Making Minimal Solvers Fast

Martin Bujnak
Bzovicka 24, 85107, Bratislava, Slovakia
martin@solvergenerator.com

Zuzana Kukelova, Tomas Pajdla
Czech Technical University, Faculty of Electrical Engineering,
Karlovo namesti 13, Prague, Czech Republic
{kukelova,pajdla}@cmp.felk.cvut.cz

Abstract

In this paper we propose methods for speeding up minimal solvers based on Gröbner bases and action matrix eigenvalue computations. Almost all existing Gröbner basis solvers spend most time in the eigenvalue computation. We present two methods which speed up this phase of Gröbner basis solvers: (1) a method based on a modified FGLM algorithm for transforming Gröbner bases which results in a single-variable polynomial followed by direct calculation of its roots using Sturm-sequences and, for larger problems, (2) fast calculation of the characteristic polynomial of an action matrix, again solved using Sturm-sequences. We enhanced the FGLM method by replacing time consuming polynomial division performed in standard FGLM algorithm with efficient matrix-vector multiplication and we show how this method is related to the characteristic polynomial method. Our approaches allow computing roots only in some feasible interval and in desired precision. Proposed methods can significantly speedup many existing solvers. We demonstrate them on three important minimal computer vision problems. [1]

1. Introduction

Many important computer vision problems were in the last few years solved using the Gröbner basis method [1, 2, 5, 16, 23, 24]. This method for solving polynomial systems become popular for creating efficient solvers to minimal problems and even the automatic generator creating source codes of such solvers was proposed by [18].

Minimal solvers, such as the 5-point relative pose solver [20, 24] are often used inside RANSAC [9] and are parts of large systems like SfM pipelines. Therefore it is very important to maximize their efficiency.

Gröbner basis solvers usually consist of Gauss-Jordan (G-J) elimination or LU decomposition of one or several matrices created using so-called elimination templates [18] and the eigenvalue computation of a multiplication (action) matrix [22].

Recently, it has been demonstrated that the numerical stability of Gröbner basis solvers can be improved e.g. by reordering columns in elimination templates, using QR or LU decomposition or by "extending the basis" [3, 6, 4]. Several methods for reducing the size of elimination templates, and in this way speeding up Gröbner basis solvers and improving their stability, were proposed [19, 18].

However, it is usually not G-J elimination or LU decomposition of elimination templates but the eigenvalue computation which takes the biggest fraction of time in these solvers. Although the elimination templates are for some solvers quite large, they are relatively sparse and thus can be efficiently eliminated using sparse methods. Therefore, this part of Gröbner basis solvers usually takes few microseconds. Usually the eigenvalue and eigenvector computation is the bottleneck taking, e.g., more than $50\mu s$ for 10×10, 140μ for 15×15 or even $250\mu s$ for 20×20 action matrices, respectively. The computation time depends on the size of the action matrix, and the size of the action matrix depends on the number of solutions of the problem.

The number of solutions of a given formulation of a problem and therefore the size of the action matrix can't be often reduced. Moreover, the action matrices are relatively dense and hence sparse solvers do not help. On the other hand, many solutions to some variables, which are obtained as eigenvalues of these action matrices, are often not feasible. They are either complex or out of range, e.g. too large or negative focal lengths, depths or radial distortion coefficients.

[1] This work has been supported by EC projects FP7-SPACE-218814 PRoVisG, FP7-SPACE-241523 PRoViScout and SGS12/191/OHK3/3T/13.

In this paper we show how to use Gröbner basis solvers in such a way that only the promising interval of solutions is examined. This is done by first transforming the problem to a polynomial in a single variable and then by calculating the roots of this polynomial only on a selected interval using efficient Sturm-sequences [13]. This way we save huge amount of work spent in calculating solutions which are usually latter dropped.

We present two different methods for obtaining a single variable polynomial. The first method is based on modified FGLM [11] algorithm for transforming the grevlex Gröbner basis to the lexicographic Gröbner basis which contains a single variable polynomial and the second is based on efficient computation of the coefficients of the characteristic polynomial of the action matrix.

The standard FGLM [11] algorithm for transforming the grevlex Gröbner basis to the lexicographic Gröbner basis performs polynomial division which may take quite a long time. Here we present a modification of this method which performs only matrix-vector multiplication and which even doesn't require a complete grevlex Gröbner basis. Such modification is much more efficient than the standard FGLM [11] algorithm and results in a significant speed up of Gröbner basis solvers.

We show how the modified "matrix FGLM" algorithm is related to Krylov's algorithm [12] for computing the coefficients of the characteristic polynomial of an action matrix. Since Krylov's algorithm [12] is suitable only for small or medium size action matrices, we finally present a method based on Danilevsky algorithm [8] for computing the coefficients of the characteristic polynomial which is very efficient and can be used even for larger problems.

We demonstrate by experiments the usefulness of both methods, the modified "matrix FGLM" algorithm and the Danilevsky algorithm [8] for computing the characteristic polynomial, by significant speedup of several important minimal computer vision problems.

We next briefly describe the Gröbner basis method for solving systems of polynomial equations.

2. Gröbner basis method

Gröbner basis method for solving systems of polynomial equations is based on polynomial ideal theory and multivariate polynomial division. It generates special bases of these ideals, called Gröbner bases [7].

Let

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

be a system of m polynomial equations in n unknowns $\mathbf{x} = (x_1, \ldots, x_n)$ which we want to solve and let this system have a finite number of solutions.

These polynomials define an *ideal* $I = \{\Sigma_{i=1}^{m} f_i\, h_i \,|\, h_i \in \mathbb{C}[x_1, ..., x_n]\}$, which is a set of all polynomials that can be generated as polynomial combinations of the initial polynomials $f_1, ..., f_m$.

In general, an ideal can be generated by many different sets of generators which all share the same solutions. There is a special set of generators though, the reduced Gröbner basis w.r.t. the lexicographic ordering, which generates the ideal I but is easy (often trivial) to solve since it contains a single variable polynomial [7]. Computing this basis and "reading off" the solutions from it is one standard method for solving systems of polynomial equations [7].

Unfortunately, for larger systems of polynomial equations, and therefore for most computer vision problems, computing the Gröbner basis w.r.t. the lexicographic ordering is not feasible. It is because a computation of Gröbner basis is in general an EXPSPACE-complete problem. It may take very long time to compute this basis and huge space may be necessary for storing intermediate results [15].

Therefore, Gröbner basis solvers usually construct a Gröbner basis G under another ordering, e.g. the graded reverse lexicographic (grevlex) ordering, which is often easier to compute. Then, the properties of the *quotient ring* $A = \mathbb{C}[x_1, \ldots, x_n]/I = \{[f] : f \in \mathbb{C}[x_1, \ldots, x_n]\}$, which is the set of equivalence classes for congruence modulo I, with cosets $[f] = f + I = \{f + h : h \in I\}$ and $[f] = [g] \Leftrightarrow f - g \in I$, are used to get the solutions to the system (1) [7]. Usually the remainder of the polynomial f on the division by the Gröbner basis G, denoted as $\overline{f}^G$, is used as a standard representative of the coset $[f] \in A$.

In the quotient ring A, the multiplication (action) matrix $\mathtt{M}_p$ [22] is constructed. It is the matrix of the linear operator $T_p \colon A \to A$ of the multiplication by a suitably chosen polynomial p w.r.t. some basis $B = \{[b_1], \ldots, [b_s]\}$ of A. Usually, the standard monomial basis $B = \left\{[\mathbf{x}^\alpha] : \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}$ of A is used. Here $\mathbf{x}^\alpha$ represents a monomial $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$.

Then we can represent the multiplication mapping T_p by representing the image $T_p([b_i])$ of every basis element $[b_i],\ i = 1 \ldots, s$ in terms of $B = \{[b_1], \ldots, [b_s]\}$

$$T_p([b_j]) = [p].[b_j] = [pb_j] = \sum_{i=1}^{s} m_{ij}\,[b_i] \tag{2}$$

with $s \times s$ multiplication (action) matrix $\mathtt{M}_p := (m_{ij})$.

The solutions to the system of equations (1) can be read off directly from the eigenvalues and the eigenvectors of the action matrix (2). Therefore this action matrix can be viewed as a generalization of the companion matrix used in solving one polynomial equation in one unknown [7].

3. Gröbner basis solvers

Since many computer vision problems share the convenient property that the monomials which appear in the set

of initial polynomials (1) are always the same irrespectively from the concrete coefficients arising from non-degenerate image measurements, it is not necessary to use general algorithms [7] for constructing Gröbner bases when solving these problems.

Therefore, most Gröbner basis solvers for computer vision problems consist of two phases. In the first "offline" phase, so-called "elimination templates" are found. These templates say which input polynomials should be multiplied with which monomials and then eliminated to obtain all polynomials from the grevlex Gröbner basis or at least all polynomials necessary for constructing an action matrix. This phase is for a one concrete problem performed only once.

The second "online" phase consist of two steps. In the first step the found elimination templates are used with concrete coefficients arising from image measurements to construct the action matrix. Then eigenvalues and eigenvectors of this action matrix are used to find solutions to initial polynomial equations and in this way to solve the input problem.

Many papers addresses the numerical stability and size of elimination templates [3, 6, 4, 18, 19]. The second step of "online solvers" was often done by finding the eigenvalues and the eigenvectors using a standard numerical eigenvalue algorithm [21].

In this paper we propose two modifications of this second step of Gröbner basis solvers, such that only feasible intervals of solutions are examined. We create a polynomial in a single variable and find its solutions by calculating the roots only in a selected interval using Sturm-sequences [13]. This significantly speeds up Gröbner basis solvers.

The first method, which we present, is based on a modified FGLM algorithm [11] for transforming the grevlex Gröbner basis to the lexicographic Gröbner basis. This modified method doesn't need a complete grevlex Gröbner basis and uses the computed action matrix and matrix-vector multiplication to obtain a single-variable polynomial. The second method computes eigenvalues of the action matrix as the roots of its characteristic polynomial.

We next present the method based on FGLM algorithm.

4. Gröbner basis conversion method

Another Gröbner basis method for solving systems of polynomial equations (1) is based on conversion of the Gröbner basis w.r.t. some feasible monomial ordering, e.g. the grevlex ordering to a lexicographic Gröbner basis containing a single-variable polynomial. There exist several algorithms for Gröbner basis conversion, e.g. the well known FGLM algorithm [11].

FGLM algorithm hasn't been used to solve minimal computer vision problems yet. It is probably because this method requires to perform quite inefficient and sometimes also numerically unstable polynomial division. Therefore, it was assumed that it is more efficient to find solutions to the initial system of polynomial equations by computing the eigenvalues and the eigenvectors of an action matrix constructed from a grevlex Gröbner basis.

We present a modified "matrix FGLM" algorithm and show that it, for some systems of equations, leads to significantly faster solvers than the standard "eigenvalue action matrix" method. Then we show how this "matrix FGLM" algorithm is related to Krylov's algorithm [12] for computing the coefficients of the characteristic polynomial. It reveals the relationship between the standard FGLM algorithm [11], an action matrix and its characteristic polynomial.

4.1. Standard FGLM algorithm

The standard FGLM algorithm [11] starts with some Gröbner basis G of the ideal I and converts it to a lexicographic Gröbner basis G_{lex}, or a Gröbner basis w.r.t. some other monomial ordering, of the same ideal. The algorithm uses only two structures, a list $G_{lex} = \{g_1, \ldots, g_k\}$ which at each stage contains a subset of the lexicographic Gröbner basis and a list B_{lex} which contains a subset of the monomial basis of the quotient ring $A = \mathbb{C}[x_1, \ldots, x_n]/I$. Both these lists are initially empty. For each monomial $\mathbf{x}^\alpha \in \mathbb{C}[x_1, \ldots, x_n]$ in increasing lexicographic ordering, starting with $\mathbf{x}^\alpha = 1$, the algorithm performs these three steps:

1. For the input $\mathbf{x}^\alpha$, compute $\overline{\mathbf{x}^\alpha}^G$

   - If $\overline{\mathbf{x}^\alpha}^G = \sum_j c_j \overline{\mathbf{x}^{\alpha(j)}}^G$, where $\mathbf{x}^{\alpha(j)} \in B_{lex}$ and $c_j \in \mathbb{C}$, i.e. if $\overline{\mathbf{x}^\alpha}^G$ is linearly dependent on the remainders on division by G of the monomials in B_{lex}, then add polynomial $g = \mathbf{x}^\alpha - \sum_j c_j \mathbf{x}^{\alpha(j)} \in I$ to G_{lex} as the last element.
   - Otherwise add $[\mathbf{x}^\alpha]$ to B_{lex} as the last element.

2. Termination Test

   If a polynomial g was added to G_{lex}, and if leading term $LT(g)$ [7] is a power of the greatest variable in the lexicographic ordering, then STOP.

3. Next Monomial

   Replace $\mathbf{x}^\alpha$ with the next monomial, w.r.t. the lexicographic ordering, which is not divisible by any of the monomials $LT(g_i)$ for $g_i \in G_{lex}$. GOTO 1.

Note that for the division by G in this algorithm the original ordering (not the lexicographic ordering) is used and that for our applications it is sufficient to stop this algorithm after finding the first polynomial from G_{lex}, i.e. the single-variable polynomial.

This means that for the ordering of the variables $x_1 > x_2 > \cdots > x_n$, the FGLM algorithm starts with the monomial $\mathbf{x}^\alpha = 1$ and continues with monomials x_n, x_n^2, x_n^3 and so on in increasing lexicographic order and computes reminders $\overline{x_n^i}^G$ on the division by G. After reaching x_n^s, where s is the number of solutions to the initial system of equations, it expresses $\overline{x_n^s}^G$ as the linear combination of $\overline{x_n^i}^G$, $i < s$. Then the coefficients of this linear combination are the coefficients of the single-variable polynomial from the lexicographic Gröbner basis G_{lex}.

4.2. Modified matrix FGLM algorithm

The standard FGLM algorithm [11] performs polynomial division which may be for some problems less efficient than the "eigenvalue action matrix" method.

Here we show how action matrix powers can be used instead of polynomial division. We propose a method, which we call the "matrix FGLM" algorithm, and which requires only matrix-vector multiplication for obtaining a single-variable polynomial. This method for many problems results in a significant speedup over the eigenvalue computation.

This method even doesn't require a complete grevlex Gröbner basis. All it needs is the action matrix $\mathtt{M}_{x_i}$ of the linear operator T_{x_i} of multiplication by some variable $[x_i]$ in $A = \mathbb{C}[x_1, \ldots, x_n]/I$ w.r.t. the standard monomial basis $B = \left\{[\mathbf{x}^\alpha] : \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}$ of A.

In this case the j^{th} column of the matrix $\mathtt{M}_{x_i}$ corresponds to the reminder $\overline{x_i b_j}^G$, where $[b_j] \in B$. For example, imagine that the basis $B = \{[xy], [x], [y], [1]\}$ and that we have the action matrix $\mathtt{M}_x = (m_{ij})_{i,j=1,\ldots,4}$. Then we can obtain the remainder of x^2 on the division by G from this action matrix as $\overline{x^2}^G = m_{12}xy + m_{22}x + m_{32}y + m_{42}$.

Moreover, since the action matrix $\mathtt{M}_{x_i}$ represents multiplication by $[x_i]$ in $A = \mathbb{C}[x_1, \ldots, x_n]/I$ we can use it to obtain reminders $\overline{x_i^k}^G$ also for higher powers k.

Let l be the largest power such that $\left[x_i^l\right] \in B$. Then we have $\overline{x_i^k}^G = x_i^k$ for $k \leq l$ and for $k > l$ we can obtain $\overline{x_i^k}^G$ in the following way.

Let $\left[x_i^l\right] = [b_q]$ for some $[b_q] \in B$, i.e. $\left[x_i^l\right]$ is the q^{th} element of B. We set

$$\mathbf{v}_l = [0 \ldots 1 \ldots 0]^\top, \tag{3}$$

$$\mathbf{v}_{j+1} = \mathtt{M}_{x_i}\mathbf{v}_j, \quad j = l, \ldots, s-1, \tag{4}$$

where 1 in $\mathbf{v}_l$ is on the q^{th} place and s is the number of solutions of our system of polynomial equations (1).

Then we have

$$\overline{x_i^k}^G = \mathbf{v}_k^\top \mathbf{b}, \quad k = l+1, \ldots, s \tag{5}$$

where $\mathbf{b} = [b_1, \ldots\ b_s]^\top$, for $[b_j] \in B$.

Moreover, these equations (3), (4) and (5) hold also for $l = 0$, i.e. for all reminders $\overline{x_i^k}^G$.

This means that we can compute the reminders $\overline{x_i^k}^G$ by simple matrix-vector multiplication. This is much more efficient that the polynomial division performed in the standard FGLM [11] algorithm describe in Section 4.

After obtaining $\overline{x_i^k}^G$ using the presented method we can continue with the standard FGLM algorithm and obtain the coefficients c_j of the single-variable polynomial by finding the coefficients of the linear combination

$$\overline{x_i^s}^G = \sum_{j=0}^{s-1} c_j \overline{x_i^j}^G. \tag{6}$$

The single-variable polynomial has then the form $x_i^s - c_{s-1}x_i^{s-1} - \cdots - c_1 x_i - c_0 = 0$.

This results in solving a simple system of s linear equations. After finding the coefficients of the single-variable polynomial we can use Sturm-sequences [13] to find its roots on the interval, which is feasible for the variable x_i, e.g. only positive values for the focal length. This gives us solutions to the variable x_i on this interval.

Solutions to the remaining variables can be obtained either by backsubstituting obtained solutions to the initial equations and solving a simplified system, by computing the eigenvectors of the action matrix $\mathtt{M}_{x_i}$, or by performing the whole presented process also on action matrices for the remaining variables.

We next show that the presented "matrix FGLM" algorithm, which we have obtained using properties of the action matrix, is equivalent to the Krylov's algorithm [12] for computing the coefficients of a characteristic polynomial.

5. Characteristic polynomial method

Let $\mathtt{A}$ be an $n \times n$ matrix. To find its eigenvalues and eigenvectors we need to solve the matrix equation $\mathtt{A}\mathbf{x} = \lambda\mathbf{x}$, where λ is the eigenvalue corresponding to the eigenvector $\mathbf{x} \neq 0$. Therefore the eigenvalues λ must satisfy the equation $\det(\mathtt{A} - \lambda\mathtt{I}) = 0$. This is an n^{th} degree monic polynomial

$$p_\mathtt{A}(\lambda) = \lambda^n + p_{n-1}\lambda^{n-1} + \cdots + p_1\lambda + p_0 \tag{7}$$

in λ called the characteristic polynomial of matrix $\mathtt{A}$. Its roots are the eigenvalues of matrix $\mathtt{A}$.

Next we describe two methods for finding the coefficients a_i of the characteristic polynomial $p_\mathtt{A}(\lambda)$ for a given matrix $\mathtt{A}$. The first one is the Krylov's method [12] which is equivalent to the "matrix FGLM" algorithm presented in Section 4.2 and the second is the Danilevsky method [8] which is numerically more stable.

5.1. Krylov's method

Krylov's method for computing the characteristic polynomial $p_{\mathtt{A}}(\lambda)$ uses the Cayley-Hamilton theorem. According to this theorem the matrix $\mathtt{A}$ satisfies its characteristic polynomial, i.e.

$$p_{\mathtt{A}}(\mathtt{A}) = \mathtt{A}^n + p_{n-1}\mathtt{A}^{n-1} + \cdots + p_1\mathtt{A} + p_0\mathtt{I}_n = \mathtt{O}_n, \quad (8)$$

where $\mathtt{I}_n$ is the $n \times n$ identity matrix.

Let $\mathbf{v} \neq 0$ be a non-zero $n \times 1$ vector, e.g. $\mathbf{v} = [1\,0\ldots0]^\top$. Then from (8) we get

$$\mathtt{A}^n\mathbf{v} + p_{n-1}\mathtt{A}^{n-1}\mathbf{v} + \cdots + p_1\mathtt{A}\mathbf{v} + p_0\mathbf{v} = 0. \quad (9)$$

This means that the vectors $\mathbf{v}_k,\ k = 1,\ldots,n$, defined as

$$\mathbf{v}_k = \mathtt{A}^k\mathbf{v} \quad (10)$$

are linearly dependent.

Therefore, to obtain the coefficients on the characteristic polynomial it is sufficient to compute vectors $\mathbf{v}_k$ (10), $i = 1,\ldots,n$ and find the coefficients of the linear combination (9) by solving a system of n linear polynomial equations in n unknowns.

This is exactly what is done in the "matrix FGLM" algorithm presented in Section 4.2. We see that the sequence (10) of vectors $\mathbf{v}_k$, called the Krylov's sequence, is equivalent to the sequence of vectors (4), used in the "matrix FGLM" algorithm, for $\mathtt{M}_{x_i} = \mathtt{A}, s = n, l = 0$, and $\mathbf{v}_0 = \mathbf{v}$. The searched coefficients a_i of the characteristic polynomial are then the coefficients $-c_i$ from (6).

Krylov's method, as well as the "matrix FGLM" algorithm presented in Section 4.2, performs $\frac{3}{2}n^2(n+1)$ multiplications and divisions and works well for small or medium size matrices. For larger matrices it may have numerical problems.

We next describe another method for computing the coefficients of the characteristic polynomial $p_{\mathtt{A}}(\lambda)$ (7) which is more stable than the Krylov's method.

5.2. Danilevsky method

A very efficient and numerically stable method for computing the coefficients of the characteristic polynomial (7) was proposed by Danilevsky [8]. The main idea of this method is in transforming the matrix

$$\mathtt{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,n} \end{bmatrix} \quad (11)$$

to the matrix $\mathtt{P}$ of the form

$$\mathtt{P} = \begin{bmatrix} -p_1 & -p_2 & \ldots & -p_{n-1} & -p_n \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ddots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix}. \quad (12)$$

The matrix $\mathtt{P}$ is known as the companion matrix of $p_{\mathtt{A}}(\lambda)$ (7), or the Frobenius form of $\mathtt{A}$.

Since the matrices $\mathtt{A}$ and $\mathtt{P}$ have the same characteristic polynomial $p_{\mathtt{A}}(\lambda)$, they are similar. This means that $\mathtt{P} = \mathtt{T}^{-1}\mathtt{A}\mathtt{T}$ for some regular transformation matrix $\mathtt{T}$.

In the Danilevsky method this transformation from the matrix $\mathtt{A}$ to $\mathtt{P}$ is done by $n-1$ similarity transformations $\mathtt{T}_i$ which successively transform the rows of $\mathtt{A}$, beginning with the last row, into corresponding rows of $\mathtt{P}$. In fact this method applies a special form of the G-J elimination to transform $\mathtt{A}$ to its Frobenius form.

Here we describe only the first step of this method and the form of the matrices $\mathtt{T}_{n-1}$ and $\mathtt{T}_{n-1}^{-1}$. In this first step we want to transform the last row of $\mathtt{A}$ to the last row of $\mathtt{P}$. Assuming $a_{n,n-1} \neq 0$, the searched transformations have the form

$$\mathtt{T}_{n-1} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ddots & \ldots & \ldots \\ -\frac{a_{n,1}}{a_{n,n-1}} & -\frac{a_{n,2}}{a_{n,n-1}} & \ldots & \frac{1}{a_{n,n-1}} & -\frac{a_{n,n}}{a_{n,n-1}} \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix} \quad (13)$$

and

$$\mathtt{T}_{n-1}^{-1} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ 1 & 1 & \ldots & 0 & 0 \\ \ldots & \ldots & \ddots & \ldots & \ldots \\ a_{n,1} & a_{n,2} & \ldots & a_{n,n-1} & a_{n,n} \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix}. \quad (14)$$

The matrix $\mathtt{T}_{n-1}^{-1}\mathtt{A}\mathtt{T}_{n-1}$ has the desired form with the last row equal to the last row of $\mathtt{P}$ so the process can continue with the next row.

In this way we can easily construct $n-1$ similarity transformations $\mathtt{T}_i$ such that

$$\mathtt{P} = \mathtt{T}_1^{-1}\mathtt{T}_2^{-1}\ldots\mathtt{T}_{n-1}^{-1}\mathtt{A}\mathtt{T}_{n-1}\ldots\mathtt{T}_2\mathtt{T}_1. \quad (15)$$

Then the coefficients of the characteristic polynomial $p_{\mathtt{A}}(\lambda)$ (7) can be extracted from the computed matrix $\mathtt{P}$. Note that it is not necessary to perform full matrix multiplications when computing $\mathtt{T}_i^{-1}\mathtt{A}\mathtt{T}_i$ since matrices $\mathtt{T}_i$ are close to the identity matrix.

More about this method and solutions to cases when some pivots are equal to zero can be found in [8, 12]. The number of multiplications and divisions performed in this method is equal to $(n-1)\left(n^2+n-1\right)$ i.e. proportional to a single G-J elimination. This method is numerically more stable than the Krylov's method presented in Section 5.1 and its accuracy can be even increased by pivoting.

Moreover, Danilevsky method can be also used to find the eigenvectors of the matrix $\mathtt{A}$ by transforming the eigenvectors of the matrix $\mathtt{P}$, which have very simple form, using the transformations $\mathtt{T}_i$.

We next briefly describe how this method can be used to solve systems of polynomial equations.

5.3. Characteristic polynomial of the action matrix

Let's consider that we have constructed the action matrix $\mathtt{M}_{x_i}$ for the variable x_i using the method described in Sections 2 and 3. We know that the eigenvalues of $\mathtt{M}_{x_i}$ give solutions to x_i and that the solutions to the remaining variables can be obtained from its eigenvectors.

Instead of computing these eigenvalues and eigenvectors using a standard numerical algorithm we can use the presented Danilevsky method. This is efficient especially when we know some constraints on the variable x_i.

After finding the coefficients of the characteristic polynomial $p_{\mathtt{M}_{x_i}}(x_i)$ of the action matrix $\mathtt{M}_{x_i}$, as described in Section 5.2, we can use Sturm-sequences [13] to find its roots on the interval, which is feasible for the variable x_i.

Solutions to the remaining variables x_j can be again obtained either by backsubstituting obtained solutions to the initial polynomial equations and solving simplified system or by computing the eigenvectors of the action matrix $\mathtt{M}_{x_i}$ corresponding to obtained solutions.

6. Minimal solvers

To show the usefulness of the methods presented in Sections 4.2 and 5.2 we have used them to create efficient fast solvers to several important minimal problems. All these problems have been previously solved using the "eigenvalue action matrix" method described in Section 2.

We next briefly describe these three problems and their existing Gröbner basis solutions which are the bases of our new fast solutions. Note that in all our solutions we only replace the eigenvalue action matrix computation, performed in the existing Gröbner basis solver, with the proposed "matrix FGLM" method or with the Danilevsky method for computing the coefficients of the characteristic polynomial, followed by the computation of the roots of the obtained single-variable polynomial using the Sturm-sequences [13].

6.1. 5-point relative pose

The problem of estimating relative pose of two calibrated cameras from five image point correspondence is one of the oldest and well-studied problems.

The state-of-the-art methods [20, 23] use the formulation which results in solving ten equations in three unknowns and gives ten solutions to this problem.

In this case, the Gröbner basis solution [24] is quite simple since it only performs G-J elimination of the 10×20 coefficient matrix representing the initial ten polynomials. Then, the action matrix is created, from the rows of this eliminated matrix. In [24] the solutions are obtained from the eigenvalues and the eigenvectors of this action matrix. This is much less efficient than the state-of-the-art solver [20], which uses special structure of the initial equations to obtain almost a closed-form solution.

However, in experiments we show that by replacing the eigenvalue computation in the Gröbner basis solver [24] with the "matrix FGLM" method or the Danilevsky method for computing the coefficients of the characteristic polynomial, followed by the computation of its roots using the Sturm-sequences [13], we obtain comparable efficiency to the Nister's solution [20].

6.2. 6-point focal length problem

The problem of estimating relative camera pose for two cameras with unknown, but equal, focal length from minimal number of six point correspondences has 15 solutions. It leads to solving ten equations in three unknowns [23].

There exist several Gröbner basis solutions to this problem [23, 3, 18] which find the solutions by computing the eigenvalues of an action matrix using a standard eigenvalue method. The solution [18] which results in the smallest "elimination template" performs G-J elimination of the 31×46 matrix. From the rows of this eliminated matrix the action matrix for the unknown focal length is created.

This action matrix is the input to both our methods presented in Sections 4.2 and 5.2.

6.3. P4P+f problem

The last problem is the problem of estimating the absolute pose of a camera with unknown focal length from four 2D-3D point correspondences. This problem results in five equations in four unknowns and has ten solutions [1].

To obtain the smallest solver for the P4P+f problem, we have used the solver generator [18] with equations proposed in [1]. In this way we have created a Gröbner basis "eigenvalue action matrix" solver which performs G-J elimination of the 52×62 matrix and for which the action matrix for the unknown focal length is created. This action matrix is again the input to both our presented methods.

7. Experiments

In this section we compare the speed and the numerical stability of the proposed solutions with the existing state-of-the-art solvers. Since all solvers are algebraically equivalent, we have evaluated them on synthetic noise free data only.

In all our experiments and performance tests we executed each algorithm $10K$ times on synthetically generated data. All scenes in these experiments were generated using 3D points randomly distributed in a 3D cube. Each 3D point was projected by a camera with random feasible orientation and position and random or fixed focal length.

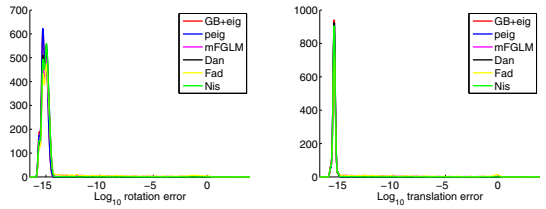

Figure 1. Numerical stability of studied 5-pt relative pose solvers. Peaks on the right hand side of both figures indicate algorithm failure.

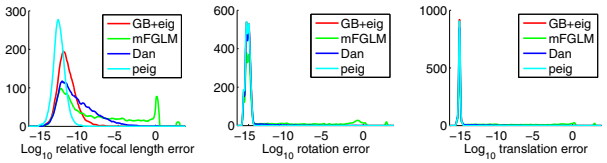

Figure 2. Numerical stability of studied 6-pt focal length solvers. Peaks on the right hand side of all figures indicate algorithm failure.

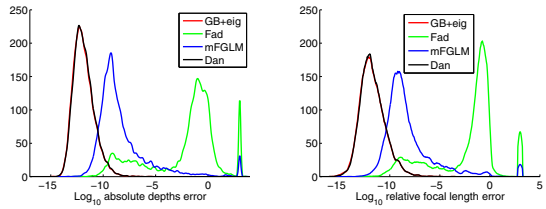

Figure 3. Numerical stability of studied P4P+f solvers. Peaks on the right hand side of both figures indicate algorithm failure.

7.1. Numerical stability

We first study the numerical stability of the proposed approaches on three selected important computer vision problems. We collected several different publicly available solvers from the Internet [14], reimplemented several state-of-the-art methods [20, 1, 23], and compared them with our new solvers based on the methods presented in Sections 4.2, 5.1, 5.2. For this comparison we have also implemented one additional method for computing the coefficients of the characteristic polynomial, i.e. the well known Faddeev-Leverrier method [10].

Figure 1 shows the comparison of several solvers to the 5-pt relative pose problem: GB+eig denotes the Gröbner basis "action matrix eigenvalue" solver [24], Nis - the Nisters solver [20], peig - the polynomial eigenvalue solver [17] , mFGLM - the solver based on the proposed "matrix FGLM" method, Fad - the solver based on Faddeev-Leverrier algorithm [10] and Dan - the solver based on the presented Danilevsky method.

The rotation error was measured as the angle in the angle axis representation of the relative rotation $\mathtt{R}\,\mathtt{R}_{gt}^{-1}$, where $\mathtt{R}_{gt}$ is the ground truth rotation and $\mathtt{R}$ the estimated rotation, and the translation error as an angle between ground-truth and estimated translation vectors.

All new fast solvers (mFGLM, Fad, Dan) behave competitively compared to the remaining solvers. Among these new solvers, the solver based on the Danilevsky method, Section 5.2, is the best in precision and as it will be seen latter in speed too. This is true also for the remaining two tested problems, the 6-pt focal length an the P4Pf problem.

For the 6-pt focal length problem, we compared the Gröbner basis "action matrix eigenvalue" solution [18] (GB-eig), with the polynomial eigenvalue solution [17] (peig) and our two modifications of the Gröbner basis solver [18] using the modified "matrix FGML" method presented in Section 4.2 (mFGLM) and the Danilevsky method, Section 5.2 (Dan). We do not show plots for the Faddeev-Leverrier algorithm [10] since it was very unstable for this problem. Figure 2 shows that the "matrix FGML" method also suffers a little bit from numerical instability.

Finally, Figure 3 shows the comparison of several P4P+f solves, i.e. the Gröbner basis "action matrix eigenvalue" solver [1] (GB-eig), and our modifications of this solver using the modified "matrix FGML" method, Section 4.2 (mFGLM), the Faddeev-Leverrier algorithm [10] (Fad) and the Danilevsky method, Section 5.2 (Dan). It is clear that the Faddeev-Leverrier method suffers from a large numerical instability and can not be used for solving this problem. The "matrix FGML" method failed several times which is visible on the right hand side of both plots. The best results are again obtained using the Danilevsky method which for all considered problems results in very stable solvers.

7.2. Speedup

In this experiment we are focusing on achieved speedup. We reimplemented most of the competing solvers in C++ and used the same math libraries in all tests. We omitted solvers which are known to be slower since our main focus is on speed. We have used Sturm sequences [13] to find real roots of the single-variable polynomial in all new solvers based on the "matrix FGML" method and the characteristic polynomial method. No special optimization e.g. CPU intrinsic such as SSE were used. All tests were performed on Intel i7 Q720 1.6Ghz notebook.

Table 1 shows results for 5-point relative pose problem:

Nister	GB+eig	Faddeev	mFGML	Danilevsky
10.6μs	61.2μs	17.2μs	13.7μs	14.2μs

Table 1. Speed comparison of 5-pt relative pose solvers.

Our reimplementation of the Nister's algorithm [20] is the fastest for the 5-pt relative pose problem. It is because in this solution almost all computations can be done in a closed form. Our reimplementation of the Gröbner basis "action matrix eigenvalue" solver [18] (GB+eig) which uses a standard eigenvalue method [21] is almost $6\times$ slower due

to eigenvalue and eigenvector computations. By replacing this eigenvalue computations with either the proposed "matrix FGML" method or the characteristic polynomial calculation using the Danilevsky method presented in Section 5.2 we achieved more than $4\times$ speed up.

Table 2 shows results for the 6-pt focal length problem. It can be seen that replacing the eigenvalue computations in the Gröbner basis solver [18] with the "matrix FGLM" method or the Danilevsky method resulted in almost $8\times$ speed up. Note that the "matrix FGML" algorithm is a little bit less stable comparing to the remaining algorithms.

GB+eig	mFGML	Danilevsky
176.3μs	21.3μs	22.6μs

Table 2. Speed comparison of 6-pt focal length solvers.

In this case we do not provide results for the Faddeev-Leverrier algorithm [10] because its numerical stability is poor and it failed to deliver any result most of the time.

Finally, the last Table 3 shows results for the absolute pose problem with the unknown focal length.

GB+eig	sparse GB	Faddeev	mFGML	Danilevsky
127.4μs	82.9μs	51.2μs	46.2μs	47.4μs

Table 3. Speed comparison of P4P+f solvers.

We also obtained significant speedup over the existing Gröbner basis "eigenvlaue action matrix" algorithm [1] (GB+eig). Here sparse GB corresponds to the Gröbner basis "eigenvlaue action matrix" algorithm [1] with the sparse G-J elimination. Again the algorithm based on the Danilevsky method outperforms all the remaining algorithms both in numerical stability and speed.

8. Conclusion

We presented several methods for speeding up minimal solvers based on Gröbner basis and action matrix computation. We showed that such solvers can be significantly sped up by replacing the eigenvalue computations with the computation of the characteristic polynomial of an action matrix followed by the calculation of its roots using Sturm-sequences. We demonstrated how effective the proposed methods are on three important computer vision problems.

Source codes of the presented solvers are available at http://cmp.felk.cvut.cz/minimal/ and http://www.solvergenerator.com/vision/.

References

[1] M. Bujnak, Z. Kukelova, T. Pajdla, A general solution to the P4P problem for camera with unknown focal length. *CVPR 2008*.

[2] M. Bujnak, Z. Kukelova, and T. Pajdla. 3D reconstruction from image collections with a single known focal length. In *ICCV 2009*.

[3] M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of Gröbner basis polynomial equation solver. *ICCV 2007*.

[4] M. Byröd, K. Josephson, and K. Åström. A Column-Pivoting Based Strategy for Monomial Ordering in Numerical Gröbner Basis Calculations. *ECCV 2008*.

[5] M. Byröd, M. Brown, and K. Åström. Minimal Solutions for Panoramic Stitching with Radial Distortion. *BMVC 2009*.

[6] M. Byröd. Numerical Methods for Geometric Vision: From Minimal to Large Scale Problems. *PhD Thesis, Centre for Mathematical Sciences, Lund University* , 2010.

[7] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry, Second edition*, volume 185. Springer Verlag, Berlin - Heidelberg - New York, 2005.

[8] A.M. Danilevskii. The numerical solution of the secular equation (Russian), *Matem. Sbornik*, 44(2), 1937.

[9] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, (1981).

[10] D. K. Faddeev and W. N. Faddeeva. Computational Methods of Linear Algebra. Freeman, San Francisco, 1963.

[11] J. Faugére, P. Gianni, D. Lazard and T. Mora. Efficient computation of zero-dimensional ideal Gröbner bases by change of ordering. *J. Symbolic Comput*. 16, 1993.

[12] A. Householder. The Theory of Matrices in Numerical Analysis. Blaisdell, Waltham, Mass., 1964.

[13] D. Hook, P. McAree, Using Sturm Sequences To Bracket Real Roots of Polynomial Equations, Graphic Gems I, Academic Press, 416-423, 1990.

[14] http://cmp.felk.cvut.cz/minimal/

[15] K. Kuehnle and E. Mayr. Exponential space computation of Groebner bases. *In Proceedings of ISSAC*. ACM, 1996.

[16] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. *CVPR 2007*.

[17] Z. Kukelova, M. Bujnak, T. Pajdla, Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems. *BMVC 2008*.

[18] Z. Kukelova and M. Bujnak and T. Pajdla. Automatic Generator of Minimal Problem Solvers. *ECCV 2008*.

[19] O. Naroditsky and K. Daniilidis, Optimizing Polynomial Solvers for Minimal Geometry Problems, *ICCV 2011*.

[20] D. Nister. An efficient solution to the five-point relative pose. *IEEE PAMI*, 26(6):756–770, 2004.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C++: The Art of Scientific Computing. Cambridge University Press, February 2002.

[22] H. J. Stetter. *Numerical Polynomial Algebra*, SIAM, 2004.

[23] H. Stewenius, D. Nister, F. Kahl, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. *CVPR 2005*.

[24] H. Stewenius, C. Engels, and D. Nister. Recent developments on direct relative orientation. *ISPRS J. of Photogrammetry and Remote Sensing*, 60:284–294, 2006.