Automatic Generator of Minimal Problem Solvers

Zuzana Kukelova[1], Martin Bujnak[1,2] and Tomas Pajdla[1]

[1] Center for Machine Perception
Czech Technical University, Prague
[2] Microsoft Corporation
`kukelova,bujnam1,pajdla@cmp.felk.cvut.cz`

Abstract. Finding solutions to minimal problems for estimating epipolar geometry and camera motion leads to solving systems of algebraic equations. Often, these systems are not trivial and therefore special algorithms have to be designed to achieve numerical robustness and computational efficiency. The state of the art approach for constructing such algorithms is the Gröbner basis method for solving systems of polynomial equations. Previously, the Gröbner basis solvers were designed ad hoc for concrete problems and they could not be easily applied to new problems. In this paper we propose an automatic procedure for generating Gröbner basis solvers which could be used even by non-experts to solve technical problems. The input to our solver generator is a system of polynomial equations with a finite number of solutions. The output of our solver generator is the Matlab or C code which computes solutions to this system for concrete coefficients. Generating solvers automatically opens possibilities to solve more complicated problems which could not be handled manually or solving existing problems in a better and more efficient way. We demonstrate that our automatic generator constructs efficient and numerically stable solvers which are comparable or outperform known manually constructed solvers. The automatic generator is available at http://cmp.felk.cvut.cz/minimal [3].

1 Introduction

Many problems can be formulated using systems of algebraic equations. Examples are the minimal problems in computer vision, i.e. problems solved from a minimal number of point correspondences, such as the five point relative pose problem [20], the six point focal length problem [18], six point generalized camera problem [19], the nine point problem for estimating para-catadioptric fundamental matrices [9], the eight point problem for estimating fundamental matrix and single radial distortion parameter for uncalibrated cameras [11], the six point problem for estimating essential matrix and single radial distortion parameter for calibrated cameras [12, 4], the nine point problem for estimating fundamental matrix and two different radial distortion parameters for uncalibrated cameras [12, 4]. These are important problems with a broad range of applications [10].

[3] This work has been supported by EC projects FP6-IST-027787 DIRAC and MRTN-CT-2004-005439 VISIONTRAIN and grants MSM6840770038DMCMIII, STINT Dur IG2003-2 062 and MSMT KONTAKT 9-06-17.

Often, polynomial systems which arise are not trivial. They consist of many polynomial equations in many unknowns and of higher degree. Therefore, special algorithms have to be designed to achieve numerical robustness and computational efficiency. The state of the art method for constructing such algorithms, the solvers, is the Gröbner basis method for solving systems of polynomial equations. It was used to solve all previously mentioned computer vision problems.

The Gröbner basis solvers in computer vision are mostly designed for concrete problems and in general consist of three key steps. In the first step, the problem is solved using a computer algebra system, e.g. Macaulay 2 or Maple, for several (many) random coefficients from a finite field. This is done using a general technique for solving polynomial equations by finding a Gröbner basis for original equations [6]. In this phase, the basic parameters of the problem are identified, such as whether there exists a finite number of solutions for general coefficients and how "hard" is to obtain the Gröbner basis, and which monomials it contains. This procedure is automatic and relies on general algorithms of algebraic geometry such as Buchberger [1] or F4 [8] algorithm. The computations are carried out in a finite field in order to avoid expensive growth of coefficients and to avoid numerical instability.

In the second step, a special elimination procedure, often called "elimination template" is generated. This elimination template says which polynomials from the ideal should be added to the initial polynomial equations to obtain a Gröbner basis or at least all polynomials needed for constructing a special matrix, called "action matrix", and thus solving the initial equations. The goal of this step is to obtain a computationally efficient and numerically robust procedure. Until now, this step has been mainly manual requiring to trace the path of elimination for random coefficient form the finite field, checking redundancies and possible numerical pitfalls, and writing down a program in a procedural language such as Matlab or C.

In the last step, the action matrix is constructed from the resulting equations and the solutions to the original problem are obtained numerically as the eigenvalues or eigenvectors of the action matrix [7].

The first and the third step are standard and well understood. It is the second step which still involves considerable amount of craft and which makes the process of solver generating rather complex and virtually impenetrable for a non-specialist. Moreover, for some problems it need not be clear how their elimination templates were generated and therefore non-specialists often use them as black boxes, are not able to reimplement them, improve them or create similar solvers for their own new problems.

In this paper we present an automatic generator of the elimination templates for Gröbner basis solvers to an interesting class of systems of polynomial equations which appear in computer vision problems. We have to accept that there is no hope to obtain an efficient and robust solver for completely general systems of polynomial equations since the general problem is known to be NP-complete and EXPSPAPCE-complete [15]. On the other hand, all known computer vision problems share the property that the elimination path associated with their solution is the same for all interesting configurations of coefficients of a given problem.

Consider, for instance, the classical 5 point problem for estimating the essential matrix from 5 point correspondences [16, 20]. In general, the elimination path used to solve the 5 point problem depends on actual image coordinates measured. Fortunately, for non-degenerate image correspondences, i.e. for those which lead to a finite number of essential matrices, the elimination path is always the same. Therefore, it is enough to find the path for one particular non-degenerate configuration of coefficients and then use the same path for all non-degenerate configuration of coefficients. The paths for degenerate configurations of coefficients may be very different and there may be very many of them but we need not consider them.

We propose an automatic generator that finds one elimination path. It can find any path. The choice of the path is controlled by the particular coefficients we choose to generate the elimination template. We demonstrate that our automatic generator constructs efficient and numerically stable solvers which are comparable or outperform known manually constructed solvers.

The input into our automatic generator is the system of polynomial equations which we want to solve with a particular choice of coefficients from $\mathbb{Z}_p$ that choose the particular elimination path. For many problems, the interesting "regular" solutions can be obtained with almost any random choice of coefficients. Therefore, we use random coefficients from $\mathbb{Z}_p$. The output from the generator is the Matlab or C code solver that returns solutions to this system of polynomial equations for concrete coefficients from $\mathbb{R}$. In online computations, only this generated solver is called.

In the next two sections we review the Gröbner basis method for solving systems of polynomial equations and the solvers based on this method. Section 5 is dedicated to our automatic procedure for generating Gröbner basis solvers. Then, we demonstrate the results of our automatic generator on some minimal problems.

2 Solving systems of polynomial equations

Our goal is to solve a system of algebraic equations

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

which are given by a set of m polynomials $F = \{f_1, ..., f_m | \, f_i \in \mathbb{C}[x_1, ..., x_n]\}$ in n variables $\mathbf{x} = (x_1, ..., x_n)$ over the field of complex numbers.

Such system of algebraic equations can be written in a matrix form

$$\mathtt{M}X = 0, \tag{2}$$

where X is a vector of all monomials which appear in these equations and $\mathtt{M}$ is a coefficient matrix. In the next we mostly consider this matrix representation of systems of equations and, for example, by Gauss-Jordan (G-J) elimination of equations we understand G-J elimination of the corresponding coefficient matrix $\mathtt{M}$.

Solving systems of algebraic polynomial equations is a very challenging problem. There doesn't exist one robust, numerically stable and efficient method for solving such systems in general case. Therefore, special algorithms have to be designed for specific problems.

The general Gröbner basis method for solving systems of polynomial equations can be quite inefficient in some cases but it was recently used successfully as the basis for efficient solvers of computer vision minimal problems. We now describe this method and the solvers based on this method.

3 Gröbner basis method

The polynomials $F = \{f_1, ..., f_m | \, f_i \in \mathbb{C}\left[x_1, ..., x_n\right]\}$ define *ideal* I, which is the set of all polynomials that can be generated as polynomial combinations of initial polynomials F

$$I = \left\{\Sigma_{i=1}^{m} f_i \, p_i \, | \, p_i \in \mathbb{C}\left[x_1, ..., x_n\right]\right\}, \tag{3}$$

where p_i are arbitrary polynomials from $\mathbb{C}\left[x_1, ..., x_n\right]$.

We can define division by an ideal I in $\mathbb{C}\left[x_1, ..., x_n\right]$ as the division by the set F of generators of I. There are special sets of generators, Gröbner bases, of the ideal I, for which this division by the ideal I is well defined in the sense that the remainder on the division doesn't depend on the ordering of the polynomials in the Gröbner basis G. This means that the remainder of an arbitrary polynomial $f \in \mathbb{C}\left[x_1, ..., x_n\right]$ on the division by G in a given monomial ordering is uniquely determined. Furthermore, $f \in I$ if and only if the reminder of f on the division by G is zero ($\overline{f}^G = 0$). This implies that $\overline{f}^G + \overline{g}^G = \overline{f+g}^G$ and $\overline{\overline{f}^G \overline{g}^G}^G = \overline{fg}^G$

Gröebner bases generate the same ideal as the initial polynomial equations and therefore have the same solutions. However, it is important that they are often easier to solve (e.g. the reduced Gröbner basis w.r.t. the lexicographic ordering contains polynomial in one variable only). Computing such basis and "reading off" the solutions from it is one standard method for solving systems of polynomial equations. Although this sounds really nice and easy the reality is much harder. The problem is that the computation of Gröbner bases is in general an EXPSPACE-complete problem, i.e. that large space is necessary for storing intermediate results. Fortunately in many specific cases, the solution to a system of polynomial equations computed using Gröbner bases can be obtained much faster.

For solving systems of polynomial equations, the most suitable ordering is the lexicographic one which results in a system of equations in a "triangular form" with one equation in one variable only. Unfortunately, computation of such Gröbner basis w.r.t. the lexicographic ordering is very time consuming and for most of the problems can not be used. Therefore in many cases a Gröbner basis G under another ordering, e.g. the graded reverse lexicographical ordering (grevlex), which is often easier to compute, is constructed. Then, other properties of this basis are used to obtain solutions to the initial system of polynomial equations.

Thanks to the property that the division by the ideal I is well defined for the Gröbner basis G, we can consider the space of all possible remainders on the division by I. This space is know as a *quotient ring* and we will denote it as $A = \mathbb{C}\left[x_1, ..., x_n\right]/I$. It is known that if I is a radical ideal [6] and the set of equations F has a finite number of solutions N, then A is a finite dimensional space with $dim(A) = N$. Now we can use nice properties of a special action matrix defined in this space, to find solutions to our system of equations (1).

Consider the multiplication by some polynomial $f \in \mathbb{C}\left[x_1, ..., x_n\right]$ in the quotient ring A. This multiplication defines a linear mapping T_f from A to itself. Since A is a finite-dimensional vector space over $\mathbb{C}$, we can represent this mapping by its matrix $\mathtt{M}_f$ with respect to some monomial basis $B = \left\{\mathbf{x}^\alpha | \overline{\mathbf{x}^\alpha}^G = \mathbf{x}^\alpha\right\}$ of A, where $\mathbf{x}^\alpha$ is a monomial $\mathbf{x}^{\alpha} = x_1^{\alpha_1} x_2^{\alpha_2} ... x_n^{\alpha_n}$ and $\overline{\mathbf{x}^{\alpha}}^{G}$ is the reminder of $\mathbf{x}^{\alpha}$ on the division by G. The action matrix $\mathtt{M}_f$ can be viewed as a generalization of the companion matrix used in solving one polynomial equation in one unknown. It is because solutions to our system of polynomial equations (1) can be easily obtained from the eigenvalues and eigenvectors of this action matrix [7].

4 Gröbner basis solver

The Gröbner basis method for solving systems of polynomial equations based on action matrices was recently used to solve many minimal problems in computer vision. The solvers to all these minimal problems are very similar and are based on the same concepts. Many minimal problems in computer vision, including all mentioned above, have the convenient property that the monomials, appearing in the set of initial generators F are always same irrespectively from the concrete coefficients arising from non-degenerate image measurements. Therefore, the leading monomials of the corresponding Gröbner basis, and thus the monomials in the basis B of the quotient ring A, are generally the same and can be found once in advance. This is an important observation which helps to solve all these problems efficiently.

The first step of these solvers is to analyze the particular problem, i.e. whether it is solvable and how many solutions there are, in a randomly chosen finite prime field $\mathbb{Z}_p$ $(\mathbb{Z}/\left<p\right>)$ with $p >> 7$. Coefficients in $\mathbb{Z}_p$ can be can be represented accurately and efficiently. It speeds up computations, minimizes memory requirements and especially avoids numerical instability. Computing with floating point approximations of the coefficients may lead to numerical instability since it may be difficult to determine when the coefficients become zero.

Next, the Gröbner basis G, and the basis B are found for many random coefficients from $\mathbb{Z}_p$. Thanks to algebraic geometry theorem [21], we know that if the bases G and B remain stable for many different random coefficients, i.e. if B consists of the same monomials, they are generically equivalent to the bases of the original system of polynomial equations with rational coefficients.

With this information in hand the solver can be created. The solver typically consists of hand made elimination templates [18, 19, 9, 11] (or one template [2–5]) that determine which polynomials from the ideal I should be added to the initial equations to obtain the Gröbner basis G or at least all polynomials needed for constructing the action matrix M_f. These elimination templates are the crucial part of the solver.

An important observation has been made in [11]. It was observed that the action matrix can be constructed without computing a complete Gröbner basis G. All we need for creating the action matrix M_f is to construct polynomials from the ideal I with leading monomials from the set $(f \cdot B) \setminus B$ and the remaining monomials from B. This fact was in some way implicitly used in previous solvers [18, 19, 9] but hasn't been fully articulated.

Consider that the basis $B = \left\{\mathbf{x}^{\alpha(1)}, \ldots, \mathbf{x}^{\alpha(N)}\right\}$ of A, which has been found once in advance by computations in $\mathbb{Z}_p$. Then, the polynomials we need for constructing the action M_f matrix are of the form

$$q_i = f\mathbf{x}^{\alpha(i)} + h_i, \tag{4}$$

with $h_i = \sum_{j=1}^{N} c_{ji} \mathbf{x}^{\alpha(j)} \in A$ and $\mathbf{x}^{\alpha(i)} \in B$. It is because to construct the action matrix M_f we need to compute $T_f\left(\mathbf{x}^{\alpha(i)}\right) = \overline{f\mathbf{x}^{\alpha(i)}}^G$ for all $\mathbf{x}^{\alpha(i)} \in B$ [7]. However, if for some $\mathbf{x}^{\alpha(i)} \in B$ and chosen f, $f\mathbf{x}^{\alpha(i)} \in A$, then $T_f\left(\mathbf{x}^{\alpha(i)}\right) = \overline{f\mathbf{x}^{\alpha(i)}}^G = f\mathbf{x}^{\alpha(i)} = \sum_{j=1}^{N} d_{ji} \mathbf{x}^{\alpha(j)}$ and we are done. For all other $\mathbf{x}^{\alpha(i)} \in B$, for which $f\mathbf{x}^{\alpha(i)} \notin A$, we consider the above mentioned polynomials q_i. For these $\mathbf{x}^{\alpha(i)}$, $T_f\left(\mathbf{x}^{\alpha(i)}\right) = \overline{f\mathbf{x}^{\alpha(i)}}^G = \overline{q_i - h_i}^G = -h_i \in A$.

Then the action matrix M_f has the form

$$M_f = \begin{pmatrix} -c_{11} & d_{12} & -c_{13} & \dots & -c_{1N} \\ -c_{21} & d_{22} & . & & . \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ -c_{N1} & d_{N2} & -c_{N3} & \dots & -c_{NN} \end{pmatrix}, \tag{5}$$

where columns containing c_{ji} correspond to the monomials $\mathbf{x}^{\alpha(i)} \in B$ for which $f\mathbf{x}^{\alpha(i)} \notin A$ and columns containing d_{ji} to the monomials $\mathbf{x}^{\alpha(i)} \in B$ for which $f\mathbf{x}^{\alpha(i)} = \sum_{j=1}^{N} d_{ji} \mathbf{x}^{\alpha(j)} \in A$.

Since the polynomials q_i are from the ideal I, they can be generated as algebraic combinations of the initial generators F. This can be done using several methods. One possible way is to start with F and then systematically generate new polynomials from I by multiplying already generated polynomials by individual variables and reducing them each time by the G-J elimination. This method was, for example, used in [11] and resulted in several G-J eliminations of coefficient matrices $\mathtt{M}_1, \dots, \mathtt{M}_l$.

Another possible way is to generate all new polynomials in one step by multiplying polynomials from F with selected monomials and reducing all generated polynomials at once using single G-J elimination of one coefficient matrix $\mathtt{M}$. This method was used in [2] and it was observed to be numerically more stable.

Such systematic generation of polynomials q_i results in many unnecessary polynomials, many of which can be eliminated afterwards in a simple and intuitive way [5]. The method starts with the matrix $\mathtt{M}$, which has the property that after its G-J elimination all polynomials q_i necessary for constructing the action matrix are obtained.

Since it is known which monomials appear in q_i for a particular problem to be solved, the number of generated polynomials can systematically be reduced in the following way:

1. For all rows from $\mathtt{M}$ starting with the last row r (i.e. with the highest degree polynomial) do
   (a) Perform G-J elimination on the matrix $\mathtt{M}$ without the row r
   (b) If the eliminated matrix contains all necessary polynomials q_i, then $\mathtt{M} := \mathtt{M} \backslash \{r\}$

All the previous steps, i.e. finding the number of solutions, basis G, basis B, the generation of elimination template(s) and the reduction of unnecessary polynomials, are performed only once in the automatic generator. The generated online solver, which the user come into contact with, takes the elimination template, the matrix $\mathtt{M}$, fills it

Fig. 1. Block diagram of the automatic generator.

with concrete coefficients arising from image measurements and performs its G-J elimination. The rows of `M` then correspond to the polynomials q_i and are used to create the action matrix. Finally, the eigenvalues or the eigenvectors of this action matrix give solutions to the problem.

5 The automatic procedure for generatig Gröbner basis solvers

In this section we describe our approach to automatic generation of such Gröbner basis solvers to general problems.

The input of this automatic generator is the system of polynomial equations with coefficients from $\mathbb{Z}_p$ and the output is the solver, the MATLAB or C code, which returns solutions to this system of polynomial equations for concrete coefficients from $\mathbb{R}$.

Our automatic generator consists of several independent modules (Figure 1). Since all these parts are independent, they can be further improved or replaced by more efficient implementations. Next we briefly describe each of these parts.

5.1 Polynomial equations parser

First, input equations are split into coefficient and monomials. For the automatic generator, we instantiate each known parameter occurring in coefficients with a random number from the $\mathbb{Z}_p$. We assign a unique identifier to each coefficient used. This is important for the code generation module to be able to track the elimination path.

5.2 Computation of the basis B and the number of solutions

This module starts with the system of polynomial equations F, which we want to solve, with random coefficients from $\mathbb{Z}_p$. For many problems, the interesting "regular" solutions can be obtained with almost any random choice of coefficients. The coefficients from $\mathbb{Z}_p$ speed up computations, minimize memory requirements and especially avoid numerical instability.

The generator first verifies if the input system of equations F has a finite number of solution, i.e. the initial polynomial equations generate a zero dimensional ideal, and how many solutions there is. If the system has a finite number of solutions, we compute the Gröbner basis G w.r.t. the grevlex [6] monomial ordering and the basis B of the quotient ring A. The output of this part of the generator is the basis B of the quotient ring A.

To obtain this information we use the existing algorithms implemented in algebraic geometry softwares Macaulay 2 or Maple. Both these softwares are able to compute in finite prime fields and provide efficient implementations of all functions that we need for our purpose. Moreover, these functions can be called directly from MATLAB and their output can be further used in the generator. Modularity of the generator allows replacing this part of the code by another existing module computing the Gröbner basis G and the basis B [6].

5.3 Single elimination template construction

The input to this third, most important, step of our automatic generator is the basis B of the quotient ring A and the polynomial f for which we want to create the action matrix. We use an individual variable i.e. $f = x_k$, called "action variable", to create the action matrix.

The goal is to generate now all necessary polynomials for constructing the action matrix M_{x_k}. The method described in Section 4 calls for generating polynomials from ideal I with leading monomials from the set $(x_k \cdot B) \setminus B$ and the remaining monomials from B, i.e. polynomials of the form $q_i = x_k \mathbf{x}^{\alpha(i)} + \sum_{j=1}^{N} c_{ji} \mathbf{x}^{\alpha(j)} \in I$.

As explained in Section 4, there are several ways how to generate these polynomials from the initial polynomial equations F. We have decided to generate them in one step by multiplying polynomials from F with selected monomials and reducing all generated polynomials at once using a single G-J elimination of one coefficient matrix.

These monomial multiples of polynomials F, which should be added to the initial equations to obtain all necessary polynomials q_i, are generated in this part of the automatic generator. This is done by systematically generating polynomials of I and testing them. We stop when all necessary polynomials q_i are obtained. The generator tries to add polynomials starting with the polynomials with as low degree as possible. Thus, it first multiplies input polynomials with the lowest degree monomials and then moves to the higher degree monomials. The polynomial generator can be described as follows:

1. Generate all monomial multiples $\mathbf{x}^{\alpha} f_i$ of degree $\leq d$ (sort them by leading term w.r.t. grevlex ordering).
2. Write the polynomials $\mathbf{x}^{\alpha} f_i$ in the form $\mathtt{M}X = 0$, where `M` is the coefficient matrix and X is the vector of all ordered monomials.
3. Simplify matrix `M` by the G-J elimination.
4. If all necessary polynomials q_i have been generated, stop.
5. Else set $d = d + 1$. Go to 1.

In this way we generate polynomials, which, after G-J elimination of their corresponding coefficient matrix `M`, contain all necessary polynomials q_i. These polynomials, i.e. the matrix `M`, is the so called elimination template. Unfortunately, we often generate many unnecessary polynomials. In the next part of our automatic generator we try to minimize the number of these unnecessary polynomials.

5.4 Reducing the size of the template

This part of the automatic generator starts with the polynomials generated in the previous step. We know that after the G-J elimination of these polynomials (i.e. of the corresponding matrix `M`) we obtain all polynomials that we need for constructing of the action matrix. Starting with the coefficient matrix `M` and with the information about the form of the necessary polynomials q_i, we systematically reduce the number of generated polynomials using the method described in Section 4. The algorithm in Section 4 removes polynomials one by one and each time calls expensive G-J elimination. This is not efficient. It has almost $o(n^4)$ complexity in the number n of polynomials used. We enhanced this algorithm in several ways: (1) we use sparse G-J elimination, since elimination template is quite sparse matrix, (2) we remove more than one polynomials at once with the heuristic - if we succeeded to remove k polynomial, then we remove $2\,k$ polynomials in the next step. If we failed to remove k polynomials, we try to remove only $\frac{1}{4}\,k$ polynomials in the next step. These two steps considerably speed up the reduction process. Moreover, we can employ the fact that polynomials in the elimination template are ordered by the degree of their leading monomials. In G-J elimination of such ordered polynomials we can exploit results from previous G-J eliminations.

5.5 Construction the action matrix

To create the template for the action matrix M_{x_k}, we identify those rows of eliminated matrix `M` (matrix `M` after the G-J elimination) which correspond to polynomials q_i. The action matrix will than contain coefficients from these rows which correspond to the monomials from the basis B and will have the form (5).

For the generated online solver we just need to know which rows and columns we need to extract. Note that the structure of the eliminated matrix is always the same for all instances of the problem.

5.6 Generating efficient online solver

The generated online solver consist of the following steps:

1. construction of the coefficient matrix (from elimination template);
2. G-J elimination;
3. action matrix extraction;
4. solution extraction from eigenvectors of the action matrix.

To build the coefficient matrix we use unique identifiers associated with coefficients of each of the input polynomials. Hence besides coefficient matrix in $\mathbb{Z}_p$ we maintain matrix with coefficient identifiers, the index matrix, and apply all operations performed on actual coefficients, i.e. adding, removing rows and linear combination of rows, to the index matrix.

Recall that in the construction of the elimination template we use the input equations and multiply them by monomials. This is nothing else than shifting identifiers associated to input polynomials in the index matrix.

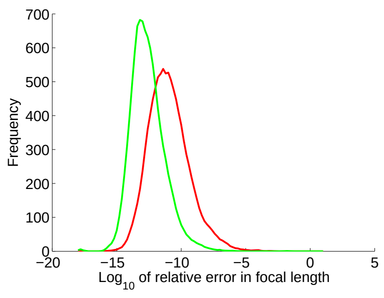

Fig. 2. The log_{10} relative errors of the focal length for 10000 runs of two solvers. Original solver [18] (Red, darker) and our generated solver (Green, lighter) on the synthetic dataset without noise.

Reducing of the polynomials and further optimizations results only in removing rows or columns in the index matrix. Hence, the code generator creates code which simply puts coefficients of the input polynomials to correct places using coefficients identifiers. Then, after G-J elimination, it reads values form precomupted rows and columns and builds the action matrix.

6 Experiments

In this section we demonstrate that our automatic generator constructs efficient and numerically stable solvers which are comparable or outperform known manually constructed solvers. For comparison, we consider five recently solved minimal problems which have a broad range of applications and can be used in a RANSAC-based estimation.

Since the automatic procedure described in this paper generates very similar Gröbner basis solvers as those proposed in the original solutions, there is no point in testing the behavior of generated solvers under noise, outliers or on real images.

Generated solvers solve the same system of polynomial equations using the same algebraic method as the original solvers. The difference is only in the number of generated polynomials and therefore the size of matrices, elimination templates, which are used to obtain solutions. For all considered problems we have obtained comparable or quite smaller elimination templates than those used in the original solvers.

We choose two problems, the well known [18] and more complex “radial distortion problem” [4], to evaluate and compare the intrinsic numerical stability of the existing solvers with our solvers generated automatically. For the remaining three problems we compare only the sizes of generated elimination templates.

The numerical stability of the solvers is compared on synthetically generated scenes without noise. These generated scenes consist of 1000 points distributed randomly within a cube. Points were projected on image planes of the two displaced cameras with the same focal lengths. We use different radial distortions in the “radial distortion problem”.

6.1 Six Point Focal Length Problem

The problem of estimating the relative camera position for two cameras with unknown focal length is a classical and popular problem in computer vision. The minimal number of point correspondences needed to solve this problem is six. This minimal problem was solved by Stewénius et. al. [18] using the Gröbner basis techniques and has 15 solutions.

In this solution the linear equations from the epipolar constraint are first used to parametrize the fundamental matrix with two unknowns, $F = F_0 + xF_1 + yF_2$. Using this parameterization, the rank constraint for the fundamental matrix and the trace constraint for the essential matrix result in ten third and fifth order polynomial equations in the three unknowns x, y and $w = f^{-2}$, where f is the unknown focal length.

The Gröbner basis solvers [18] starts with these ten polynomial equations which can be represented in a 10×33 matrix. In the first step two new polynomials are added to the initial polynomial equations and eliminated by G-J elimination. After this four new polynomials are added and eliminated. Finally two more polynomials are added and eliminated. The resulting system then contains the Gröbner basis and can be used to construct the action matrix. The resulting solver therefore consists of three G-J eliminations of three matrices of size $12 \times 33, 16 \times 33$ and 18×33.

More recently, another Gröbner basis solver to this problem was proposed in [2]. This solver uses only one G-J elimination of a 34×50 matrix and uses special technique for improving the numerical stability of Gröbner basis solvers based on changing the basis B. In this paper it was shown that this solver gives more accurate results than the original solver [18].

Our automatic generator starts with the ten initial polynomial equations in three unknowns. For this problem we have generated three different solvers for all three variables (action matrices for three different action variables x, y and w).

For the action variable w our generator first generates all monomial multiples of initial ten polynomial equations up to total degree eight. This results in the 236×165 matrix which contains all necessary polynomials q_i. After the reduction step only 41 polynomials in 60 monomials remained.

For the action variables x and y our generator generates all monomial multiples of initial ten polynomial equations up to total degree seven. This results in the 125×120 matrix. In the reduction step 94 polynomials out of these 125 are removed resulting in 31 polynomials in 50 monomials. After the G-J elimination of the corresponding 31×50 matrix (in fact 31×46 matrix is sufficient thanks to removing columns that do not affect G-J elimination) all necessary polynomial are obtained and the action matrix M_x (M_y) is created. Our generated solver results in a little bit smaller matrix than the solver proposed in [2].

Since we did not have the source code of the solver proposed in [2], we have compared our generated solver with the original solver proposed by Stewénius [18].

The log_{10} relative errors of the focal length for 10000 runs of both solvers (original Stewénius solver (Red) and our automatically generated solver (Green)) are shown in Figure 2. Our generated solver gives a little bit more accurate results than Stewéniuse's original solver.

As we have already mentioned, we have no source code to the solver proposed in [2], but according to the results presented in this paper our generated solver gives

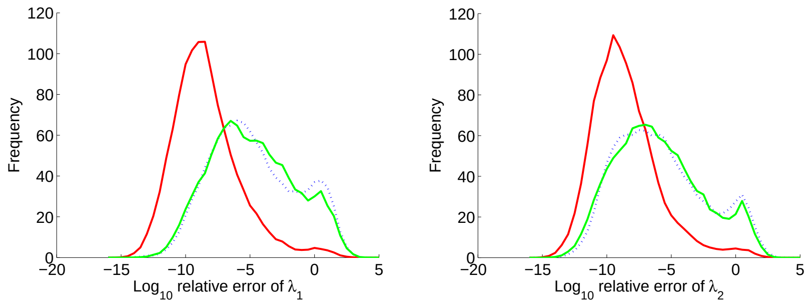

Fig. 3. The log_{10} relative errors of the radial distortion parameters λ_1 (Left) and λ_2 (Right) for 1000 runs of three different solvers. Original solver with the basis selection [4] (Red, darker), original solver without the basis selection (Blue, dotted) and our generated solver (Green, lighter) on the synthetic dataset without noise .

very similar results (log_{10} relative errors around $10^{-13} - 10^{-14}$) as this solver which uses further special technique for improving the numerical stability.

6.2 Nine Point Radial Distortion Problem

The minimal problem of simultaneous estimation of fundamental matrix and two different radial distortion parameters for two uncalibrated cameras and nine image point correspondences has been successfully solved in floating point arithmetic only recently [4]. This problem has 24 solutions and results in ten polynomial equations in ten unknowns. These equations can be simplified to four equations in four unknowns. The existing Gröbner basis solver [4] starts with these four equations and to obtain the action matrix first generates all monomial multiples of these initial equations up to total degree eight. This gives 497 equations. Using "fine tuning" authors reduce the number of used equations to 393 equations in 390 monomials. After the G-J of the corresponding 393×390 matrix all necessary polynomials for constructing the action matrix $M_{f_{31}}$ are obtained.

Our automatic procedure starts with simplified four polynomial equations in four unknowns. First, the generator also generates all monomial multiples of the initial polynomial equations up to total degree eight. In the reduction step, 318 out of these 497 polynomials are removed, resulting in 179 polynomials in 212 monomials for action variables f_{31} and also for λ_2. After the G-J elimination of the corresponding 179×212 matrix (in fact 179×203 matrix is sufficient) all necessary polynomial q_i are obtained and the action matrix $M_{f_{31}}$ (M_{λ_2}) is constructed.

We have compared our generated solver with the original solver proposed in [4] which uses special technique for improving the numerical stability based on changing the basis B and also with the "one elimination solver" (the same solver as in [4]) but without this basis selection.

The log_{10} relative errors of the two radial distortion parameters λ_1 and λ_2 for 1000 runs of these solvers, i.e. the solver [4] with the basis selection (Red, darker), the solver [4] without the basis selection (Blue, dotted) and our generated solver (Green, lighter), are shown in Figure 3.

	Original solver	Our generated solver
5pt relative pose problem [20]	1 elimination 10×20	1 elimination 10×20
6pt focal length problem [18, 2]	3 eliminations [18] $12 \times 33, 16 \times 33$ and 18×33 1 elimination [2] 34×50	1 elimination 31×46
6pt radial distortion problem [4]	1 elimination 320×363	1 elimination 238×290
8pt radial distortion problem [11]	3 eliminations 8×22,11×30, 36×50	1 elimination 32×48
9pt radial distortion problem [4]	1 elimination 393×390	1 elimination 179×203

Table 1. The comparison of the size of the elimination templates used in our generated solvers with the size of the elimination templates used in the original solvers.

The best results gives the original solver [4] with basis selection. A classical Gröbner basis solver (without this basis selection) gives very similar results as our automatically generated solver. Although these results are still very good (log_{10} relative errors around 10^{-6}) they can be further enhanced using the same technique for improving the numerical stability as it was used in [4].

6.3 Elimination template(s) size

We have compared the size of the elimination templates used in our generated solvers with the size of the elimination templates used in the original solvers for three other minimal problems, (i) five point relative pose problem [20], (ii) problem of estimating epipolar geometry and single distortion parameter for two uncalibrated cameras and eight point correspondences [11], and (iii) problem of estimating epipolar geometry and single distortion parameter for two calibrated cameras and six point correspondences [4]. The results for these three minimal problems together with the results for the two previously discussed problems are shown in Table 1. For all these problems we have obtained smaller or the same size elimination templates than those used in original solvers. Smaller templates in faster solvers.

6.4 Computation time

We have implemented the generator in MATLAB. Computation times demand on the problem. For several problems we have tested, the generator running time was from nine seconds to two minutes. Running times of the resulting automatically generated online solvers were in milliseconds.

7 Conclusion

We have proposed an automatic procedure for generating Gröbner basis solvers for an interesting problems which appear in computer vision and elsewhere. This automatic generator can be easily used even by non-experts to solve their own new problems. The input to the generator is a system of polynomial equations with a finite number of solutions and the output is the Matlab or C code which computes solutions to this system for concrete coefficients. We have demonstrated the functionality of our generator on several minimal problems. Our generator constructs efficient and numerically stable solvers which are comparable or outperform known manually constructed solvers in acceptable time. The automatic generator is available at http://cmp.felk.cvut.cz/minimal.

References

1. B. Buchberger Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal *PhD Thesis*, Mathematical Institute, University of Innsbruck, Austria, 1965.
2. M. Byröd, K. Josephson, and K. Aström. Improving numerical accuracy of gröbner basis polynomial equation solver. In International Conference on Computer Vision, 2007.
3. M. Byröd, K. Josephson, and K. Aström. Fast Optimal Three View Triangulation. ACCV 2007, (2) pp. 549-559, 2007.
4. M. Byröd, Z. Kukelova, K. Josephson, T. Pajdla, K. Åström, Fast and robust numerical solutions to minimal problems for cameras with radial distortion, CVPR 2008.
5. M. Bujnak, Z. Kukelova, T. Pajdla, A general solution to the P4P problem for camera with unknown focal length, CVPR 2008.
6. D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, 1992.
7. D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer-Verlag, 2005.
8. J.-C. Faugere. A new efficient algorithm for computing gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
9. C. Geyer, and H. Stewenius. A nine-point algorithm for estimating para-catadioptric fundamental matrices. *CVPR 2007*, Minneapolis, 2007
10. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.
11. Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. *CVPR 2007*, Minneapolis, 2007
12. Z. Kukelova and T. Pajdla. Two minimal problems for cameras with radial distortion. In *OMNIVIS 2007*, Rio de Janeiro, 2007
13. H. Li. A simple solution to the six-point two-view focal-length problem. *ECCV 2006*.
14. H. Li and R. Hartley. Five-point motion estimation made easy. *ICPR 2006*.
15. E. W. Mayr. Some complexity results for polynomial ideals. Journal of Complexity, vol. 13, n. 3, pp. 303–325, 1997
16. D. Nister. An efficient solution to the five-point relative pose. *IEEE PAMI*, 26(6):756–770, 2004.
17. H. Stewénius. *Gröbner basis methods for minimal problems in computer vision*. PhD thesis, Lund University, 2005.
18. H. Stewénius, D. Nister, F. Kahl, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. In *CVPR 2005*, pp. 789–794.
19. H. Stewénius, D. Nister, M. Oskarsson, and K. Astrom. Solutions to minimal generalized relative pose problems. *OMNIVIS 2005*.
20. H. Stewénius, C. Engels, and D. Nister. Recent developments on direct relative orientation. *ISPRS J. of Photogrammetry and Remote Sensing*, 60:284–294, 2006.
21. C. Traverso. Gröbner trace algorithms. In *ISSAC*, pages 125–138, 1988.