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

Byröd, Martin; Josephson, Klas; Åström, Karl

Published in:
International Journal of Computer Vision

DOI:
10.1007/s11263-009-0235-z

2009

Noname manuscript No.
(will be inserted by the editor)

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

Martin Byröd · Klas Josephsson · Kalle Åström

the date of receipt and acceptance should be inserted later

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

1 Introduction

Numerous geometric problems in computer vision involve the solution of systems of polynomial equations. This is particularly true for so called minimal structure and motion problems, *e.g.* [11, 29, 42]. Solutions to minimal structure and motion problems can often be used in RANSAC algorithms to find inliers in noisy data [17, 43, 44]. For such applications one needs to efficiently solve a large number of minimal structure and motion problems in order to find the best set of inliers. There is thus a need for fast and numerically stable algorithms for solving particular systems of polynomial equations. Recent applications of polynomial techniques in computer vision include solving for fundamental and essential matrices and radial distortion [9], panoramic stitching [4] and pose with unknown focal length [5].

Another area of recent interest is global optimization used *e.g.* for optimal triangulation, resectioning and fundamental matrix estimation. Global optimization is a promising, but difficult pursuit and different lines of attack have been tried, *e.g.* branch and bound [1], L_∞-norm methods [21,26] and methods using linear matrix inequalities (LMIs) [27]. An alternative way to find the global optimum is to calculate stationary points directly, usually by solving some polynomial equation system [22, 41]. So far, this has been an approach of limited applicability since calculation of stationary points is numerically difficult for larger problems. By using the methods presented in this paper it is possible to handle a somewhat larger class of problems, thus offering an alternative to the above mentioned optimization methods. An example of this is optimal three view triangulation which has previously not been solved in a practical way [41]. We show that this problem can now be solved efficiently with an algorithm implemented in standard IEEE double precision.

Traditionally, researchers have hand-coded elimination schemes in order to solve systems of polynomial equations. Recently, however, new techniques based on algebraic geometry and numerical linear algebra have been used to find all solutions, *cf.* [37]. The outline of such algorithms is that one first studies a specific geometric problem and finds out what structure the Gröbner basis of the ideal I has for that

Martin Byröd · Klas Josephson · Kalle Åström
Centre for Mathematical Sciences, Lund University, Sölvegatan 18 22100 Lund, Sweden, E-mail: byrod@maths.lth.se

Klas Josephson
E-mail: klasj@maths.lth.se

Kalle Åström
E-mail: kalle@maths.lth.se

problem, how many solutions there are and what the degrees of monomials occurring in the Gröbner basis elements are. For each instance of the problem with numerical data, the process of forming the Gröbner basis follows the same steps and the solution to the problem can be written down as a sequence of pre determined elimination steps using numerical linear algebra.

Currently, the limiting factor in using these methods for larger and more difficult cases is numerical problems. For example in [41] it was necessary to use emulated 128 bit numerics to make the system work, which made the implementation very slow. This paper improves on the state of the art of these techniques making it possible to handle larger and more difficult problems in a practical way.

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

Firstly, we show how the sensitive elimination step can be avoided, by using an overly large/redundant basis for $\mathbb{C}[\mathbf{x}]/I$ to construct the action matrix. This method yields the right solutions along with a set of false solutions that can then easily be filtered out by evaluation in the original equations. Note that this basis is not a true basis in the sense that it is linearly dependent.

Secondly, we show how a change of basis in the quotient space $\mathbb{C}[\mathbf{x}]/I$ can be used to improve the numerical precision of the Gröbner basis computations. This approach can be seen as an attempt at finding an optimal reordering or even linear combination of the monomials and we investigate what conditions such a reordering/linear combination needs to satisfy. We develop the tools needed to compute the action matrix in a general linear basis for $\mathbb{C}[\mathbf{x}]/I$ and propose two strategies for selecting a basis which enhances the stability of the solution procedure.

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

Factorization with column pivoting is a very well studied technique and there exist highly optimized and reliable implementations of these algorithms in *e.g.* LAPACK [2], which makes this technique accessible and relatively straightforward to implement.

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

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

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

1.1 Related Work

The area of polynomial equation solving is currently very active. See *e.g.* [10] and references therein for a comprehensive exposition of the state of the art in this field.

One of the oldest and still used methods for non-linear equation solving is the Newton-Raphson method which is fast and easy to implement, but relies heavily on initialization and finds only a single zero for each initialization. In the univariate case, a numerically sound procedure to find the complete set of roots is to compute the eigenvalues of the companion matrix. However, if only real solutions are needed, the fastest way is probably to use Sturm sequences [24].

In several variables a first method is to use resultants [13], which using a determinant construct enables the successive elimination of variables. However, the resultant grows exponentially in the number of variables and is in most cases not practical for more than two variables. In some cases, a related technique known as the hidden variable method can be used to eliminate variables, see *e.g.* [33]. An alternative way of eliminating variables is to compute a lexicographical Gröbner basis for the ideal generated by the equations which can be shown to contain a univariate polynomial representing the solutions [13]. This approach is however often numerically unstable.

A radically different approach is provided by homotopy continuation methods [45]. These methods typically work in conjunction with mixed volume calculations by constructing a simple polynomial system with the same number of zeros as the actual system that is to be solved. The simple system with known zeros is then continuously deformed into the actual system. The main drawback of these methods is the computational complexity with computation time ranging in seconds or more.

At present, the best methods for geometric computer vision problems are based on eigendecomposition of a certain matrices (action matrices) representing multiplication in the quotient space $\mathbb{C}[\mathbf{x}]/I$. The action matrix can be seen as a direct generalization of the companion matrix in the univariate case. The factors that make this approach attractive is that it (i) is fast and numerically feasible, (ii) handles more than two variables and reasonably large numbers of solutions (up to about a hundred) and (iii) is well suited to tuning for specific applications. To the authors best knowledge, this method was first used in the context of computer vision by Stewenius *et al.* [37] even though Gröbner basis methods were mentioned in [23].

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

2 Review of Algebraic Geometry for Equation Solving

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

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

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

We denote by V the zero set of (1). In general V need not be finite, but in this paper we will only consider zero dimensional V, *i.e.* V is a point set.

The general field of study of multivariate polynomials is algebraic geometry. See [13] and [12] for a nice introduction to the field and for proofs of all claims made in this section. In the language of algebraic geometry, V is an affine algebraic variety and the polynomials f_i generate an *ideal* $I = \{g \in \mathbb{C}[\mathbf{x}] : g = \Sigma_i h_i(\mathbf{x}) f_i(\mathbf{x})\}$, where $h_i \in \mathbb{C}[\mathbf{x}]$ are any polynomials and $\mathbb{C}[\mathbf{x}]$ denotes the set of all polynomials in $\mathbf{x}$ over the complex numbers.

The motivation for studying the ideal I is that it is a generalization of the set of equations (1). A point $\mathbf{x}$ is a zero of (1) iff it is a zero of I. Being even more general, we could ask for the complete set of polynomials vanishing on V. If I is equal to this set, then I is called a radical ideal.

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

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

2.1 The Action Matrix

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

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

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 [32]. For V finite, the space $\mathbb{C}[\mathbf{x}]/I$ is finite dimensional. Moreover, if I is radical, then the dimension of $\mathbb{C}[\mathbf{x}]/I$ is equal to $|V|$, *i.e.* the number of solutions [13]. For some $p \in \mathbb{C}[\mathbf{x}]$ consider now the operation $T_p : f(\mathbf{x}) \to p(\mathbf{x}) f(\mathbf{x})$. 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 a corresponding system of polynomial equations. The main issue is now how to compute $\mathbf{m}_p$. This is done by selecting a basis $\mathcal{B}$ for $\mathbb{C}[\mathbf{x}]/I$ and then computing $[p \cdot b_i]$ for each $b_i \in \mathcal{B}$. To do actual computations in $\mathbb{C}[\mathbf{x}]/I$ we need to represent each equivalence class $[f]$ by a well defined representative polynomial. The idea is to use multivariate polynomial division and represent $[f]$ by the remainder under division of f by I. Fortunately, for any polynomial ideal I, this can always be done and the tool for doing so is a Gröbner basis G for I [13]. The Gröbner basis for I is a canonical set of generators for I with the property that multivariate division by G, denoted $\overline{f}^G$, always yields a well defined remainder. By well defined we mean that for any $f_1, f_2 \in [f]$, we have $\overline{f_1}^G = \overline{f_2}^G$. The Gröbner basis is computed relative a monomial order and will be different for different monomial orders. As a consequence, the set of representatives for $\mathbb{C}[\mathbf{x}]/I$ will be different, whereas the space itself remains the same.

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

Example 1 The following two equations describe the intersection of a line and a circle

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

A Gröbner basis for this system is

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

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

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

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

2.3 A Note on Algebraic and Linear Bases

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

2.4 Floating Point Gröbner Basis Computations

The well established Buchberger's algorithm is guaranteed to compute a Gröbner basis in finite time and works well in exact arithmetic [13]. However, due to round-off errors, it easily becomes unstable in floating point arithmetic and except for very small examples it becomes practically useless. The reason for this is that in the Gröbner basis computation, leading terms are successively eliminated from the generators of I by pairwise subtraction of polynomials, much like Gaussian elimination. This leads to cancellation effects where it becomes impossible to tell whether a certain coefficient should be zero or not.

A technique introduced by Faugere *et al.* in [16] is to write the system of equations on matrix form

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

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

This technique takes us further, but for larger more demanding problems it is necessary to study a particular class of equations (*e.g.* relative orientation for omnidirectional cameras [18], fundamental matrix estimation with radial distortion [30], optimal three view triangulation [41], etc.) and use knowledge of what the structure of the Gröbner basis should be to design a special purpose Gröbner basis solver [37]. The typical work flow has been to study the particular problem at hand with the aid of a computer algebra system such as Maple or Macaulay2 and extract information such as the leading terms of the Gröbner basis, the monomials to use as a basis for $\mathbb{C}[\mathbf{x}]/I$, the number of solutions, etc. and work out a specific set of larger (gauss-jordan) elimination steps leading to the construction of a Gröbner basis for I.

Although, these techniques have permitted the solution to a large number of previously unsolved problems, many difficulties remain. Most notably, the above mentioned elimination steps (if at all doable) are often hopelessly ill conditioned [31, 41]. This is in part due to the fact that one has focused on computing a complete and correct Gröbner basis respecting a properly defined monomial order, which we show is not necessary.

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

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

Drawing on these observations, we investigate in detail the exact matrix operations needed to compute $\overline{f}$ and thus obtain a procedure which is both faster and more stable, enabling the solution of a larger class of problems than previously possible.

3 A New Approach to the Action Matrix Method

In this section we present a new way of looking at the action matrix method for polynomial equation solving. The advantage of the new formulation is that it yields more freedom in how the action matrix is computed. It should however be noted that a fundamental limitation still remains. As is the case with all other floating point methods, we do not present an algorithm which is guaranteed to work for all situations. To some extent, the method relies on heuristics and will only work if certain conditions can be fulfilled. As we show later, it does however expand the domain of application compared to previous methods. We start with a few examples that we will use to clarify these ideas.

Example 2 In the five point relative orientation problem for calibrated cameras, *cf*. [14, 29, 34, 38], the calculation of the essential matrix using 5 image point correspondences leads to 10 equations of degree 3 in 3 unknowns. These equations involve 20 monomials. By writing the equations as in (9) and using a total degree ordering on the monomials we get a coefficient matrix $\mathbf{C}$ of size 10×20 and a monomial vector $\mathbf{X} = [\mathbf{x}^{\alpha_1} \dots \mathbf{x}^{\alpha_n}]^T$ with 20 monomials. It turns out that the first 10×10 block $\mathbf{C}_1$ of $\mathbf{C} = [\mathbf{C}_1 \; \mathbf{C}_2]$ is in general of full rank and thus the first 10 monomials $\mathbf{X}_1$ can be expressed in terms of the last 10 monomials $\mathbf{X}_2$ as

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

This makes it possible to regard the monomials in $\mathbf{X}_2$ as representatives of a linear basis for $\mathbb{C}[\mathbf{x}]/I$. It is now straightforward to calculate the action matrix for T_x (the multiplication operator for multiplication by x) since monomials in the linear basis are either mapped to monomials in the basis or to monomials in $\mathbf{X}_1$, which can be expressed in terms of the basis using (10). □

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

Based on these observations we again emphasize two important facts: (i) We are not interested in finding the Gröbner basis or a basis for the remainder space relative to some Gröbner basis *per se*; it is enough to get a well defined mapping $\overline{f}$ and (ii) it suffices to calculate $\overline{f}$ on the elements $x \cdot \mathbf{x}^{\alpha_i}$, *i.e.* we do not need to be able to compute $\overline{f}$ for all $f \in \mathbb{C}[\mathbf{x}]$. These statements and their implications will be made more precise further on.

Example 3 Consider the equations

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

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

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

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

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

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

Drawing on the concepts illustrated in the above two examples we define a *solving basis*, similar to $\mathcal{B}$ in Example 3. The overall purpose of the definition is to rid our selves of the need of talking about a Gröbner basis and properly defined monomial orders, thus providing more room to derive numerically stable algorithms for computation of the action matrix and similar objects.

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

3.1 Solving Bases

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

Definition 1 Consider a finite subset $\mathcal{B} \subset \mathbb{C}[\mathbf{x}]$ of the set of polynomials over the complex numbers. If for each $b_i \in \mathcal{B}$ and some $p \in \mathbb{C}[x]$ we express pb_i as a linear combination of basis elements as

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

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

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

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

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

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

for $\mathbf{x} \in V$. □

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

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

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

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

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

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

If on the other hand the inverse image is not finite for some v_i so that we get a parameter family $\mathbf{x}$ corresponding to this eigenvector, then the correct solution can typically not be obtained without further use of the equations (1) as illustrated in the following example.

Example 4 Consider the polynomial system

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

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

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

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

At a first glance, Theorem 2 might not seem very useful since solving for x from $\mathbf{b}(x) = v_i$ potentially involves solving a new system of polynomial equations. However, it provides a tool for ruling out choices of $\mathcal{B}$ which are not practical to work with. Moreover, there is usually much freedom in the choice of $\mathcal{B}$. In general, $\mathcal{B}$ can be a set of polynomials. However, it is often practical to work with a basis of monomials. We get a particularly convenient situation if the coordinate variables x_i are included in $\mathcal{B}$ as seen in the following straight forward result:

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

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

This fact suggests that we should always try to include the coordinate variables in $\mathcal{B}$ to make for easy extraction of the final solutions. In practice, this is nearly always easy to do. And even if for some reason a few variables have to be left out, we can often still express each variable x_k as a linear combination of the basis elements $b_i(\mathbf{x})$ for $\mathbf{x} \in V$ by making use of the original equations. We thus again obtain a well defined inverse to the mapping in Definition 2.

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

Theorem 3 *If the ideal I generated by (1) is radical, then a complete solving basis $\mathcal{B}$* w.r.t *to p for (1) with the property that all eigenvalues of $\mathbf{m}_p$ are distinct spans $\mathbb{C}[x]/I$.*

Proof Since I is radical, $\mathbb{C}[x]/I$ is isomorphic to $\mathbb{C}[V]$, the ring of all polynomial functions on V. Moreover, since V is finite, all functions on V are polynomial and hence $\mathbb{C}[V]$ can be identified with $\mathbb{C}^r$, where $r = |V|$. Consider now the matrix $B = \left[\mathbf{b}(x_1), \dots, \mathbf{b}(x_r)\right]$. Each row of B corresponds to a (polynomial) function on V. Hence, if we can show that B has row rank r, then we are done. Due to Theorem 1, all $\mathbf{b}(x_i)$ are eigenvectors of $\mathbf{m}_p$ corresponding to eigenvalues $p(x_i)$. By the assumption of distinct eigenvalues we have $p(x_i) \neq p(x_j)$ whenever $\mathbf{b}(x_i) \neq \mathbf{b}(x_j)$. Since $\mathcal{B}$ is a complete solving basis we have $\mathbf{b}(x_i) \neq \mathbf{b}(x_j)$ whenever $x_i \neq x_j$. This means that the r points in V correspond to distinct eigenvalues and hence, since eigenvectors corresponding to different eigenvalues are linearly independent, B has column rank r. For any matrix row rank equals column rank and we are done. □

The above theorem provides a correspondence between solving bases and linear bases for $\mathbb{C}[\mathbf{x}]/I$ and in principle states that under some extra restrictions, a solving basis is simply a certain choice of linear basis for $\mathbb{C}[\mathbf{x}]/I$ and then the matrix $\mathbf{m}_p$ turns into the transposed action matrix.

However, relaxing these restrictions we get something which is not necessarily a basis for $\mathbb{C}[\mathbf{x}]/I$ in the usual sense, but still serves our needs in terms of equation solving. More specifically, using the concept of a solving basis provides two distinctive advantages.

(i) For a radical polynomial system with r zeros, $\mathbb{C}[\mathbf{x}]/I$ is r-dimensional, so a basis for $\mathbb{C}[\mathbf{x}]/I$ contains r elements. This need not be the case for a solving basis, which could well contain more than r elements, but due to Theorem 2 still provides the right solutions. This fact is exploited in Section 4.

(ii) Typically, the arithmetics in $\mathbb{C}[\mathbf{x}]/I$ has been computed using a Gröbner basis for I, which directly provides a monomial basis for $\mathbb{C}[\mathbf{x}]/I$ in form of the set of monomials which are not divisible by the Gröbner basis. In this work we move focus from Gröbner basis computation to the actual goal of expressing the products pb_i in terms of a set of linear basis elements and thus no longer need to adhere to the overly strict ordering rules imposed by a particular monomial order. This freedom is exploited in Sections 5.1 and 5.2.

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

3.2 Solving Basis Computations using Numerical Linear Algebra

We now describe the most straight forward technique for deciding whether a candidate basis $\mathcal{B}$ w.r.t. one of the variables

x_k, can be used as a solving basis and simultaneously calculate the action of T_{x_k} on the elements of $\mathcal{B}$.

We start by generating more equations by multiplying the original set of equations by a hand crafted (problem dependent) set of monomials. This yields additional equations, which are equivalent in terms of solutions, but hopefully linearly independent from the original ones. In Example 4, we could multiply by *e.g.* $\{x, y, 1\}$, yielding $xy^2 - 2x, x^3 - x, y^3 - 2y, x^2y - y, y^2 - 2, x^2 - 1$. The general question of which and how many additional equations to generate is a tough one and there exist no watertight answers. This has to do with the fact that computing a Gröbner basis is an NP hard problem in general. The rule of thumb is to start out with a small set of equations and then sequentially adding more equations until the computations go through. However, some additional insight into the difficulty of the problem can be obtained by studying it symbolically using a computer algebra system, *e.g.* Macaulay2 [3].

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

$$\mathbf{C} = \begin{bmatrix}\mathbf{C}_{\mathcal{E}}\ \mathbf{C}_{R}\ \mathbf{C}_{B}\end{bmatrix}, \tag{16}$$

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

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

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

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

or equivalently

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

which means that the $\mathcal{R}$-monomials can be expressed as a linear combination of the basis monomials. Thus $\mathcal{B}$ is a solving basis and the matrix $\mathbf{m}_{x_k}$ can easily be constructed as in (12). In other words, given an enlarged set of equations and a choice of linear basis $\mathcal{B}$, the full rank of $\mathbf{C}_{\mathcal{R}2}$ is sufficient to solve (1) via eigendecomposition of $\mathbf{m}_{x_k}$. The above method is summarized in Algorithm 1 and given the results of Section 3.1 we now have the following:

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

Proof The postcondition that $\mathbf{C}_{\mathcal{R}2}$ is of full rank ensures that $\mathcal{B}$ is a solving basis and together with the preconditions, Theorem 2 and Corollary 1 then guarantees the statement. □

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

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

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

with $\mathcal{M} = \{x^2, xy, y^2, x, y, 1\}$ and since we chose $\mathcal{B} = \{y, 1\}$, we get $\mathcal{R} = \{xy, x\}$ and $\mathcal{E} = \{x^2, y^2\}$. After Step 11 and 12 of Algorithm 1 we have $\mathbf{C}_{\mathcal{R}2} = \left[\begin{smallmatrix}2 & 0 \\ 0 & 1\end{smallmatrix}\right]$ and $\mathbf{C}_{\mathcal{B}2} = \left[\begin{smallmatrix}0 & -1 \\ -1 & 0\end{smallmatrix}\right]$ and inserting into (19) we obtain

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

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

A typical problem that might occur is that some eigenvalues of $\mathbf{m}_{x_k}$ are equal, which means that two or more zeros have equal x_k-coordinate. Then the corresponding eigenvectors can not be uniquely determined. This problem can be resolved by computing $\mathbf{m}_{x_k}$ 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

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

```
Require: List of equations $F = \{f_1, \ldots, f_m\}$, set of basis monomials $\mathcal{B}$ containing the coordinate variables $x_1, \ldots x_s$, $m$ lists of monomials $\{L_i\}_{i=1}^m$.
Ensure: $\mathbf{C}_{\mathcal{R}2}$ is of full rank, eigenvalues of $\mathbf{m}_{x_k}$ are distinct.
 1: $F_{\text{ext}} \leftarrow F$
 2: for all $f_i \in F$ do
 3:     for all $\mathbf{x}^{\boldsymbol{\alpha}_j} \in L_i$ do
 4:         $F_{\text{ext}} \leftarrow F_{\text{ext}} \bigcup \{\mathbf{x}^{\boldsymbol{\alpha}_j} \cdot f_i\}$
 5:     end for
 6: end for
 7: Construct coefficient matrix $\mathbf{C}$ from $F_{\text{ext}}$.
 8: $\mathcal{M} \leftarrow$ The set of all monomials occurring in $F_{\text{ext}}$.
 9: $\mathcal{R} \leftarrow x_k \cdot \mathcal{B} \setminus \mathcal{B}$
10: $\mathcal{E} \leftarrow \mathcal{M} \setminus (\mathcal{R} \bigcup \mathcal{B})$
11: Reorder and partition $\mathbf{C}$: $\tilde{\mathbf{C}} = [\mathbf{C}_{\mathcal{E}}\ \mathbf{C}_R\ \mathbf{C}_B]$.
12: LU-factorize to obtain $\mathbf{C}_{\mathcal{R}2}$ and $\mathbf{C}_{\mathcal{B}2}$ as in (17).
13: Use (19) to express $x_k \cdot \mathbf{x}^{\boldsymbol{\alpha}_i}$ in terms of $\mathcal{B}$ and store the coefficients in $\mathbf{m}_{x_k}$.
14: Compute eigenvectors of $\mathbf{m}_{x_k}$ and read off the tentative set of solutions.
15: Evaluate in $F$ to filter out possible false zeros.
```

similar to (19). The reason for this is that without proper care, $\mathbf{C}_{\mathcal{R}2}$ tends to become very ill conditioned (condition numbers of 10^{10} or higher are not uncommon). This was also the reason that extremely slow emulated 128 bit numerics had to be used in [41] to get a working algorithm.

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

3.3 Constructing the Action Matrix

Given a solving basis $\mathcal{B}$ w.r.t. x_k and the corresponding vector $\mathbf{X}_{\mathcal{B}}$ for the polynomial system we wish to solve, it is easy to construct the corresponding action matrix. In short, this section fleshes out Step 13 of Algorithm 1. After performing Algorithm 1, the solving basis property of $\mathcal{B}$ together with (19) guarantees $x_k\mathcal{B} \setminus \mathcal{B} = \mathcal{R}$ and the following:

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

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

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

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

4 Using Redundant Solving Bases - The Truncation Method

As mentioned in Section 3, the sub matrix $\mathbf{C}_{\mathcal{R}2}$ which appears in Equation 17 is a large cause of numerical problems in the equation solving process. A typical situation with an ill conditioned or rank deficient $\mathbf{C}_{\mathcal{R}2}$ is that there are a few problematic monomials where the corresponding columns in $\mathbf{C}$ are responsible for the deteriorated conditioning of $\mathbf{C}_{\mathcal{R}2}$. A straightforward way to improve the situation is to simply include the problematic monomials in $\mathcal{B}$, thus avoiding the need to express these in terms of the other monomials. In practice this means that some columns of $\mathbf{C}_{\mathcal{R}}$ are moved into $\mathbf{C}_{\mathcal{B}}$. This technique is supported by Theorem 2, which guarantees that we will find the original set of solutions among the eigenvalues/eigenvectors of the larger $\mathbf{m}_p$ found using this redundant basis. The price we have to pay is performing an eigenvalue decomposition on a larger matrix.

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

An example of how the redundant solving basis technique can be used is provided by the problem of L_2-optimal triangulation from three views [41]. The optimum is found among the up to 47 stationary points, which are zeros of a polynomial system in three variables. In this example an enlarged set of 255 equations in 209 monomials were used to get a Gröbner basis. Since the solution dimension r is 47 in this case, the 47 lowest order monomials were used as a basis for $\mathbb{C}[\mathbf{x}]/I$ in [41], yielding a numerically difficult situation. In fact, as will be shown in more detail in the experiments section, this problem can be solved by simply including more elements in $\mathcal{B}$. In this example, the complete permissible set contains 154 monomials. By including all of these in $\mathcal{B}$ leaving 55 monomials to be expressed in terms of $\mathcal{B}$, we get a much smaller and in this case better conditioned elimination step. As mentioned above, this leads to a larger eigenvalue decomposition, but all true solutions can still be found among the larger set of eigenvalues/eigenvectors. This is illustrated in Figure 1, where the set of eigenvalues computed from $\mathbf{m}_{x_k}$ for one instance are plotted in the complex plane together with the actual solutions of the polynomial system.

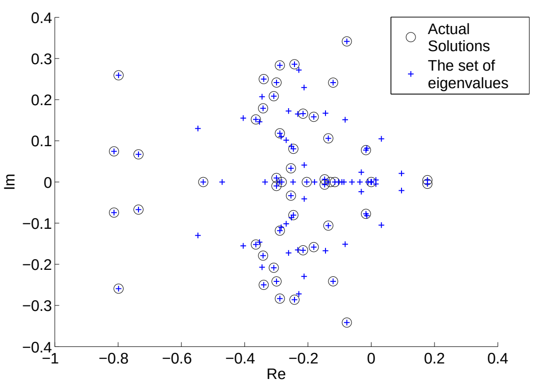

Fig. 1 Eigenvalues of the action matrix using the redundant basis method and actual solutions to the polynomials system plotted in the complex number plane. The former are a strict superset of the latter.

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 $\mathbf{C}_{\mathcal{R}2}$ and form $[\mathbf{C}_{\mathcal{E}}\ \mathbf{C}_{\mathcal{R}}\ \mathbf{C}_{\mathcal{P}}]$. Any selection of basis monomials $\mathcal{B} \subset \mathcal{P}$ will then correspond to a matrix $\mathbf{C}_{\mathcal{B}}$ consisting of a subset of the columns of $\mathbf{C}_{\mathcal{P}}$.

By performing Gaussian elimination we again obtain (17), but with $\mathcal{B}$ replaced by $\mathcal{P}$, letting us get rid of the $\mathcal{E}$-monomials by discarding the top rows. Furthermore, the $\mathcal{R}$-monomials will all have to be expressed in terms of the $\mathcal{P}$-monomials so we continue the elimination putting $\mathbf{C}_{\mathcal{R}2}$ on triangular form, obtaining

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

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

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

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

and then proceed using standard elimination. The goal must thus be to make this choice so as to minimize the condition number $\kappa(\begin{bmatrix} c_{\pi(1)} & \dots & c_{\pi(p-r)} \end{bmatrix})$ of the first $p-r$ columns of the permuted matrix. In its generality, this is a difficult combinatorial optimization problem. However, the task can be approximately solved in an attractive way by QR factorization with column pivoting [19]. With this algorithm, $\mathbf{C}_{\mathcal{P}2}$ is factorized as

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

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

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

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

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

Finally

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

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

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

Furthermore, QR factorization with column pivoting is a widely used and well studied algorithm and there exist free, highly optimized implementations [2], making this an accessible approach.

Standard QR factorization successively eliminates elements below the main diagonal by multiplying from the left with a sequence of orthogonal matrices (usually Householder transformations). For matrices with more columns than rows (under-determined systems) this algorithm can produce a rank-deficient $\mathbf{U}$ which would then cause the computations in this section to break down. QR with column pivoting solves this problem by, at iteration k, moving the column with greatest 2-norm on the last $m-k+1$ elements to position k and then eliminating the last $m-k$ elements of this column by multiplication with an orthogonal matrix Q_k. See [19] for more about QR factorization and column pivoting.

5.2 The SVD Method

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

As in Section 5.1 we start out by selecting an as large as possible (redundant) solving basis and perform preliminary matrix operations forming (23), where the aim is now to construct a linearly independent basis from $\mathcal{P}$. We now do this by performing an SVD on $\mathbf{C}_{\mathcal{P}2}$, writing

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

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

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

$$\begin{bmatrix} \mathbf{U}_{\mathcal{R}} & \mathbf{C}_{\mathcal{P}1}\mathbf{V} \\ \mathbf{0} & \Sigma \end{bmatrix} \begin{bmatrix} \mathbf{X}_{\mathcal{R}} \\ \mathbf{V}^T\mathbf{X}_{\mathcal{P}} \end{bmatrix} = 0. \tag{30}$$

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

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

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

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

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

Computing the action matrix is complicated slightly by the fact that we are now working with a polynomial basis rather than a monomial one. To deal with this situation we introduce some new notation. To each element e_k of $\tilde{\mathcal{P}} = \mathcal{P}' \bigcup \mathcal{B}$ we assign a vector $v_k = \left[0 \ldots 1 \ldots 0\right]^T \in \mathbb{R}^{|\tilde{\mathcal{P}}|}$, with a one at position k. Similarly, we introduce vectors $u_k \in \mathbb{R}^{|\mathcal{M}|}, w_k \in \mathbb{R}^{|\mathcal{B}|}$ representing elements of $\mathcal{M}$ and $\mathcal{B}$ respectively. Further we define the linear mapping $R : \mathrm{span}(\mathcal{M}) \to \mathrm{span}(\tilde{\mathcal{B}})$, which using (32) associates an element of $\mathrm{span}(\mathcal{M})$ with an element in $\mathrm{span}(\tilde{\mathcal{B}})$. We now represent R by a $|\mathcal{B}| \times |\mathcal{M}|$ matrix

$$\mathbf{R} = \left[-\mathbf{C}_{\mathcal{B}}^T\mathbf{U}_{\mathcal{R}}^{-T}\ \mathbf{0}\ \mathbf{I}\right], \tag{33}$$

acting on the space spanned by the vectors u_k.

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

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

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

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

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

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

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

where $\mathbf{L} = \left[\begin{smallmatrix} \mathbf{0} \\ \mathbf{I} \end{smallmatrix}\right]$ simply interprets the $w_k \in \mathbb{R}^{|\mathcal{B}|}$ vectors as $\mathbb{R}^{|\tilde{\mathcal{P}}|}$-vectors. The matrix $\tilde{\mathbf{m}}_p$ derived here is the transpose of the corresponding matrix in Section 3.1.

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

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

5.3 Basis Selection and Adaptive Truncation

We have so far seen three techniques for dealing with the case when the sub matrix $\mathbf{C}_{\mathcal{P}2}$ is ill conditioned. By the method in Section 4 we avoid operating on $\mathbf{C}_{\mathcal{P}2}$ altogether. Using, the QR and SVD methods we perform elimination, but in a numerically much more stable manner. One might now ask whether it is possible to combine these methods. Indeed it turns out that we can combine either the QR or the SVD method with a redundant solving basis to get an adaptive truncation criterion yielding even better stability in some cases. The way to do this is to choose a criterion for early stopping in the factorization algorithms. The techniques in this section are related to truncation schemes for rank-deficient linear least squares problems, *cf.* [28].

A neat feature of QR factorization with column pivoting is that it provides a way of numerically estimating the conditioning of $\mathbf{C}_{\mathcal{P}2}$ simultaneously with elimination. By design, the QR factorization algorithm produces an upper triangular matrix $\mathbf{U}$ with diagonal elements u_{ii} of decreasing absolute value. The factorization proceeds column wise, producing one $|u_{ii}|$ at a time. If $\operatorname{rank}(\mathbf{U}) = r$, then $|u_{rr}| > 0$ and $u_{r+1,r+1} = \cdots = u_{nn} = 0$. However, in floating point arithmetic, the transition from finite $|u_{ii}|$ to zero is typically gradual passing through extremely small values and the rank is consequently hard to determine. For robustness it might therefore be a good idea to abort the factorization process early. We do this by setting a threshold τ for the ratio $|\frac{u_{11}}{u_{ii}}|$ and abort the factorization once the value exceeds this threshold. A value of $\tau \approx 10^8$ has been found to yield good results[1]. Note that this produces an equivalent result to carrying out the full QR factorization and then simply discarding the last rows of $\mathbf{U}$. This is practical since off-the-shelf packages as LAPACK [2] only provide full QR factorization, even though some computational effort could be spared by modifying the algorithm so as not to carry out the last steps.

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

In the context of the SVD we get a similar criterion by looking at the singular values instead and set a threshold for $\frac{\sigma_1}{\sigma_i}$, which for $i = \operatorname{rank}(\mathbf{C}_{\mathcal{P}2})$ is exactly the condition number of $\mathbf{C}_{\mathcal{P}2}$.

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

6 Using Eigenvalues Instead of Eigenvectors

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

Fortunately, we can make use of our knowledge of the eigenvectors to devise a scheme for quickly finding the eigenvalues of any action matrix on $\mathbb{C}[\mathbf{x}]/I$. From Section 2 we know that the right eigenvectors of an action matrix is the vector of basis elements of $\mathbb{C}[\mathbf{x}]/I$ evaluated at the zeros of I. This holds for *any* action matrix and hence all action matrices have the same set of eigenvectors. Consider now a problem involving the two variables x_i and x_j. If we have constructed $\mathbf{m}_{x_i}$, the construction of $\mathbf{m}_{x_j}$ requires almost no extra time. Now perform an eigenvalue decomposition $\mathbf{m}_{x_i} = \mathbf{V}\mathbf{D}_{x_i}\mathbf{V}^{-1}$. Since $\mathbf{V}$ is the set of eigenvectors for $\mathbf{m}_{x_j}$ as well, we get the eigenvalues of $\mathbf{m}_{x_j}$ by straightforward matrix multiplication and then element wise division from

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

This means that with very little extra computational effort over a single eigenvalue decomposition we can obtain the eigenvalues of all action matrices we need. The observations in this section also suggest a slightly easier way of filtering out false solutions obtained using the method in Section 4. If the coordinate variables x_i are present in the basis, they correspond to both eigenvalues *and* elements in the eigenvectors. Any discrepancy here implies a false solution which can immediately be discarded.

7 Experiments

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

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

7.1 Optimal Three View Triangulation

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

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

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

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

The synthetic data used in the validation was generated with three randomly placed cameras at a distance around 1000 from the origin and a focal length of around 1000. The unknown world point was randomly placed in a cube with side length 1000 centered at the origin. The methods have been compared on 100,000 test cases and the code has been made available for download[2].

7.1.1 Numerical Experiments

The first experiment investigates what improvement can be achieved by simply avoiding the problematic matrix elimination using the techniques of Section 4. For this purpose we choose the complete set of permissible monomials $\mathcal{P}$ as a redundant basis and perform the steps of Algorithm 1. In this case we thus get a redundant basis of 154 elements and a 154×154 multiplication matrix to perform eigenvalue decomposition on. In both cases the eigenvectors are used to find the solutions. The results of this experiment are shown in Figure 2. As can be seen, this relatively straightforward technique already yields a large enough improvement in numerical stability to give the solver practical value.

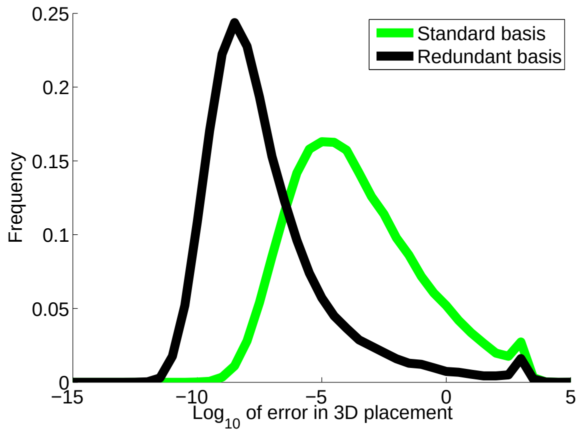

Fig. 2 Histogram of errors over 100,000 points. The improvement in stability using the redundant basis renders the algorithm feasible in standard IEEE double precision.

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. It is unclear what causes these errors. However, by doing some extra work using the QR method of Section 5.1 to select a true basis as a subset of $\mathcal{P}$, we improve stability further in general and in particular completely resolve the issue with large errors, *cf*. Figure 3.

In Figure 4, the performance of the QR method is compared to the slightly more stable SVD method which selects

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

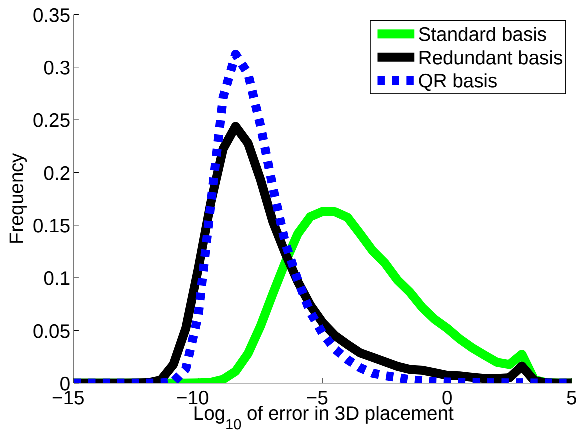

Fig. 3 Histogram of errors for the standard, redundant basis and QR methods. The QR method improves stability in general and in particular completely removes the small set of large errors present in both the standard and redundant basis methods.

a polynomial basis for $\mathbb{C}[\mathbf{x}]/I$ from the monomials in $\mathcal{P}$. In this case, errors are typically a factor ~ 5 smaller for the SVD method compared to the QR method.

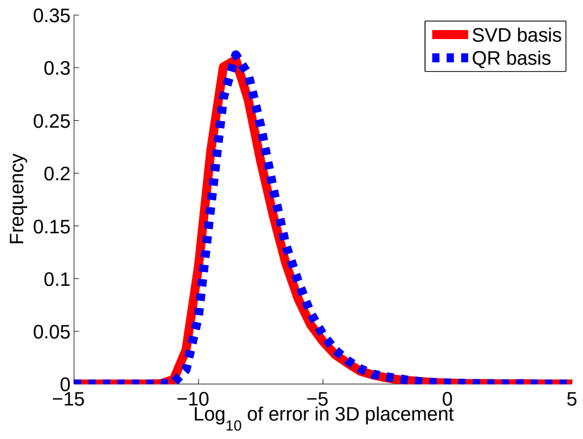

Fig. 4 Comparison between the SVD and QR methods. The SVD method improves somewhat over the QR method at the cost of the computationally more demanding SVD factorization.

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

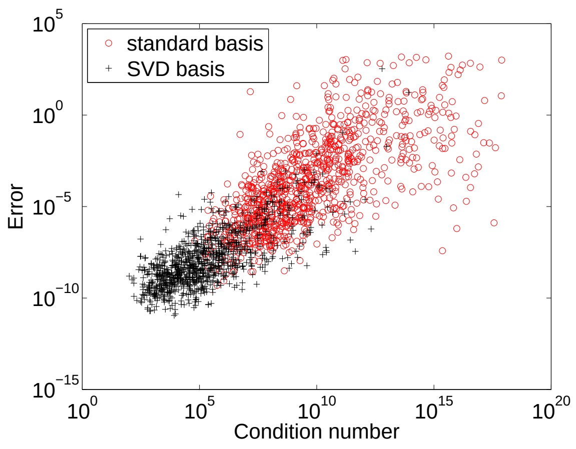

Fig. 5 Error versus condition number for the part of the matrix which is inverted in the solution procedure.

As mentioned in Section 6, it might be beneficial to use the eigenvalues instead of eigenvectors to extract solutions.

When solving this problem using eigenvalues there are two extra eigenvalue problems of size 50×50 that have to be solved. The impact of the switch from eigenvectors to eigenvalues can be seen in Figure 6. For this example we gain some stability at the cost of having to perform three eigenvalue decompositions (one for each coordinate) instead of only one. Moreover, we need to sort the eigenvalues using the eigenvectors to put together the correct triplets.

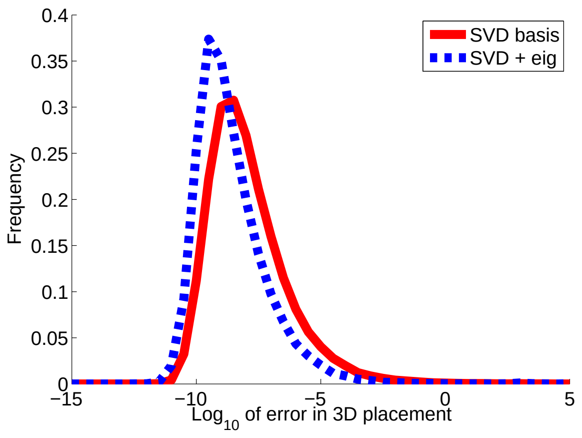

Fig. 6 Error histograms showing the difference in precision between the eigenvalue and eigenvector methods.

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.

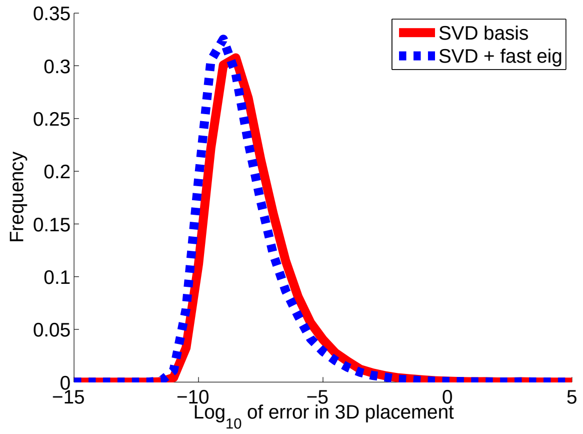

Fig. 7 This graph shows the increase in performance when the fast eigenvalue method is used instead of the eigenvector method.

Finally we study the combination of basis selection and early stopping yielding a redundant Gröbner basis for the three view triangulation problem. The basis size was determined adaptively as described in Section 5.3 with a threshold $\tau = 10^8$. Table 1 shows the distribution of basis sizes obtained when this method was used. Since the basis is chosen minimal in 94% of the cases for the SVD method and 95% for the QR method the time consumption is almost identical to the original basis selection methods, but as can be seen in Table 2 the number of large errors are reduced for the QR method. This is probably due to the fact that truncation is carried out only when the matrices are close to being singular. This effect is not present for the SVD method.

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

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

To conclude the numerical experiments on three view triangulation two tables with detailed error statistics are given. The acronyms *STD, QR, SVD* and *TRUNC* respectively denote the standard method, QR method, SVD method and redundant basis method. The suffixes *eig, fast* and *var* respectively denote the eigenvalue method, the fast eigenvalue method (Section 6) and the use of a variable size basis (Section 5.3). Table 2 shows how many times the error gets larger the some given levels for several solvers. As can be seen, the QR method with adaptive basis size is the best method for reducing the largest errors but the SVD method with use of the eigenvalues is the best in general. Table 3 shows the median and the 95:th percentile errors for the same methods as in the previous table. Notable in here is that the 95:th percentile is improved with as much as factor 10^7 and the median with a factor 10^5. The SVD method with eigenvalues is shown to be the best but the QR method gives almost as good results.

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

Table 2 Number of errors out of 100,000 experiments larger than certain levels. The QR method with adaptive basis size yields the fewest number of large errors but the SVD method with eigenvalues is the best in general.

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

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

The time measurements were done on an Intel Core 2 2.13 GHz machine with 2 GB memory. Each algorithm was executed with 1000 different coefficient matrices constructed from the same type of scene setups as previously. The same set of coefficient matrices was used for each method. The result is given in Table 4. Our results show that the QR method with adaptive truncation is approximately four times faster than the SVD method but 40% slower than the standard method. The reason that the redundant basis method is more than twice as slow as the QR method is the larger eigenvalue decomposition, which dominates the computation time.

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

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

7.1.3 Triangulation of Real Data

Finally, the algorithm is evaluated under real world conditions. The Oxford dinosaur [15] is a familiar image sequence of a toy dinosaur shot on a turn table. The image sequence consists of 36 images and 4983 point tracks. For each point visible in three or more views we select the first, middle and last view and triangulate using these. This yields a total of 2683 point triplets to triangulate. The image sequence contains some erroneous tracks that we deal with by removing any points reprojected with an error greater than two pixels in any frame. The whole sequence was processed in approximately 45 seconds in a Matlab implementation on an Intel Core 2 2.13 GHz CPU with 2 GB of memory by the QR method with variable basis size. The resulting point cloud is shown in Figure 8.

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

7.2 Localization with Hybrid Features

In this experiment, we study the problem of finding the pose of a calibrated camera with unknown focal length. One minimal setup for this problem is three point-correspondences with known world points and one correspondence to a world line. The last feature is equivalent to having a point correspondence with another known calibrated camera. These types of mixed features are called hybrid features and were introduced in [25], where the authors propose a parameterization of the problem but no solution was given apart from showing that the problem has 36 solutions.

The parameterization in [25] gives four equations in four unknowns. The unknowns are three quaternion parameters and the focal length. The equation derived from the line correspondence is of degree 6 and those obtained from the 3D points are of degree 3. The coefficient matrix $\mathbf{C}$ is then constructed by expanding all equations up to degree 10. This means that the equation derived from the line is multiplied with all monomials up to degree 4, but no single variable in the monomials is of higher degree than 2. In the same manner the point correspondence equations are multiplied with monomials up to degree 7 but no single variable of degree more than 5. The described expansion gives 980 equations in 873 monomials.

The next step is to reorder the monomials as in (16). In this problem $\mathbf{C}_{\mathcal{P}}$ corresponds to all monomials up to degree 4 except f^4 where f is the focal length, which gives 69 columns in $\mathbf{C}_{\mathcal{P}}$. The part $\mathbf{C}_{\mathcal{R}}$ corresponds to the 5:th degree monomials that appear when the monomials in $\mathcal{P}$ are multiplied with the first of the unknown quaternion parameters.

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

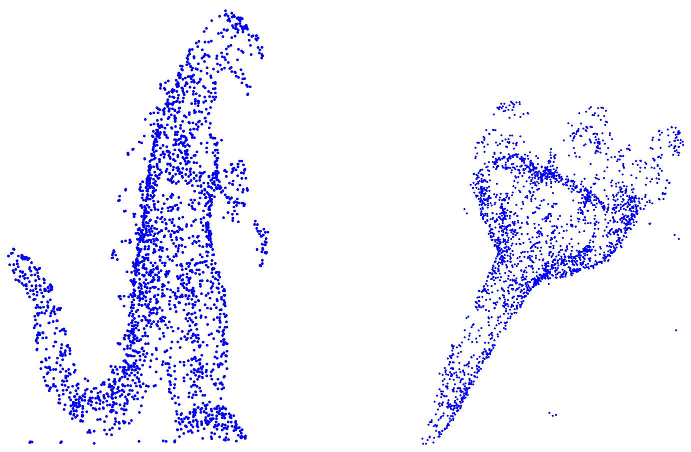

Fig. 8 The Oxford dinosaur reconstructed from 2683 point triplets using the QR method with variable basis size. The reconstruction was completed in approximately 45 seconds by a Matlab implementation on an Intel Core 2 2.13 GHz CPU with 2 GB of memory.

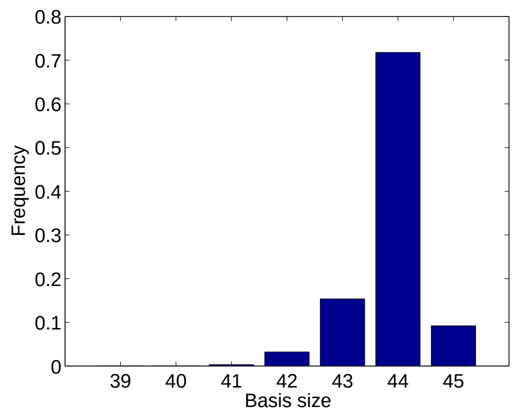

Fig. 9 The basis size for localization with hybrid features. The number of solutions are 36 and since we always add three dimensions to the truncated ideal the minimal possible basis size is 39.

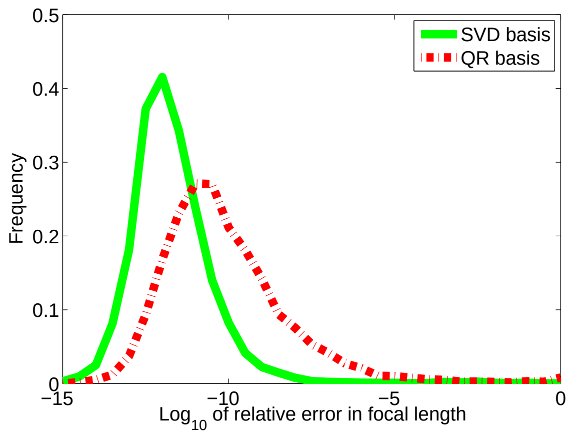

Fig. 10 The error for localization with hybrid features. The standard method is omitted since we did not manage to construct a standard solver due to numerical problems.

emphasize that we were not able to construct a solver with the standard method and hence no error distribution for that method is available.

7.3 Relative Pose with Unknown Focal Length

Relative pose for calibrated cameras is a well known problem and the standard minimal case for this is five points in two views [38]. There are in general ten solutions to this problem. For the same problem but with unknown focal length, the corresponding minimal case is six points in two views, which was solved by Stewénius *et al.* using Gröbner basis techniques [39].

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

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

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

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

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

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

The validation data was generated with two cameras of equal focal length of around 1000 placed at a distance of around 1000 from the origin. The six points were randomly placed in a cube with side length 1000 centered at the origin. The standard, SVD, and QR methods have been compared on 100,000 test cases and the errors in focal length are shown in Figure 11. In this case the QR method yields slightly better results than the SVD method. This is probably due to loss in numerical precision when the solution is transformed back to the original basis.

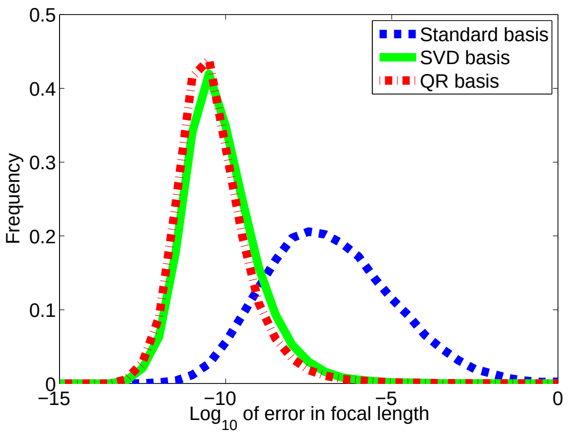

Fig. 11 The error in focal length for relative pose with two semi calibrated cameras with unknown but common focal length.

7.4 Relative Pose for Generalized Camera

Generalized cameras provide a generalization of the standard pin-hole camera in the sense that there is no common focal point through which all image rays pass, *cf*. [36]. Instead the camera captures arbitrary image rays or lines. Solving for the relative motion of a generalized camera can be done using six point correspondences in two views. This is a minimal case which was solved in [40] with Gröbner basis techniques. The problem equations can be set up using quaternions to parameterize the rotation, Plücker representation of the lines and a generalized epipolar constraint which captures the relation between the lines. After some manipulations one obtains a set of sixth degree equations in the three quaternion parameters v_1, v_2 and v_3. For details, see [40]. The problem has 64 solutions in general.

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

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

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

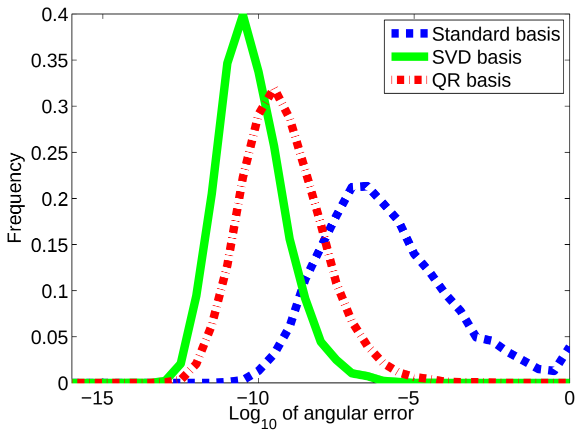

Fig. 12 The angular error in degrees for relative pose with generalized cameras.

8 Discussion

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

The technique for solving polynomial equations that we use in this paper can be summarized as follows. The original equations are first expanded by multiplying the polynomials with a set of monomials. The resulting equations is expressed as a product of a coefficient matrix $\mathbf{C}$ and a monomial vector $\mathbf{X}$. Here we have some freedom in choosing which monomials to multiply with. We then try to find a solving basis $\mathcal{B}$ for the problem. For a given candidate basis $\mathcal{B}$ we have shown how to determine if $\mathcal{B}$ constitutes a solving basis. If so then numerical linear algebra is used to construct the action matrix and get a fast and numerically stable solution to the problem at hand. However, we do not know (i) what monomials we should multiply the original equations with and (ii) what solving basis $\mathcal{B}$ should be used to get the simplest and most numerically stable solutions. Are 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.

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. E. Anderson and et al. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
3. D. Bayer and M. Stillman. Macaulay. www.math.columbia.edu/~bayer/Macaulay/, 1994. An open source computer algebra software.
4. M. Brown, R. Hartley, and D. Nister. Minimal solutions for panoramic stitching. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR07)*, Minneapolis, June 2007.
5. M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.
6. M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. In *Asian Conference on Computer Vision*, 2007.
7. M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of gröbner basis polynomial equation solvers. In *Proc. 11th Int. Conf. on Computer Vision*, Rio de Janeiro, Brazil, 2007.
8. M. Byröd, K. Josephson, and K. Åström. A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations. In *The 10th European Conference on Computer Vision*, 2008.
9. M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. In *Proc. Conf. Computer Vision and Pattern Recognition, Anchorage, USA*, 2008.
10. E. Cattani, D. A. Cox, G. Chèze, A. Dickenstein, M. Elkadi, I. Z. Emiris, A. Galligo, A. Kehrein, M. Kreuzer, and B. Mourrain. *Solving Polynomial Equations: Foundations, Algorithms, and Applications (Algorithms and Computation in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
11. M. Chasles. Question 296. *Nouv. Ann. Math.*, 14(50), 1855.
12. D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer Verlag, 1998.
13. D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 2007.
14. M. Demazure. Sur deux problemes de reconstruction. Technical Report 882, INRIA, 1988.
15. Visual geometry group, university of oxford. http://www.robots.ox.ac.uk/∼vgg.
16. J.-C. Faugère. A new efficient algorithm for computing gröbner bases (f_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
17. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–95, 1981.
18. C. Geyer and H. Stewénius. A nine-point algorithm for estimating para-catadioptric fundamental matrices. In *Proc. Conf. Computer Vision and Pattern Recognition, Minneapolis, USA*, June 2007.
19. G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.
20. D. Grayson and M. Stillman. Macaulay 2. Available at http://www.math.uiuc.edu/Macaulay2/, 1993-2002. An open source computer algebra software.
21. R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Proc. Conf. Computer Vision and Pattern Recognition, Washington DC*, pages 504–509, Washington DC, USA, 2004.
22. R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, 1997.
23. R. Holt, R. Huang, and A. Netravali. Algebraic methods for image processing and computer vision. *IEEE Transactions on Image Processing*, 5:976–986, 1996.
24. D. G. Hook and P. R. McAree. Using sturm sequences to bracket real roots of polynomial equations. *Graphics gems*, pages 416–422, 1990.
25. K. Josephson, M. Byröd, F. Kahl, and K. Åström. Image-based localization using hybrid feature correspondences. In *The second international ISPRS workshop BenCOS 2007, Towards Benchmarking Automated Calibration, Orientation, and Surface Reconstruction from Images*, 2007.
26. F. Kahl. Multiple view geometry and the l_∞-norm. In *ICCV*, pages 1002–1009, 2005.
27. F. Kahl and D. Henrion. Globally optimal estimates for geometric reconstruction problems. In *Proc. 10th Int. Conf. on Computer Vision, Bejing, China*, pages 978–985, 2005.
28. I. Karasalo. A criterion for truncation of the QR-decomposition algorithm for the singular linear least squares problem. *BIT Numerical Mathematics*, 14(2):156–166, June 1974.
29. E. Kruppa. Zur Ermittlung eines Objektes aus Zwei Perspektiven mit innerer Orientierung. *Sitz-Ber. Akad. Wiss., Wien, math. naturw. Kl. Abt*, IIa(122):1939–1948, 1913.
30. Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *CVPR*, 2007.
31. Z. Kukelova and T. Pajdla. Two minimal problems for cameras with radial distortion. In *Proceedings of The Seventh Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*, 2007.
32. D. Lazard. Resolution des systemes d'equations algebriques. *Theor. Comput. Sci.*, 15:77–110, 1981.
33. H. Li. A simple solution to the two-view focal-length algorithm. In *Proc. 9th European Conf. on Computer Vision, Graz, Austria*, 2006.
34. D. Nistér. An efficient solution to the five-point relative pose problem. In *Proc. Conf. Computer Vision and Pattern Recognition*, volume 2, pages 195–202. IEEE Computer Society Press, 2003.
35. J. Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *Photogrammetric Record*, 15(88):589–599, Oct. 1996.
36. R. Pless. Using many cameras as one. In *Proc. Conf. Computer Vision and Pattern Recognition, Madison, USA*, 2003.
37. H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, Apr. 2005.
38. H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60:284–294, June 2006.
39. H. Stewénius, F. Kahl, D. Nistér, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *Proc. Conf. Computer Vision and Pattern Recognition, San Diego, USA*, 2005.
40. H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision*, Beijing China, Oct. 2005.
41. H. Stewénius, F. Schaffalitzky, and D. Nistér. How hard is three-view triangulation really? In *Proc. Int. Conf. on Computer Vision*, pages 686–693, Beijing, China, 2005.
42. E. H. Thompson. A rational algebraic formulation of the problem of relative orientation. *Photogrammetric Record*, 14(3):152–159, 1959.
43. P. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15(8):591–605, 1997.
44. P. Torr and A. Zisserman. Robust computation and parametrization of multiple view relations. In *Proc. 6th Int. Conf. on Computer Vision, Mumbai, India*, pages 727–732, 1998.
45. J. Verschelde. Phcpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software*, 25(2):251–276, 1999.