

A Column-Pivoting Based Strategy for Monomial Ordering in Numerical Gröbner Basis Calculations

Martin Byröd, Klas Josephson and Kalle Åström

Centre For Mathematical Sciences,
Lund University, Lund, Sweden
{byrod, klasj, kalle}@maths.lth.se

Abstract. This paper presents a new fast approach to improving stability in polynomial equation solving. 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. An interesting approach to stabilising the computations is to study basis selection for the quotient space $\mathbb{C}[\mathbf{x}]/I$. In this paper, the exact matrix computations involved in the solution procedure are clarified and using this knowledge we propose a new fast basis selection scheme based on QR-factorization with column pivoting. We also propose an adaptive scheme for truncation of the Gröbner basis to further improve stability. The new basis selection strategy is studied on some of the latest reported uses of Gröbner basis methods in computer vision and we demonstrate a fourfold increase in speed and nearly as good over-all precision as the previous SVD-based method. Moreover, we get typically get similar or better reduction of the largest errors.¹

1 Introduction

A large number of geometric computer vision problems can be formulated in terms of a system of polynomial equations in one or more variables. A typical example of this is minimal problems of structure from motion [1, 2]. This refers to solving a specific problem with a minimal number of point correspondences. Further examples of minimal problems are relative motion for cameras with radial distortion [3] or for omnidirectional cameras [4]. Solvers for minimal problems are often used in the inner loop of a RANSAC engine to find inliers in noisy data, which means that they are run repeatedly a large number of times. There is thus a need for fast and stable algorithms to solve systems of polynomial equations.

Another promising, but difficult pursuit in computer vision (and other fields) is global optimization for *e.g.* optimal triangulation, resectioning and fundamental matrix estimation. See [5] and references therein. In some cases these

¹ This work has been funded by the Swedish Research Council through grant no. 2005-3230 'Geometry of multi-camera systems' and grant no. 2004-4579 'Image-Based Localization and Recognition of Scenes'.

optimization problems can be solved by finding the complete set of zeros of polynomial equations [6, 7].

Solving systems of polynomial equations is known to be numerically very challenging and there exist no stable algorithm for the general case. Instead, specific solver algorithms are developed for each application. The state-of-the-art method for doing this is calculations with Gröbner bases. Typically, one obtains a floating point version of Buchberger’s algorithm [8] by rewriting the various elimination steps using matrices and matrix operations [9]. These techniques have been studied and applied to vision problems in a number of cases [3, 10, 4]. However, for larger and more demanding problems Gröbner basis calculations are plagued by numerical problems [11, 6].

A recently introduced, interesting approach to stabilisation of Gröbner basis computations is to study basis selection for the quotient space $\mathbb{C}[\mathbf{x}]/I$ [12], where I is the ideal generated by the set of equations. The choice of basis has been empirically shown to have a great impact on numerical performance and by adaptively selecting the basis for each instance of a problem one can obtain a dramatic increase in stability. In [12], a scheme based on singular value decomposition (SVD) was used to compute an orthogonal change of basis matrix. The SVD is a numerically very stable factorization method, but unfortunately also computationally rather expensive. Since the involved matrices tend to be large (around hundred rows and columns or more), the SVD computation easily dominates the running time of the algorithm.

In this paper, we propose a new fast strategy for selecting a basis for $\mathbb{C}[\mathbf{x}]/I$ 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 a crucial elimination step and obtain a simultaneous selection of basis and triangular factorization. With this method, we demonstrate an approximately fourfold increase in speed over the previous SVD based method while retaining good numerical stability.

Moreover, the technique of truncating the Gröbner basis to avoid large errors introduced in [13] fits nicely within the framework of column pivoting. Since the pivot elements are sorted in descending order, we get an adaptive criterion for where to truncate the Gröbner basis 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.

Factorization with column pivoting is a well studied technique and there exist highly optimized and reliable implementations of these algorithms in *e.g.* LAPACK [14], which makes this technique accessible and straight forward to implement. Matlab code for one of the applications, optimal triangulation from three views, is available at <http://www.maths.lth.se/vision/downloads>.

2 Review of Gröbner Basis Techniques for Polynomial Equation Solving

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

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

$$f_1(\mathbf{x}) = 0, \dots, f_m(\mathbf{x}) = 0, \quad (1)$$

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

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

where the $h_k \in \mathbb{C}[\mathbf{x}]$ are any polynomials. The reason for studying the ideal I is that it has the same set of zeros as (1).

Consider now the space of equivalence classes modulo I . This space is denoted $\mathbb{C}[\mathbf{x}]/I$ and referred to as the *quotient space*. Two polynomials f and g are said to be equivalent modulo I if $f = g + h$, where $h \in I$. The logic behind this definition is that we get true equality, $f(\mathbf{x}) = g(\mathbf{x})$ on zeros of (1).

To do calculations in $\mathbb{C}[\mathbf{x}]/I$ it will be necessary to compute unique representatives of the equivalence classes in $\mathbb{C}[\mathbf{x}]/I$. Let $[\cdot] : \mathbb{C}[\mathbf{x}] \rightarrow \mathbb{C}[\mathbf{x}]/I$ denote the function that takes a polynomial f and returns the associated equivalence class $[f]$. We would now like to compose $[\cdot]$ with a mapping $\mathbb{C}[\mathbf{x}]/I \rightarrow \mathbb{C}[\mathbf{x}]$ that associates to each equivalence class a unique representative in $\mathbb{C}[\mathbf{x}]$. The composed map $\mathbb{C}[\mathbf{x}] \rightarrow \mathbb{C}[\mathbf{x}]$ should in other words take a polynomial f and return the unique representative \bar{f} for the equivalence class $[f]$ associated with f . Assume for now that we can compute such a mapping. This operation will here be referred to as reduction modulo I .

A well known result from algebraic geometry now states that if the set of equations (1) has r zeros, then $\mathbb{C}[\mathbf{x}]/I$ will be a finite-dimensional linear space with dimension r [8]. Moreover, an elegant trick based on calculations in $\mathbb{C}[\mathbf{x}]/I$ yields the complete set of zeros of (1) in the following way: Consider multiplication by one of the variables x_k . This is a linear mapping from $\mathbb{C}[\mathbf{x}]/I$ to itself and since we are in a finite-dimensional space, by selecting an appropriate basis, this mapping can be represented as a matrix \mathbf{m}_{x_k} . This matrix is known as the *action matrix* and the eigenvalues of \mathbf{m}_{x_k} correspond to x_k evaluated at the zeros of (1) [8]. Moreover, the eigenvectors of \mathbf{m}_{x_k} correspond the vector of basis monomials/polynomials evaluated at the same zeros and thus the complete set of solutions can be directly read off from these eigenvectors. The action matrix can be seen as a generalization of the companion matrix to the multivariate case.

Given a linear basis $\mathcal{B} = \{[e_i]\}_{i=1}^r$ spanning $\mathbb{C}[\mathbf{x}]/I$, the action matrix \mathbf{m}_{x_k} is computed by calculating $\overline{x_k e_i}$ for each of the basis elements \bar{e}_i . Performing this

operation is the difficult part in the process. Traditionally, the reduction has been done by fixing a monomial ordering and then computing a Gröbner basis G for I , which is a canonical set of polynomials that generate I . Computing \bar{f} is then done by polynomial division by G (usually written \bar{f}^G).

We now make two important observations: (i) We are not interested in finding the Gröbner basis *per se*; it is enough to get a well defined mapping \bar{f} and (ii) it suffices to calculate reduction modulo I on the elements $x_k e_i$, *i.e.* we do not need to know what \bar{f} is on all of $\mathbb{C}[\mathbf{x}]$. Note that if for some i , $x_k e_i \in \mathcal{B}$ then nothing needs to be done for that element. With this in mind, we denote by $\mathcal{R} = x_k \mathcal{B} \setminus \mathcal{B}$ the set of elements f for which we need to calculate representatives \bar{f} of their corresponding equivalence classes $[f]$ in $\mathbb{C}[\mathbf{x}]/I$.

Calculating the Gröbner basis of I is typically accomplished by Buchberger's algorithm. This works well in exact arithmetic. However, in floating point arithmetic Buchberger's algorithm very easily becomes unstable. There exist some attempts to remedy this [15, 16], but for more difficult cases it is necessary to study a particular class of equations (*e.g.* relative orientation for omnidirectional cameras [4], optimal three view triangulation [6], etc.) and use knowledge of what the structure of the Gröbner basis should be to design a special purpose Gröbner basis solver [9].

In this paper we move away from the goal of computing a Gröbner basis for I and focus on computing \bar{f} for $f \in \mathcal{R}$ as mentioned above. However, it should be noted that the computations we do much resemble those necessary to get a Gröbner basis.

2.1 Computing Representatives for $\mathbb{C}[\mathbf{x}]/I$

In this section we show how representatives for $\mathbb{C}[\mathbf{x}]/I$ can be efficiently calculated in floating point arithmetic. The reason why Buchberger's algorithm breaks down in floating arithmetic is that eliminations of monomials are performed successively and this causes round-off errors to accumulate to the point where it is impossible to tell whether a certain coefficient should be zero or not. The trick introduced by Faugere [15] is to write the list of equations on matrix form

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

where $\mathbf{X} = [\mathbf{x}^{\alpha_1} \dots \mathbf{x}^{\alpha_n}]^t$ is a vector of monomials with the notation $\mathbf{x}^{\alpha_k} = x_1^{\alpha_{k1}} \dots x_s^{\alpha_{ks}}$ 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.

By combining this approach with knowledge about a specific problem obtained in advance with a computer algebra system such as Macaulay2 [17] it is possible to write down a fixed number of expansion/elimination steps that will generate the necessary polynomials.

In this paper, we use a linear basis of monomials $\mathcal{B} = \{\mathbf{x}^{\alpha_1}, \dots, \mathbf{x}^{\alpha_r}\}$ for $\mathbb{C}[\mathbf{x}]/I$. Recall now that we need to compute $\bar{x_k \mathbf{x}^{\alpha_i}}$ for $x_k \mathbf{x}^{\alpha_i} \notin \mathcal{B}$, *i.e.* for \mathcal{R} . This is the aim of the following calculations.

Begin by multiplying the equations (1) by a large enough set of monomials producing an equivalent (but larger) set of equations. We will come back to what *large enough* means. Thereafter, stack the coefficients of the new equations in an expanded coefficient matrix \mathbf{C}_{exp} , yielding

$$\mathbf{C}_{\text{exp}}\mathbf{X}_{\text{exp}} = 0. \tag{4}$$

Now partition the set of all monomials \mathcal{M} occurring in the expanded set of equations as $\mathcal{M} = \mathcal{E} \cup \mathcal{R} \cup \mathcal{B}$ and order them so that $\mathcal{E} > \mathcal{R} > \mathcal{B}$ holds for all monomials in their respective sets. The monomials \mathcal{E} (\mathcal{E} for excessive) are simply the monomials which are neither in \mathcal{R} nor in \mathcal{B} . This induces a corresponding partitioning and reordering of the columns of \mathbf{C}_{exp} :

$$[\mathbf{C}_{\mathcal{E}} \ \mathbf{C}_{\mathcal{R}} \ \mathbf{C}_{\mathcal{B}}] \begin{bmatrix} \mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0. \tag{5}$$

The \mathcal{E} -monomials are not in the basis and do not need to be reduced so we eliminate them by an LU decomposition on \mathbf{C}_{exp} yielding

$$\begin{bmatrix} \mathbf{U}_{\mathcal{E}_1} & \mathbf{C}_{\mathcal{R}_1} & \mathbf{C}_{\mathcal{B}_1} \\ 0 & \mathbf{U}_{\mathcal{R}_2} & \mathbf{C}_{\mathcal{B}_2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{E}} \\ \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0, \tag{6}$$

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

$$[\mathbf{U}_{\mathcal{R}_2} \ \mathbf{C}_{\mathcal{B}_2}] \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = 0, \tag{7}$$

from which we get the elements of the ideal I we need since equivalently, if the submatrix $\mathbf{U}_{\mathcal{R}_2}$ is of full rank, we have

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

and then the \mathcal{R} -monomials can be expressed uniquely in terms of the \mathcal{B} -monomials. As previously mentioned, this is precisely what we need to compute the action matrix \mathbf{m}_{x_k} in $\mathbb{C}[\mathbf{x}]/I$. In other words, the property of $\mathbf{U}_{\mathcal{R}_2}$ as being of full rank is sufficient to get the operation \bar{f} on the relevant part of $\mathbb{C}[\mathbf{x}]$. Thus, in designing the set of monomials to multiply with (the first step in the procedure) we can use the rank of $\mathbf{U}_{\mathcal{R}_2}$ as a criterion for whether the set is large enough or not. However, the main problem in these computations is that even if $\mathbf{U}_{\mathcal{R}_2}$ is in principle invertible, it can be *very* ill conditioned.

A technique introduced in [12], which alleviates much of these problems uses basis selection for $\mathbb{C}[\mathbf{x}]/I$. The observation is that the right linear basis for $\mathbb{C}[\mathbf{x}]/I$ induces a reordering of the monomials, which has the potential to drastically improve the conditioning of $\mathbf{U}_{\mathcal{R}_2}$. Since \mathbf{C}_{exp} depends on the data, the choice of linear basis cannot be made on beforehand, but has to be computed adaptively

each time the algorithm is run. This leads to the difficult optimisation problem of selecting a linear basis so as to minimize the condition number of $\mathbf{U}_{\mathcal{R}_2}$. In [12] this problem was addressed by making use of SVD providing a numerically stable, but computationally expensive solution.

The advantage of the above exposition is that it makes explicit the dependence on the matrix $\mathbf{U}_{\mathcal{R}_2}$, both in terms of rank and conditioning. In particular, the above observations leads to the new fast strategy for basis selection which is the topic of the next section and a major contribution of this paper.

3 Column Pivoting as Basis Selection Strategy

In the one-variable case the monomials are given a natural ordering by their degree. In the multivariate case, there are several ways to order the monomials. To specify representatives for $\mathbb{C}[\mathbf{x}]/I$, one traditionally fixes one of these. The monomial order then automatically produces a linear basis for $\mathbb{C}[\mathbf{x}]/I$ in form of the set of monomials which are not divisible by the Gröbner basis in that monomial order.

For Buchberger's algorithm to make sense a monomial order is required to respect multiplication, *i.e.* $\mathbf{x}^\alpha > \mathbf{x}^\beta \Rightarrow x_k \mathbf{x}^\alpha > x_k \mathbf{x}^\beta$. Interestingly, when we relax the requirement of getting a strict Gröbner basis and compute \bar{f} as outlined in the previous section, this property is unnecessarily strict. The crucial observation is that we can choose any linear basis for $\mathbb{C}[\mathbf{x}]/I$ we like, as long as we are able to compute well defined representatatives for the equivalence classes of $\mathbb{C}[\mathbf{x}]/I$. Thus, instead of letting the monomial order dictate the linear basis, we would like to do it the other way around and start by choosing a set of basis monomials \mathcal{B} .

After noting that we have some freedom in choosing \mathcal{B} , the first question is which monomials \mathcal{P} (for permissible) in \mathcal{M} are eligible for inclusion in the linear basis? Since we have to reduce the set $x_k \mathcal{B} \setminus \mathcal{B}$ to $\mathbb{C}[\mathbf{x}]/I$ we obviously have to require $x_k \mathcal{P} \subset \mathcal{M}$. Moreover, by making the construction leading up to (8), but replacing \mathcal{B} by \mathcal{P} we see that again the resulting $\mathbf{U}_{\mathcal{R}_2}$ needs to be of full rank to be able to guarantee reduction modulo I for all elements.

With these properties in place we aim at selecting \mathcal{P} as large as possible 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 again performing an LU factorization and discarding the top rows to get rid of the \mathcal{E} -monomials, we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}_2} & \mathbf{C}_{\mathcal{P}_2} \\ \mathbf{0} & \mathbf{C}_{\mathcal{P}_3} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}} \end{bmatrix} = \mathbf{0}, \quad (9)$$

in analogy with (7), where we now get zeros below $\mathbf{U}_{\mathcal{R}_2}$ since the larger $\mathbf{C}_{\mathcal{P}}$ means that we can still eliminate further. This is where the basis selection comes to play.

As noted above we can choose which monomials of the p monomials in \mathcal{P} to put in the basis and which to reduce. This is equivalent to choosing a permutation

Π of the columns of $\mathbf{C}_{\mathcal{P}_3}$ so that

$$\mathbf{C}_{\mathcal{P}_3}\Pi = [c_{\pi(1)} \cdots c_{\pi(p)}]. \quad (10)$$

The goal must thus be to make this choice so as to minimize the condition number $\kappa([c_{\pi(1)} \cdots c_{\pi(p-r)}])$ of the first $p-r$ columns of the permuted matrix. In its generality, this is a difficult combinatorial optimization problem. However, the task can be approximately solved in an attractive way by QR factorization with column pivoting [18]. With this algorithm, $\mathbf{C}_{\mathcal{P}_3}$ is factorized as

$$\mathbf{C}_{\mathcal{P}_3}\Pi = \mathbf{Q}\mathbf{U}, \quad (11)$$

where \mathbf{Q} is orthogonal and \mathbf{U} is upper triangular. By solving for $\mathbf{C}_{\mathcal{P}_3}$ in (11) and substituting into (9) followed by multiplication from the left with $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}^t \end{bmatrix}$ and from the right with $\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \Pi \end{bmatrix}$, we get

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}_2} & \mathbf{C}_{\mathcal{P}_2}\Pi \\ \mathbf{0} & \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \Pi^t \mathbf{X}_{\mathcal{P}} \end{bmatrix} = \mathbf{0}, \quad (12)$$

We observe that \mathbf{U} is not quadratic and emphasize this by writing $\mathbf{U} = [\mathbf{U}_{\mathcal{P}_3} \ \mathbf{C}_{\mathcal{B}_2}]$, where $\mathbf{U}_{\mathcal{P}_3}$ is quadratic upper triangular. We also write $\mathbf{C}_{\mathcal{P}_2}\Pi = [\mathbf{C}_{\mathcal{P}_4} \ \mathbf{C}_{\mathcal{B}_1}]$ and $\Pi^t \mathbf{X}_{\mathcal{P}_2} = [\mathbf{X}_{\mathcal{P}'} \ \mathbf{X}_{\mathcal{B}}]^t$ yielding

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}_2} & \mathbf{C}_{\mathcal{P}_4} & \mathbf{C}_{\mathcal{B}_1} \\ \mathbf{0} & \mathbf{U}_{\mathcal{P}_3} & \mathbf{C}_{\mathcal{B}_2} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \\ \mathbf{X}_{\mathcal{B}} \end{bmatrix} = \mathbf{0} \quad (13)$$

and finally

$$\begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{X}_{\mathcal{P}'} \end{bmatrix} = - \begin{bmatrix} \mathbf{U}_{\mathcal{R}_2} & \mathbf{C}_{\mathcal{P}_4} \\ \mathbf{0} & \mathbf{U}_{\mathcal{P}_3} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{C}_{\mathcal{B}_1} \\ \mathbf{C}_{\mathcal{B}_2} \end{bmatrix} \mathbf{X}_{\mathcal{B}} \quad (14)$$

is the equivalent of (8) and amounts to solving r upper triangular equation systems which can be efficiently done by back substitution.

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

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

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

3.1 Adaptive Truncation

A further neat feature of QR factorization with column pivoting is that it provides a way of numerically estimating the conditioning of $\mathbf{C}_{\mathcal{P}}$ simultaneously with the elimination. In [13], it was shown that for reductions with a Gröbner basis, the Gröbner basis could be truncated yielding a larger representation of $\mathbb{C}[\mathbf{x}]/I$ (more than r basis elements), while retaining the original set of solutions. The advantage of this is that the last elements of the Gröbner basis often are responsible for a major part of the numerical instability and making use of the observation in [13], the last elements do not have to be computed.

As discussed earlier we do not calculate exactly a Gröbner basis, but the method of [13] is straightforward to adapt to the framework of this paper. However, both rank and conditioning of $\mathbf{C}_{\mathcal{P}}$ might depend on the data and we would therefore like to decide adaptively where to truncate, *i.e.* when to abort the QR factorization.

As a consequence of how the QR algorithm is formulated, the elements u_{kk} on the main diagonal of \mathbf{U} will be sorted in decreasing absolute value. In exact arithmetic, if the rank is q , then $u_{kk} = 0$ for $k > q$. In floating point this will not be the case due to round-off errors. However, we can set a threshold τ and abort the elimination process once $|u_{kk}|/|u_{11}| < \tau$. The remaining columns (monomials) are then transferred to the basis which is correspondingly expanded.

Apart from being numerically sound, this strategy also spares some computational effort compared to setting a fixed larger basis. Truncating the set of polynomials means a higher dimensional representation of $\mathbb{C}[\mathbf{x}]/I$, which means we have to solve a larger eigenvalue problem. As will be shown in the experiments, the basis can usually be kept tight and only needs to be expanded in some cases.

4 Experiments

The purpose of this section is to verify the speed and accuracy of the QR-method. To this end, three different applications are studied. The first example is relative pose for generalised cameras, first solved by Stewénus *et al.* in 2005 [19]. The second one is the previously unsolved minimal problem of pose estimation with unknown focal length. The problem was formulated by Josephson *et al.* in [20], but not solved in floating point arithmetic. The last problem is optimal triangulation from three views [6].

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 all three examples we compare with the “standard” method, by which we mean to fix a monomial order (typically grevlex) and use the basis dictated by that order together with straightforward gauss jordan elimination to express

monomials in terms of the basis. Previous works have often used several expansion / elimination rounds. We have found this to have a negative effect on numerical stability so to make the comparison fair, we have implemented the standard method using a single elimination step in all cases.

For the adaptive truncation method, the threshold τ for the ratio between the k :th diagonal element and the first was set to 10^{-8} .

4.1 Relative Pose for Generalised Cameras

A generalised camera is a camera with no common focal point. This *e.g.* serves as a useful model for several ordinary cameras together with fixed relative locations [21]. For generalised cameras there is a minimal case for relative pose with two cameras and six points. This problem was solved in [19] and has 64 solutions. In [12] this problem was used to show how the SVD-method improved the numerics. We follow the methods of the later paper to get a single elimination step. This gives an expanded coefficient matrix of size 101×165 with the columns representing monomials up to degree eight in three variables. For details see [19] and [12].

The examples for this experiment were generated by picking six points from a normal distribution centered at the origin. Then six randomly chosen lines through these point were associated to each camera. This made up two generalised cameras with a relative orientation and translation.

Following this recipe, 10000 examples were generated and solved with the standard, QR- and SVD-method. The angular errors between true and estimated motion were measured. The results are shown in Figure 1.

The method with variable basis size was also implemented, but for this example the $\mathbf{U}_{\mathcal{R}_2}$ (see Equation 7) part of the coefficient matrix was always reasonably conditioned and hence the basis size was 64 in all 10000 test examples. There were no large errors for neither the SVD nor the QR method.

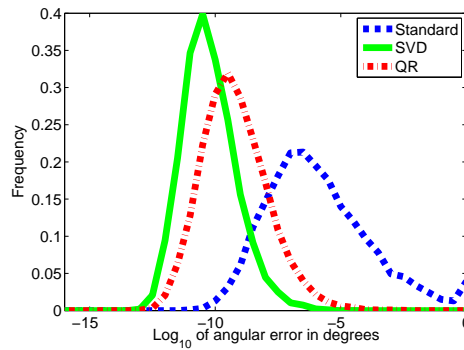


Fig. 1. Error distributions for the problem of relative pose with generalised cameras. The SVD-method yields the best results but the faster QR-method is not far behind and also eliminates all large errors.

4.2 Localisation with Hybrid Features

This problem was introduced in [20]. The problem is to find 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 camera. These types of mixed features are called hybrid features. In [20], the authors propose a parameterisation of the problem but no solution was given apart from showing that the problem has 36 solutions.

The parameterisation in [20] 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}_{exp} 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 according to (5). In this problem $\mathbf{C}_{\mathcal{P}}$ corresponds to all monomials up to degree 4 except f^4 where f is the focal length, this gives 69 columns in $\mathbf{C}_{\mathcal{P}}$. The part $\mathbf{C}_{\mathcal{R}}$ corresponds to the 5:th degree monomials that appears when the monomials in \mathcal{B} 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 an invertible $\mathbf{U}_{\mathcal{R}_2}$. 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 over what was produced by the adaptive criterion was beneficial for the stability. In this experiment, we added three basis elements to the automatically produced basis. To get a working version of the SVD solver we had to adapt the truncation method to the SVD case as well. We did this by looking at the ratio of the singular values.

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 10000 times.

In Figure 2 (right) 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.

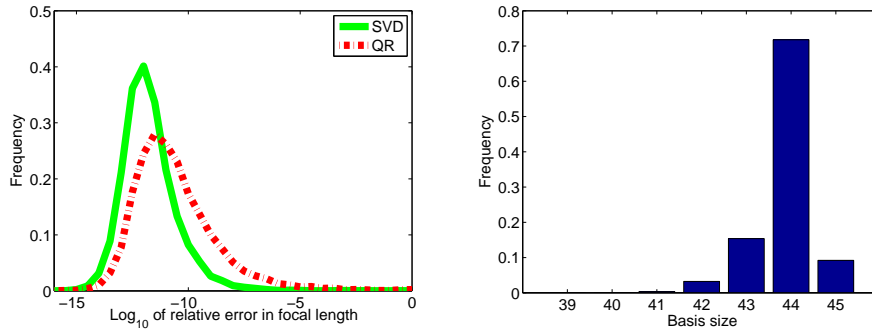


Fig. 2. Left: Relative error in focal length for pose estimation with unknown focal length. Both the SVD- and QR-methods uses adaptive truncation. Right: The size of the adaptively chosen basis for the QR-method. For the SVD-method the size differs from this in less than 3% of the cases and by at most one element.

Figure 2 (left) 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.

4.3 Optimal Triangulation from Three Views

The last experiment does not concern a geometrical minimal case, but instead deals with an optimisation problem. Given noisy image measurements in three views, the problem is to find the world point that minimises the sum of squares of reprojection errors. This is the statistically optimal estimate under Gaussian noise.

We find the global minimum by calculating the complete set of stationary points of the reprojection error function. This was first done in [6], where the standard Gröbner basis method was used. However, because of numerical problems they were forced to use extremely slow, emulated 128 bit numerics to get accurate results. In [12] the SVD-method was later used to enable calculations in standard double precision. It should be mentioned that a more recently introduced and probably more practical global optimisation method for triangulation is given in [22]. Still though, this problem serves as an interesting test bed for equation solving.

For details on the construction of the coefficient matrix see [12, 6]. The coefficient matrix constructed with this method is of size 225×209 and the number of solutions is 50. The QR-method was implemented as described earlier and the method with variable size basis was used. For reference, we implemented the method of [6] in standard double precision, with some small modifications to get a single elimination step (this made it slightly more stable).

The unknown point was randomly placed in a cubic box with side 1000 centered around the origin. The three cameras were placed approximately on a sphere with distance 1000 from origin and the focal lengths were also set to around 1000. The error in 3D placement over 10000 iterations is shown in Figure 3. It can be seen that the QR-method is almost as accurate as the SVD-method.

One important property of a solver is that the number of large errors is small. Thus in Table 1 the number of large errors are shown. The results show that the QR-method is better at suppressing large errors, probably due to the variable size of the basis.

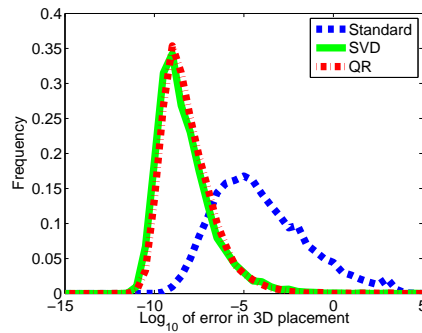


Fig. 3. The distribution of the error in 3D placement of the unknown point using optimal three view triangulation. The experiment was run 10000 times. The QR-method gives nearly identical results compared to the SVD-method.

Error	> 1	$> 10^{-1}$	$> 10^{-2}$	$> 10^{-3}$
QR	10	15	28	54
SVD	31	39	52	79

Table 1. Number of errors larger than some levels. This shows that the QR-method gives fewer large errors probably due to the variable size of the basis.

Basis size	50	51	52	53	54	55	≥ 56
#	9471	327	62	34	26	17	58

Table 2. Number of times a certain basis size appears in 10000 iterations. The largest basis size obtained in the experiment was 66.

4.4 Speed Comparison

In the problem of optimal three view triangulation the execution times for the three different algorithms were measured. Since the implementations were done in Matlab it was necessary to take care to eliminate the effect of Matlab being an interpreted language. To do this only the time after construction of the coefficient matrix was taken into account. This is because the construction of the coefficient matrix essentially amounts to copying coefficients to the right places which can be done extremely fast in *e.g.* a C language implementation.

In the routines that were measured no subroutines were called that were not built-in functions in Matlab. The measurements were done with Matlab's 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, these were constructed from the same type of scene setup as in the previous section. The same set of coefficient matrices was used for each method. The result is given in Table 3. Our results show that the QR-method with adaptive truncation is approximately four times faster than the SVD-method but 40% slower than the standard method. It should however be noted that here, the standard method is by far too inaccurate to be of any practical value.

Method	Time per call / ms	Relative time
SVD	41.685	1
QR	10.937	0.262
Standard	8.025	0.193

Table 3. Time consumed in the solver part for the three different methods. The time is an average over 1000 calls.

5 Conclusions

In this paper we have presented a new fast strategy for improving numerical stability of Gröbner basis polynomial equation solvers. The key contribution is a clarification of the exact matrix operations involved in computing an action matrix for $\mathbb{C}[\mathbf{x}]/I$ and the use of numerically sound QR factorization with column pivoting to obtain a simultaneous basis selection for $\mathbb{C}[\mathbf{x}]/I$ and reduction to upper triangular form. We demonstrate a nearly fourfold decrease in computation time compared to the previous SVD based method while retaining good numerical stability. Moreover, since the method is based on the well studied, freely available QR algorithm it is reasonably simple to implement and not much slower than using no basis selection at all.

The conclusion is thus that whenever polynomial systems arise and numerical stability is a concern, this method should be of interest.

References

1. Chasles, M.: Question 296. *Nouv. Ann. Math.* **14** (1855)
2. Kruppa, E.: Zur Ermittlung eines Objektes aus Zwei Perspektiven mit innerer Orientierung. *Sitz-Ber. Akad. Wiss., Wien, math. naturw. Kl. Abt IIa* (1913) 1939–1948
3. Kukulova, Z., Pajdla, T.: A minimal solution to the autocalibration of radial distortion. In: *CVPR*. (2007)
4. Geyer, C., Stewénius, H.: A nine-point algorithm for estimating para-catadioptric fundamental matrices. In: *CVPR*, Minneapolis, USA. (2007)
5. Hartley, R., Kahl, F.: Optimal algorithms in multiview geometry. In: *ACCV*. (2007) 13–34
6. Stewénius, H., Schaffalitzky, F., Nistér, D.: How hard is three-view triangulation really? In: *Proc. Int. Conf. on Computer Vision*, Beijing, China (2005) 686–693
7. Hartley, R., Sturm, P.: Triangulation. *Computer Vision and Image Understanding* **68** (1997) 146–157
8. Cox, D., Little, J., O’Shea, D.: *Ideals, Varieties, and Algorithms*. Springer (2007)
9. Stewénius, H.: *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University (2005)
10. Stewénius, H., Kahl, F., Nistér, D., Schaffalitzky, F.: A minimal solution for relative pose with unknown focal length. In: *Proc. Conf. Computer Vision and Pattern Recognition*, San Diego, USA. (2005)
11. Kukulova, Z., Pajdla, T.: 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)
12. Byröd, M., Josephson, K., Åström, K.: 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)
13. Byröd, M., Josephson, K., Åström, K.: Fast optimal three view triangulation. In: *Asian Conference on Computer Vision*. (2007)
14. Anderson, E., et al.: *LAPACK Users’ Guide*. Third edn. Society for Industrial and Applied Mathematics, Philadelphia, PA (1999)
15. Faugère, J.C.: A new efficient algorithm for computing gröbner bases (f_4). *Journal of Pure and Applied Algebra* **139** (1999) 61–88
16. Faugère, J.C.: A new efficient algorithm for computing gröbner bases without reduction to zero (f_5). In: *ISSAC ’02*, New York, NY, USA, ACM Press (2002) 75–83
17. Grayson, D., Stillman, M.: *Macaulay 2*. <http://www.math.uiuc.edu/Macaulay2> (1993–2002)
18. Golub, G.H., van Loan, C.F.: *Matrix Computations*. 3rd edn. The Johns Hopkins University Press (1996)
19. Stewénius, H., Nistér, D., Oskarsson, M., Åström, K.: Solutions to minimal generalized relative pose problems. In: *OMNIVIS*, Beijing China (2005)
20. Josephson, K., Byröd, M., Kahl, F., Åström, K.: Image-based localization using hybrid feature correspondences. In: *BenCOS 2007*. (2007)
21. Pless, R.: Using many cameras as one. In: *Proc. Conf. Computer Vision and Pattern Recognition*, Madison, USA. (2003)
22. Lu, F., Hartley, R.: A fast optimal algorithm for l_2 triangulation. In: *ACCV*. (2007) 279–288