are not well-defined. Though this should never be a problem in practice, it may complicate the theoretic discussion. We eliminate this complication by requiring inliers to have at least some distance d_{min} to the camera.

We also need an equivalent of Theorem 3. The fact that we are performing a search over camera positions complicates matters somewhat. Because we are considering multiple camera positions at once, we are not evaluating the left hand side of (21) exactly, but as we will show, the approximation errors tends to zero. Consider a set of camera positions

$$S = \{t : |t - S_c| < r\} \tag{23}$$

To get a bound on the angular uncertainty of $X - t$ given that $t \in S$, we consider a cone with apex at X circumscribing S. If $|X - S_c| > r$, the opening angle γ of this cone will be

$$\tan(\gamma/2) = \frac{r}{|X - S_c|} \tag{24}$$

Now 2γ is a bound on the approximation error introduced in (21) due to the radius of S. As the radius tends to zero, so does this error. The only complication is the case when $X = S_C$, see Remark 1.

Theorem 5. *Consider 3D points X_i and image unit vectors x_i ordered such that X_i corresponds to x_i. For any bounded set $S \subset \mathbb{R}^3$, in a finite number of iterations, the algorithm outlined in Section 7.1 will determine if there is an isometric transformation R and a camera centre $t \in S$ such that*

$$\angle(R(X_i - t), x_i) \leq \epsilon. \tag{25}$$

Proof. Consider the function

$$h(\tilde{x}_1, \ldots, \tilde{x}_N, t) = \max_{i,j} |\angle(\tilde{x}_i, \tilde{x}_j) - \angle(X_i - t, X_j - t)| \tag{26}$$

on the set $\angle(\tilde{x_i}, x_i) \leq \epsilon$ and $t \in S$. This is a continuous function, defined on a compact set. Thus it takes a minimum value $\delta \geq 0$ in this set. If there is an R satisfying (25) then clearly $\delta = 0$. Otherwise, when the sum of the correspondence uncertainty and the approximation uncertainty in (24) is less than $\delta/2$, the inconsistency will be detected. □

8.4 Initialization

To start the parameter search over camera positions, we require some initial bounds on the camera position, for example $t_i \in [-10, 10]$ for $i = 1, 2, 3$ or $|t| < 20$. In most cases we can get this from our knowledge of the scene. In localization, checking which features are visible tells us approximately where we are. If the aim is recognition we can assume that the distance to the object is relatively small - otherwise it would not be recognizable anyhow.

In some cases these bounds can effectively acquired using the pumpkin constraints of the previous section. Like in Section 8.2.1, let S be the central circle of a pumpkin. The pumpkin surface can be defined as $\{x : |x - S| = d\}$. Since it is easy to get bounds on the coordinates of points on the central circle, $x_i < m_i$ this yields bounds on the camera centre $|t_i| < m_i + d$. In presence of outliers some of these bounds can be false, but recall we are only interested in an approximate bounding box that is certain to contain the optimal camera centre.

8.5 Updating the Lower Bound

If we fix the camera position to the centre of the current box, it remains only to estimate the rotation but that is exactly the problem that we discussed in Section 6.2 and the same algorithm is applicable here.

9 Application: 3D-3D Registration

9.1 Problem Formulation

Given two sets of of 3D points $\{x_i\}$ and $\{y_i\}$ and a set of hypothetical correspondences, C, we are interested in the following problem.

Problem 4. *For a prescribed threshold $\epsilon > 0$, find the rotation R, translation t and scaling factor ρ that maximizes $|I|$ where $I \subset C$ is the largest consistent set of correspondences (i, j) such that*

$$|\rho R(x_i + t) - y_j| \leq \epsilon. \tag{27}$$

To solve this problem, we use the method from Section 9. The search space is quite simply a subset of $\mathbb{R}$, representing the possible scaling factors. We will also work with the simpler case of 3D-3D registration with known scaling factor.

9.2 The Constraints

Consider two hypothetical correspondences (i, j) and (m, n). If they are both correct then $\rho|x_i - x_m| = |y_j - y_n|$ provided that there is no noise. This enforces that distances between corresponding points must be equal in the two point clouds. Thus two inliers in the optimal solution of Problem 4 must satisfy

$$\big|\rho|x_i - x_m| - |y_j - y_n|\big| \leq 2\epsilon, \tag{28}$$

for noise threshold ϵ.

As in Section 6 a set of correspondences can be pairwise consistent even though they are not consistent according to Definition 1. To verify that a given set of correspondences is consistent we instead use an approximate method to estimate the transformation and then check that all correspondences are consistent with this transformation. For the experiments we used the method from [22].

9.3 Sufficiency

Theorem 6. *Consider two sets of 3D points ordered such that x_i corresponds to y_i. If for all i, j*

$$\rho|x_i - x_j| = |y_i - y_j|, \tag{29}$$

then there exists a rotation or a reflection (or a combination of the two) R and a translation t such that

$$\rho R(x_i + t) = y_i \text{ for all } i. \tag{30}$$

Proof. We can assume $y_1 = 0$. Choose $t = -x_1$. Let $\tilde{x}_i = \rho(x_i - x_1)$ and note that $|\tilde{x}_i| = |y_i|$. We seek R such that $R\tilde{x}_i = y_i$. Using (29) we get

$$2\tilde{x}_i\tilde{x}_j = |\tilde{x}_i - \tilde{x}_j|^2 - |\tilde{x}_i|^2 - |\tilde{x}_j|^2 = |y_i - y_j|^2 - |y_i|^2 - |y_j|^2 = 2y_iy_j, \tag{31}$$

so the requirements of Theorem 2 are satisfied and this completes the proof. □

This shows that the constraints are sufficient in the noise-free case. A result similar to that in Theorem 5 holds as well, with a very similar proof.

Remark 2. In many applications, for example, 3D-3D surface registration, there are also orientation constraints available. Surface normals at corresponding points should match. Given an angular threshold ϵ for consistency, such constraints are trivial to incorporate and will (generally) speed up the execution time of the algorithm since outliers are easier to discard in the vertex cover algorithm.

10 Results

Algorithms for 2D-3D registration and 3D-3D registration were implemented in C and they are both publicly available for downloading. The implementations follow Algorithm 9. The method for splitting correspondences to get guaranteed convergence was not implemented. Reported execution times are for a computer with a 3.0 GHz Intel DualCore CPU and 3 GB RAM.

10.1 2D-3D Registration

Implementation details. To make the experiments easier to repeat, this section will describe some of the details on how the implementations was done. The overall structure is that of Algorithm 9. Refining of hypotheses as in Section 6.1 was hardly ever necessary in practice, so this was not used for this application. Initial bounds on the camera position are given as input parameters. An upper bound on the minimal vertex cover is also assumed to be given.

Let us look at steps 2 and 3 of Algorithm 9 in greater detail. First, a factor-2 approximation is performed for the vertex cover problem. To minimize computations, only those constraints required for the factor-2 approximation are evaluated. If the box cannot be discarded at this stage, all constraints are evaluated. For each correspondence Algorithm 3 is used in order to prove that the correspondence in question is an outlier. If this does not discard the box either we need to decide whether it is more beneficial to divide the box and continue with the parameter search or to continue with branch and bound in order to discard the box. If there is a lot of outliers branch and bound might be necessary, but for lower rates of outliers this might slow down the execution. One way to get round this is to use branch and bound only if the rate of outliers is high. This method was chosen for the experiments using a threshold of 30%. To reduce the time spent on branch and bound, a maximum branch depth for this problem was also set.

Another question is when to stop the algorithm. It was experimented with different stopping criteria, for example, to consider the number of remaining correspondences not in the optimal solution. The optimal correspondences are uniquely determined if this number to be zero, but correspondences very close to being consistent can be hard to discard so it might be necessary to allow a positive number. Another stopping criterion could be the remaining volume in parameter space, reflecting the uncertainty in camera position. The experiments section states what stopping criteria have been used for the different experiments.

Recognition. The data sets introduced in [28] were used, more specifically, the bear and the Bournvita objects. The aim was not a complete recognition system but rather to show that the algorithm is robust enough to work with challenging real world data. It is also an example of how to use an optimal algorithm to evaluate features for recognition.

Figure 7: A bear with the 3D model reprojected onto the image.

The goal is still to find the largest consistent set of correspondences. The size shows how well the 3D model matches the image.

First SIFT features were extracted from the test images. The features were matched to the stereo model with most overlapping view and for comparison also to three other models, one of the same object from a different view and two of other objects.

The threshold of the ratio in the SIFT feature matching was set to 0.8. Since we were interested in the recognition problem, we ran the algorithm until we had tight bounds on the optimal inlier set. Figure 8 shows the lower and upper bounds in two examples. For all but one image the difference between lower and upper bound was ≤ 5 and in this image there were between 252 and 270 inliers. The error threshold used was 0.0015 radians. To initialize our algorithm we put a lower bound on the size of object in the image yielding a bound on the distance from the camera to the object.

The sizes of the inlier sets are presented in Figs. 9–10. In 2 out of 22 bear images, there were less than 20 correspondences in the optimal set when matched against the bear stereo models. For the Bournvita images, all test images had optimal sets larger than 20 correspondences. There was not a single case with more than 20 correspondences in other cases.

In Figure 7 the 3D model is re-projected for one of the test images in the data set.

The same setup was used in a RANSAC engine where three points were randomly chosen to estimate the camera position. After 10,000 iterations for each test image, 3 out of 22 bear images did not get more than 20 inliers and 1 out of 9 images of the Bournvita objects. So RANSAC may give lower number of correspondences in some cases but more

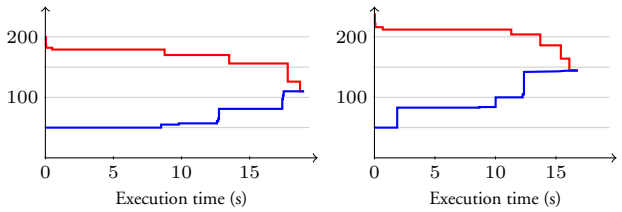

Figure 8: Upper and lower bounds for the solution as a function of time in vertex cover algorithm. (Test images 15 and 16 from [28].)

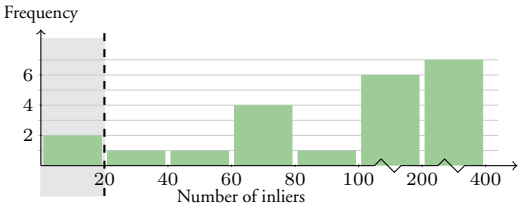

Figure 9: Size of the inlier sets for 22 bear images from [28]. Note the axis. Matching these images to stereo models of the other objects gave inlier sets exclusively in the shaded area.

importantly, no bounds on the solutions are given.

Shopping street. The 2D-3D registration was also tested on a data set consisting of 95 images of a shopping street covering approximately 100 meters. An example of these images is shown in Figure 14. The experiments were performed in a leave-one-out fashion. One image was removed and a model consisting of 3D points and camera matrices were constructed from the remaining views using Photo Tourism [39]. Then the proposed method was used to find the largest possible inlier set for the removed image. This procedure was then repeated for each of the 95 images.

To initialize the algorithm, we prescribe that the camera is reasonably close to street, that is in a cube with side 50m. To get a bound on the optimal solution 50 RANSAC iterations using a minimal solver was performed. This bound was then used as an initial guess for the 2D-3D registration algorithm. The algorithm was run until the remaining volume is less than $0.1\ \mathrm{m}^3$, corresponding to a cube with side length 0.5 meters.

10.2 3D-3D Registration

Implementation details. As for the 2D-3D registration problem we describe the implementation details, to make it easier to repeat the experiments. The implementation

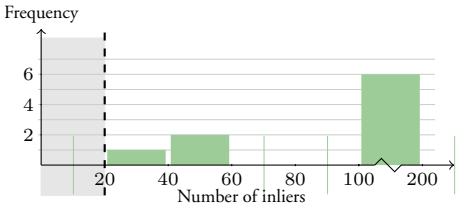

Figure 10: Size of the inlier sets for the 10 Bournvita images from [28]. Note the axis. Matching these images to models of the other objects gave inlier sets exclusively in the shaded area.

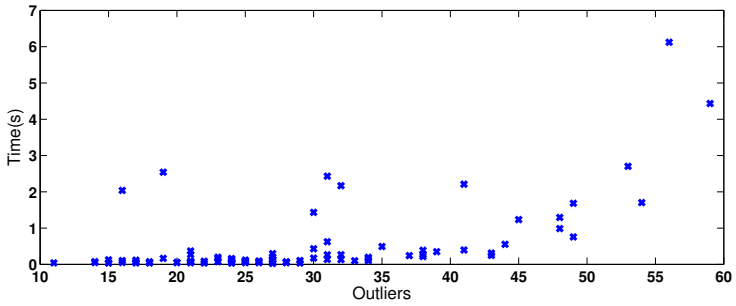

Figure 11: Results for the shopping street experiment. Running times for different rate of outliers. For each image 100 random feature points were used.

follows the outline of Algorithm 9. As for the 2D-3D problem, it was not necessary to do the refinement described in Section 6.1, so it was not implemented. The initial bound on the possible scales are given as parameters to the function, in the experiments a scaling factor between 0.1 and ten was allowed.

Step 2 and 3 of Algorithm 9 are performed as in the 2D-3D registration problem, but Step 5 is not used. Intervals in the parameter space are split at the geometric average of their lower and upper bound, since this is natural for a scaling factor.

As in the 2D-3D registration problem the question of when to stop the algorithm does not have an obvious answer. For the experiments a maximum search depth of ten levels was predefined.

Street view experiments. For this experiment we use a sequence of 300 street view images, downloaded from `eniro.se`. All images are equipped with GPS coordinates of the camera positions. For pairs of images a 3D model was built using SIFT features in a RANSAC engine assuming planar motion. Using the presented algorithm for 3D-3D

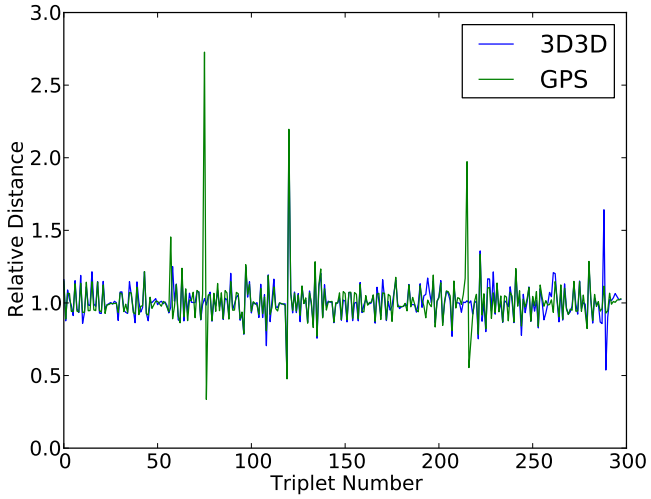

Figure 12: Comparison between GPS measurements of translations and results obtained using our method.

registration, the relative position of consecutive 3D models was calculated.

To verify the results these relative movement was compared to the GPS coordinates. The relative distance between the first two images and the two last images was calculated and compared with the relative distances from the GPS data. Figure 12 shows this comparison and Figure 13 gives some timing info.

Known scale. We tried our algorithm on a 500-point 3D model of the Stanford bunny (see Figure 14). The same data set was used in [31], though there the translation was assumed known and only the rotation was estimated. To mimic the experiments from [31] we computed a random rotation and translation and added uniform noise of magnitude 0.1 to the transformed points. The error threshold of Problem 1 was set to 0.3 (approximately 1% of the bunny length). We then set the algorithm to seek a solution matching at least 90 % of the points. In all cases the optimal solution was found with the basic algorithm without splitting any hypotheses.

In this experiment the total number of hypothetical correspondences was 250 000 and the rate of true correspondences 0.2%, since no correspondences were given and hence all points were matched against all points.[1] Our mean execution time was 0.77 seconds.

[1]This means that a basic RANSAC algorithm would have required on average 125 million iterations to converge.

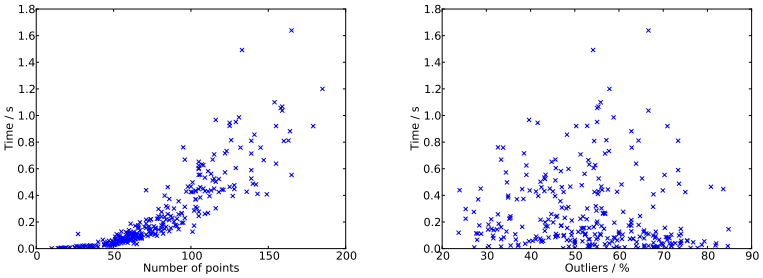

Figure 13: The plots show how computation times depend on the number of points (left) and the rate of outliers (right).

Figure 14: Left: The Stanford Bunny used in the 3D-3D experiments. Right: One of the images of the shopping street used in the 2D-3D experiments.

Essential to get this low average is that it is never necessary to set up the complete vertex cover graph.

The algorithm in [31] obtains similar results on this data set as our algorithm, but they assume that the translation is known and reported execution times are much higher. The ICP and the SoftAssign algorithms [5, 16] work only when the initial rotation is within an angle of less than 50 degrees of the correct solution for this data set.

To get more realistic examples, the 3D-3D registration algorithm was also tested on the data introduced in [28]. An example of those images is shown in Figure 7. Three-dimensional models were constructed from each stereo pair in the training set. Then the registration was performed to the two neighboring stereo pairs for each model. The bear, ball, vase and Bournvita images were used.

To generate the 3D models the training images of the data set were downsampled a factor four. Then SIFT features [32] were extracted and matched for each stereo pair. The matching was performed by comparing the ratio between the best and the second

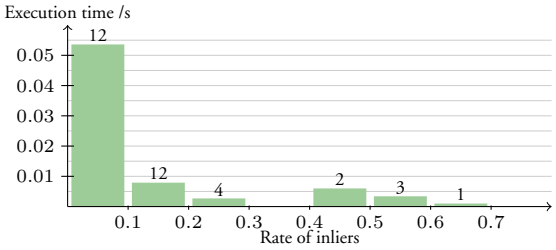

Figure 15: Execution times for 3D-3D registration with 34 stereo pairs from [28] versus the ratio of inliers. The bars show the mean execution time and the numbers give the number of examples in each interval. The optimal solutions have between 6 and 100 inliers.

best match, as proposed by Lowe, with a threshold at 0.6. To construct the actual 3D-model, points were triangulated using the optimal method described in [20]. Points with a reprojection error larger than two pixels were removed.

In the matching between two stereo models all SIFT features with a dot product greater than 0.75 were considered to be hypothetical matches. This results in many multiple hypotheses. The algorithm was set to seek a solution with 20 inliers. If none was found this number was divided by two and the algorithm restarted. In all cases an optimal solution was found and verified, but in a few examples we did not prove uniqueness. This could have been done using the algorithm discussed in Section 6.1.

In Figure 1 a concatenation of two stereo models of the bear is shown. In Figure 15 the execution times for the model concatenation is given for different ratio of outliers.

11 Conclusions

The vertex cover formalism is not a new idea for correspondence problems, but it has previously only been used with limited success. Even though it is an NP-hard problem, we have shown that it is possible to achieve competitive results with sophisticated use of approximation algorithms in combination with today's computers. For several state-of-the-art methods, we have compared the performance to our approach in terms of number of inliers and running times with good results.

We have also extended the use of graph methods for registration, by showing for a number of problem classes how convergence to an optimal solution can be guaranteed.

Bibliography

[1] K. Åström, O. Enqvist, C. Olsson, F. Kahl, and R. Hartley. An L_∞ approach to structure and motion problems in 1d-vision. In *Int. Conf. Computer Vision*, Rio de Janeiro, Brazil, 2007.

[2] K. B. Atkinson. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, 1996.

[3] S. Axler. *Linear Algebra Done Right*. Springer-Verlag, 1997.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[5] P. Besl and N. McKay. A method for registration two 3-d shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[6] R. Bolles and R. Cain. Recognizing and locating partially visible objects: The local-feature-focus method. *Int. Journal Robotics Research*, 1(3):57–82, 1982.

[7] T. Breuel. Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 90(3):258–294, 2003.

[8] M. Bujnak and R. Sara. A robust graph-based method for the general correspondence problem demonstrated on image stitching. In *Int. Conf. Computer Vision*, Rio de Janeiro, Brazil, 2007.

[9] T. Caetano, T. Caelli, D. Schuurmans, and D. Barone. Graphical models and point pattern matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(10):1646–1663, 2006.

[10] T. Cass. Polynomial-time geometric matching for object recognition. *Int. Journal Computer Vision*, 21(1–2):37–61, 1999.

[11] A. Cross and E. Hancock. Graph matching using a dual step em algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(11):1236–1253, 1998.

[12] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous pose and correspondence determination. *IJCV*, 59(3):259–284, 2004.

[13] J. Eckhoff. Helly, radon and carathéodory type theorems. In *Handbook of Convex Geometry*, pages 389–448. North-Holland, Amsterdam, 1993.

[14] O. Enqvist and F. Kahl. Two view geometry estimation with outliers. In *British Machine Vision Conf.*, London, UK, 2009.

[15] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–395, 1981.

[16] S. Gold, A. Rangarajan, C. Lu, and E. Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. *Pattern Recognition*, 31:957–964, 1998.

[17] E. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, 1990.

[18] J. A. Grunert. Das pothenot'sche problem, in erweiterter gestalt; nebst bemerkungen über seine anwendung in der geodäsie. *Archiv der Mathematik und Physik*, 1:238–248, 1841.

[19] R. Hartley and F. Kahl. Global optimization through searching rotation space and optimal estimation of the essential matrix. In *Int. Conf. Computer Vision*, Rio de Janeiro, Brazil, 2007.

[20] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.

[21] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(11):1168–1180, 1989.

[22] B. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using ortonormal matrices. *Journal of the Optical Society of America A*, 5, 1988.

[23] D. Huttenlocher and S. Ullman. Object recognition using alignment. In *Int. Conf. Computer Vision*, pages 102–111, London, UK, 1987.

[24] D. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. Journal Computer Vision*, 5(2):195–212, 1990.

[25] D. Jacobs. Matching 3-d models to 2-d images. *Int. Journal Computer Vision*, 21(1–2):123–153, 1997.

[26] F. Jurie. Solution of the simultaneous pose and correspondence problem using gaussian error model. *Computer Vision and Image Understanding*, 73(3):357–373, 1999.

[27] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations (Symposium Proceedings)*, 1972.

[28] A. Kushal and J. Ponce. Modeling 3d objects from stereo views and recognizing them in photographs. In *ECCV*, pages 563–574, Graz, Austria, 2006.

[29] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, pages 1482–1489, Rio de Janeiro, Brazil, 2007.

[30] H. Li. A practical algorithm for L_∞ triangulation with outliers. In *Conf. Computer Vision and Pattern Recognition*, Minneapolis, USA, 2007.

[31] H. Li and R. Hartley. The 3D – 3D registration problem revisited. In *ICCV*, Rio de Janeiro, Brazil, 2007.

[32] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[33] J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(2):187–199, 2003.

[34] M. Marques, M. Stosic, and J. Costeira. Subspace matching: Unique solution to point matching with geometric constraints. In *Int. Conf. Computer Vision*, Kyoto, Japan, 2009.

[35] C. Olson. A general method for geometric feature matching and model extraction. *Int. Journal Computer Vision*, 45(1):39–54, 2001.

[36] C. Olsson, O. Enqvist, and F. Kahl. A polynomial-time bound for matching and registration with outliers. In *Conf. Computer Vision and Pattern Recognition*, Anchorage, USA, 2008.

[37] C. Olsson, F. Kahl, and M. Oskarsson. Optimal estimation of perspective camera pose. In *Int. Conf. Pattern Recognition*, volume II, pages 5–8, Hong Kong, China, 2006.

[38] K. Sim and R. Hartley. Removing outliers using the L_∞-norm. In *Conf. Computer Vision and Pattern Recognition*, pages 485–492, New York City, USA, 2006.

[39] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM SIGGRAPH*, 25(3):835–846, 2006.

[40] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *European Conf. Computer Vision*, pages 596–609, Marseille, France, 2008.

[41] P. Tu, T. Saxena, and R. Hartley. Recognizing objects using color-annotated adjacency graphs. In *Lecture Notes in Computer Science; Shape, Contour and Grouping in Computer Vision*, pages 246–263, 1999.

[42] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[43] M. Zaslavskiy, F. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009.

PAPER VI

Submitted to *European Conference on Computer Vision*, 2010

City-Scale Localization Using Geometrical Visual Words

Klas Josephson, Linus Svärm, Olof Enqvist, Fredrik Kahl

Abstract

In this paper a new method for global localization using a single image query is proposed. We advance current state-of-the-art by one order of magnitude (from town to city scale). The approach is based on a regular bag-of-words search, but unlike the standard framework, we encode geometry in the design and in the process of image-based localization search.

Recent work has shown that by adding geometry to weak appearance features, powerful and discriminant pose features can be obtained. Our work follows this line of research. Each pair of features, together with the viewing angles between them, form a geometrical visual word. We show how to construct a 3D city model of triangulated 3D features and how to appropriately encode all potential viewpoints in the database. Real experiments are reported on a complete city model with promising results.

1 Introduction

The global localization problem has seen a large increase in potential during the last decade. In 2002 state of the art systems in robotics still used lasers for navigation, e.g. [15]. Since then improvements in global localization systems have been rapid. This is largely due to great advances in large scale structure from motion and image search. The success of the large scale structure from motion systems, such as Photo Tourism [22] and fast city scale reconstruction methods like Building Rome in a Day [1], are founded on the improved methods for wide baseline matching. Cornerstones of the wide baseline matching methods used today are interest point detectors, such as Maximally Stable Extremal Region (MSER) [11], and region descriptors, such as the SIFT descriptor [10]. Evaluations of several detectors and descriptors can be found in [12] and [13][1].

The second important path of development for large scale localization problems is image retrieval. The most common approach to this problem is to use visual words [21].

[1] Also see www.featurespace.org for up do date feature evaluation.

This has made it possible to efficiently search millions of images to find one similar to the query image, e.g., [17]. Image retrieval methods have been used in several previous papers [26, 7, 20] to perform global localization.

Another method used for image retrieval is to use the GIST descriptor [18] that uses global images features instead of local features. In [14] they use the GIST descriptor to make large scale localization where the images cover an area of approximately 20 kilometers.

There are also other methods that do not use local features for the localization problem. One of these is [16] that uses an epitomic representation of the locations to be able to find the positions. The method presented in this paper can be used to classify an image with respect to different image types, such as corridor or kitchen, but also be to find a specific instance of let us say, a kitchen.

Recent work has focused on adding geometry to the image retrieval systems. Examples of this are [2, 24, 8]. In the first of these papers a min-hashing technique was used with a geometrical part that finds closely placed feature points both in the query image and in the image database. This geometrical part was shown to improve results. In the second paper a different approach was used but still considering closely placed points. All points detected by a difference of gaussians detector [10] that were covered by a MSER were bundled together, which meant that geometry was directly encoded into the image retrieval. In [8], they used weak geometrical consistency that favor that the scale and rotation of the feature points have been transformed consistently between the search image and the database images.

In this paper, we develop a system for localization recognition based on a 3D city model. The bag-of-words approach is adopted, but there are several issues that differ from a standard image retrieval system. The main features of our system are:

(i) To our knowledge, this is the largest localization system to-date. Our model covers approximately 20 times more road scenery than in [20] and [14] and approximately 1000 times larger area than in [7].

(ii) Our system is designed for *localization* retrieval. The result of a database query is the *position* of the viewer rather than that of nearby images. The idea is similar to Irschara et al. [7] where the database is also created for localization retrieval by the use of so called *synthetic views*. However, in their approach no spatial information was utilized during the (online) query phase.

(iii) 3D geometry is directly encoded into the visual words. By combining two 3D features simultaneously, and the viewing angle between them, a powerful and discriminant localization feature is obtained. In addition, unlike many image retrieval systems that have large vocabulary sizes (in the order of millions), we can use a much smaller vocabulary. This makes the system more robust to feature quantization.

Figure 1: The image to the left is an example of an image that was used to create the three-dimensional model to the right.

There are several other methods proposed to do global localization of images [25, 19]. These papers use different methods to make image based global localization, but none of these can handle problems at a city scale level as can the method proposed in this paper.

2 Overview of the Localization Method

To facilitate localization retrieval, we rely on synthetic views. A synthetic view represents a small area patch where we keep track of the three-dimensional feature points visible from the patch. We also calculate the viewing angle between each pair of visible features. After assigning a visual word to each feature point, each pair of points are bundled together with their viewing angle to form a new feature that we call a *geometrical visual word*. The synthetic views, with their geometrical visual words, can then facilitate search over positions, much like ordinary bag-of-words facilitate search over images.

To build the database with synthetic views in a way that facilitates queries, quite a few steps need to be performed. To get an overview over these steps, and to see how they fit together, see Algorithm 1. To begin with, a three-dimensional model of the search area is needed. In this paper we use about 95,000 street view images of a city to construct this model. To generate the synthetic views, we need to know from what locations a certain feature point is visible. So in the same step we generate a polygon to each feature point, describing the area from where it is visible. See Algorithm 1.A. To be able to match feature points, we train a vocabulary for visual words. In this paper we use 1600 visual words. To generate the synthetic views, we subdivide the city into a dense grid of small patches, each eight by eight meters. In total we get roughly 1.2 million synthetic views. The details of how they are built, can be seen in Algorithm 1.C, and in the coming sections.

In the online query phase we start by using the bag-of-words approach to find a few promising locations. These are then re-ranked by using stricter geometric conditions. The layout of these steps can be seen in Algorithm 2, and Section 4 will go into the details.

Algorithm 1 Construction of the Model

Required

A set of images, taken with known cameras, over the city.

A. Construct Three-Dimensional Model

1. For each pair of close placed images:
2. Extract feature points using MSER and encode them using SIFT.
3. Triangulate matching feature points using the method of [5]
4. For each such point:
5. Generate a polygon representing its area of visibility.

B. Train Vocabulary for Visual Words

1. Select descriptors of all triangulated points.
2. Select half as many descriptors of non-triangulated points.
3. Cluster these into 1600 clusters. Each cluster represents a visual word.
4. Assign to each triangulated point a visual word.

C. Generate Synthetic Views

1. Subdivide the city into a dense grid of small patches (each 8 by 8 meters).
2. For each patch:
3. For each point within 100 meters from the patch:
4. Intersect the patch with the points polygon of visibility.
5. If there is a non-empty intersection, the point is considered visible.
6. For each pair of visible points:
7. Calculate the range of angles under which the two are visible.
8. The visual words of the two points, coupled with the calculated angles, form a geometrical visual word.
9. Store the geometrical visual word for the patch.

The set of geometrical visual words for a given patch constitutes a synthetic view.

D. Build the Inverted Files

1. Create a list for each pair of visual words.
2. For each patch:
3. For each geometrical visual word in current patch:
4. Save patch-ID and angle interval in the list specific for its visual words.

Algorithm 2 Querying the Model

Required
An image, taken with a camera with known inner calibration.

A. Process Image
1. Extract feature points using MSER and encode them using SIFT.
2. Assign to each feature point a visual word using the trained clusters.
3. For all pairs of points:
4. Measure the angle between the points.
5. Form a geometrical visual word using the visual words and the angle.

B. Rank Patches
1. Keep track of the number of votes given to each patch.
2. For each geometrical visual word in image:
3. Use the inverted file for that visual word pair.
4. For each patch-ID in the inverted file:
5. If the angle in the image lie in the angle-range for the patch:
6. Give current patch a vote.
7. For each patch:
8. Normalize its vote by the number of features in the patch.

C. Re-rank Result
1. Re-rank the top voted patches using a vertex cover based method.

3 Construction of the Model

To perform localization we first need a three-dimensional model of the search area. Our model consists of a three-dimensional point cloud and to each point a polygon of visibility. Model points can be calculated by using systems like Photo Tourism [22] or other similar systems, e.g. [6]. These systems often handle uncontrolled image collections. In our work the data is more controlled so instead of using Photo Tourism like methods, a special purpose algorithm is used, designed to fit our data.

3.1 Dataset

The dataset of images used, consists of approximately 95,000 downloaded images of a city[2]. These images are street view images taken with roughly an interval of five meters.

[2]Contact the authors on how to download the dataset.

Every image has a size of 1288×642 pixels and covers the complete visual sphere. An example image is shown in Figure 1. Each pixel in the images represents a constant shift in inclination and azimuth, respectively. This makes it possible to calculate the inner calibration of the cameras, since every pixel in the image can be associated with a direction on the image sphere, which is exactly the same as a known inner calibration [5].

Not only the inner calibration was possible to find in the downloaded data, also the position and rotation were available. The position was retrieved from GPS coordinates associated to the images. The GPS coordinates are given in a reference system that is right angled and where the distances can be interpreted in meters. Further on, the rotation between two arbitrary images in the data was zero, since all images were cropped in the south. This data was not in all cases entirely correct, but we made no efforts to adjust the positions.

3.2 Keypoint Selection and Triangulation

We now have a complete set of images with all camera parameters known. This made it easy to triangulate points from the images. We began by finding interest points in all images. This was done using the MSER keypoint detector [11], where we used a threshold to only include regions larger than 100 pixels. This bounds the number of features to a few hundreds per image. After that the SIFT descriptor was applied to the interest points. To perform these calculations we used the code by Mikolajczyk [12]. With the known keypoints the triangulation was performed on pairwise images. To perform the two-view triangulation, the method of Hartley and Sturm [4] was used. Part of the 3D-model is plotted in Figure 1.

3.3 Construction of Visibility Polygons

To make the search over positions in the city we place a dense grid over the city, with each square having the side length of eight meters. For every single patch we calculate which features are visible.

A polygon representing visibility is constructed using the directions to the two cameras, which the point was reconstructed from, together with the distance to the camera furthest away. See Figure 2 for details.

3.4 Training Vocabulary for Visual Words

A vocabulary for visual words is used to match feature points in the model and the query image. Usually, query images contain lots of features that are not robust enough to triangulate, e.g. leaves. By using a vocabulary based on only triangulated features, such unstable features are forced to match features that have been triangulated, and thus adds noise to the search. To avoid this, we use features from both the three-dimensional points,

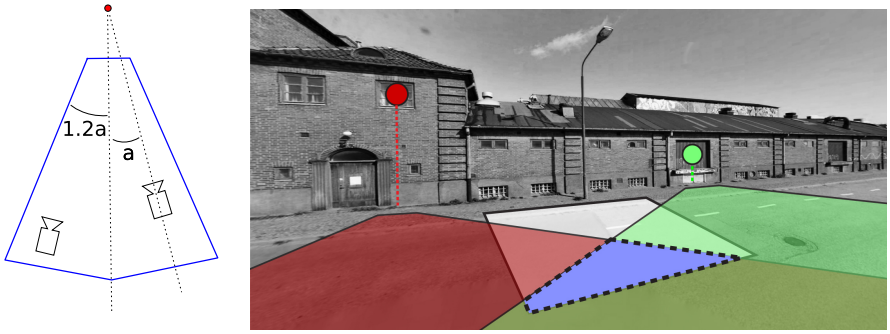

Figure 2: *Left*: Construction of the polygon of visibility. The red dot shows the model point and the blue polygon the area of visibility. The distance between the point and the corners furthest away from the point, is the maximum value of 20 meters and the distance to the most distant camera that the point was triangulated from. The distance between the point and the closest corners of the polygons are fixed to three meters. The field of view is the angles to the cameras where the point is visible scaled with 1.2 compared with the central angle of the visibility cone. This value is not allowed to be greater than $\pi/2.1$. *Right*: Intersection of visibility polygons and a patch. The red and the green areas are visibility polygons, the white area represents a patch. The area inside the dashed line is where the two feature points are regarded visible at the same time for the given patch. See Section 3.6.

and points that have not been triangulated. When clustering, twice the number of triangulated points are used compared to the number of non-triangulated ones. By using a majority of triangulated features for training, we should get a vocabulary well suited for representing the features we are interested in.

The clustering is done using VLFeat[23]. We choose to cluster the data into 1600 clusters. Results in the image retrieval research suggests, that the vocabulary should contain a million to several million words, to achieve good results. The reason for our much lower number is twofold. First of all, a lower number of features gives a greater robustness and generality and the possibility to assign a correct word to a new feature is increased. Second of all, we will use pairs of features that in practice increase the number of words to roughly 1.3 million words. The angles between two features will also be used in the end to further increase specificity.

3.5 Model Reduction

With word and visibility polygon known for each model point, a model reduction is performed. This is done since we during triangulation did not search for longer point

tracks between several images; only pairs of images were considered. One downside of this is that the same physical point can be triangulated twice or more. As a way of reducing the amount of duplicates, all feature points close to eachother having the same word are merged, provided that their polygons of visibility intersect.

The result of this process is that groups of closely placed features are merged to single points, with visibility polygons that cover the same area as all of the original polygons combined. The process will thereby reduce the number of multiple points describing the same physical point. In our experiment the number of model points are reduced by almost 10% by this merging step.

3.6 Generating Synthetic Views

To generate a synthetic view, pairs of features and the corresponding viewing angle for each pair is considered. Given such a pair of features and a patch in the search space, see Figure 2, we need to determine under which angles these features can be seen. More precisely, if X and Y are the 3D positions of the two features, and C is a thought camera centre within the patch, we want to determine bounds on the angle XCY.

To do so, the polygons representing the visibility of these two features are intersected with eachother and with the current patch. This results in a polygon P_{2D} representing that part of the current patch where both points are visible.

Consider a coordinate system such that the ground plane is perpendicular to the z-axis. The region P_{2D} restricts the camera in the xy plane, but the sought angle will also depend on the z-coordinate of the camera centre C. Using the model images as a reference, we assume that the z-coordinate of the query images will deviate with at most two meters. Since the model images are captured using cameras mounted on top of a car, very few query images should fail to satisfy this constraint.

This means that for any

$$C \in P = \{(x, y, z) : (x, y) \in P_{2D}, z \in [-2, 2]\} \tag{1}$$

we want to determine bounds for the angle XCY.

3.7 Calculating Angular Bounds for Polyhedron

We approach the problem of finding the angular bounds in the following manner.

1. Inscribe P in a number of spheres - we use four spheres.
2. For each sphere, determine bounds $[l_i, u_i]$ on XCY if C lies in the sphere.
3. Take the union of intervals $[l_i, u_i]$.

It remains to show how to perform Step 2.

Problem 1. *Given two 3D points X, Y and a spherical set of camera centers, $S = \{C : |C - S_c| \leq r\}$, determine upper and lower bounds for the angle XCY.*

First we note that if either X or Y lies in the sphere, then any angle is possible. In the general case, we can use the following lemma.

Lemma 1. *The maximum and minimum for XCY are attained in the plane defined by X, Y and the sphere centre S_c.*

Proof. Let S_{2D} be the intersection of the mentioned plane and the sphere S. Now, it is clear that rotating C around the axis given by X and Y will not change the angle. Thus any point not in the mentioned plane can be rotated into this plane without changing the problem. But any $C \in S$ will be rotated into the S_{2D}, which completes the proof. □

This means that we only need to consider the planar case.

Lemma 2. *The planar case can be solved by finding all circles through X and Y that is tangential to S_{2D}.*

Proof. We use the same reasoning as in [3]. The angle XCY is equal for all C on a circular arc through X and Y and the angle is determined by the radius of the circular arc. Thus we seek the largest and smallest circles through X and Y such that they intersect S_{2D} and it is easy to see that these are also tangential to S_{2D}. □

These circles can be found by solving a simple quadratic equation. For brevity, we exclude the derivation.

3.8 Building the Inverted Files

With all visible pairs of points known for every patch, along with the angle intervals for all these pairs, it is possible to create a search database to find the right patch. A geometric visual word is constructed for every pair of features together with the bounds on the angle interval.

All the geometric visual words are stored in inverted files [17] to make a fast search after the best patch possible. The inverted files hold, for every given pair of visual words, the patches were these are visible together with the angular interval for each patch.

4 Search Routine

Given a query image, the search for its camera position is divided into two steps. First we make a bag-of-words search [17] over all synthetic views. After that, a re-ranking is performed to improve the result.

4.1 Bag-of-Words Search Step

In the query image, interest points are extracted using MSER and described using SIFT. All the descriptors are then assigned words from the clustered vocabulary of the model. To get the angle between every pair of features, it is necessary to transform the image coordinates to the visual sphere. To do so, the inner calibration of the camera needs to be known. With the coordinates on the image sphere given, the angle between every pair of interest points are calculated.

With the list of word pairs and the angles between them, the inverted files are used to rank all patches. The search algorithm iterates over all pairs of words in the search image. For each word pair a list of patches are given where these are visible. Together with the patch, the angular interval is also returned. If the image angle is inside this interval, one vote is added to this patch.

To balance the result depending on how many points that are visible in a square, the value obtained after all words have been tested, is divided by the maximum of the square root of the number of point pairs in every patch and 50.

4.2 Improved Patch Ranking

The bag-of-word step of the search algorithm manages, in most cases, to rank the correct patch within the top one hundred of all patches. To improve the result we perform a second step of ranking on these top results.

As our new measure of how well the image matches an area patch in the city, we want to count the number of correspondences of the feature points. By correspondence we mean a match between a specific feature point x_i in the image and a specific point X_i in the model, where only points having the same word may match.

Furthermore, we require that the all pairs of correspondences are consistent in terms of geometry. This means that the angle between the two image points must lie within the interval of possible angles of the two corresponding model points. Two correspondences are also considered inconsistent if they try to match the same point in one of the point sets, to different points in the other set.

To find a consistent set of points a graph with all hypothetical correspondences as vertices is built. Edges are added between all inconsistent correspondences. The task of finding the largest set of geometrically consistent correspondences is now equivalent to removing as few vertices as possible in such a way that all edges are removed. This is the vertex cover problem; one of Karp's 21 NP-complete problems [9].

Since this is a NP-complete problem, we confine ourselves to finding a sub-optimal solution. Intuitively, wrong correspondence should get a lot of edges since they probably are inconsistent with most other correspondences. This motivates Algorithm 3, which is a greedy algorithm for efficiently finding a solution. The remaining vertices from the algorithm represent consistent correspondences.

The number of remaining correspondences is divided by the number of visual points

Algorithm 3 Greedy Vertex Cover Approximation

1: Keep track of the number of edges to each vertex.
2: Iterate until no edges remain.
3: Select the vertex with most number of edges.
4: Remove selected vertex.
5: Remove affected edges and reduce the edge-count of all affected vertices

in the patch. This is done to compensate for the uneven distribution of points among patches. If the number of points is lower than 200, we choose to divide by 200 to avoid favoring patches with few points but relatively many inliers.

5 Experiments

Before the results are presented some details of the implementation of the method are described to simplify a reimplementation.

5.1 Implementation Details

Dataset Pre-processing. On some streets, the car that captured the photos, has passed several times. This results in a much higher density of images in those areas. To get a more uniform distribution of images for the model, we remove images from these dense areas. To decide which images to remove, a density measure is calculated for each image. The density measure ρ for image k is defined as:

$$\rho(k) = \sum_{i \in I_k} \frac{6}{1 + d(k, i)}, \tag{2}$$

where I_k is all images closer than eight meters from image k and $d(k, i)$ is the distance in meters between image k and i. This value is calculated for all images and the one with the highest density is removed. The density measure of affected images is recalculated. This removal process is repeated as long as any image has a density measure larger than an empirically chosen threshold of 3.7. By doing this approximately 11,000 of the 95,000 images were removed.

Methods to Remove Incorrect Three-Dimensional Points. To reduce the number of incorrect three-dimensional points, several actions were applied. First only image pairs with a baseline between four and 15 meters were used to get a stable model. When the triangulation was done, only points with small reprojection errors were kept. Also positive depths were required in both images. Points further than one hundred meters or closer than 3 meters to any of the two cameras are removed.

Clustering. We used two levels of hierarchical clustering with 40 splits in each level, resulting in 1600 visual words.

Point Merging. During point merging, presented in Section 3.5, points that are within half a meter of each other are considered. The merging works as follows. The new point is given the position equal to the mean of the points to merge. Select the viewing angles of the new points so that the viewing directions of all merged points are covered. Finally set the viewing distance to the maximum of the viewing distance of all merged points.

Angle Thresholds. To reduce the amount of data necessary to store, we only save pairs of features that can be visible with a viewing angle between them less than 90 degrees. This reduces the amount of data to store significantly, since half of the image sphere is out of reach for this value. A second motivation for this threshold is that the field of view for a regular camera usually does not cover more than 90 degrees.

5.2 Results

To evaluate our method, 4202 images were used that had not been used for model construction. These images were coupled with GPS information, which was used as ground truth. During the evaluation, two results were investigated. First the initial bag-of-words search and then improvements gained through re-ranking. In our experiments the area where the localization is done covers approximately eight times ten kilometers, with over 1.2 million patches. But since the bounding box of the city covers more than the urban areas, not all patches have visible points inside. In our case roughly 460,000 of the patches have visible feature such that they are electable by the localization method.

The first set of experiments was aimed at testing the performance of the first search step. For a test image, each patch is ranked based on the number of votes. As a measure of the result, we used the share of all test images with its ground truth patch within the top-k ranked ones, for $k \in \{1, 5, 10, 25, 50, 100\}$. Further, it was examined if the results improved when ground truth was set to be a larger area than one patch. The experiments also examined if the results improved when an error margin of five pixels were allowed when measuring angles in the query image. The results from the first set of experiments are summarized in Table 1. The results shows that for most of the test images, close localization in the top five ranked patches can be achieved without any re-ranking of the result. One can also see that allowing for some error in angle measurements improve results.

The next set of experiments were focused on re-ranking the initial results. Since the method with error margins worked best in the first phase, we use that as base for the re-ranking. The 100 best results of the first localization step were used in the re-ranking. In these tests, the ranking of the first patch that had its center within 16 meters of the GPS coordinates of the image was counted as a correct localization. The results

Top-ranked	Hard 0-patches	Hard 3-patches	Soft 0-patches	Soft 3-patches
1	0.162	0.420	0.172	0.439
5	0.346	0.541	0.371	0.558
10	0.411	0.599	0.433	0.607
25	0.481	0.661	0.499	0.668
50	0.524	0.706	0.546	0.712
100	0.570	0.753	0.592	0.756

Table 1: Result without re-ranking. Two different methods have been tried. The first method, called hard, is when no error margins are used. In the method called soft an error of five pixels is accepted. 0-patches and 3-patches stands for how far from the correct patch we consider a correct search. In 0-patches, only the exact patch is accepted, and in 3-patches a difference of three patches from the correct patch is allowed.

of the re-ranking are plotted to the left in Figure 3. The dashed red line is the result before re-ranking and the solid blue after. As can be seen, re-ranking improves the result significantly. To the right in the same figure is a plot showing the result when different number of patches are re-ranked. The different levels are, 10, 25, 50 and 100 patches. The solid blue line is the same as in the plot to the left and represents when the top 100 is re-ranked. Performance of localization improves if more patches is used in the re-ranking. Hence it is important to re-rank as many patches as possible, which makes time an issue.

5.2.1 Memory and Time Requirements

The inverted files we used for the experiments, covering the entire city, in total required 3.6 GB of memory. The entire amount of data necessary for the re-ranking step was 16 GB. The execution time for one inverted file query was on average 1.5 seconds using a python implementation, excluding time for feature extraction. The average execution time for re-ranking was 1.85 seconds per image, when re-ranking the top 100 patches. Both the inverted file query and the procedure of re-ranking can be fully run in parallell for faster execution.

6 Conclusions

We have introduced a novel method for global localization on a city-scale level. The main contributions of the paper are, (i), a method to create synthetic views that holds a rich level of geometric information that can be used in a fast initial localization step, and (ii), a new re-ranking procedure that finds a set of geometrically consistent matches between query image and model. These two contributions combined makes it possible to do global localization on a larger scale than before.

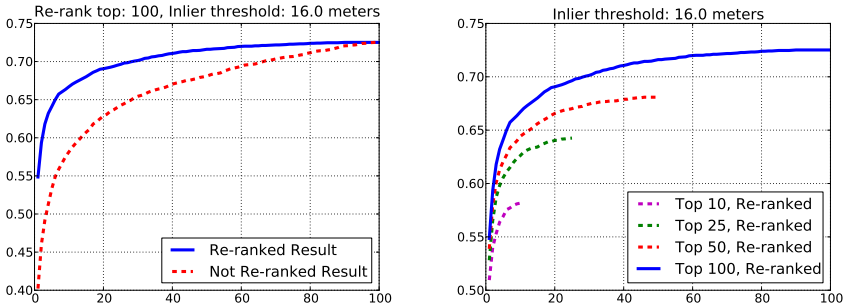

Figure 3: Result of re-ranking. In the left figure is a comparison of results without re-ranking and when the top 100 has been re-ranked. The y-axis is the share of images that have been ranked close enough to ground truth. The x-axis represents that the x-best patches are considered. To the right is a comparison when different number of patches have been re-ranked.

Experiments show that the method is able to find the correct location in 55% of all cases. And among the top twenty ranked positions, the correct position is present for almost 70% of the query images.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proc. 12th Int. Conf. on Computer Vision, Kyoto, Japan*, 2009.

[2] O. Chum, M. Perd'och, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17–24, 2009.

[3] O. Enqvist, K. Josephson, and F. Kahl. Optimal correspondences from pairwise constraints. In *Proc. 12th Int. Conf. on Computer Vision, Kyoto, Japan*, 2009.

[4] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, 1997.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition.

[6] M. Havlena, A. Torii, J. Knopp, and T. Pajdla. Randomized structure from motion based on atomic 3d models from camera triplets. In *Proc. Conf. Computer Vision and Pattern Recognition, Miami, Florida, USA*, pages 2874–2881. IEEE Computer Society, 2009.

[7] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *Proceedings of Computer Vision and Pattern Recognition Conference*, pages 2599–2606, 2009.

[8] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, volume I of *LNCS*, pages 304–317, oct 2008.

[9] R. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations (Symposium Proceedings)*, 1972.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[11] 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.

[12] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Proc. Conf. Computer Vision and Pattern Recognition*, 2003.

[13] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.

[14] A. Murillo Arnal and J. Kosecka. Experiments in place recognition using gist panoramas. In *Proceedings of The Ninth Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*, 2009.

[15] P. Newman, J. J. Leonard, J. D. Tardos, and J. Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Proc. Int. Conf. on Robotics and Automation, Washington D.C., USA*, pages 1802–1809, 2002.

[16] K. Ni, A. Kannan, A. Criminisi, and J. Winn. Epitomic location recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:2158–2167, 2009.

[17] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, June 2006.

[18] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. In *Visual Perception, Progress in Brain Research*, 2006.

[19] D. Robertson and R. Cipolla. An image-based system for urban navigation. In *BMVC*, pages 819–828, 2004.

[20] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proc. Conf. Computer Vision and Pattern Recognition, Minneapolis, USA*, pages 1–7, 2007.

[21] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, 2003.

[22] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, November 2008.

[23] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[24] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling features for large scale partial-duplicate web image search. In *Proc. Conf. Computer Vision and Pattern Recognition, Miami, Florida, USA*, pages 25–32, 2009.

[25] W. Zhang and J. Kosecka. Image based localization in urban environments. In *3DPVT*, pages 33–40, 2006.

[26] Z. Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H. S. Sawhney. Real-time global localization with a pre-built visual landmark database. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, volume 0, pages 1–8. IEEE Computer Society, 2008.