# New Results on Triangulation, Polynomial Equation Solving and Their Application in Global Localization

Josephson, Klas

2008

# New Results on Triangulation, Polynomial Equation Solving and Their Application in Global Localization

Klas Josephson

Lund Institute of Technology
Lund University

Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Licentiate Theses in Mathematical Sciences 2008:7
ISSN 1404-028X

ISBN 978-91-633-2870-1
LUTFMA-2029-2008



Printed in Sweden by MediaTryck, Lund 2008

# Preface

This thesis considers the problem of image based localization. This means that the objective is to find the position and the direction from which an image was taken. To achieve this two subproblems are first studied, triangulation and polynomial equations. In the last paper a complete localization system is constructed using improvements to the pose problem.

The work of this thesis has been founded by the Swedish Research Council through grant no. 2004-4579 'Image-Based Localisation and Recognition of Scenes'.

The thesis consists of the following three papers:

- K. Josephson, F. Kahl, Triangulation of Points, Lines and Conics, to appear in *Journal of Mathematical Imaging and Vision*, 2008.
- M. Byröd, K. Josephson, K. Åström, Fast and Stable Polynomial Equation Solving for Computer Vision, manuscript for *International Journal of Computer Vision*, 2008.
- K. Josephson, M. Byröd, F. Kahl, K. Åström, Localization with Hybrid Features, submitted to *Journal of Mathematical Imaging and Vision*, 2008.

The first paper considers optimal triangulation in arbitrary many views. The second paper concerns solving systems of polynomial equations. This can for example be used for fast and optimal three view triangulation. The last paper considers localization with hybrid features. In this paper the methods of paper two are necessary.

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

- K. Josephson, F. Kahl, Triangulation of Points, Lines and Conics, *Proc. 15th Scandinavian Conference on Image Analysis*, Aalborg, Denmark, 2007.
- 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. International 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.

- M, Byröd, Z. Kukelova, K. Josephson, T. Pajdla, K. Åström, Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion, *Proc. at Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, USA, 2008.

- Z. Kukelova, M, Byröd, K. Josephson, T. Pajdla, K. Åström, Fast and Robust Numerical Solutions to Minimal Problems for Cameras with Radial Distortion, submitted to *Computer Vision and Image Understanding*, 2008.

- M, Byröd, K. Josephson, K. Åström, Fast Optimal Three View Triangulation, submitted to *IPSJ Transactions on Computer Vision and Applications*, 2008

## Acknowledgements

First of all, I would like to thank my supervisor Fredrik Kahl for always being available for discussions and for his careful examination of this manuscript as well as other manuscripts. Also Magnus Oskarsson and Olof Barr 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 and for many discussions on the subject. My gratitude also goes to the Mathematical Imaging Group at the Centre for Mathematical Sciences and especially to Martin Byröd that has been my main collaborator since he joined the research group. 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. The football team at the Centre for Mathematical Sciences also needs to be acknowledged, together with a wish for better results this season.

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

# Introduction

## 1 Background

The scope of this thesis is geometrical computer vision. The geometrical foundation of this field is old and probably begun in 1855 by Chasles [5]. In his paper Chasles looks at the problem of finding the geometry between two cameras if seven point correspondences are known between two images. The next important paper in the field was written by Kruppa in 1913 [20]. 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 [33]. After that the field really took off and the amount of publications has constantly increased.

One of the fundamental problems in geometrical computer vision is the structure and motion problem. Given one or many images of the same object, this refers to the problem to say something about the structure of the imaged object and/or the camera motion between the images. The structure and motion problems can be stated in several different ways depending what is known about the different entities. In some 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 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, for example, 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 information of a 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 is small or if both are large.

This thesis considers several of these structure and motion problems. The first paper deals with the triangulation problem. This is a well studied area in computer vision [18], but still more work is needed especially in the area of globally optimal methods. Globally optimal methods were developed for two and three views for points [17, 32] but for more views or other structures, such as lines and conics, things get more complicated.

The second paper deals with Gröbner basis methods. Gröbner basis methods are used to solve systems of polynomial equations which naturally arise in structure and motion problems. The notion of Gröbner bases was introduced by Buchberger in his thesis from 1965 [2] and builds on methods in algebraic geometry. The Gröbner basis methods give an algorithmic way to solve systems of polynomial equations, but these algorithms require exact arithmetics. To overcome this problem Faugère [10] introduced the F4 method that applies methods from numerical linear algebra. This can be used in simpler problems but the numerical accuracy become problematic even for this method. Due to this problem, specific solvers have been presented inspired by the F4 method. In the second paper of this thesis an improved method to this problem is presented that can handle more difficult problems than previously.

## 1.1 Localization

"Where am I and what am I seeing?" is the question in the localization problem. Many localization system have been constructed over the time *e.g.* [27, 3]. Common for these two are that they only solve the problem in two dimensions. In this thesis the focus instead is on localization in three dimensions. In three dimensions GPS is a competing technic, but the GPS has several drawbacks. For example it is not possible to get the direction from a GPS and it needs an open sky to work.

To solve the localization problem many steps are necessary to carry out. Two of these steps are those addressed in the first two papers. The problem of triangulation is necessary to solve to be able to build a model by the training data and the Gröbner Basis method has shown to be an important step if minimal cases are to be used. The improvements on the Gröbner Basis methods are also possible to apply for better solving the triangulation problem.

The last paper of the thesis builds a complete localization system. To do this several more problems need to be addressed. One of these is the correspondence problem. This problem is not directly addressed in this thesis instead commonly used methods are carried out. The focus is on a set of new minimal cases that are derived and used in the process of localization. By showing how to solve these new minimal cases, we give new contributions towards solving the general localization problem.

Next will be a short introduction to the most fundamental concepts of geometrical computer vision and algebraic geometry. 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 algebraic geometry the concepts of ideal and variety are explained. This is followed by the basics on Gröbner bases, and an introduction to the important concept the action matrix. For the localization problem the correspondence problem is shortly reviewed along with outliers handling using RANSAC.

# 2 Geometry in Computer Vision

## 2.1 The Pinhole Camera

One of the foundations of geometrical computer vision is the pinhole camera. 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. There $\mathbf{C}$ is the focal point and $\pi$ 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.

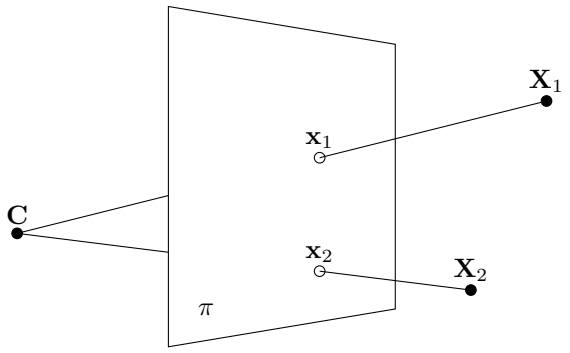



Figure 1: The geometry of a pinhole camera. $\mathbf{X}_i$ are points in the three dimensional space projected to the image points, $\mathbf{x}_i$ on the image plane $\pi$. $\mathbf{C}$ is the focal point, also referred to as the camera center.

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

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

In this equation $\lambda$ is the distance between the focal point and the world point and both the image and the world point are in homogeneous/extended coordinates. $P$ is the camera matrix of size $3 \times 4$ and holds both the intrinsic and extrinsic parameters of the camera. To understand why (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 looks as in Figure 2.

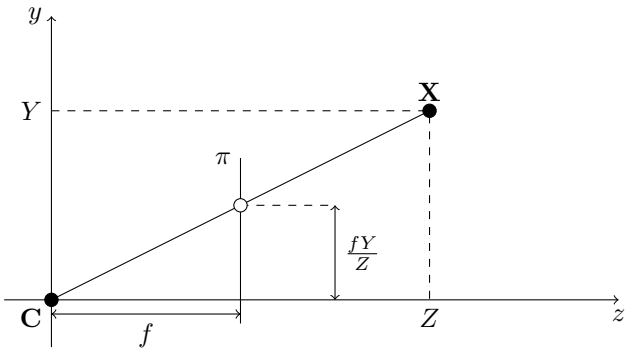



Figure 2: A pinhole camera with the image plane, $\pi$, placed at $z = f$ where $f$ is the focal length of the camera. The focal point $\mathbf{C}$ is placed in the origin and the world point $\mathbf{X}$ is projected to the image plane.

As can be seen in Figure 2 a point $\mathbf{X} = (0, Y, Z)^T$ is mapped 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,

$$(X, Y, Z)^T \mapsto (\frac{fX}{Z}, \frac{fY}{Z})^T. \tag{2}$$

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

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3}$$

The principal point is the orthogonal projection of the focal point on the image plane. During this derivation it is 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 (3) can be adjusted to account for different principal points by putting

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{4}$$

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

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}. \tag{5}$$

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, $\gamma$. If the pixels are square than $\gamma$ 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,

$$K = \begin{bmatrix} f & s & p_x \\ & \gamma f & p_y \\ & & 1 \end{bmatrix}. \tag{6}$$

This is the full calibration matrix that is used to model a pinhole camera. The only thing 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,

$$P = K[R \mid \mathbf{t}], \tag{7}$$

where $R$ is an orthogonal matrix and $t$ a translation vector. This is the the camera matrix for an arbitrary pinhole camera. With this camera (1) represents a projection through a pinhole camera in homogenous coordinates. This model of a camera is used throughout this thesis. For a more exhausted derivation of the camera equation see [18].

## 2.2 Two View Geometry

Geometrical computer vision often includes several cameras, or several images with different views with 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 to 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 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 a epipolar line in the other image. Furthermore the epipolar line always intersects the epipole.

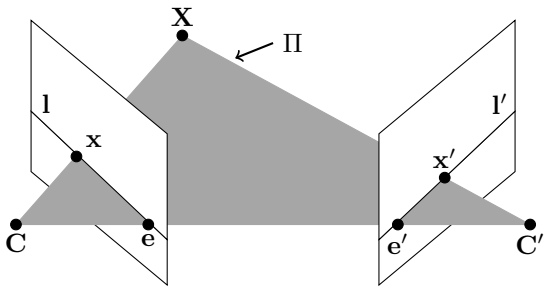



Figure 3: A geometry with two cameras and a point $\mathbf{X}$ projected down to $\mathbf{x}$ and $\mathbf{x}'$, respectively, in the cameras. The baseline between the two camera centra, $\mathbf{C}$ and $\mathbf{C}'$, intersect the image planes in the epipoles, $\mathbf{e}$ and $\mathbf{e}'$. The lines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

### 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 generate the corresponding epipolar line in the other image and the fundamental matrix $F$ will map points to lines in homogeneous coordinates according to,

$$\mathbf{l}' = F\mathbf{x}. \tag{8}$$

According 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

$$\mathbf{x}'^T\mathbf{l}' = \mathbf{x}'^T F\mathbf{x} = 0, \tag{9}$$

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 (9). 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.

The fundamental matrix has seven degrees of freedom. The nine elements of the fundamental matrix encodes these seven degrees of freedom since it is homogenous and that $\det(F) = 0$ 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 has only 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 but since the essential matrix only is given up to scale it only remains five degrees of freedom.

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

$$\hat{\mathbf{x}}'^T E \hat{\mathbf{x}} = 0, \tag{10}$$

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,

$$E = K'^T F K. \tag{11}$$

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 zero [18].

In [28] Nistér uses the property,

$$2EE^T E - \mathrm{tr}(EE')E = 0, \tag{12}$$

introduced by Demazure [8] and later also used by [9], 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 [22, 30]. This method is also used in 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.

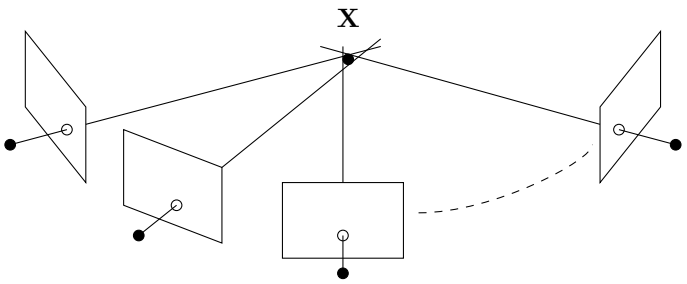



Figure 4: Sketch of multiple view triangulation. If there were no noise, all lines would intersect in a single point, but due to noise this is not true. The problem of triangulation is to find the point $\mathbf{X}$ that is the best approximation to such a point given some type of measure.

The triangulation problem is well studied and hence a lot of literature can be found, see [18, 26] and the references therein. In the absence of noise this problem is trivial and can be solved with two views by constructing the matrix,

$$A = \begin{bmatrix} x\mathbf{p}_3^T - \mathbf{p}_1^T \\ y\mathbf{p}_3^T - \mathbf{p}_2^T \\ x'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ y'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}, \tag{13}$$

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 is the reprojection errors in $L_2$-norm. In recent years also the $L_\infty$-norm have bee given more interest, see *e.g.* [16, 19] but in this thesis the focus is on the $L_2$-norm.

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 [18]. This method is used even though it is well known that the error function will not be convex 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 [17]. The solution is given by calculating the stationary points to the cost function. This leads to solving a polynomial of degree 6. The three view case was solved in 2005 by Stewénius *et al.* [32]. Also this problem is solved by calculating the stationary points to 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 to the second paper of this thesis due to the numerical difficulties they had. The problem of three view triangulation is used as a benchmarking problem in our paper.

If more than three views are used there are to our knowledge no published global methods on closed form. You may argue that the solutions to two and three views are neither closed but they can be solved by an eigenvalue problem. Instead of these methods other approaches need to be taken. This was done in 2006 by Agarwal *et al.* [1]. This method is based on a branch and bound algorithm where the bounding is accomplished by fractional programing. Branch and bound algorithms have theoretically an exponential complexity but if both the branching and bounding are done wisely the complexity can in real examples be considerably lower. In the paper by Agarwal *et al.* it is shown that this indeed is the case with their solution.

### 2.3.1 Triangulation of Other Geometric Structures

In the paper by Agarwal *et al.* only points are considered. But not only points are of interest in the problem of triangulation. Other structures such as lines, conics and arbitrary patches, can be interesting to triangulate. The triangulation of lines in two views gives an exactly determined line in the three dimensional space, since a line in space has four degrees of freedom.

In the first paper of this thesis an extension to the method used by Agarwal is given that deals with lines and conics that also gives the globally optimal triangulated structures.

# 3 Algebraic Geometry

Due to the structure of the camera equation (1) minimal problems in geometrical computer vision often lead to solving systems of polynomial equations. This is also true if equation (12) 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. With use of a quaternion $q = (a, b, c, d)^T$ and a translation vector $t = (x, y, z)^T$ the camera matrix can then be described as follows:

$$P = \begin{pmatrix} a^2+b^2-c^2-d^2 & 2bc-2ad & 2ac+2bd & x \\ 2ad+2bc & a^2-b^2+c^2-d^2 & 2cd-2ab & y \\ 2bd-2ac & 2ab+2cd & a^2-b^2-c^2+d^2 & z \end{pmatrix}. \quad (14)$$

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,

$$\begin{aligned} f_1(\mathbf{x}) &= 0, \\ &\vdots \\ f_m(\mathbf{x}) &= 0. \end{aligned} \quad (15)$$

To solve systems like this many methods exist, see [4] and references therein. The method studied to solve these problems in this thesis is through algebraic geometry, see [6, 7] for a 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 often are used in the kernel 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 $f_i = 0$ 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 (15) is that a point $\mathbf{x}$ is a zero of (15) 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, that is [7], if $f^m \in I$ for some $m \geq 1$ then $f \in I$, then the converse is also true and the two structures are isomorphic.

The most important property is that the dimension of $\mathbb{C}[\mathbf{x}]/I$ is equal to the number of solutions of (15).

## 3.1 The Action Matrix

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 to a polynomial of degree three. If the polynomial is

$$f(x) = x^3 + a_2x^2 + a_1x + a_0, \tag{16}$$

then the roots to can be found through the so-called companion matrix,

$$\begin{pmatrix} -a_2 & 1 & 0 \\ -a_1 & 0 & 1 \\ -a_0 & 0 & 0 \end{pmatrix}. \tag{17}$$

The eigenvalues of the companion matrix are equal to the zeros of (16). This is easily seen since the characteristic polynomial of (17) is equal to (16). 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 [6, 23]. Consider again the system of polynomial equations in (15), 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 elements evaluated on $V$. To understand this consider an arbitrary polynomial $p(\mathbf{x})$. It can be represented as $p(\mathbf{x}) = c^T\mathbf{b}$, 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 $c$,

$$[p \cdot c^T\mathbf{b}] = [(\mathbf{m}_p c)^T\mathbf{b}] = [c^T\mathbf{m}_p^T\mathbf{b}]. \tag{18}$$

Since its 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,

$$p(\mathbf{x})\mathbf{b}(\mathbf{x}) = \mathbf{m}_p^T\mathbf{b}(\mathbf{x}), \quad \text{for } \mathbf{x} \in V. \tag{19}$$

This is recognized as an eigenvalue problem for $\mathbf{m}_p^T$. Hence the eigenvalues of $\mathbf{m}_p^T$ is 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 equivalent 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 [7]. The key idea is to use multivariate polynomial division. A Gröbner basis exists for each ideal and yields a well defined reminder 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 [7] that is guaranteed to find this basis. This method unfortunately only works with exact arithmetics. The Gröbner basis is also dependent of monomial order that is needed to define a consistent division of multivariate polynomials.

The reason way Buchberger's algorithm only works with exact arithmetics is that in each step a check is done if a remainder is zero. With floating points arithmetics it is impossible to be certain about that since the calculations always are plagued by round of errors. There has been work to overcome this problem by Faugère [10]. The main idea of his work is to do many reductions in one step using well studied methods of linear algebra. This is done by putting the system in (15) on the form,

$$C\mathbf{X} = 0, \tag{20}$$

where $\mathbf{X} = [\mathbf{x}^{\alpha_1}, \ldots, \mathbf{x}^{\alpha_n}]^T$ is a vector holding all occurring monomials and $\mathbf{C}$ is a coefficient matrix. With this notion all operation may be done on the coefficient matrix $\mathbf{C}$ 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 [29], autocalibration with radial distortion [21], relative pose with unknown but common focal length [30], optimal three view triangulation [32], 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 used 128 bits mantissa on the floats to be able to get accurate results, which resulted in an extremely slow solver.

## 4 Global Localization

The problem of global localization is also known as the kidnapped robot problem. The reason for this is that the problem to solve is to find the location of the robot where the

only knowledge is some kind of model. In this thesis it is done by using an image so the problem can be formulated as:

**Problem 4.1** (Localization)**.** *Given a query image and a model of the 3D scene structure, find where the image was taken and how the camera was oriented relative to the model.*

In general it is not necessary that the query is in form of an image, even other methods have been used such as laser sensors [3].

To solve the localization problem several methods from computer vision have to be used. The triangulation problem, that is the subject for the first paper of this thesis, is an important step in the construction of a model. The Gröbner Basis methods have also been showed to be useful in this type of problem. Also other problems have to be solved among those is the correspondence problem. The correspondence is the problem of finding common points between a query image and the model. This problem is not directly considered in this thesis instead a widely used method is applied, the method includes solving many minimal geometrical problems.

With the triangulation and correspondence problems and several minimal cases that are solved by using Gröbner Basis technics it is possible to build a global localization system. This is the theme of the last paper of this thesis. In that paper several new minimal cases are derived that are shown useful in global localization.

In the next two subsections a short introduction to the correspondence problem and to minimal problems are given.

## 4.1 The Correspondence Problem

The correspondence problem is, given two or more images of the same scene find a set of points which can be identified as the same points in another image. This problem is not directly addressed in this thesis. Instead one of the commonly used methods in the computer vision community is used. This means that interest points are located with the Maximally Stable Extremal Regions (MSER) method [25]. This method looks for regions that have similar intensity. These regions are then used as interest points, where the points is the center of the regions. For each such point a descriptor is calculated that makes it possible to locate the point in the model. The descriptor used is the SIFT descriptor [24]. 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 are used for matching.

When the matching between model and query image has been done there will be a large number of outliers in a vast majority of the cases. One method to handle the outliers is to use the RANSAC method (RANdom SAmple Consensus). RANSAC was introduced by Fischler and Bolles [11]. RANSAC 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 be true inliers. On this larger set it is now possible to apply other methods in order to calculate a better solution, typically involving local non-linear optimization. In Figure 5 an example of fitting lines to data with outliers using RANSAC is shown.

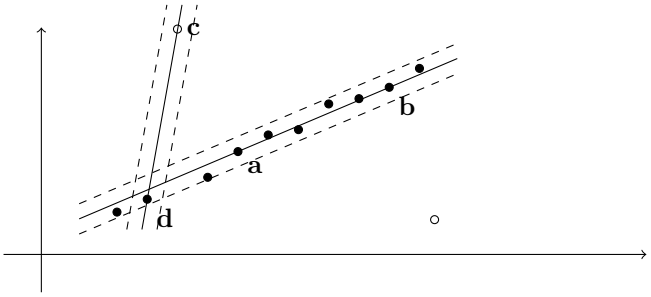



Figure 5: Estimation of a line using RANSAC: The open points are outliers and the solid points inliers. A random minimal set of two points is chosen and an exact fitting is made to this minimal set. One random set is $(\mathbf{a}, \mathbf{b})$ and the other $(\mathbf{c}, \mathbf{d})$. In both cases a 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 10 inliers whereas $(\mathbf{c}, \mathbf{d})$ only results in two. Hence the inlier set of $(\mathbf{a}, \mathbf{b})$ is chosen.

## 4.2 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. One well know minimal case for pose problems is if the camera is calibrated, *i.e.* $K$ in equation (6) is known. In this case it is only necessary to have three correspondences between the image and known world points to decide the pose of the camera, see [15]. On the other hand, this does not give a single solution, as in Figure 5, instead it can give up to four possible solutions.

The given minimal case is applicable if the model holds three-dimensional points. Another way to attack the problem is to let the model hold other cameras with points in those images. Then the problem becomes to decide the relative pose between two cameras.

**Problem 4.2** (Relative Pose)**.** *Given a model with known cameras and known image points in those cameras, find where a query image was taken and how the camera was orientated.*

This problem was as mentioned before almost solved by Kruppa [20] in 1913 and corrected by Thomson in 1959 [33] given that calibrated cameras are used. If uncalibrated cameras are used, the problem was solved by Chasles [5] in 1855. This problem requires at least seven corresponding points and can give up to three possible solutions.

It is not necessary that the camera is either calibrated or uncalibrated. There are also levels in between. One of these cases of semi-calibrated camera is when the inner calibration of the camera is known except for the focal length. In this case it is still possible to solve the pose problem with a minimal setup, which was shown by Stewénius *et al*. in [30]. The minimal setup in this case is six corresponding points and it yields 15 solutions.

In the problem of relative pose it is not necessary that all correspondences refer to the same camera. If this is not the case Stewénius *et al*. [31] solved the problem and showed that there can be up to 64 solutions if 6 correspondences are used when the unknown camera is calibrated.

# 5 Summary of the Papers

## PAPER I — Triangulation of Points, Lines and Conics

This paper approaches one of the most fundamental problems in computer vision, the triangulation problem. The paper continues on a work by Agarwal *et al*. [1] presented in 2006, where a method was presented that guaranteed a globally optimal solution for triangulation of points in arbitrary many views, if the error measure was the $L_2$-norm of the reprojection errors. The method is based on a branch and bound algorithm where the under estimators are calculated through fractional programming.

In the paper the method is extended in two directions. First it is shown how to handle uncertainties in measurements of the image objects. It is also shown how the method can be applied to lines and conics. In the line case Plücker coordinates [18] are used which leeds to a more complicated relaxation in the bounding step of the algorithm.

**Author contribution:** This paper is the result of work carried out by myself and my supervisor Fredrik Kahl. I acted as first author and carried out the experiments.

## PAPER II — Fast and Stable Polynomial Equation Solving for 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 [32] Stewénius *et al*. were forced to use emulated floats with higher accuracy than usual, which resulted in a very slow algorithm. Another example of numerical instability is the paper [22] where Kukelova *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 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.

## PAPER III — Localization Using Hybrid Feature Correspondences

In the last paper of this thesis the pose problem is considered. Several new minimal cases are considered with different degree of calibration of the camera. 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 by Gröbner basis methods and in some cases the improved methods of Paper II were necessary to achieve a solution.

**Author contribution:** This is a joint work of myself, Martin Byröd and our supervisors Fredrik Kahl and Kalle Åström. I acted as first author and carried out most of the experiment with aid primary by Martin.

# Bibliography


[1] Sameer Agarwal, Manmohan Krishna Chandraker, Fredrik Kahl, David J. Kriegman, and Serge Belongie. Practical global optimization for multiview geometry. In *Proc. 9th European Conf. on Computer Vision, Graz, Austria*, pages 592–605, 2006.

[2] B. Buchberger. *On finding a vector space basis of the residue class ring modulo a zero dimensional polynomial ideal (German)*. PhD thesis, University of Innsbruck, Austria, 1965.

[3] W. Burgard, A.B. 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.

[4] Eduardo Cattani, David A. Cox, Guillaume Chèze, Alicia Dickenstein, Mohamed Elkadi, Ioannis Z. Emiris, André Galligo, Achim Kehrein, Martin Kreuzer, and Bernard Mourrain. *Solving Polynomial Equations: Foundations, Algorithms, and Applications (Algorithms and Computation in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

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

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

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

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

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

[10] 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.

[11] 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.

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

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

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

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

[20] 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.

[21] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[22] 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.

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

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 2004.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

[redacted]

# PAPER I

To appear in *Journal of Mathematical Imaging and Vision*, 2008.

# Triangulation of Points, Lines and Conics

Klas Josephson and Fredrik Kahl

## Abstract

*The problem of reconstructing 3D scene features from multiple views with known camera motion and given image correspondences is considered. This is a classical and one of the most basic geometric problems in computer vision and photogrammetry. Yet, previous methods fail to guarantee optimal reconstructions - they are either plagued by local minima or rely on a non-optimal cost-function. A common framework for the triangulation problem of points, lines and conics is presented. We define what is meant by an optimal triangulation based on statistical principles and then derive an algorithm for computing the globally optimal solution. The method for achieving the global minimum is based on convex and concave relaxations for both fractionals and monomials. The performance of the method is evaluated on real image data.*

## 1 Introduction

Triangulation is the problem of reconstructing 3D scene features from their projections. Naturally, since it is such a basic problem in computer vision and photogrammetry, there is a huge literature on the topic, in particular, for point features, see [7, 13]. The standard approach for estimating point features is a two-step approach:

(i) Use a linear least-squares algorithm to get an initial estimate.

(ii) Refine the estimate (so called *bundle adjustment*) by minimizing the sum of squares of reprojection errors in the images.

This methodology works fine in most cases. However, it is well-known that the cost-function is indeed non-convex and one may occasionally get trapped in local minima [6]. The goal of this paper is to derive the statistically optimal cost-function, cf. [12], and an algorithm which gives the globally optimal solution with a guarantee. The algorithm may serve as a benchmarking tool for other methods which are non-optimal. On the

other hand, as we shall see, the developed algorithm shows good empirical performance on real data and it can be used for reliably computing optimal estimates in a structure from motion system.

In [6], the two-view triangulation problem for points was treated. The solution to the optimal problem was obtained by solving a sixth degree polynomial. This was generalized for three views in [16], but the resulting polynomial system turns out to be of very high degree and the solution method based on Gröbner bases becomes numerically unstable. Even though the state-of-the-art methods have improved in this area [2], this approach is not generalizable for an arbitrary number of views. In [11] convex linear matrix inequalities relaxations are used to approximate the non-convex cost-function (again, in the point case), but no guarantee of actually obtaining the global minimum is provided. For line and conic features, the literature is limited to closed-form solutions with algebraic cost-functions and to local optimization methods, see [7] and the references therein. Global optimization techniques have also been applied to other problems in multiple view geometry, see [10] as well as the survey paper [5].

In this paper, we present a common framework for the triangulation problem for any number of views and for three different feature types, namely, points, lines and conics. An algorithm is presented which yields the global minimum of the statistically optimal cost-function. Our approach is most closely related to the work in [9], where fractional programming is used to solve a number of geometric reconstruction problems including triangulation for points. Our main contributions are the following. First, we show how a covariance-weighted cost-function - which is the statistically correct thing to consider - can be minimized globally. For many feature detectors, e.g., [4, 19], it is possible to obtain information of position uncertainty of the estimated features. Second, we present a unified framework for the triangulation problem of points, lines and conics and the corresponding optimal algorithms. Finally, from an algorithmic point of view, we apply convex and concave relaxations of monomials in the optimization framework in order to handle the Plücker constraints for line features. To our knowledge, global optimization over the Plücker manifold has not been done previously in multiple view geometry.

The paper is organized as follows. Section 2 contains a recapitulation on projective geometry and how points, lines and conics projection can be written in a similar manner. In Section 3 we give expressions for the geometrical reprojection error and state the problem we solve. Further on in Section 4 the optimization using a branch and bound algorithm is described, including calculation of a lower bound for the objective function. In Section 5 evaluation of the method is presented with experiments on three different data set.

## 2 Projective Geometry

In the triangulation of points, lines and conics, it is essential to have the formulation of the projection from the three dimensional space to the two dimensional image space in the

same way as the standard projection formulation used in the point case. For that reason we begin with a short recapitulation of the projection of points with a standard pinhole camera. After that the methods to reformulate the projection of lines and quadrics into similar equations are considered. For more reading on projective geometry see [7].

## 2.1 Points

A perspective/pinhole camera is modeled by,

$$\lambda x = PX, \quad \lambda > 0, \tag{1}$$

where $P$ denotes the camera matrix of size $3 \times 4$. Here $X$ denotes the homogeneous coordinates for the point in the 3D space, $X = [U\ V\ W\ 1]^T$, and $x$ denote the coordinates in the image plane, $x = [u\ v\ 1]^T$. The scalar $\lambda$ can be interpreted as the depth, hence $\lambda > 0$ if the point appears in the image. This property is useful in the optimization performed in this paper.

## 2.2 Lines

Lines in three dimensions have four degrees of freedom - a line is determined by the intersection of the line with two predefined planes. The two intersection points on the two planes encode two degrees of freedom. Even if lines only have four degrees of freedom, there are no universal way of representing every line in $\mathbb{P}^4$. One alternative way to represent a line is to use Plücker coordinates. With Plücker coordinates, the line is represented in an even higher dimensional space $\mathbb{P}^5$. The over parameterization is hold back by a quadratic constraint that has to be fulfilled for every line.

The Plücker coordinates to a line in three dimensional space can be obtained from the Plücker matrix,

$$L = AB^T - BA^T, \tag{2}$$

where $A$ and $B$ are two arbitrary 3D points in homogeneous coordinates on the line. It is easy to see that $L$ is a skew-symmetric $4 \times 4$ matrix. Further on are the six Plücker coordinates defined as the elements[1],

$$\mathcal{L} = \{l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}\} \tag{3}$$

of $L$.

From the fact that $L$ has rank 2 and hence $\det L = 0$ it follows that the coordinates satisfy the constraint

$$l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0. \tag{4}$$

[1] Alternative definitions are also used.

The Plücker coordinates are also easy to deal with when changing coordinate systems. If a points $X$ transforms according to $X' = HX$, the construction of the Plücker matrix gives,

$$\begin{aligned} L' &= (HA)(HB)^T - (HB)(HA)^T = \\ &= H(AB^T)H^T - H(BA^T)H^T = \\ &= H(AB^T - BA^T)H^T = HLH^T. \end{aligned} \tag{5}$$

Thus the Plücker matrix transforms as $L' = HLH^T$. Further, it is easy to show that the coordinates are independent of the choice of points defining the line.

One advantage with the Plücker representation of lines is that it is possible to construct a Plücker camera, $P_{\mathcal{L}}$, that makes it possible to write the projection formula in a similar way as in the point case. The Plücker camera equation looks like,

$$\lambda l = P_{\mathcal{L}} \mathcal{L}, \tag{6}$$

where $P_{\mathcal{L}}$ is the so called Plücker camera. The Plücker camera is given by

$$P_{\mathcal{L}} = \begin{pmatrix} p_2 \wedge p_3 \\ p_3 \wedge p_1 \\ p_1 \wedge p_2 \end{pmatrix}, \tag{7}$$

where $p_i^T$ denote the rows of the point camera matrix and $p_i \wedge p_j$ denotes the Plücker line constructed by $p_i$ and $p_j$. The Plücker camera is hence a $3 \times 6$ matrix with elements that are quadratic in the elements of the standard camera matrix. Further, $l$ denotes the projected image line, represented by a $3 \times 1$ vector.

One difference, however, in the projection equation for lines compared with that of points is that it is not possible to interpret the scale $\lambda$ as depth.

## 2.3 Conics

As for lines, we are interested in writing the projection of a quadric to an image conic in the form of the projection formula for points. Before that we first make a short recapitulation on conics and quadrics.

A general conic in the plane is defined by the quadratic form

$$x^T c x = 0. \tag{8}$$

Here $x$ are points in homogeneous coordinates on the edge of the conic and $c$ is a $3 \times 3$ symmetric matrix. In a similar manner a quadric in space is defined by the quadratic form,

$$X^T C X = 0, \tag{9}$$

where $X$ is a point in homogeneous coordinates and $C$ a $4 \times 4$ symmetric matrix.

When quadrics are projected through a pinhole camera (1), is it a lot easier to work with the duals to the conics and quadrics. These duals are the envelopes of the structures. For conics, the envelope consists of lines and for quadrics, the envelope consist of all planes tangent to the quadric locus. Provided the quadrics and conics are non-degenerate, one can show that the equations for the duals are

$$U^T L U = 0, \tag{10}$$

where $U$ are homogeneous plane coordinates and $L = C^{-1}$. Similar for conics, one gets

$$u^T l u = 0, \tag{11}$$

where $u$ are homogeneous line coordinates and $l = c^{-1}$. The projection for the envelope forms are

$$\lambda l = P L P^T \quad \lambda \neq 0. \tag{12}$$

Now we want to reformulate (12) so it appears in a similar form as the point projection formula. This is indeed possible.

**Lemma 2.1.** *The projection of a quadric,*

$$\lambda l = P L P^T, \tag{13}$$

*can be written*

$$\lambda \tilde{l} = P_C \widetilde{L}. \tag{14}$$

*where $\tilde{l}$ and $\widetilde{L}$ denote column vectors of length 6 and 10 obtained from stacking the elements in $l$ and $L$. $P_C$ is an $6 \times 10$ matrix. The entries in $P_C$ are quadratic expressions in $P$.*

*Proof.* Equation (13) can be written as a tensor product, $\lambda \hat{l} = (P \otimes P)\hat{L}$, where $\hat{l}$ and $\hat{L}$ denote the stacked columns of $l$ and $L$. Due to the symmetry in $l$ and $L$ it can be rewritten as (14). □

As for the line case, it is not possible to make the interpretation that the scalar $\lambda$ of the projected conic corresponds to the depth.

# 3 Triangulation

In triangulation the goal is to reconstruct a three dimensional scene from measurements in $N$ images, $N \geq 2$. The camera matrices $P_i$, $i = 1 \ldots N$, are considered to be known for each image. In the point case, the camera matrix can be written $P = (p_1, p_2, p_3)^T$, where $p_j$ is a 4–vector. Let $(u, v)^T$ denote the image coordinates and $X = (U, V, W, 1)^T$ the extended coordinates of the point in 3D. This gives the reprojection error

$$r = \left( u - \frac{p_1^T X}{p_3^T X},\ v - \frac{p_2^T X}{p_3^T X} \right). \tag{15}$$

Further, $\sum_{i=1}^{N} \|r_i\|_2^2$ is the objective function to minimize if the smallest reprojection error is to be achieved in $L_2$-norm. To use the optimization algorithm proposed in this paper (see next section), it is necessary to write the error in each image as a rational function $f/g$ where $f$ is convex and $g$ concave.

The residual $r$ in (15) can be rewritten as

$$r = \left( \frac{u\, p_3^T X - p_1^T X}{p_3^T X}, \frac{v\, p_3^T X - p_2^T X}{p_3^T X} \right), \tag{16}$$

is it easy to see that the $L_2$-norm of the residual can be written as $\|r\|^2 = ((a^T X)^2 + (b^T X)^2)/(p_3^T X)^2$, where $a$, $b$ are 4-vectors determined by the image coordinates and the elements of the camera matrix. By choosing $f = ((a^T X)^2 + (b^T X)^2)/(p_3^T X)$ (with the domain $p_3^T X > 0$) and $g = p_3^T X$, one can show that $f$ is indeed convex and $g$ concave. It is straight forward to form the same residual vectors in the line and conic cases - the only difference is that the dimension is different.

## 3.1 Error Distribution

In the remainder of this paper, it will be assumed that the errors on the residual elements are Gaussian. This is true when triangulating points given that the errors that appear in the image plane are normally distributed. For lines and conics this will not be entirely true [8] but we argue that it is an good approximation when the magnitude of the errors is small.

In order to validate this claim, the following experiments were performed: 100,000 random images were created consisting of one conic and one line. These structures were than sampled with 21 and 63 points for the line and the conic, respectively. Gaussian noise was then added to these points with a standard deviation corresponding to one pixel in an image with a resolution $1000 \times 1000$ pixels. In the next step, a line and a conic were fitted to these points using least-squares. In the line case, the scale was adjusted such that the first two coordinates have a norm of one and translated such that the distance to the origin was one in order to avoid passing through the origin (since this would mean that the third coordinate would be zero).

As can be seen in Figure 1 the elements in the line vector residual is not perfectly normally distributed but it is an good approximation. In the conic case, Figure 2 shows that two of the elements are closer to be perfect normally distributed. These two elements correspond to the center of the conic divided by a scale factor. The other three elements are close to normal distributions even though they are not as close as the other two.

## 3.2 Incorporation of Covariance

The residual vector (16) may in dimension independent notation be written

$$r = \left( \frac{x_1 p_n^T X - p_1^T X}{p_n^T X}, \ldots, \frac{x_{n-1} p_n^T X - p_{n-1}^T X}{p_n^T X} \right). \tag{17}$$

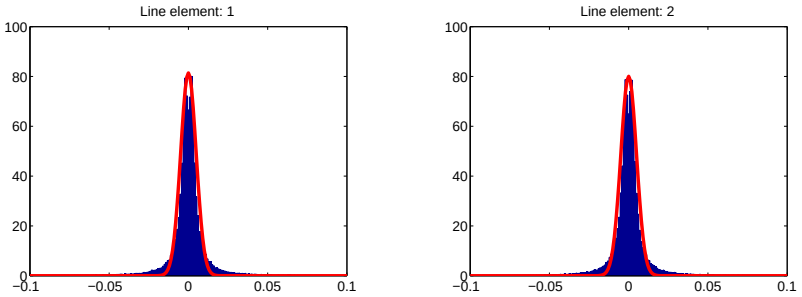



Figure 1: The distributions of the errors in the two residuals for line case when Gaussian noise is added to 21 sample points of the true line. A Gaussian distribution is overlaid to the histogram to compare with.

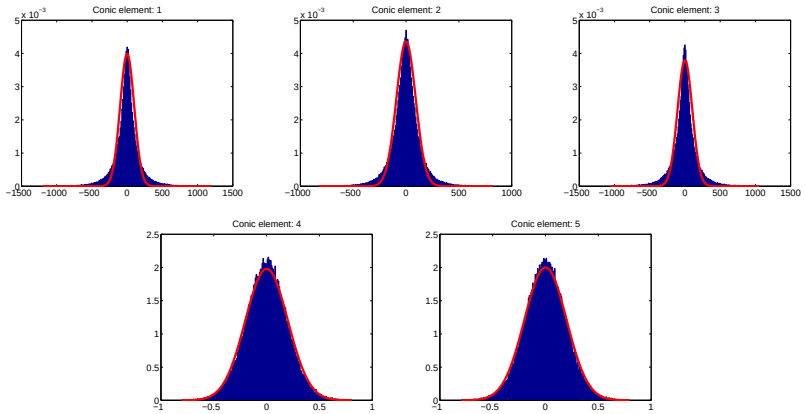



Figure 2: The distributions of the errors in the five elements of the conic residual vector when Gaussian noise is added to 63 sample points. A Gaussian distribution is overlaid to the histogram to compare with.

The statistically optimal cost-function is to weight the residual vector by its covariance [12] (at least to a first order approximation). Incorporating covariance weighted error transforms to,

$$\|Lr\| = \left\| L \left( \frac{x_1 p_n^T X - p_1^T X}{p_n^T X}, \dots \right) \right\|, \tag{18}$$

where $L$ is the cholesky factorization of the inverse covariance matrix to the structure in each image. Notice that we have chosen to normalize by the last coordinate and in that case the covariance becomes a $2 \times 2$ symmetric matrix in the point and line cases and a $5 \times 5$ in the conic case. The reason why the covariance matrix is one dimension lower than the image vector is that there is no uncertainty in the last element of the extended image coordinates.

### 3.3 Problem Formulation

For the triangulation of points, lines or conics, the optimization problem to solve can be formulated as follows:

$$\min \sum_{i=1}^{n} \|L_i r_i\|^2. \tag{19}$$

The only thing which differs (except for dimensions) in the different cases is that in the line case it is necessary to fulfill the quadratic Plücker constraint (4) for the coordinates of the three dimensional structure.

## 4 Branch and Bound Optimization

Branch and bound algorithms are used to find the global optimum for non-convex optimization problems. The result of the algorithm is a provable upper and lower bound of the optimum and it is possible to get arbitrary close to the optimum.

On a non-convex, scalar-valued, objective function $\Phi$ at the domain $Q_0$ the branch and bound algorithm works by finding a lower bound to the function $\Phi$ on the domain $Q_0$. The lower bounding function should ideally be a convex function to make it possible to find a minimum reliably. If the difference between the optimum for the bounding function and the lowest value of the function $\Phi$ - calculated so far - is small enough, say $\epsilon$, then the optimum is considered to be found. Otherwise the domain $Q_0$ is splitted into subdomains and the procedure is repeated in these domains. The algorithm repeats until the gap between the bounding function and the value of the objective function is smaller than the predefined value $\epsilon$. Hence, the global optimum is attained within a preset $\epsilon > 0$ which can be chosen arbitrarily small.

If the lower bounding function on a subdomain has its minimum higher than a known value of the objective function in another subdomain, it is possible to neglect the first subdomain since we know that the optimum in that region is greater than the lowest value obtained so far.

## 4.1 Bounding

The goal of the bounding function $\Phi_{\text{lb}}$ is that it should be (i) a close under-estimator to the original objective function $\Phi$ and (ii) easy to compute the lowest value $\Phi_{\text{lb}}$ in a given domain. For practical reasons, convex functions are good candidates as bounding functions. Further, as the domain of the bounding functions is partitioned into smaller regions, the approximation gap to the objective function must converge (uniformly). A good choice of $\Phi_{\text{lb}}$ is the convex envelope, defined as follows.

**Definition 1** (Convex Envelope)**.** Let $f : S \rightarrow \mathbb{R}$, where $S \subset \mathbb{R}^n$ is a nonempty convex set. The convex envelope of $f$ over $S$ (denoted **convenv** $f$) is a convex function such that (*i*) **convenv** $f(x) \leq f(x)\ \forall\ x \in S$ and (*ii*) for any other convex function $u$, satisfying $u(x) \leq f(x)\ \forall\ x \in S$, we have **convenv** $f(x) \geq u(x)\ \forall\ x \in S$.

In Figure 3, an example of a convex envelope of a one dimensional function is plotted.

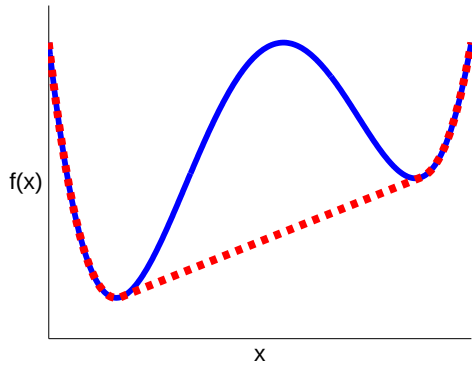



Figure 3: Illustration of the convex envelope; the blue solid line is a function $f(x)$ and the dashed red line is its convex envelope **convenv** $f(x)$.

The concave envelope is defined in a similar manner. It can often be hard to compute the convex envelope (or concave envelope), in fact, it may be as hard as computing the global optimum itself.

**Fractional Relaxation**

Fractional programming is used to minimize/maximize a sum of $p \geq 1$ fractions subject to convex constraints. In this paper we are interested of minimizing

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{p} \frac{f_i(x)}{g_i(x)} \\ \text{subject to} \quad & x \in D, \end{aligned} \tag{20}$$

where $f_i$ and $g_i$ are convex and concave, respectively, functions from $\mathbb{R}^n \rightarrow \mathbb{R}$, and the domain $D \subset \mathbb{R}^n$ is a convex set. On top of this it is assumed that $f_i$ and $g_i$ are positive and that one can compute a priori bounds on $f_i$ and $g_i$. Even under these assumptions it can be shown that the problem is $\mathcal{NP}$-complete [3].

Instead of studying equation (20) more let us rewrite it. Assume that we have bounds on the functions $f_i(x)$ and $g_i(x)$, where the intervals are $[l_i,\ u_i]$ and $[L_i,\ U_i]$, respectively. Let the $2p$-dimensional rectangle $[l_1,\ u_1] \times \cdots \times [l_p,\ u_p] \times [L_1,\ U_1] \times \cdots \times [L_p,\ U_p]$ be denoted $Q_0$. With auxiliary variables $t = (t_1, \ldots, t_p)^T$ and $s = (s_1, \ldots, s_p)^T$ it can be showed that the global optimum to (20) is identic as to the following optimization problem,

$$\begin{aligned} \min_{x,t,s} \quad & \sum_{i=1}^{p} \frac{t_i}{s_i} \\ \text{subject to} \quad & f_i(x) \leq t_i \quad g_i(x) \geq s_i \\ & x \in D \qquad (t,s) \in Q_0. \end{aligned} \tag{21}$$

In [18] it has been shown that the problem of finding the convex envelope to a single fraction $t/s$ as the parts in the sum in (21), where $t \in [l,u]$ and $s \in [L,U]$, is given as the solution of the following *Second Order Cone Program* (SOCP):

$$\begin{aligned} \text{convenv}\left[\frac{t}{s}\right] \quad = \quad & \text{minimize} \ \ r \\ \text{subject to} \quad & \left\| \begin{array}{c} 2\lambda\sqrt{l} \\ r' - s' \end{array} \right\| \leq r' + s' \\ & \left\| \begin{array}{c} 2(1-\lambda)\sqrt{u} \\ r - r' - s + s' \end{array} \right\| \leq r - r' + s - s' \\ \lambda L \leq s \leq \lambda U \quad & (1-\lambda)L \leq s - s' \leq (1-\lambda)U \\ r' \geq 0 \quad & r - r' \geq 0 \\ l \leq t \leq u \quad & L \leq s \leq U \end{aligned} \tag{22}$$

where $\lambda = \frac{u-t}{u-l}$ for ease the notation and $r, r', s'$ are auxiliary scalar variables.

When $\Phi$ is a sum of ratios as in (21) a bound for the function can be calculated as the sum of the convex envelopes of the individual fractions. When summarizing the convex envelopes the sum will, however, generally not be the convex envelope to the function. Still the function will be a lower bound and it fulfills the requirements of a bounding function. This way of calculating $\Phi_{\text{lb}}$ by solving $p$ SOCP problems can be done efficiently [17].

A more exhaustive description on fractional programming in multiple view geometry can be found in [9] where point triangulation (without covariance weighting) is treated.

**Monomial Relaxation**

In the line case the Plücker coordinates have to fulfill the Plücker constraint (4). This gives extra constraints in the problem to find lower bounds.

If we make the choice in the construction of the Plücker coordinates that the first point lies on the plane $z = 1$ and the second on the plane $z = 0$, remember that the Plücker coordinates are independent of the construction points, the two points $X = (x_1,\ x_2,\ 1,\ 1)^T$ and $Y = (y_1,\ y_2,\ 0,\ 1)^T$ gives the following Plücker coordinates for the line (2.2),

$$\mathcal{L} = (x_1y_2 - x_2y_1,\ -y_1,\ x_1 - y_1,\ -y_2,\ y_2 - x_2,\ 1)^T. \tag{23}$$

This parameterization involves that the last coordinate is one and that only the first one is nonlinear to the points of intersection. Thanks to this it is only necessary to make a relaxation of the first coordinate (in addition to the fractional terms).

The choosen parameterization does not work if the line is parallel to the plane $z = 0$. If this is the case we change the coordinate system.

In [14] the convex and the concave envelopes are derived for a monomial $y_1y_2$. The convex envelope is given by

$$\mathbf{convenv}(y_1y_2) = \max \left\{ \begin{array}{c} y_1y_2^U + y_1^U y_2 - y_1^U y_2^U \\ y_1y_2^L + y_1^L y_2 - y_1^L y_2^L \end{array} \right\}. \tag{24}$$

Similarly the concave envelope is

$$\mathbf{concenv}(y_1y_2) = \min \left\{ \begin{array}{c} y_1y_2^L + y_1^U y_2 - y_1^U y_2^L \\ y_1y_2^U + y_1^L y_2 - y_1^L y_2^U \end{array} \right\}. \tag{25}$$

Given bounds on $x_1, x_2, y_1$ and $y_2$ in the parameterization of a line, it is possible to propagate the bounds to the monomials $x_1y_2$ and $x_2y_1$.

## 4.2 Branching

It is necessarily to have a good strategy when branching. If a bad strategy is chosen the complexity can be exponentially but if a good choice is made it is possible to achieve a lower complexity - at least in practice.

A standard branching strategy for fractional programming [1] is to branch on the denominator $s_i$ of each fractional term $t_i/s_i$. This limits the practical use of branch and bound optimization to at most about 10 dimensions [15] but in the case of triangulation the number of branching dimensions can be limited to a fixed number (at most the degree of freedom of the geometric primitive). Hence, in the point case is it enough to branch on three dimensions, in the line case four and in the cases of conics nine dimensions maximally.

In the line case, we choose not to branch on the denominators. Instead the coordinates of the points where the line intersect with the planes $z = 0$ and $z = 1$ are used for the parameterization (4.1). This gives four dimensions to split at, independent of the number of images. It is also important to choose a coordinate system such that the numerical values of these parameters are kept at a reasonable magnitude. When the decision of which rectangle to split is taken there are two choices to make. The first is on which dimension to split and the second where to split in the chosen dimension. A reasonable pick of dimension is the one with the largest interval. The decision to split can be made in several different ways. One alternative is to make use of the bounding function. The hypothesis is that the domain with the minimum bounding function is the best candidate for the minimum of the objective function. To get as good estimation as possible close to that point the splitting location can be chosen close to the minimum of the bounding function. Another way to split is to split the dimension into two equal parts. Both these splitting procedure can be shown to be convergent [1].

## 4.3 Initialization

It is necessary to have an initial domain $Q_0$ for the branch and bound algorithm. The method used for this is similar in the point and conic case but different in the line case due to the Plücker constraint.

### Points and Conics

In order to get a bound on the denominators, we assume a bound on the maximal reprojection error. Thus the bounds are constructed from a user given maximal reprojection error. The bounds on the denominators $g_i(x)$ can then be calculated by the following optimization problem,

$$\begin{aligned} \text{for } i = 1, \ldots, p, \qquad & \min/\max \;\; g_i(x) \\ \text{subject to} \qquad & \frac{f_j(x)}{g_j(x)} \leq \gamma \quad j = 1, \ldots, p, \end{aligned} \tag{26}$$

where $\gamma$ is the user given bound on the reprojection error. This is a quadratic convex programming problem. In the experiments, $\gamma$ is set to 3.

**Lines**

In the case of lines, the Plücker constraint makes things a bit more problematic. Instead we choose a more geometric way of getting bounds on the coordinates of the two points defining the line.

For each image line $l$ we first set the scale such that the norm of the first two coordinates equals one and then we make a translation to avoid the line to pass through the origin. Two parallel lines ($l_1$ and $l_2$) are then constructed with $\gamma$ pixels apart (one on each side of $l$). Then, we make the hypothesis that the two points defining the optimal 3D line (with our choice of coordinate systems) are located on the planes $z = 0$ and $z = 1$, respectively. Now, in order to find bounds on $x_1, x_2, y_1, y_2$, see equation (23), we need to solve the optimization problems,

$$\begin{array}{ll} \text{max/min} & x_i \\ \text{subject to} & l_{1i}^T P_i X \leq 0 \\ & l_{2i}^T P_i X \geq 0, \end{array} \tag{27}$$

where $X = [x_1, x_2, 0, 1]^T$ and the corresponding for $y_i$ with $X$ substituted to $Y = [y_1, y_2, 1, 1]^T$ . Again, it is important to choose the coordinate system such that the planes $z = 0$ and $z = 1$ are located appropriately. In addition, to avoid getting an unbounded feasible region, the maximum depth is limited to the order of the 3D point furthest away. In the experiments, we set $\gamma$ to 5 pixels.

# 5 Experiments

The implementation is made in Matlab using a toolbox called SeDuMi [17] for the convex optimization steps.

The splitting of dimensions has been made by taking advantage of the information where the minimum of the bounding function is located, as described in Section 4.2.

While testing the various cases, we have found that the relaxation performed in the line case - a combination of fractions and monomials - the bounds on the denominators is a critical factor for the speed of convergence. To increase the convergence speed, a local gradient descent step is performed on the computed solution in order to quickly obtain a good solution which can be employed to discard uninteresting domains.

Two public sets of real data[2] were used for the experiments with points and lines. One of a model house with a circular motion and one of a corridor with a mostly forward moving camera motion. The model house has 10 frames and the corridor 11. In these two sequences there were no conics. In Figure 4 samples of the data sets for points and lines are given. A third real data sequence was used for conic triangulation, see Figure 6.

[2] http://www.robots.ox.ac.uk/∼vgg/data.html


Figure 4: Image sets used for the experiments.

Points and lines were reconstructed and then the reprojection errors between different methods were compared. The other methods compared are bundle adjustment and a linear method [7]. The covariance structure for the lines was computed by fitting a line to measured image points. In the reconstruction only the four first frames were used. In the house scene, there are 460 points and in the corridor 490. The optimum was considered to be found if the gap between the global optimum and the under-estimator was less than 10 %. The results are presented in Tables 1 and 2.

| Data set | Optimal | | Bundle | | Linear | |
|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std |
| House | 0.15 | 0.14 | 0.15 | 0.14 | 0.16 | 0.15 |
| Corridor | 0.13 | 0.11 | 0.13 | 0.11 | 0.13 | 0.11 |

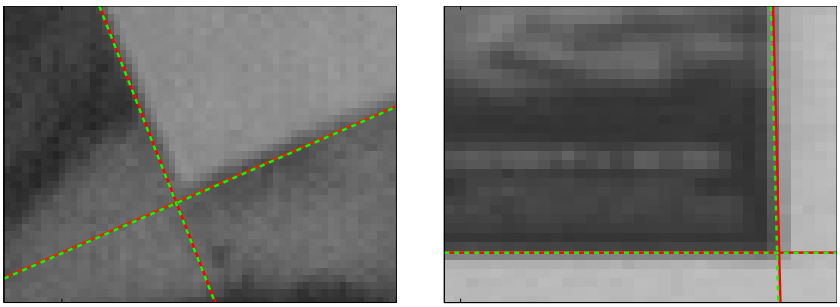
Table 1: Reprojection errors for points with three different methods on two data sets.

| Data set | Optimal | | Bundle | | Linear | |
|---|---|---|---|---|---|---|
| | Mean | Std | Mean | Std | Mean | Std |
| House | 1.40 | 0.92 | 1.41 | 0.93 | 1.62 | 1.03 |
| Corridor | 3.42 | 4.29 | 3.30 | 4.34 | 4.02 | 5.45 |

Table 2: Reprojection errors for lines with three different methods on two data sets.

In the house scene, the termination criterion was reached already in the first iteration for all points and for most of them the bounding functions was very close to the global minimum (less than the 10 % required). In the corridor scene, the average number of iterations were 3 and all minimas were reached within at most 23 iterations.

In the line case, the under-estimators works not as well as in the point case. This is due



(a) (b)

Figure 5: The result from reprojection of lines. The green dashed line is the original and the red solid line is the reprojected. Image (a) is from the house scene and (b) is from the corridor.

to the extra complexity of the Plücker coordinates. This makes it necessary to perform more iterations before it is possible to state that the global optimum is reached within a given certainty. In the house scene with the circular camera motion the breakpoint is reached within at most 120 iterations for all the tested 12 lines. However, for the corridor sequence with a weaker camera geometry (at least for triangulation purposes) it is not even enough with 500 iterations for 6 of the tested 12 lines. Even if a lot of iterations are needed to certify the global minimum, the location of the optimum in most cases is reached within less than a handful of iterations.

It can be seen in Tables 1 and 2 that both a linear method and bundle adjustment work fine for these problems. However, in some cases the bundle adjustment reprojection errors get higher than the errors for the optimal method. This shows that bundle adjustment (which is based on local gradient descent) sometimes gets stuck in a local minimum. The opposite also occurs in some cases: bundle adjustment gets better results than the optimal method. The reason for this is twofold. The first reason is that a threshold is used in the optimal method for the gap between the optimal value and the found value of the objective function. This gap is fixed to 10% in our experiments which leaves a margin. The second reason is that this margin of error is not attained in all line experiments where we instead reach the maximum iteration number.

The result can also be seen in Figure 5 where two lines from each data set are compared with reprojected line.

## 5.1 Conics

For conics, some example images can be found in Figure 6 with marked image conics. The covariance structure was estimated by fitting a conic curve to measured image points. The

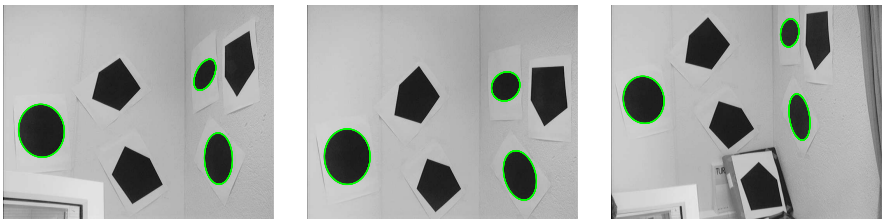


Figure 6: The images with the data used for conics.

corresponding 3D quadrics were first computed with the optimal and a linear method. The result of the reprojected conics from these two methods are imaged in Figure 7.

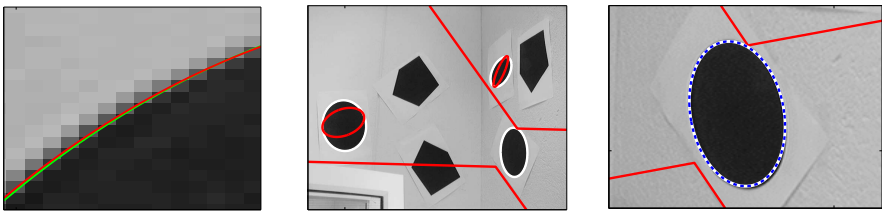
The number of iterations performed to reach the global minimum with a gap less then 5 % of the bounding function for the three conics were 3, 6 and 14. As can be seen from the images, the quadrics in the data set are planar and hence the condition number of the corresponding $4 \times 4$ matrix should be zero. For the three estimated quadrics with the optimal method, the condition numbers are $1.2 \cdot 10^{-3}$, $3.7 \cdot 10^{-7}$ and $8.8 \cdot 10^{-6}$. This can be compared with the result for the linear estimate with condition numbers of $3.7 \cdot 10^{-4}$, $4.1 \cdot 10^{-5}$ and $1.1 \cdot 10^{-4}$.

The result of the optimal method was also compared with bundle adjustment. In the local optimization the result of the linear method were used as initialization. The norm of the residual from the three estimated quadrics are shown in Table 3. As can be seen there the result of the optimal method and bundle adjustment is identical. This shows that bundle adjustment reached the global optimum in all the experiments.

| Conic # | Optimal | Bundle | Linear |
|---|---|---|---|
| 1 | 39.5 | 39.5 | 1190 |
| 2 | 5.00 | 5.00 | 160 |
| 3 | 15.3 | 15.3 | 2660 |

Table 3: The norm of the residuals when quadrics were reconstructed. It is obvious that the linear method isn't good enough and that bundle adjustment reaches the global optimum.

Figure 7 (a) shows the reprojected conic compared with the original for one of the conics. The fitting is very good and it is obvious from Figure 7 (b) that the linear result is far from acceptable. In part (c) of Figure 7 a comparison of the optimal method and bundle adjustment is shown. As can be seen there the result is almost identical.



(a) (b) (c)

Figure 7: The result of the reprojected conics of the data set in Figure 6. In image (a) a part of the reconstruction with optimal method is viewed. The light green line is the reprojection and the dark red the original conic. In (b) the red lines are the reprojection after linear method and the white when the optimal method were used. In (c) the image in (b) is zoomed in. The added blue dashed line is the result from bundle adjustment initialized with the result from the linear method (red), as can be seen this line almosst perfectly coincide with the result from the optimal method.

# 6 Discussion

A unified treatment of the triangulation problem has been described using covariance propagation. In addition to traditional local algorithms and algorithms based on algebraic objective functions, globally optimal algorithms have been presented for the triangulation of points, lines and conics. For most cases, local methods work fine and they are generally faster in performance. However, none of the competing methods have a guarantee of globality.

The performed experiments show that bundle adjustment works well. This conclusion may come as no surprise. It has already been observed in the two-view case for points [6]. Now it is shown that this is true for any number of views for point features. Perhaps the main contribution of this paper is to serve as a benchmarking algorithm of other algorithms since it gives a way to evaluate the performance of other methods.

A future line of research is to include more constraints in the estimation process, for example, planar quadric constraints. This opens up the possibility to perform optimal auto-calibration using the image absolute conic. An other line of research is to improve computational efficiency for global optimal triangulation problems.

# 7 Acknowledgments

This work has been funded by the Swedish Research Council (grants no. 2004-4579, no. 2005-3230 and no. 2007-6476), by the Swedish Foundation for Strategic Research (SSF) through the programme Future Research Leaders, and by the European Research Council (GlobalVision grant no. 209480).

# Bibliography


[1] Harold P. Benson. Using concave envelopes to globally solve the nonlinear sum of ratios problem. *Journal of Global Optimization*, 22:343–364, 2002.

[2] Martin Byröd, Klas Josephson, and Kalle Åström. Improving numerical accuracy of gröbner basis polynomial equation solvers. In *International Conference on Computer Vision*, 2007.

[3] R. W. Freund and F. Jarre. Solving the sum-of-ratios problem by an interior-point method. *J. Glob. Opt.*, 19(1):83–102, 2001.

[4] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[5] R. Hartley and F. Kahl. Optimal algorithms in multiview geometry. In *Asian Conf. Computer Vision*, Tokyo, Japan, 2007.

[6] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

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

[8] Stephan Heuel. *Uncertain Projective Geometry: Statistical Reasoning for Polyhedral Object Reconstruction*, volume 3008 of *Lecture Notes in Computer Science*. Springer, 2004.

[9] F. Kahl, S. Agarwal, M.K. Chandraker, D.J. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. *International Journal of Computer Vision*, 2007.

[10] F. Kahl and R. Hartley. Multiple view geometry under the $L_\infty$-norm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2007. To appear.

[11] F. Kahl and D. Henrion. Globally optimal estimates for geometric reconstruction problems. *Int. Journal Computer Vision*, 74(1):3–15, 2007.

[12] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science, Amsterdam, The Netherlands, 1996.

[13] J.C. McGlone, E.M. Mikhail, J. Bethel, and R. Muellen. *Manual of Photogrammetry*. American Society of Photogrammetry and Remote Sensing , Falls Church, VA, 2004.

[14] Hong Seo Ryoo and Nikolaos V. Sahinidis. Analysis of bounds for multilinear functions. *Journal of Global Optimization*, 19(4):403–424, 2001.

[15] Siegfried Schaible and Jianming Shi. Fractional programming: the sum-of-ratios case. *Optimization Methods and Software*, 18:219–229, 2003.

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

[17] J.F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[18] Mohit Tawarmalani and Nikolaos V. Sahinidis. Semidefinite relaxations of fractional programs via novel convexification techniques. *Journal of Global Optimization*, 20:133–154, 2001.

[19] Bill Triggs. Detecting keypoints with stable position, orientation, and scale under illumination changes. In *European Conf. Computer Vision*, volume 4, pages 100–113, Prague, Czech, 2004.

[redacted]

# PAPER II

Manuscript for *International Journal of Computer Vision* , 2008.

# Fast and Stable Polynomial Equation Solving for Computer Vision

Martin Byröd, Klas Josephson and Kalle Åström

## Abstract

*This paper presents several new results on techniques for solving polynomial problems 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 show that an extension of current state of the art by not only considering reduction based on a Gröbner basis can yield dramatic improvements in numerical stability. Furthermore, 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 stabality 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 particularily true for so called minimal structure and motion problems, *e.g*. [5, 22, 35]. Solutions to minimal structure and motion problems can often be used in RANSAC algorithms to find inliers in noisy data [10, 36, 37]. For such applications one needs to solve a large number of minimal structure and motion problems as fast as possible in order to find the best set of inliers. There is thus a need for fast and numerically stable algorithms for solving particular systems of polynomial equations.

Another area of recent interest is global optimization used for *e.g*. optimal triangulation, resectioning and fundamental matrix estimation. Global optimization is a promising, but difficult pursuit and different lines of attack have been tried, *e.g*. branch and

bound [1], $L_\infty$ norm methods [14, 19] and methods using linear matrix inequalities (LMIs) [20].

An alternative way to find the global optimum is to calculate stationary points directly (usually by solving some polynomial equation system) [15, 34]. 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 should be possible to handle a somewhat larger class of problems, thus offering an alternative to the above mentioned optimization methods. An example of this is optimal three view triangulation which has previously not been solved in a practical way [34]. We show that this problem can now be solved in a reasonably efficient way 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*. [30]. The outline of such algorithms is that one first studies a specific geometric problem and finds out what structure the Gröbner basis 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 [34] 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 pin-point the main source of these numerical problems (the conditioning of a crucial elimination step) and propose a range of techniques for dealing with this issue. The main novelty is a new approach to the action matrix method for equation solving, relaxing the need of adhering to a properly defined monomial order and a complete Gröbner basis. This unlocks substantial freedom, which 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 completely, by using an overly large/redundant (linearly dependent) basis for $\mathbb{C}[\mathbf{x}]/I$ to construct the action matrix. This method yields the right solutions along with a set of false solutions that can then easily be filtered out by evaluation in the original equations.

Secondly, we show how a 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 [25], which makes this technique accessible and relatively straight forward to implement.

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

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

The paper is organized as follows. We give an overview of the classical theory of algebraic geometry underlying the ideas presented in this paper in Section 2 after a brief discussion of related techniques for polynomial equation solving in Section 1.1. Thereafter, in Section 3.1, we present the theoretical underpinnings of the new numerical techniques introduced in Sections 4, 5, 5.3 and 6. In Section 7 we evaluate the speed and numerical stability of the proposed techniques on a range of typical 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 and only few of the available methods have yet found their way into the computer vision community. See *e.g*. [4] 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 essentially works by gradient descent. Newton-Raphson is fast and easy to implement, but depends heavily on initialisation and finds only a single zero for each initialisation. 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 [17].

In several variables, a first method is to use resultants [6], which using a determinant construct enables the successive elimination of variables. However, the resultant grows exponentially in the number of variables and is in most cases not practical for more than two variables. An alternative way of eliminating variables is to compute a lexicographical Gröbner basis for the ideal generated by the equations which can be shown to contain a univariate polynomial representing the solutions [6]. This approach is however often numerically unstable.

A radically different approach is provided by homotopy continuation methods [38]. 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 continously 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 high degrees (up to around 10) and (iii) is well suited to tuning for specific applications. To the authors best knowledge, this method was first used in the context of computer vision by Stewenius *et al*. [30] eventhough Gröbner basis methods were mentioned in [16].

The work presented in this paper is based on preliminary results presented in [2, 3] 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 [6] 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* $I = \Sigma_i h_i(\mathbf{x}) f_i(\mathbf{x})$, 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 which vanish on $V$. If $I$ is equal to this set, then $I$ is called a radical ideal.

We say that two polynomials $f, g$ are equivalent modulo $I$ iff $f - g \in I$ and denote this by $f \sim g$. With this definition we get the quotient space $\mathbb{C}[\mathbf{x}]/I$ of all equivalence classes modulo $I$ and let $[\cdot]$ denote the natural projection $\mathbb{C}[\mathbf{x}] \to \mathbb{C}[\mathbf{x}]/I$, *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. Moreover, if $V$ is a point set, then clearly $\mathbb{C}[V]$ 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

$$p(x) = x^3 + a_2 x^2 + a_1 x + a_0, \tag{2}$$

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 [26]. 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 [6]. For some $p \in \mathbb{C}[\mathbf{x}]$ consider now the operation $T_p : f(\mathbf{x}) \to p(\mathbf{x}) f(\mathbf{x})$. 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 (1). 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 $I$ [6]. 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 is $\mathbf{m}_p$ is to compute the action $\overline{pb_i}^G$ for each basis element $b_i$, which is easily done if $G$ is available.

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

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

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 $T_x[1] = [x]$ 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 [6]. 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 [9] is to write the system of equations (1) on matrix form

$$\mathbf{C}\mathbf{X} = 0, \tag{9}$$

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 [11], fundamental matrix estimation with radial distortion [23], optimal three view triangulation [34], etc.) and use knowledge of what the structure of the Gröbner basis should be to design a special purpose Gröbner basis solver [30]. 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 [34, 24]. 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 $f \in \mathbb{C}[\mathbf{x}]$ and specifically note that we only need to compute $\overline{f}$ for $f \in \mathcal{R}$. It should however be noted that the computations we do much resemble those necessary to get a Gröbner basis.

A first advantage of not having to compute a Gröbner basis is that we can replace the gauss-jordan elimination by standard linear equation solving techniques which are both faster and numerically more sound. Furthermore we are not bound by any particular monomial order which as we will see, when used right, buys considerable numerical stability.

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. We start with a few examples that we will use to clarify these ideas.

**Example 3.** In the five point relative orientation problem for calibrated cameras, *cf*. [22, 7, 27, 31], 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 of the monomials we get a coefficient matrix $\mathbf{C}$ of size $10 \times 20$ 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 $10 \times 10$ block $\mathbf{C}_1$ of $\mathbf{C} = \begin{bmatrix}\mathbf{C}_1 & \mathbf{C}_2\end{bmatrix}$ 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 straightforwad 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 $\mathbf{X}_2$ can be thought of as a basis for the space of remainders after division with a Gröbner basis for one choice of monomial order and this is how these computations have typically been viewed. However, the calculations above are not really dependent on any properly defined monomial order and it seems that they should be meaningful irrespective of whether a true monomial order is used or not. Moreover, we do not use all the Gröbner basis properties.

Based on these observations we emphasize two important facts: (i) We are not interested in finding the Gröbner basis or a basis for the remainder space relative to some Gröbner basis *per se*; it is enough to get a well defined mapping $\overline{f}$ 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 4.** 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 $T_y(1) = y$, $T_y(x) = xy \sim x-y+1$ and $T_y(y) = y^2 \sim xy \sim x-y+1$ which gives a multiplication matrix

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

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

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

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

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

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

## 3.1 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 5.** Consider a finite subset $\mathcal{B} \subset \mathbb{C}[\mathbf{x}]$ of the set of polynomials over the complex numbers. If for each $b_i \in \mathcal{B}$ and some $p \in \mathbb{C}[x]$ we can write

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

for some (not necessarily unique) coefficients $m_{ij}$ and where equality means equality on $V$, 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 6.** *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

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

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 (1) can be obtained from an eigenvalue decomposition of $\mathbf{m}_p$. We next explore some conditions under which this is possible. A starting point is the following definition

**Definition 7.** A solving basis $\mathcal{B}$ is called a *complete solving basis* if the inverse image of the mapping $x \to \mathbf{b}(x)$ 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 8.** *Let $\mathcal{B}$ be a complete solving basis for (1) and $\mathbf{m}_p$ as above and assume that for all eigenvalues $\lambda_i$ we have $\lambda_i \neq \lambda_j$ for $i \neq j$. Then the complete set of solutions to (1) can be obtained from the set of eigenvectors $\{v_i\}$ of $\mathbf{m}_p$.*

*Proof.* The vector $\mathbf{b}(\mathbf{x})$ evaluated on $V$ is an eigenvector of $\mathbf{m}_p$. The number of eigenvectors and eigenvalues of $\mathbf{m}_p$ is finite so we can compute all eigenvectors $\{v_i\}$ and get $\{\mathbf{b}(\bar{\mathbf{x}})\}$ for all $\bar{\mathbf{x}} \in V$ as a subset of these. Applying $\mathbf{b}^{-1}$ to $v_i$ for all $i$ thus yields a finite set of points containing $V$. Evaluation in (1) allows us to filter out the points of this set which are 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 9.** 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 $\mathbf{b}(x,y) = \begin{bmatrix} x & 1 \end{bmatrix}^T$, 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 8 might not seem very useful since solving for $x$ from $\mathbf{b}(x) = v_i$ potentially involves solving a new system of polynomial equations. However, it provides a tool for ruling out choices of $\mathcal{B}$ which are not practical to work with. Moreover, there is usually much freedom in the choice of $\mathcal{B}$. In general, $\mathcal{B}$ is a set of polynomials. However, it is often practical work with a basis of monomials. For each $b_i$ we then have $b_i(x) = x_1^{\alpha_{i1}} \cdots x_s^{\alpha_{is}}$ and get the following result

**Corollary 10.** *If $\mathcal{B}$ consists of monomials and the $r \times s$ matrix $A$ with $A_{ij} = \alpha_{ij}$ is of rank $s$, then all solutions to (1) are obtained from the eigenvectors of $\mathbf{m}_{x_k}$.*

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

$$\log(b_i(\mathbf{x})) = \Sigma_j \alpha_{ij} \log(\tilde{x}_j), \tag{16}$$

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

$$\log(\mathbf{b}(\mathbf{x})) = A \begin{bmatrix} \log(\tilde{x_1}) \\ \vdots \\ \log(\tilde{x_s}) \end{bmatrix}. \tag{17}$$

If $\operatorname{rank}(A) = s$ then in particular $A$ is full rank and we can solve linearly for $\log(\tilde{\mathbf{x}})$. Theorem 8 then yields the conclusion. □

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

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

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

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

**Example 12.** Consider the polynomial system (11) from Example 4. Subtracting $f_1$ and $f_2$ and dividing by 2 we get a third polynomial $f_3 = x - y$. Thus $\mathcal{B} = \{y, 1\}$ constitutes a solving basis w.r.t. $x$ since $T_x(1) = x = y$ (on $V$) and $T_x(y) = xy = x - y + 1 = 1$ (on $V$). The vector of monomials $\mathbf{b}(x, y) = \begin{bmatrix} y & 1 \end{bmatrix}^T$ is not invertible since it does not give any information about the $x$ coordinate. However, we can use $f_3 = x - y = 0$ to get the solutions from from the eigenvectors. □

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

**Theorem 13.** *If the ideal $I$ generated by (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 6, all $\mathbf{b}(x_i)$ are eigenvectors of $\mathbf{m}_p$ corresponding to eigenvalues $p(x_i)$. By the assumption of distinct eigenvalues we have $p(x_i) \neq p(x_j)$ whenever $\mathbf{b}(x_i) \neq \mathbf{b}(x_j)$. Since $\mathcal{B}$ is a complete solving basis we have $\mathbf{b}(x_i) \neq \mathbf{b}(x_j)$ whenever $x_i \neq x_j$. This means that the $r$ points in $V$ correspond to distinct eigenvalues and hence, since eigenvectors corresponding to different eigenvalues are linearly independent, $B$ has column rank $r$. For any matrix row rank equals column rank and we are done. □

The above theorem provides an obvious 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 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 provides a method for solving (1). 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 8 still provides the right solutions. This fact is exploited in Section 4.

(ii) Typically, the artithmetic 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 9, we could multiply by *e.g.* $\{x, y, 1\}$, yielding $xy^2 - 2x, x^3 - x, y^3 - 2y, x^2y - y, y^2 - 2, x^2 - 1$.

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

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_\mathcal{E} & \mathbf{C}_R & \mathbf{C}_B \end{bmatrix}, \tag{18}$$

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

$$\begin{bmatrix} \mathbf{U}_{\mathcal{E}1} & \mathbf{C}_{\mathcal{R}1} & \mathbf{C}_{\mathcal{B}1} \\ \mathbf{0} & \mathbf{C}_{\mathcal{R}2} & \mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_\mathcal{E} \\ \mathbf{X}_\mathcal{R} \\ \mathbf{X}_\mathcal{B} \end{bmatrix} = 0. \tag{19}$$

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

$$\begin{bmatrix} \mathbf{I} & \mathbf{C}_{\mathcal{R}2}^{-1}\mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_\mathcal{R} \\ \mathbf{X}_\mathcal{B} \end{bmatrix} = 0, \tag{20}$$

or equivalently

$$\mathbf{X}_\mathcal{R} = -\mathbf{C}_{\mathcal{R}2}^{-1}\mathbf{C}_{\mathcal{B}2}\mathbf{X}_\mathcal{B}, \tag{21}$$

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 summarised in Algorithm 1 and given the results of Section 3.1 we now have the following

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

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

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

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{bmatrix} = 0, \tag{22}$$

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 (33) we obtain

$$\begin{bmatrix} xy \\ x \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{2} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y \\ 1 \end{bmatrix}, \tag{23}$$

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

**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.

```
F_ext ← F
for all f_i ∈ F do
    for all x^{α_j} ∈ L_i do
        F_ext ← F_ext ⋃ {x^{α_j} · f_i}
    end for
end for
Construct coefficient matrix C from F_ext.
M ← The set of all monomials occurring in F_ext.
R ← x_k · B \ B
E ← M \ (R ⋃ B)
Reorder and partition C: C̃ = [C_E C_R C_B].
LU-factorize to obtain C_{R2} and C_{B2} as in (19).
Use (21) to express x_k · x^{α_i} in terms of B and store the coefficients in m_{x_k}.
Compute eigenvectors of m_{x_k} and read off the tentative set of solutions.
Evaluate in F to filter out possible false zeros.
```

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 uniquelly determined. This problem can be resolved by computing

$\mathbf{m}_{x_k}$ for several $k$ and then forming a random linear combination $\mathbf{m}_{a_1x_1+\cdots+a_sx_s} = a_1\mathbf{m}_{x_1} + \cdots + a_s\mathbf{m}_{x_s}$, 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 (21). 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 [34] to get a working algorithm.

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

# 4 Using Redundant Solving Bases – The Truncation Method

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 straight forward 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. This technique is supported by Theorem 8, which guarantees that we will find the original set of solutions among the eigenvalues/eigenvectors of the larger $\mathbf{m}_p$ found using this redundant basis. The price we have to pay is performing an eigenvalue decomposition on a larger matrix.

Not all monomials from $\mathcal{M}$ can be included in the basis $\mathcal{B}$ and in general it is a difficult question exactly which monomials can be used. 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 [34]. 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 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 [34], yielding a numerically difficult situation. In fact, as will be shown in more detail in the experiments section, this problem can solved by simply including more elements in $\mathcal{B}$. In this example, 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.

The redundant 154-element solving basis and the $154 \times 154$-matrix $\mathbf{m}_{x_k}$, was con-

structed w.r.t. one of the variables $x_k$ and the set of eigenvalues computed from $\mathbf{m}_{x_k}$ for one instance are plotted in the complex plane in Figure 1 together with the actual solutions of the polynomial system.

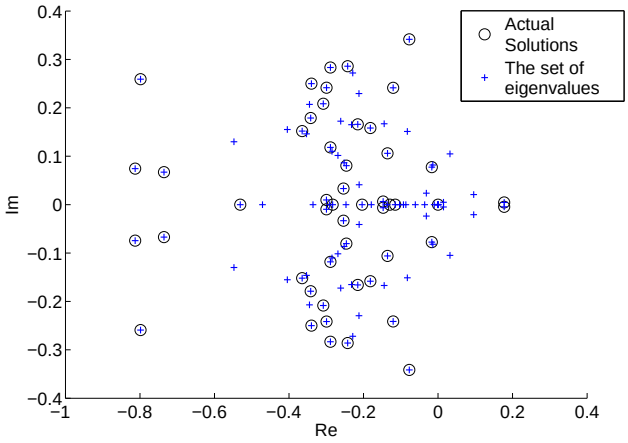



Figure 1: Eigenvalues of the action matrix using the redundant basis method and actual solutions to the polynomials system plotted in the complex number plane. The former are a strict superset of the latter.

# 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 $\mathbf{C}_{\mathcal{R}}$ 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 (19), 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

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1} \\ \mathbf{0} & \mathbf{C}_{\mathcal{P}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \tag{24}$$

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 the elimination might be very ill conditioned. This is where the basis selection comes to play.

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

$$\mathbf{C}_{\mathcal{P}2}\Pi = \begin{bmatrix} c_{\pi(1)} & \dots & c_{\pi(p)} \end{bmatrix} \tag{25}$$

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 [12]. With this algorithm, $\mathbf{C}_{\mathcal{P}2}$ is factorized as

$$\mathbf{C}_{\mathcal{P}2}\Pi = \mathbf{Q}\mathbf{U}, \tag{26}$$

where $\mathbf{Q}$ is orthogonal and $\mathbf{U}$ is upper triangular. By solving for $\mathbf{C}_{\mathcal{P}2}$ in (26) and substituting into (24) 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

$$\begin{bmatrix}\mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1}\Pi\\ \mathbf{0} & \mathbf{U}\end{bmatrix}\begin{bmatrix}\mathbf{X}_{\mathcal{R}}\\ \Pi^T\mathbf{X}_{\mathcal{P}}\end{bmatrix} = 0. \tag{27}$$

We observe that $\mathbf{U}$ is in general not quadratic and write $\mathbf{U} = \begin{bmatrix}\mathbf{U}_{\mathcal{P}'2} & \mathbf{C}_{\mathcal{B}2}\end{bmatrix}$, where $\mathbf{U}_{\mathcal{P}'2}$ is quadratic 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

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

Finally

$$\begin{bmatrix}\mathbf{X}_{\mathcal{R}}\\ \mathbf{X}_{\mathcal{P}'}\end{bmatrix} = -\begin{bmatrix}\mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}'1}\\ \mathbf{0} & \mathbf{U}_{\mathcal{P}'2}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{C}_{\mathcal{B}1}\\ \mathbf{C}_{\mathcal{B}2}\end{bmatrix}\mathbf{X}_{\mathcal{B}} \tag{29}$$

is analogous to (21) and ammounts to solving $r$ upper triangular equation systems which can be efficiently done by back substitution.

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

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

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

## 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 (24), 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

$$\mathbf{C}_{\mathcal{P}2} = \mathbf{U}\Sigma\mathbf{V}^T, \tag{30}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and $\Sigma$ 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 (24), we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1}\mathbf{V} \\ \mathbf{0} & \Sigma \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{V}^T\mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \tag{31}$$

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

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{0} & \tilde{\mathbf{C}}_{\mathcal{P}1} \\ \mathbf{0} & \Sigma' & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0, \tag{32}$$

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

$$\begin{cases} \mathbf{X}_{\mathcal{P}'} &= \mathbf{0} \\ \mathbf{X}_{\mathcal{R}} &= -\mathbf{U}_{\mathcal{R}}^{-1}\tilde{\mathbf{C}}_{\mathcal{P}1}\mathbf{X}_{\mathcal{B}} \end{cases} \tag{33}$$

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 $\tilde{\mathbf{X}}_{\mathcal{P}}$ 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 $\mathbf{X}$ and $\mathbf{X}_{\mathcal{B}}$ respectively. Further we define the linear mapping $R : \mathrm{span}(\mathcal{M}) \to \mathrm{span}(\tilde{\mathcal{B}})$, which using (33) 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

$$\mathbf{R} = \begin{bmatrix} -\tilde{\mathbf{C}}_{\mathcal{P}'}^T\mathbf{U}_{\mathcal{R}}^{-1T} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{34}$$

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

$$(\mathbf{M}_p)_{ij} = I(x^{\alpha_i} = p \cdot x^{\alpha_j}), \tag{35}$$

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 $\mathbf{V}$ we get

$$\tilde{\mathbf{M}}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}^T \end{bmatrix} \mathbf{M}_p \mathbf{V}. \tag{36}$$

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

$$\tilde{\mathbf{m}}_p = \mathbf{R}\tilde{\mathbf{M}}_p\mathbf{L}, \tag{37}$$

where $\mathbf{L} = [\begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix}]$ 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 $\mathbf{V}_q^{-1} = \mathbf{R}\mathbf{V}^T\mathbf{L}$. And 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*. [21].

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 eliminiation. 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 $\tau$ for the ratio $|\frac{u_{11}}{u_{ii}}|$ and abort the factorization once the value exceeds this threshold. A value of $\tau \approx 10^8$ has been found to yield good results[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 only provide full QR factorization, eventhough some computational effort could be spared by modifying the algorithm so as not to carry out the last steps.

Compared to setting a fixed (redundant) basis size, this approach is beneficial since both rank and conditioning of $\mathbf{C}_{\mathcal{P}2}$ might depend on the data. By the above method we decide adaptively where to truncate and *i.e.* how large the linear basis 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 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 elementwise division from

$$\mathbf{m}_{x_j}\mathbf{V} = \mathbf{V}\mathbf{D}_{x_j}. \tag{38}$$

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.

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

# 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 (standard, truncated, SVD and QR) are tested on optimal three view triangulation first studied by Stewénius *et al*. in [34]. 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: localization with hybrid features [18], relative pose with unknown but common focal length [32] and relative pose for generalized cameras [33]. 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 comparision 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. A solution to this problem was presented by Stewénius *et al*. in [34]. 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 [34] 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 [34] 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 [34] 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 [13]. 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. easiest way to get rid of the 11 false solutions at the origin is to remove the corresponding columns and rows from the action matrix.

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.

### 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 $154 \times 154$ multiplication matrix to perform eigenvalue decomposition on. In both cases the eigenvectors are used to find the solutions. The results of this experiment are shown in Figure 2. As can be seen, this relatively straight forward technique already yields a large enough improvement in numerical stability to give the solver practical value.

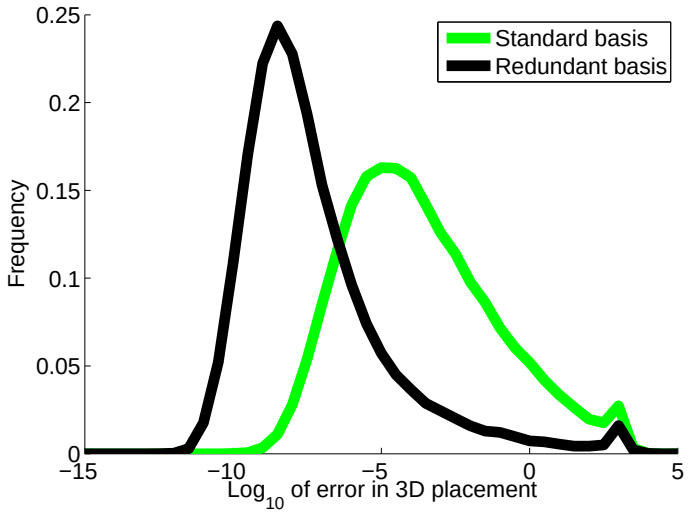



Figure 2: Histogram of errors over 100,000 points. The improvement in stability using the redundant basis renders the algorithm feasible in standard IEEE double precision.

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. 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.

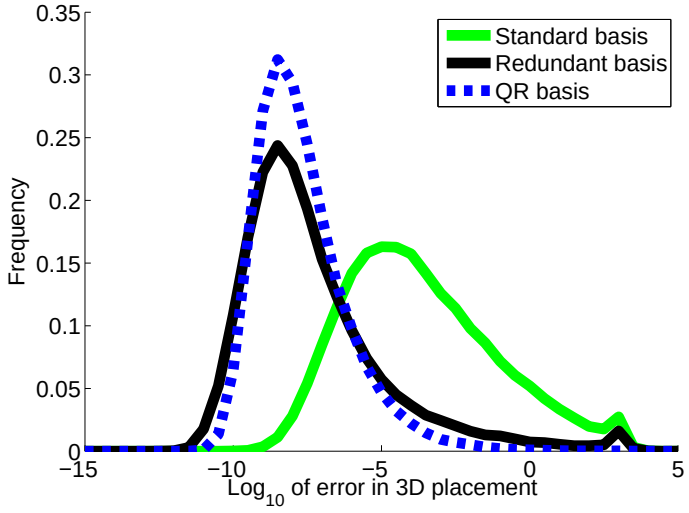



Figure 3: Histogram of errors for the standard, redundant basis and QR methods. The QR method improves stability in general and in particular completely removes the small set of large errors present in both the standard and redundant basis methods.

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 $\sim 5$ smaller for the SVD method compared to the QR method.

The reason that a good choice of basis improves the numerical stability is that the condition number in the elimination step can be lowered considerably. Using the basis selection methods, the condition number is decreased by about a factor $10^5$. Figure 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 we have a focal length of about 1000 this means that we have error $\approx 10^{-16}\kappa = \epsilon_{mach}\kappa$. This observation justifies our strategy of minimizing

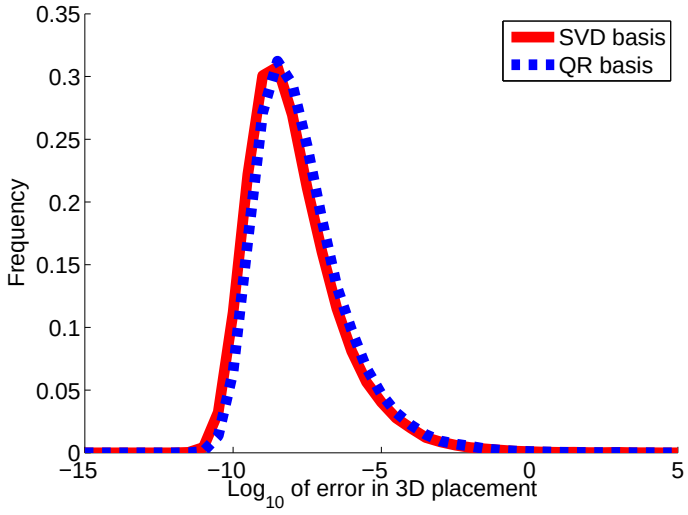



Figure 4: Comparison between the SVD and QR methods. The SVD method improves somewhat over the QR method at the cost of the computationally more demanding SVD factorization.

the condition number.

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 is two extra eigenvalues problems of size $50 \times 50$ 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 considerably.

Finally we study the combination of basis selection and truncation of the 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 size of the basis when this method was used. Since the basis is chosen minimal in 94% of the cases for the SVD-method and 95% for the QR method the time consumption is almost identical to the original basis selection methods, but as can be seen in Table 2 the number of large errors are reduced. This is probably due to the fact that truncation is carried out only

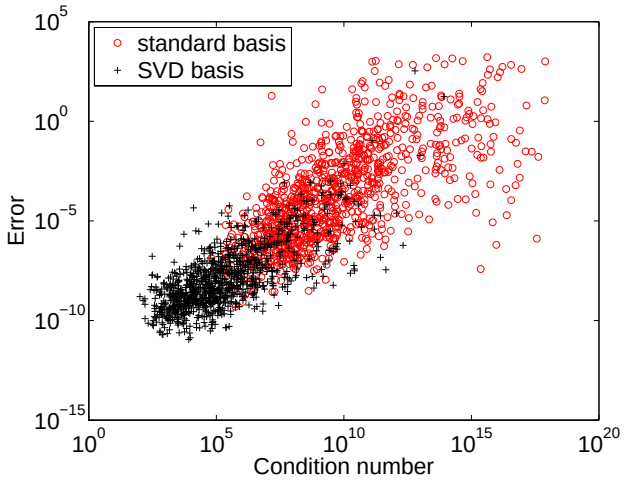



Figure 5: Error versus the condition number for the part of the matrix which is inverted in the solution procedure.

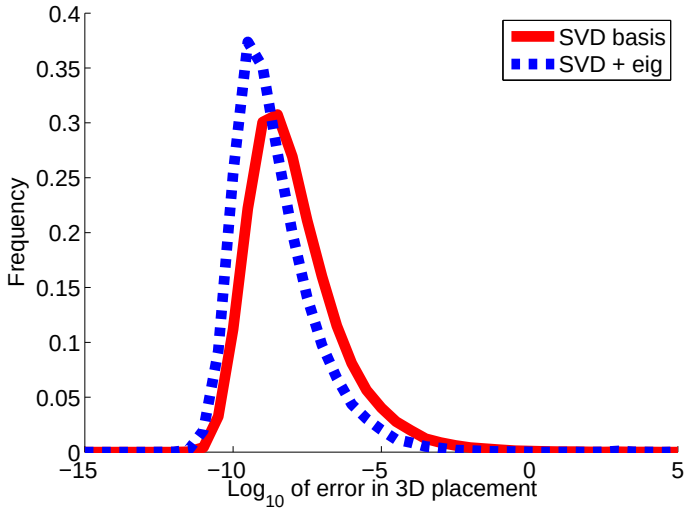



Figure 6: Error histograms showing the difference in precision between the eigenvalue and eigenvector methods.

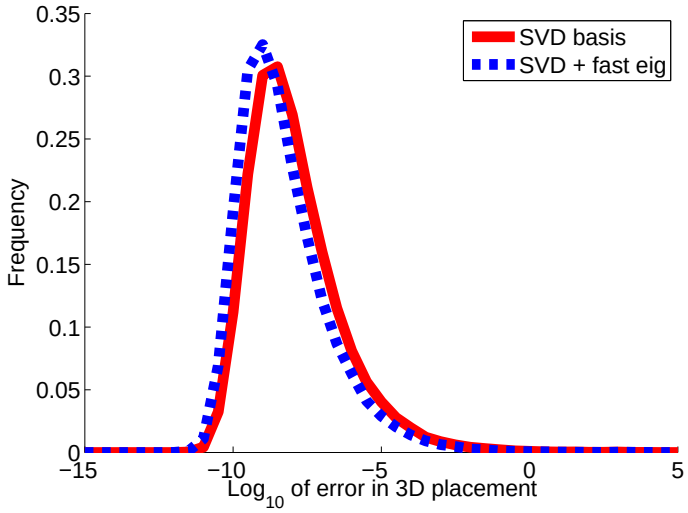



Figure 7: This graph shows the increase in performance when the fast eigenvalue method is used insted of the eigenvector method.

when the matrices are close to being singular.

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

Table 1: Basis sizes for the QR and SVD methods with variable basis size. The table shows the percentage of times the basis size occurred during 100,000 experiments.

To conclude the numerical experiments on three view triangulation two tables with detailed error statistics are given. The acronyms *STD, QR, SVD* and *TRUNC* respectively denote the standard method, QR method, SVD method and redundant basis method. The suffixes *eig, fast* and *var* respectively denote the eigenvalue method, the fast eigenvalue method (Section 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. This is interesting for example when RANSAC is used. As can be seen, the QR-method with adaptive basis size is the best method for reducing the largest errors but the SVD-method with use of the eigenvalues is the best in general. Table 3 shows the median and the 95:th percentile errors for the same methods as in the previous table. Notable in here is that the 95:th percentile is improved with as much as factor $10^7$ and the median with a factor $10^5$. The SVD-method with eigenvalues is shown to be the best but the QR-method gives almost as good results.

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

Table 2: Number of errors out of 100,000 experiments larger than certain levels. The QR-method with adaptive basis size yields the fewest number of large errors.

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

Table 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.

### 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 ammounts 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 is approximately three times faster than the SVD-method but 50% slower than the standard method.

### 7.1.3 Triangulation of Real Data

Finally, the algorithm is evaluated under real world conditions. The Oxford dinosaur [8] is a familiar image sequence of a toy dinosaur shot on a turn table. The image sequence

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

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

consists of 36 images and 4983 point tracks. For each point visible in three or more views we select the first, middle and last view and triangulate using these. This yields a total of 2683 point triplets to triangulate. The image sequence contains some erroneous tracks which we deal with by removing any points reprojected with an error greater than two pixels in any frame. The whole sequence was processed in approximately 80 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).

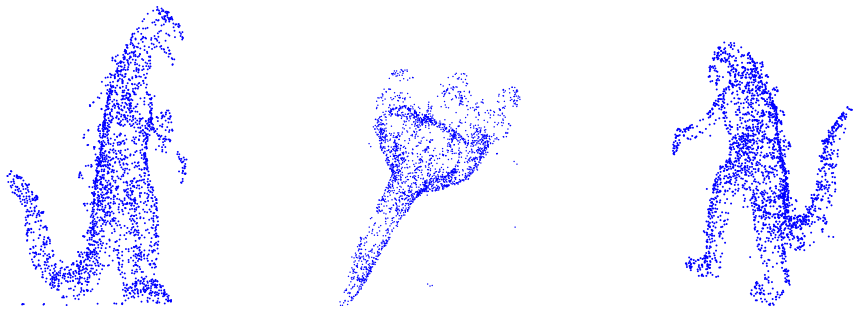


Figure 8: The Oxford dinosaur reconstructed from 2683 point triplets using the QR-method with variable basis size. The reconstruction was completed in approximately 80 seconds by a Matlab implementation on an Intel Core 2 2.13 GHz CPU with 2 GB of memory.

## 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 calibrated camera. These types of mixed features are called hybrid features and were introduced in [18], where the authors propose a parameterization of the problem but no solution was given apart from showing that the problem has 36 solutions.

The parameterization in [18] 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 (18). 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 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.

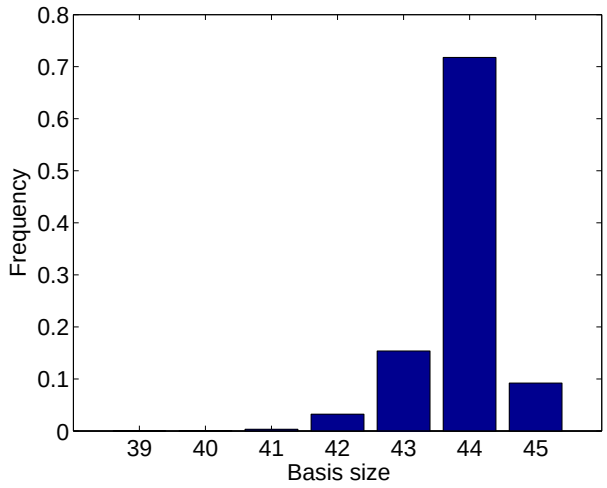



Figure 9: The basis size for localization with hybrid features. The number of solutions are 36 and since we always add three dimensions to the truncated ideal the minimal possible basis size is 39.

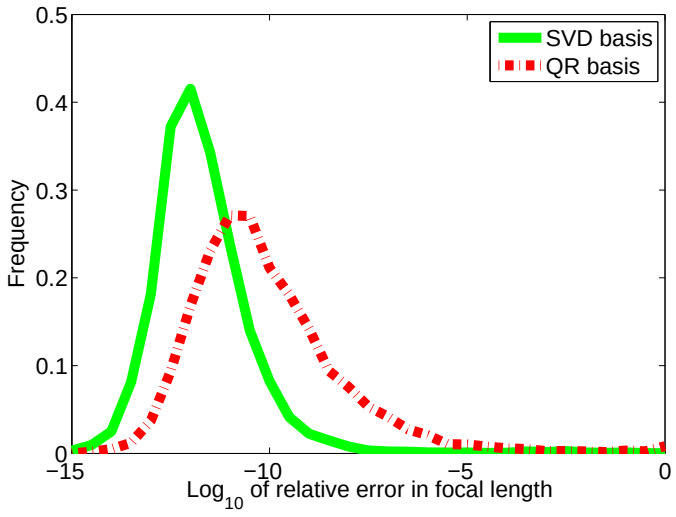



Figure 10: The error for localization with hybrid features. The standard method is omitted since we did not manage to construct a standard solver due to numerical problems.

## 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. 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 [32], which was solved by Stewénius *et al*. using Gröbner basis techniques.

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

$$F = F_0 + F_1 l_1 + F_2 l_2. \tag{39}$$

Then putting $f^{-2} = p$ one obtains nine equations from the constraint on the essential matrix [28]

$$2EE^tE - \mathrm{tr}(EE^t)E = 0. \tag{40}$$

A 10th equation is then obtained by making use of the fact that the fundamental matrix i 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. It turns out that it is possible to eliminate only one of the four monomials $l_2^3p^4$, $l_2^2p^4$, $l_2p^4$ and $p^4$ from all of the equations. However, we can discard the equations where these monomials cannot be eliminated and then proceed as usual. We choose to eliminate $l_2^2p^4$, but this choice is arbitrary.

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*. [29]. 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 [33] 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

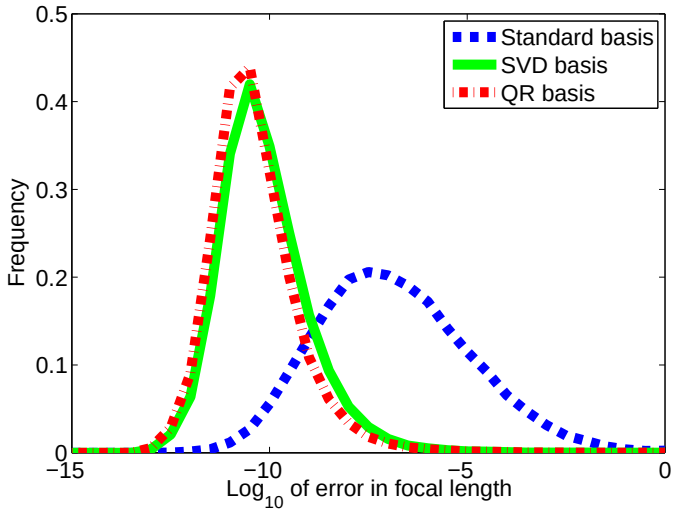



Figure 11: The error in focal length for relative pose with two semi calibrated cameras with unknown but common focal length.

in the three quaternion parameters $v_1, v_2$ and $v_3$. For details, see [33]. The problem has 64 solutions in general.

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

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

The four different methods have been compared on a data set of 10,000 randomly generated test cases. The results from this experiment are shown in Figure 12. As can be seen, a good choice of basis yields drastically improved numerical precision over the standard method.

# 8 Conclusions

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

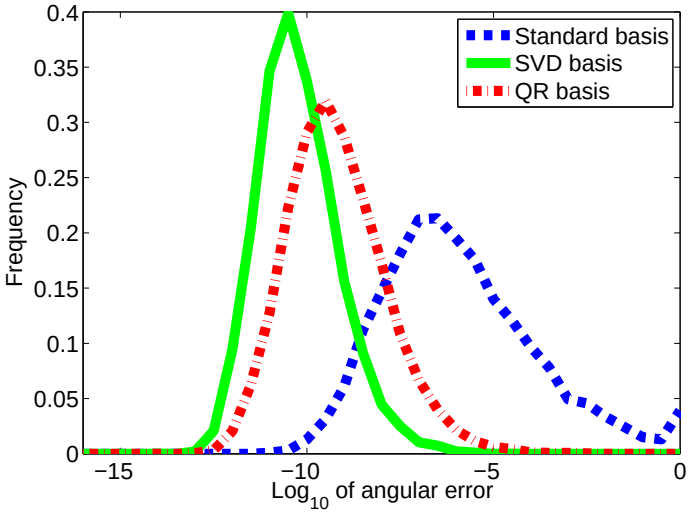



Figure 12: The angular error for relative pose with generalized camera.

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 we can use numerical linear algebra to construct the action matrix and get a fast and numirically stable solution to the problem at hand. However, we do not know (i) what monomials we should multiply the original equations with and (ii) what solving basis $\mathcal{B}$ should be used to get the simplest and most numerically stable solutions. Are there algorithmic methods for answering these questions? For a given expansion $\mathbf{CX}$ can one determine if this allows for a solving basis? A concise theoretical understanding and practical algorithms for these problems would certainly be of great aid in the work on polynomial problems and is a highly interesting subject for future research.

# Bibliography


[1] Sameer Agarwal, Manmohan Krishna Chandraker, Fredrik Kahl, David J. Kriegman, and Serge Belongie. Practical global optimization for multiview geometry. In *Proc. 9th European Conf. on Computer Vision, Graz, Austria*, pages 592–605, 2006.

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

[3] 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.

[4] Eduardo Cattani, David A. Cox, Guillaume Chèze, Alicia Dickenstein, Mohamed Elkadi, Ioannis Z. Emiris, André Galligo, Achim Kehrein, Martin Kreuzer, and Bernard Mourrain. *Solving Polynomial Equations: Foundations, Algorithms, and Applications (Algorithms and Computation in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

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

[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, 1988.

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

[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] 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.

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

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

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

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[24] 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.

[25] Lapack - linear algebra package. http://www.netlib.org/lapack.

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

[27] 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.

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

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

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

[31] 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.

[32] 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.

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

[34] 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.

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

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

[37] 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.

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

# PAPER III

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 efficient use of 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. The whole approach has been validated on both synthetic and real data, and we demonstrate improvements compared to previous work. 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, 17, 23], or by some other means.

The key idea of this paper is to use a mixture of 2D and 3D features simultaneously for localization. If one were to rely solely on 3D matches, one is restricting the set of possible

correspondences to relatively few correspondences and a relatively rich 3D model would be required in order to be successful. On the other hand, using only 2D features requires relatively many correct correspondences to generate a single hypothesis. In addition, with existing methods such as the seven point algorithm of two views [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 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. We demonstrate how hybrid feature correspondences (combinations of 2D and 3D features) can be used for improved image-based localization.
2. A complete list of minimal hybrid cases is given and for each case, we also give the number of possible solutions possible.
3. Algorithms for efficiently computing the solutions of the minimal cases are given. Further, the behavior and stability on synthetic data is evaluated for some cases.

## 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., [19]. 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 [22] 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 hypotheti-

cal part-models that need further verification. Possible such pre-processing schemes are developed in [21].

The wealth of research in the SfM field is, of course, related to the present work, in particular, the work concerned with RANSAC [27] and wide baseline matching [29, 18]. The same approach as proposed in this paper can be used to solve the wide baseline matching problem to build up 3D models [23].

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. [8]. The corresponding calibrated case was in principle solved in 1913 [14]. 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, 28].

## 2 Problem Formulation

To solve the localization problem 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 solutions 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 [16]. This involves solving minimal structure and motion problem with hybrid correspondence sets.

## 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 $3 \times 3$ rotation matrix and $t$ is a 3-elements translation vector. 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 known 3D-2D correspondences $(U_i, u_i)$ as above and 2D-2D correspondences $(v_i, u_i)$ with features $v_i$ in other views. Here we will assume that the camera matrices of the other views are known, so that a 2D-2D correspondence can be thought of as a 3D-2D correspondence where the unknown 3D

point $U_i$ lies on a line expressed in Plücker coordinates. 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 2D-3D correspondence imposes two constraints.

**Calibrated Cameras** For calibrated cameras there are six degrees of freedom, three for orientation and three for position. One way of parameterizing the camera matrix is to use a 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 algorithms 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. [14]. 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 we demonstrate that there are up to 32 solutions. 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. [26], 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

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

Potential minimal cases are

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

*The (3,2) case.* This is studied in this paper and we demonstrate that there are up to 40 solutions. 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 [25]. 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 we demonstrate that there are 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 correspondence to different view it should be solvable, but to our knowledge it is still an open problem.

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

*The (1,5) case.* This can be solved by as follows. Using the five 3D-2D correspondences, the camera matrix can be determined up to a one-parameter family $P = P_1 + \nu P_2$, where $P_1$ and $P_2$ are given $3 \times 4$ matrices and $\nu$ is an unknown scalar. The remaining 2D correspondence can be parameterized as a point on a line $U = C + \mu D$ for some unknown parameter $\mu$. The projection equation gives $\lambda u \times u = u \times PU = u \times (P_1 + \nu P_2)(U_1 + \mu U_2) = 0$. Thus we obtain two, second degree constraints in $\nu, \mu$. Using resultants, it follows easily that there are two solutions for the unknowns $\nu, \mu$.

*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 point in the known camera and then using the seven point algorithm to solve for relative orientation [8]. 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 then 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 Sec-

tion 5, 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/`.

| 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. The remaining cases have not be solved to a sufficient detail yet.

# 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. 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 [24] 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 solutions. There are several techniques for doing this. The theory of mixed volumes [9] 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` [30] 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 [9]. 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 $\mathbb{Z}_p[\mathbf{x}]$ and computing the Gröbner basis there.

Apart from providing theoretical information regarding the problem structure, Gröbner bases are also a popular and efficient tool for constructing numerical solvers for polynomial equation systems. A given polynomial system $f_1(x) = \cdots = f_m(x) = 0$, generates an ideal $I = \{g(x) : g = \Sigma_k h_k f_k, h_k \in \mathbb{C}[\mathbf{x}]\}$, consisting of all linear combinations of the $f_k$, where the coefficients $h_k$ are arbitrary polynomials. It is convenient on a theoretical level to work with $I$ since it is more general than the original system of equations, but has exactly the same zeros. A Gröbner basis $G$ for $I$ is a special set of generators which allows multivariate polynomial division by $I$ denoted $\bar{f}^G$. Consider now the quotient space $\mathbb{C}[\mathbf{x}]/I$ of equivalence classes of polynomials modulo $I$ ($f$ and $g$ are equivalent modulo $I$ *iff* $f - g \in I$) and let $[f]$ denote the equivalence class of $f$. We get a convenient representative for $[f]$ to work with by computing $\overline{f}^G$ since $\overline{f_1}^G = \overline{f_2}^G$ for $f_1, f_2 \in [f]$.

For an ideal with a finite number of solutions $\mathbb{C}[\mathbf{x}]/I$ is a finite-dimensional space with the same number of dimensions as solutions [9]. In this space, the multiplication mappings $T_{x_k} : f \mapsto x_k f$ are especially interesting. By selecting a basis for $\mathbb{C}[\mathbf{x}]/I$, we can represent $T_{x_k}$ as an $r \times r$ matrix $\mathbf{m}_{x_k}$, known as the action matrix. The eigenvalues of this matrix are then the variable $x_k$ evaluated at the zeros of $I$. Moreover, the vector of basis elements of $\mathbb{C}[\mathbf{x}]/I$, which in general is a vector of polynomials, evaluated at the zeros of $I$ equals the eigenvectors of $\mathbf{m}_{x_k}$. For proof of this see [9].

Given a set of basis elements $\mathcal{B} = \{b_1(\mathbf{x}), \ldots b_r(\mathbf{x})\}$ whose equivalence classes span $\mathbb{C}[\mathbf{x}]/I$, all we need to do now is to compute $\overline{x_k b_i}^G$ for each $b_i$. In other words, $x_k b_i$ needs to be expressed in terms of the basis elements $\{b_j\}$. This can be done as follows. First we multiply the system of polynomials by a set of monomials, yielding a larger set of equations. These equations are all members of the ideal and are hence equivalent in terms of solutions, but hopefully linearly independent. We now write the equations on matrix form

$$\mathbf{C}\mathbf{X} = 0, \tag{3}$$

where $\mathbf{C}$ is a matrix of coefficients and $\mathbf{X} = [\mathbf{x}^{\alpha_1}, \ldots, \mathbf{x}^{\alpha_n}]^T$ is a vector of monomials with the notation $\mathbf{x}^{\alpha_i} = x_1^{\alpha_{i1}} \ldots x_s^{\alpha_{is}}$. In the basic case, we now partition the set of monomials $\mathcal{M}$ into three subsets $\mathcal{M} = \mathcal{E} \cup \mathcal{R} \cup \mathcal{B}$, where $\mathcal{B}$ is a set of monomials forming a basis for $\mathbb{C}[\mathbf{x}]/I$, $\mathcal{R} = x_k\mathcal{B} \setminus \mathcal{B}$ is the set of monomials reached by multiplication by $x_k$ and which needs to be expressed in terms of $\mathcal{B}$ and finally $\mathcal{E} = \mathcal{M} \setminus (\mathcal{R} \cup \mathcal{B})$ is the set of monomials which are not involved in forming $\mathbf{m}_{x_k}$. We then get

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

Since the $\mathcal{E}$ monomials are not used for computing the action matrix we eliminate them by putting $\mathbf{C}_\mathcal{E}$ on upper triangular form

$$\begin{bmatrix} \mathbf{U}_\mathcal{E} & \mathbf{C}_{\mathcal{R}1} & \mathbf{C}_{\mathcal{B}1} \\ \mathbf{0} & \mathbf{C}_{\mathcal{R}2} & \mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_\mathcal{E} \\ \mathbf{X}_\mathcal{R} \\ \mathbf{X}_\mathcal{B} \end{bmatrix} = 0 \tag{5}$$

and the keeping only the lower part, yielding

$$\begin{bmatrix} \mathbf{C}_{\mathcal{R}2} & \mathbf{C}_{\mathcal{B}2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_\mathcal{R} \\ \mathbf{X}_\mathcal{B} \end{bmatrix} = 0. \tag{6}$$

or equivalently

$$\mathbf{X}_\mathbf{R} = \mathbf{C}_{\mathcal{R}2}^{-1}\mathbf{C}_{\mathcal{B}2}\mathbf{X}_\mathcal{B} \tag{7}$$

allowing us to express all the necessary monomials in terms of monomials in the basis.

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

## 4.1 Calibrated Cameras

A calibrated camera can be parameterized using quaternions as shown in (1). Assume 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, these can be transformed into

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

This gives us the following constraints

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

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 $3 \times 3$ 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. Since $a \in \mathbb{R}$ the probability for this is zero, but there might be problems if $a$ is close to zero. However, as the synthetic experiments will show this is no serious problem.

Assume now that we have two correspondences between image points and points that have been seen in only one other model image. This gives two points on the viewing line $C_i$ and $D_i$ associated to a point $v_i$ in the query image. If the line is represented with Plücker coordinates [13] and the camera is converted to the correspondent Plücker camera the constraints above converts to a single equation. It is further on easy to see that every nonzero element in the Plücker camera has a common factor of $1 + b^2$. After removing the common factor, the constraint polynomials $(p_1, p_2)$ are of order 2 in $b$ and order 4 in $c$.

The dimension of the quotient space $\mathbb{C}[b, c]/I$ is 16 with $I = \langle p_1, p_2 \rangle$ which can be checked with computer algebra [30]. By multiplying the polynomials with $\{1, b, c, bc\}$ we obtain 8 equations in 24 monomials. It is then possible to express 8 of the monomials in terms of the remaining 16 monomials

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

which then form a basis for the quotient space $\mathbb{C}[b, c]/I$. From this it is straightforward to construct the $16 \times 16$ action matrix $\mathbf{m}_c$ for the linear mapping $\mathbb{C}[b, c]/I \ni p(c) \mapsto cp(c) \in \mathbb{C}[b, c]/I$. From the eigenvalue decomposition of the matrix $\mathbf{m}_c$ the 16 (some possibly complex) solutions are obtained. Similar calculations give that there are 32 solutions for the (4,1) case.

## 4.2 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 [25] 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 are with

$$U_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, U_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, u_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \tag{8}$$

it is possible to eliminate $y = 0$ and $x = zf = g(b, c, d, f)$. We fix the scale by setting $a = 1$. For both the $(1, 3)$ case and the $(3, 2)$ case we get polynomial constraints in the four remaining unknowns $(b, c, d, f)$. Calculations with computer algebra [30] show that there are 36 solutions for the $(1, 3)$ case, 40 solutions to the $(3, 2)$ case and 112 in the $(5, 1)$ 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 $\lambda$ 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,

$$B = \begin{pmatrix} 0 & -1 & x_2 \\ -1 & 0 & x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}, \tag{9}$$

from which three equations can be generated by putting,

$$BPU_3 = 0. \tag{10}$$

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

The four constructed equations will now be 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 will be 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 will have size $4 \times 82$. 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 $980 \times 873$.

After $\mathbf{C}$ is expanded, we divide the monomials into the three sets $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{B}$ as outlined above. 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 [6, 7, 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.

# 5 Synthetical Experiments

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

The algorithm has been run on 10,000 randomly generated cases as described above. To evaluate the accuracy of the 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

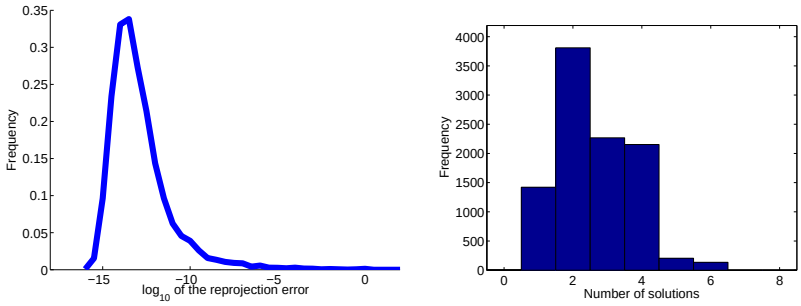



Figure 1: Statistics from the evaluation of the solver for the (2,2) case for calibrated cameras. The solver was run on 10,000 randomly generated cases. Left: Histogram over the error in matrix norm between the estimated camera $P'$ and the true camera $P$. The error is plotted on a logarithmic scale. 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 unfortunately positioned (two or more real solutions irrespective of the sign of the depths were found in all cases). In three of the cases seven solutions were found and in one case eight plausible solutions were found. The average number of plausible solutions was 2.6 and the average number of real solutions was 6.4. In some of the cases all 16 solutions were real.

## 5.2 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 run 10,000 times.

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

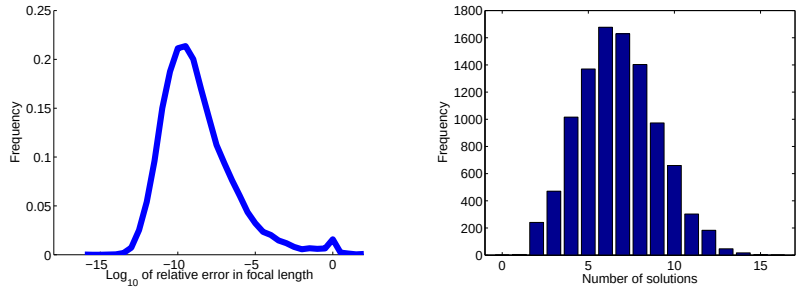



Figure 2: Statistics from the evaluation of the solver for the (1,3) case for calibrated cameras with unknown focal length. The solver was run on 10,000 randomly generated cases. Left: Histogram over the error focal length between the estimated camera and the true camera. The error is plotted on a logarithmic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

# 6 Localization System Experiments

To test the new methods on real data a complete localization system was constructed. Since this paper treats the pose problem, and not the construction of the model, there will only be a short description of how the model was built. For more on model construction [23] is recommended. Localization with some of the new minimal cases was then tested and compared with two previous methods. In the calibrated case the previous method is the (0,3)-solver [12] and in the uncalibrated case we have compared with a linear resection algorithm using six correspondences.

## 6.1 Construction of the Model

The model was built from a set of unordered images with nothing known about the positions of the cameras. The calibrations of the cameras were known for these images. On all images SIFT descriptors [16] were calculated. The descriptors were then used together with the knowledge of the calibration to find the relative motion between every pair of cameras, with enough corresponding points, using the five point algorithm [20]. By using the pairwise connection between all images the best corresponding triplet was found. By best triplet we here mean the set of three images with most corresponding points common for all three views. Two of the images in the triplet were then used as the foundation of the model to fixate the unknown scale.

After that a third camera was added. This was done by using the known relative motion to one of the first two cameras. Since this relative motion was known it is enough with one single point to get the scale. To find the scale a RANSAC engine was used. The remaining cameras were then added stepwise choosing the camera with most known

correspondences to points that were known in the 3D-space. The placement was then performed in the same way as with the third camera. After each camera was added bundle adjustment was applied to points and cameras. In the beginning all points and all cameras were used in the bundle adjustment step, but to speed up the process only the new camera and new 3D points were optimized when the size of the model increased. The model construction is not focused on removing outliers even though RANSAC is used, which will result in many outliers remaining in the model. The outliers will be handled in the localization phase.

The complete model holds the following data so it can be used in a hybrid localization system:

**SIFT:** All SIFT features found in all images.

**Points:** The normalized coordinates of all SIFT features in all images.

**Structure:** The coordinates of all triangulated points in the 3D-space.

**Cameras:** All positions of the cameras in the model.

**Tracks:** Groups of feature points that correspond to a single 3D point.

In addition to this the model also holds mappings between these objects.

For the experiments two models were constructed of the same area but with slightly different properties. The first model, called Model 1 in the following, is 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 2.

| | 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 2: 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.

Figure 3: On the left is one of the images used to construct the models and on 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.

## 6.2 Testing of Localization

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 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 balance the fact that the number of 2D points in the model is about ten times as large as the number of 3D points. The thresholds for the reprojection error in the RANSAC algorithm is fixed to 0.6% of the image size as in [23]. For the 2D points this value is reduced by a factor ten since the 3D points are first triangulated using the 2D points. As a final step of the localization, bundle adjustment is performed on the camera position using the 3D inliers.

**Test of the (2,2) Solver**

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.

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.

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 3. 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.

| | 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 3: 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.

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 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 (1,3) Solver**

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.

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.

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 4. The results are very similar to those of the previous case.

| | 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 4: 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.

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 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.

**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 with enough of correspondences was localized. All images that got at least one extra inlier in 3D were then used in kernel voting [15] 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.

# 7 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 use of Gröbner basis techniques. In two of the cases we have also presented and evaluated 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

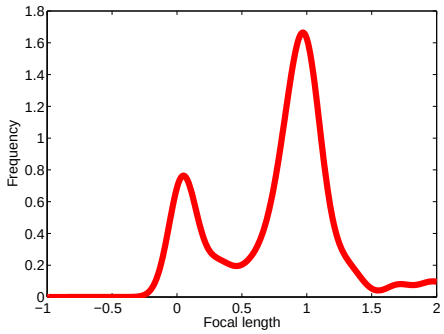



Figure 6: The result of the kernel voting for the focal length. The true focal length in this experiment is one. In the kernel voting only estimations with more the minimal number, which is three, of inliers are used in the voting process.

stable algorithm. We also present a solver for the (1,3) case for cameras with unknown focal length. This problem is much more complicated than the (2,2) problem but we can still present a numerically sound algorithm that is fast with Gröbner basis methods. Both these methods are tested in a complete localization system and they are shown to improve on the current state of the art. 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 *every* localization system using hybrid cases. In future work we intend to apply the idea of hybrid features to evaluate the performance on publicly available benchmarks for example Zurich Building Image Database to compare with the state of the art.

# Bibliography


[1] D. Bayer and M. Stillman. Macaulay. www.math.columbia.edu/˜bayer/Macaulay/, 1994. An open source computer algebra software.

[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.B. 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. Fast and stable polynomial equation solving and its application to computer vision. Licentiate thesis, Centre for Mathematical Sciences LTH, Lund University, Sweden, 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. Chasles. Question 296. *Nouv. Ann. Math.*, 14(50), 1855.

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

[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] 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.

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

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal Computer Vision*, 2004.

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

[18] 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.

[19] P.M. Newman, J.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.

[20] 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.

[21] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Conf. Computer Vision and Pattern Recognition*, volume II, pages 2161–2168, New York City, USA, 2006.

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

[23] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 2007.

[24] 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.

[25] H. Stewénius, D. Nistér, F. Kahl, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *Conf. Computer Vision and Pattern Recognition*, volume 2, pages 789–794, San Diego, USA, 2005.

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

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

[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.

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

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