LUND UNIVERSITY

Improving numerical accuracy of Grobner basis polynomial equation solvers

Byröd, Martin; Josephson, Klas; Åström, Karl

Link to publication

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Improving Numerical Accuracy of Gröbner Basis Polynomial Equation Solvers

Martin Byröd
byrod@maths.lth.se

Klas Josephson
klasj@maths.lth.se

Kalle Åström
kalle@maths.lth.se

Centre for Mathematical Sciences,
Lund University, Lund, Sweden

Abstract

This paper presents techniques for improving the numerical stability of Gröbner basis solvers for polynomial equations. Recently Gröbner basis methods have been used succesfully to solve polynomial equations arising in global optimization e.g. *three view triangulation and in many important minimal cases of structure from motion. Such methods work extremely well for problems of reasonably low degree, involving a few variables. Currently, the limiting factor in using these methods for larger and more demanding problems is numerical difficulties. In the paper we (i) show how to change basis in the quotient space* $\mathbb{R}[\mathbf{x}]/I$ *and propose a strategy for selecting a basis which improves the conditioning of a crucial elimination step, (ii) use this technique to devise a Gröbner basis with improved precision and (iii) show how solving for the eigenvalues instead of eigenvectors can be used to improve precision further while retaining the same speed.*

We study these methods on some of the latest reported uses of Gröbner basis methods and demonstrate dramatically improved numerical precision using these new techniques making it possible to solve a larger class of problems than previously.

1. Introduction

Numerous geometrical problems in vision involve the solution of systems of polynomial equations. This is particularily true for so called minimal structure and motion problems, *e.g.* [2, 12, 18]. Solutions to minimal structure and motion problems can often be used in RANSAC algorithms to find inliers in noisy data [5, 19, 20]. 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 funda-

mental matrix estimation. Global optimization is a promising, but difficult pursuit and different lines of attack have been tried, *e.g.* branch and bound [1], L_∞ norm methods [7, 10] and methods using linear matrix inequalities (LMIs) [11].

An alternative way to find the global optimum is to calculate stationary points directly (usually by solving some polynomial equation system) [8, 16]. 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 [16]. We show that this problem can now be solved in a reasonably efficient way using 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, *c.f.* [16]. The outline of such algorithms is that one studies a class of geometric problems and finds out what structure the Gröbner basis for the ideal I has for that problem and what the degree is. This degree is the same as the number of solutions to the problem. For each instance of the problem with numerical data, the process of forming the Gröbner basis and the solution to the problem is performed using numerical linear algebra.

Currently, the limiting factor in using these methods for larger and more difficult cases is numerical problems in the solver. For example in [16] it was necessary to use emulated 128 bit numerics to make the system work, which makes 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 show how a change of basis in the quotient space $\mathbb{R}[\mathbf{x}]/I$ can be used to improve the numerical precision of a Gröbner basis computation. We develop the tools needed to compute the action matrix in a general basis and propose a strategy to select a basis which enhances

the conditioning of a crucial elimination step in the solution procedure. Finally, we use this technique to devise a Gröbner basis with lower errors in the final result. In addition to these techniques, we show how solving for the eigenvalues instead of eigenvectors can be used to improve the precision further while retaining the same speed.

2. Using Gröbner Bases to Solve Systems of Polynomial Equations

In this section we briefly discuss how Gröbner basis techniques can be used for solving systems of multivariate polynomial equations. We introduce some concepts and notation used to develop the results in the subsequent sections.

The goal is to find the solutions to a system of polynomial equations on the following form

$$
\begin{aligned}
c_{11}\varphi_1 + c_{12}\varphi_2 + \cdots + c_{1n}\varphi_n &= 0, \\
\vdots \\
c_{m1}\varphi_1 + c_{m2}\varphi_2 + \cdots + c_{mn}\varphi_n &= 0,
\end{aligned}
\tag{1}
$$

where $\varphi_1, \ldots, \varphi_n$ are a given set of monomials. This can be written using matrix notation as

$$
\mathbf{C} \begin{pmatrix} \varphi_1 \\ \vdots \\ \varphi_n \end{pmatrix} = 0.
\tag{2}
$$

The m polynomials in the left hand side of the equations (1) in p variables generate an ideal I in the polynomial ring $\mathbb{R}[\mathbf{x}]$, where $\mathbb{R}[\mathbf{x}]$ denotes the set of multivariate polynomials in $\mathbf{x} = (x_1, \ldots, x_p)$ with real coefficients. To find the roots of this system we study the quotient ring $\mathbb{R}[\mathbf{x}]/I$. If the system of equations has r roots, then $\mathbb{R}[\mathbf{x}]/I$ is a linear vector space of dimension r. In this ring, multiplication with x_k is a linear mapping. The matrix $\mathbf{m}_{x_k}$ representing this mapping (in some basis) is referred to as the action matrix and is a generalization of the companion matrix for one-variable polynomials. From algebraic geometry it is known that the zeros of the equation system can be obtained from the eigenvectors of the action matrix just as the eigenvalues of the companion matrix yields the zeros of a one-variable polynomial [3].

There are basically three steps involved in computing the action matrix. First, we need to choose a set of polynomials $\{e_i(x_1, \ldots, x_p)\}_{i=1}^r$, forming a basis in the quotient ring $\mathbb{R}[\mathbf{x}]/I$. The most straightforward choice here is a set of *monomials*. In previous work on computing zeros of polynomial equations this is typically what has been used. However, we show in this paper that a general basis of *polynomials* often yields much better numerical accuracy. Secondly, we need to calculate the effect of multiplication with x_k on each of these basis elements. In general, the element $x_k e_i$ will not immediately be a linear combination of the basis

elements. Therefore, thirdly, we need to be able to reduce elements of $\mathbb{R}[\mathbf{x}]$ modulo I. By reduction modulo I we mean to replace an element in $\mathbb{R}[\mathbf{x}]$ by a representative of the corresponding equivalence class in $\mathbb{R}[\mathbf{x}]/I$ uniquely defined by the monomial ordering used. These steps can be summarized as:

1. Choose a basis of polynomials for $\mathbb{R}[\mathbf{x}]/I$.

2. Interpret these basis elements as elements of $\mathbb{R}[\mathbf{x}]$ and multiply by one of the variables x_k.

3. Reduce the result modulo I.

We will start with steps 2 and 3 and then come back to the question of how to choose a basis.

To facilitate the discussion of how to change basis in the quotient space we make use of some additional notation not normally used when working with action matrices. For instance we write $\mathbf{M}_{x_k}$ for the action matrix on the subspace of $\mathbb{R}[\mathbf{x}]$ spanned by the occurring monomials whereas $\mathbf{m}_{x_k}$ is the action matrix in the quotient space $\mathbb{R}[\mathbf{x}]/I$. The set of monomials which occur in the equations is denoted $\mathcal{M}$ and the set of basis monomials is $\mathcal{M}_b$. The linear hulls generated by $\mathcal{M}$ and $\mathcal{M}_b$ are denoted $\mathrm{lh}(\mathcal{M})$ and $\mathrm{lh}(\mathcal{M}_b)$. By construction, these linear hulls are subspaces of $\mathbb{R}[\mathbf{x}]$.

For now, we choose a basis consisting of monomials only which will make life easier for us. In the next section we will see how to deal with a general basis of polynomials.

With a monomial basis, multiplication with x_k becomes very easy. Any monomial will simply be mapped to one of the n other monomials. The $n \times r$ action matrix

$$
\mathbf{M}_{x_k} : \mathrm{lh}(\mathcal{M}_b) \to \mathrm{lh}(\mathcal{M})
\tag{3}
$$

will consist of columns of $n - 1$ zeros and a single 1.

Our next step is to reduce $\mathbf{M}_{x_k}$ to a mapping on our quotient space. Reduction modulo I is a linear mapping from an infinite dimensional space to a finite dimensional space. Now, if we restrict the reduction mapping to the subspace $\mathrm{lh}(\mathcal{M})$ of $\mathbb{R}[\mathbf{x}]$, we get a linear mapping between two finite dimensional spaces. We represent this mapping as an $r \times n$ matrix $\mathbf{P} : \mathrm{lh}(\mathcal{M}) \to \mathbb{R}[\mathbf{x}]/I$ where r is the dimension of our quotient ring and n is the number of occurring monomials. The construction of $\mathbf{P}$ is an important part of the solution procedure.

For elements *in* the basis, the modulo mapping $\mathbf{P}$ will simply be an identity mapping. However, for elements *outside* the basis we need to work a bit. What we need is a Gröbner basis for I. Actually, we do not exactly need a Gröbner basis, but we need a method for calculating the remainder after division with I in a unique way.

Here, we make use of our equations $\mathbf{C}\varphi = 0$. What we do is we put $\mathbf{C}$ on diagonal form

$$
\mathbf{C}_{red} = \begin{bmatrix} \mathbf{I} & \mathbf{T} \end{bmatrix},
\tag{4}
$$

with respect to the monomials which are not in the basis by performing normal row operations on $\mathbf{C}$. Here, $\mathbf{I}$ is the identity matrix and $\mathbf{T}$ is an $m \times r$ matrix produced by the row operations. After elimination, one line of $\mathbf{C}_{red}$ might look like this

$$0 \quad \ldots \quad 1 \quad \ldots \quad 0 \quad a'_{i,n-r+1} \quad \ldots \quad a'_{i,n}, \quad (5)$$

where the 1 is at position i. This corresponds to the equation

$$\varphi_i \quad + \quad a'_{i,n-r+1}\varphi_{n-r+1} \quad + \quad \ldots \quad + \quad a'_{i,n}\varphi_n = 0, \quad (6)$$

which lets us express φ_i in terms of the r monomials $\{\varphi_k\}_{k=n-r+1}^n$. This is precisely what we need since we can now replace monomials *not in the basis* by a linear combination of monomials *in the basis*.

In general, we cannot always perform these operations directly on the original matrix $\mathbf{C}$ since it might not have enough rows or enough rank to allow us to reduce all the monomials we would like to reduce. We can then add new equations by multiplying the old equations by a selection of monomials. These new equations will be equivalent to the old equations in terms of solutions (they are new elements of the ideal), but they will hopefully be linearly independent making the desired elimination possible. There is a general method for doing this which is guaranteed to produce a full Gröbner basis known as *Buchberger's algorithm* [3].

Using (4) and (6) we can now form the modulo mapping

$$\mathbf{P} = \begin{bmatrix} -\mathbf{T}^t & \mathbf{I} \end{bmatrix}. \quad (7)$$

This allows us to construct the final action matrix representing multiplication with x_k in $\mathbb{R}[\mathbf{x}]/I$.

$$\mathbf{m}_{x_k} = \mathbf{P}\mathbf{M}_{x_k}. \quad (8)$$

By performing an eigenvalue decomposition of this matrix, we can read off the zeros of the equation system in the eigenvectors [3].

The problem one encounters in this method is that the process of expanding $\mathbf{C}$ from (2) by adding more equations in many cases yield very large and ill conditioned matrices. This is especially true for systems with many variables, high degrees and many solutions. In (4) where $\mathbf{C}$ is put on diagonal form, we generate $\mathbf{T}$ by solving r linear equation systems. If the part of $\mathbf{C}$ which we use to eliminate is ill conditioned, (potentially large) numerical errors will be introduced in $\mathbf{T}$. The main contribution of this paper is an approach for dealing with this problem.

The key observation is that although the basis of single monomials we used above is conceptually simple to work with, it might not be the best choice numerically. Indeed, in some cases this type of basis turns out to yield very poor performance in terms of numerical stability. We would therefore like to change basis in our quotient space. This is the purpose of the next section.

3. Changing Basis in $\mathbb{R}[\mathbf{x}]/I$

Before discussing the details of how to change basis in $\mathbb{R}[\mathbf{x}]/I$, we make an important point. We are *not* completely free in our choice of basis. Since we are working with a finite set of monomials and want to study the effect of multiplication by x_k on a certain basis involving these monomials, we have to make sure that we stay within our original set of monomials $\mathcal{M}$.

As before, $\mathcal{M}$ denotes our set of n monomials. What we do is we partition $\mathcal{M}$ into two sets $\mathcal{M}'$ and $\mathcal{M}''$ with n' and n'' elements respectively, where $\mathcal{M}' = \{\varphi \in \mathcal{M} : x_k\varphi \in \mathcal{M}\}$ is the set of monomials which stay in $\mathcal{M}$ under multiplication with x_k. $\mathcal{M}''$ is the remaining set of monomials. We call $\mathcal{M}'$ the set of permissible monomials. For simplicity, we assume that the monomials are ordered so that $\mathcal{M}''$ contains the first n'' monomials of $\mathcal{M}$ and $\mathcal{M}'$ contains the rest. In other words, we can only construct our basis using permissible monomials, since otherwise we will not be able to calculate the effect of multiplication with x_k.

Assume now that we have a basis of *polynomials* for our equations which can be obtained from an *orthogonal* change of basis on the subset of permissible *monomials*. That is, our new basis is

$$\tilde{\boldsymbol{\varphi}} = \mathbf{V}^t \begin{bmatrix} \varphi_{n''+1} \\ \vdots \\ \varphi_n \end{bmatrix}, \quad (9)$$

where $\mathbf{V}$ is an orthogonal matrix and $\{\varphi_k\}_{k=n''+1}^n$ is the set of n' permissible monomials. From the set of n' new basis polynomials we select a subset of r polynomials which will be the basis for $\mathbb{R}[\mathbf{x}]/I$.

We would now like to apply this change of basis to our calculation of the action matrix $\mathbf{m}_{x_k}$. Here we have to note that we are working with two different spaces $\mathrm{lh}(\mathcal{M})$ of all monomials and $\mathrm{lh}(\mathcal{M}')$ of the monomials that can safely be multiplied by x_k. $\mathbf{V}$ applies to $\mathrm{lh}(\mathcal{M}')$ so we define

$$\mathbf{V}_e = \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{V} \end{bmatrix}, \quad (10)$$

by extending $\mathbf{V}$ with an $n'' \times n''$ identity matrix.

In the new basis, we now get

$$\tilde{\mathbf{M}}_{x_k} = \mathbf{V}_e^t \mathbf{M}_{x_k} \mathbf{V}, \quad (11)$$

where $\mathbf{M}_{x_k} : \mathrm{lh}(\mathcal{M}') \rightarrow \mathrm{lh}(\mathcal{M})$ is the non-reduced action matrix.

Changing basis according to $\tilde{\boldsymbol{\varphi}} = \mathbf{V}^t\boldsymbol{\varphi}$ also affects the coefficient matrix $\mathbf{C}$. Using the new basis, we have $0 = \mathbf{C}\boldsymbol{\varphi} = \tilde{\mathbf{C}}\tilde{\boldsymbol{\varphi}} = \tilde{\mathbf{C}}\mathbf{V}^t\boldsymbol{\varphi}$ yielding $\mathbf{C} = \tilde{\mathbf{C}}\mathbf{V}^t$ and $\tilde{\mathbf{C}} = \mathbf{C}\mathbf{V}$.

By putting $\tilde{\mathbf{C}}$ on diagonal form as in the previous section, we get $\tilde{\mathbf{C}}_{red} = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{T}} \end{bmatrix}$. From this we obtain $\tilde{\mathbf{T}}$ and a transformed modulo mapping $\tilde{\mathbf{P}} = \begin{bmatrix} -\tilde{\mathbf{T}}^t & \mathbf{I} \end{bmatrix}$ as before. Finally, to handle the technicality that $\mathbf{M}_{x_k}$ applies to

lh($\mathcal{M}'$) whereas $\mathbf{m}_{x_k}$ is defined on $\mathbb{R}[\mathbf{x}]/I$, we define a lift mapping $\mathbf{L} : \mathbb{R}[\mathbf{x}]/I \to \text{lh}(\mathcal{M}')$, which simply interprets the elements in the quotient ring as elements in lh($\mathcal{M}'$). In matrix notation we get (an $n' \times r$ matrix) $\mathbf{L} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}$. We can now write an expression for the action matrix in our new basis

$$\tilde{\mathbf{m}}_{x_k} = \tilde{\mathbf{P}} \tilde{\mathbf{M}}_{x_k} \mathbf{L}. \tag{12}$$

An eigendecomposition of $\tilde{\mathbf{m}}_{x_k}^t$ yields a set of eigenvectors $\tilde{v}$ in our new basis. It remains to inverse transform these eigenvectors to obtain eigenvectors of $\mathbf{m}_{x_k}^t$. To do this, we need to construct the change of basis matrix $\mathbf{V}_q$ in the quotient space. Using $\tilde{\mathbf{P}}$ and $\mathbf{L}$, we get $\mathbf{V}_q^{-1} = \tilde{\mathbf{P}} \mathbf{V}^t \mathbf{L}$. And from this we get $v = \mathbf{V}_q^{-t} \tilde{v}$ in our original basis.

4. How to Choose a Basis

Now that we know how to change basis for the computation of $\mathbf{m}_{x_k}$ the question is what is the optimal choice of basis? As we have mentioned previously, a numerical bottle neck in the calculations is the elimination step where the coefficient matrix $\mathbf{C}$ is put on diagonal form. What is done in this step is that the columns of $\mathbf{C}$ is split into two parts $\mathbf{C} = \begin{bmatrix} \mathbf{C}_{nb} & \mathbf{C}_b \end{bmatrix}$, where $\mathbf{C}_b$ denotes the columns corresponding to the basis elements of $\mathbb{R}[\mathbf{x}]/I$ and $\mathbf{C}_{nb}$ denotes the non-basis columns. Then $\mathbf{C}_{nb}$ is put on diagonal form, expressing the nonbasis monomials in terms of the basis monomials. A basic result from linear algebra (see *e.g.* [9]) states that the relative error in the solution of a linear equation system $Ax = b$ with errors in the right hand side can be estimated in terms of the condition number $\kappa(A)$ of the matrix as

$$\left| \frac{\delta x}{x} \right| \le \kappa(A) \left| \frac{\delta b}{b} \right| \tag{13}$$

and a similar result holds for the coefficients of A. Thus, in this step, the numerical accuracy depends on the condition number of $\mathbf{C}_{nb}$. Typically, for a system with r solutions, $\mathbf{C}$ will be an $m \times n$ matrix of rank $n - r = m$. The condition number of $\mathbf{C}$ is usually not too large, but if one is not careful in the choice of basis for $\mathbb{R}[\mathbf{x}]/I$, $\mathbf{C}_{nb}$ often becomes very ill conditioned.

Making an orthogonal change of basis in the space of monomials means multiplying $\mathbf{C}$ with an orthogonal matrix $\mathbf{V}$ from the right. This does not affect the condition number of $\mathbf{C}$ but can enhance the conditioning of $\mathbf{C}_{nb}$ considerably. A tempting strategy is to use a singular value decomposition $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^t$ and then take $\mathbf{V}$ as the change of basis matrix so as to concentrate the rank of $\mathbf{C}$ to the columns which are to be used for elimination. However, recall from the previous section that we are not allowed to make a change of basis which mixes the permissible monomials with the non-permissible monomials, thus we will have to content our selves with matrices $\mathbf{V}_e$ on the form (10) which leave the columns of $\mathbf{C}$ corresponding to the monomials $\mathcal{M}''$ untouched.

This can be formulated as an optimization problem on the condition number of $\mathbf{C}_{nb}$.

$$\mathbf{V}_e^* = \underset{\mathbf{V} \in O(n')}{\operatorname{argmin}} \kappa(\mathbf{C} \cdot \mathbf{V}_e\{1, \dots, n - r\}), \tag{14}$$

where $O(n')$ is the set of orthogonal $n' \times n'$ matrices and $\mathbf{V}_e\{1..n - r\}$ denotes the first $n - r$ columns of $\mathbf{V}_e$.

A scheme which finds the optimum for this problem would probably be iterative and hence too slow for our purposes. We therefore propose the following heuristic strategy using a partition of the columns of $\mathbf{C}$ into $\mathbf{C}''$ corresponding to the columns representing non-permissible monomials and $\mathbf{C}'$ representing permissible monomials.

1. Write $\mathbf{C}'$ as $\mathbf{C}' = \mathbf{C}'^{//} + \mathbf{C}'^{\perp}$, where $\mathbf{C}'^{\perp}$ is the projection of the column vectors of $\mathbf{C}'$ onto the orthogonal complement of the subspace spanned by the columns of $\mathbf{C}''$.

2. Decompose $\mathbf{C}'^{\perp}$ as $\mathbf{C}'^{\perp} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^t$.

3. Discard $\mathbf{C}'^{\perp}$ but use $\mathbf{V}$ from this decomposition to form $\mathbf{V}_e$ and $\tilde{\mathbf{C}} = \mathbf{C}\mathbf{V}_e$.

The heuristic argument for this scheme is as follows. We would like to change basis so that the columns $1, \dots, n - r$ of $\tilde{\mathbf{C}}$ become "as linearly independent as possible". We are not allowed to touch the columns $1, \dots, n''$ of $\mathbf{C}$ so to produce the extra $n - r - n''$ columns needed for $\mathbf{C}_{nb}$, we work with the columns $n'' + 1, \dots, n$ of $\mathbf{C}$, *i.e.* $\mathbf{C}'$. A straightforward singular value decomposition of $\mathbf{C}'$ yields a set of orthogonal columns, but there is no guarantee that these will be linearly independent of the columns of $\mathbf{C}''$. What we would like to do is to orthogonalize, but only on the part which is orthogonal to the column space of $\mathbf{C}''$. Therefore we make the singular value decomposition only on $\mathbf{C}'^{\perp}$.

5. Adding Equations

The model for how we add equations (thereby enlarging $\mathbf{C}$) and then eliminate is Buchberger's algorithm for computing a Gröbner basis. The reason we cannot use Buchberger's algorithm directly is numerical problems. Buchberger's algorithm works perfectly under exact arithmetic. However, in floating point arithmetic it becomes extremely difficult to use due to accumulating round off errors. In Buchberger's algorithm, adding equations and eliminating is completely interleaved. We aim for a process where we first add all equations we will need and then do the full elimination in one go. This is possible by first studying a particular problem using exact arithmetic[1] to determine the

---

[1]Usually with the aid of some algebraic geometry software as Macaulay 2 [6]

number of solutions and what total degree we need to go to. Using this information, we hand craft a set of monomials which we multiply our original equations with to generate new equations and hence new rows in **C**.

6. Using Eigenvalues Instead of Eigenvectors

In the literature, the preferred method of extracting solutions using eigenvalue decomposition is usually to look at the eigenvectors. It is also possible to use the eigenvalues, but this seemingly requires us to solve p 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 $\mathbb{R}[\mathbf{x}]/I$. From Section 2 we know that the right eigenvectors of an action matrix is the vector of basis elements of $\mathbb{R}[\mathbf{x}]/I$ 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

$$\mathbf{m}_{x_j}\mathbf{V} = \mathbf{V}\mathbf{D}_{x_j}. \qquad (15)$$

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.

7. Experimental Validation

We are now ready to try the techniques developed in the previous sections on some data. In the following, we consider two recently solved minimal cases of structure from motion, which can be used in a RANSAC engine. We also consider the problem of optimal three view triangulation, which can be solved by calculating all stationary points of the likelihood function. Since the techniques described in this paper improve the numerical stability of the solver itself, but do not affect the conditioning of the actual problem, there is no point in considering the behavior under noise. Hence we will use synthetically generated examples without noise to compare the intrinsic numerical stability of the different methods.

In our experiments we have compared four different versions for solving the system of equations; (i) The standard method with a monomial basis used in *e.g.* [15, 16, 17], (ii) The new method with an svd-based change of basis introduced in this paper, (iii) The standard method using eigenvalues instead of eigenvectors and (iv) a combination of using eigenvalues and changing basis.

As we will see, the best results are obtained when the new change of basis method is combined with the use of eigenvalues. By the trick introduced in Section 6, we extract solutions from eigenvalues in more or less the same speed as using eigenvectors. Taking the trouble of actually computing the eigenvalues of the action matrix for each variable gives still better performance, but only marginally and since this is notably slower we will only consider the faster method.

7.1. Relative Pose for Generalized Cameras

Generalized cameras provide a generalization of the standard pin-hole camera in the sense that there is no common focal point through which all image rays pass, *c.f.* [14]. 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 [15] with Gröbner basis techniques. The problem equations can be set up using quaternions to parameterize the rotation, Plücker representation of the lines and a generalized epipolar constraint which captures the relation between the lines. After some manipulations one obtains a set of sixth degree equations in the three quaternion parameters v_1, v_2 and v_3. For details, see [15]. 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 some intermediate removal of linearly dependent equations 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 30.000 randomly generated test cases. The results from this experiment are shown in Figure 1. Table 1 gives the 95th percentile of the error distribution. This metric is relevant *e.g.* if the solver is to be used in a RANSAC algorithm. As can be seen, the combination of conditioning by a good choice of basis and using eigenvalues instead of eigenvectors yields drastically improved numerical precision over the state-of-the-art method.

Method	95th percentile error
standard basis	1.52×10^{-2}
standard + eigenvalues	6.51×10^{-4}
svd basis	9.42×10^{-5}
svd + eigenvalues	6.20×10^{-10}

Table 1. Evaluation of numerical stability for relative pose for generalized cameras. We compare the angular error of the estimated rotation matrix. The technique for numerical conditioning with a change of basis combined with using eigenvalues yields an improvement in numerical precision by approximately a factor 10^8 over the state-of-the-art 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 new method, the condition number is decreased by about a factor 10^5.

Figure 1 also shows a scatter plot of error versus condition number for the generalized camera solver. The new 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^{-16} which means that we have err $\approx 10^{-16}\kappa = \epsilon_{mach}\kappa$. This observation agrees well with (13) and justifies our strategy of minimizing the condition number.

7.2. Relative Pose with Unknown Focal Length

Relative pose for calibrated cameras is a well known problem and the standard minimal case for this is five points in two views. There are in general ten solutions to this problem which was in principle shown by Kruppa [12] in 1913, and corrected by Demazure [4] in 1988. For the same problem but with unknown focal length, the corresponding minimal case is six points in two views [17], which was solved in 2005 by Stewénius *et al.* also using Gröbner basis techniques.

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

$$F = F_0 + F_1 l_1 + F_2 l_2. \tag{16}$$

Then putting $f^{-2} = p$ one obtains nine equations from the constraint on the essential matrix [13]

$$2EE^t E - \mathrm{tr}(EE^t)E = 0. \tag{17}$$

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 $\mathbf{C}$ by multiplying these ten equations by p so that the degree of p reaches a maximum of four. This gives

Method	95th percentile error
standard basis	3.38×10^{-7}
svd basis	3.13×10^{-10}
svd + eigenvalues	3.81×10^{-12}

Table 2. Evaluation of numerical stability for relative pose for regular cameras with unknown focal length. We compare the estimation error in relative focal length. The combined svd and eigenvalue method yields an improvement in numerical precision by approximately a factor 10^5 over the state-of-the-art method.
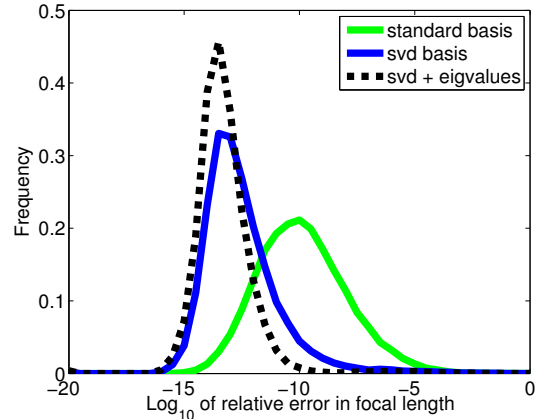

Figure 2. Histogram over the error in relative focal length estimated in the solver for relative pose for standard cameras with unknown focal length.

34 equations in a total of 50 monomials. It turns out that it is possible to eliminate only one of the four monomials $l_2^3 p^4$, $l_2^2 p^4$, $l_2 p^4$ 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 $l_2^2 p^4$, 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 methods have been compared on 30.000 test cases and the relative errors on the focal length are shown in Table 2 and Figure 2.

The improvement here is not as large as for the generalized camera. This is probably due to the smaller reduction that is needed to obtain the Gröbner basis. Still the improvement in precision is as large as a factor 10^5.

7.3. Optimal Three View Triangulation

The last experiment is on optimal triangulation from three views. The problem is formulated as finding the world point that minimizes the sum of squares of the reprojection error. This means that we are minimizing the likelihood function, thus obtaining a statistically optimal estimate. A
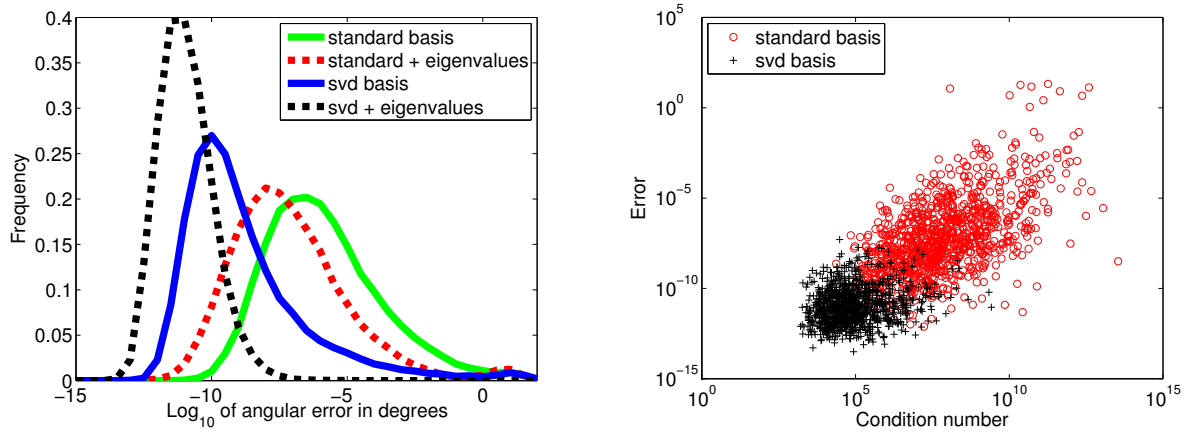
Figure 1. *Left*: Histogram over the angular error in degrees of the estimated rotation matrix in the solver for relative pose for generalized cameras. As can be seen, the new method shows drastically improved numerical precision. *Right*: The angular error plotted versus the condition number of $\mathbf{C}_{nb}$. As can be seen we have a linear trend in the logarithmic scale with slope approximately one and a y-axis intersection at about 10^{-16}. This means that we have err $\approx 10^{-16}\kappa = \epsilon_{mach}\kappa$.

solution to this problem was presented by Stewénius *et al.* in [16]. 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 [16] 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 now possible to take the step to double-precision arithmetics.

To construct the solver for this example some changes in the algorithm of [16] 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 [16] 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 [6]. The 11 solutions at the origin can be ignored and the other three can easily be filtered out in a later stage.

To build the solver we use the nine equation from the saturated ideal (3 of degree 5 and 6 of degree 6) and multiply with x, y and z up to degree 9. This gives 225 equations in 209 different monomials. The easiest way to get rid of the

Method	95th percentile error
standard basis	1.51×10^1
svd basis	2.63×10^{-5}
svd + eigenvalues	1.27×10^{-5}

Table 3. Evaluation of numerical stability for optimal three view triangulation. We compare the estimation error in 3D placement of the point. The combined svd and eigenvalue method yields an improvement in numerical precision by approximately a factor 10^6 over the state-of-the-art method.

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 30.000 test cases and the errors are shown in Table 3 and Figure 3.

In this example the precision is improved by approximately a factor 10^6. With this improvement it is know possible to use IEEE double-precision and get good results.

8. Conclusions

In this paper we have shown how the precision of numerical Gröbner basis algorithms for solving systems of polynomial equations can be dramatically improved by clever choice of basis in the quotient space $\mathbb{R}[\mathbf{x}]/I$. The methods have been verified on three examples of problems which involve such systems of equations and we demonstrate improvements in numerical precision roughly by a factor 10^5 to 10^8. The price we have to pay for this is the extra computational cost of the svd used to select the new basis. This
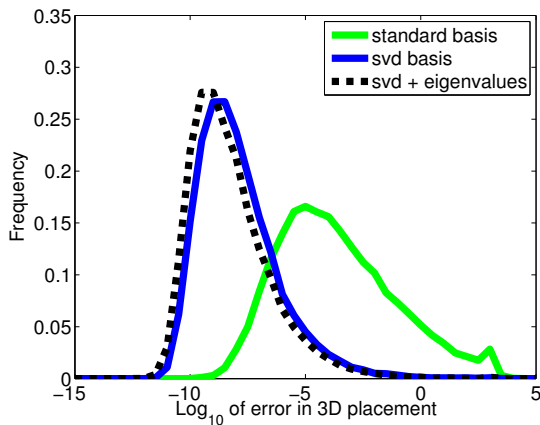
Figure 3. Histogram over the error in 3D placement of the unknown point obtained using optimal three view triangulation.

will be large or even dominating if **C** is large. However, using a change of basis might absolutely necessary to get a usable solver. In [16] where 128 bit arithmetics had to be used for three view triangulation, the computation time was 30s for one problem instance (due to the expensive emulation of high precision multiplication). Our experiments indicate that using the new techniques allowing standard double precision, it should be possible to solve the same problem in roughly 50 milliseconds. Consequently, it should now also be possible to approach larger and more difficult problems in general.

As was seen in Section 7, the deciding factor for the numerical precision is the condition number and we thus aim at minimizing this. However, the heuristic approach we use does most likely not yield the optimum and it is still an open question how close we get. It would be interesting to use *e.g.* gradient descent to see how much further it would be possible to decrease the condition number and what this would do for the precision.

Acknowledgment

References

[1] S. Agarwal, M. K. Chandraker, F. Kahl, D. J. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. In *Proc. 9th European Conf. on Computer Vision, Graz, Austria*, pages 592–605, 2006.

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

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

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

[5] 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.

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

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

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

[9] M. T. Heath. *Scientific Computing : An introductory Survey*. McGraw-Hill, 1996.

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

[11] 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.

[12] E. Kruppa. Zur Ermittlung eines Objektes Zwei Perspektiven mit innerer Orientierung. *Sitz-Ber. Akad. Wiss., Wien, math. naturw. Kl. Abt*, IIa(122):1939–1948, 1913.

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

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

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

[16] 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.

[17] H. Stewńius, 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.

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

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

[20] 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.