This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Efficient Solvers for Minimal Problems by Syzygy-based Reduction

Viktor Larsson Kalle Åström Magnus Oskarsson
Centre for Mathematical Sciences
Lund University
{viktorl,kalle,magnuso}@maths.lth.se *

Abstract

In this paper we study the problem of automatically generating polynomial solvers for minimal problems. The main contribution is a new method for finding small elimination templates by making use of the syzygies (i.e. the polynomial relations) that exist between the original equations. Using these syzygies we can essentially parameterize the set of possible elimination templates.

We evaluate our method on a wide variety of problems from geometric computer vision and show improvement compared to both handcrafted and automatically generated solvers. Furthermore we apply our method on two previously unsolved relative orientation problems.

1. Introduction

One of the success stories of computer vision is using robust estimation schemes such as RANSAC [13] in multiple view geometry estimation. With a hypothesis and test framework, one can efficiently handle large amounts of outliers in the measured data. A key element in such a framework is the ability to model the problem using a small or minimal subset of data points – a so-called *minimal problem*. A classic example is the 5-point algorithm for estimating the relative pose between two cameras, given only image point measurements, [25, 12, 41, 47]. The underlying problems in multiple view geometry naturally lead to systems of polynomial equations in several variables. In order to devise tractable algorithms, robust and fast solvers of polynomial equations are needed. The predominant way to solve minimal problems in computer vision is using methods based on Gröbner bases. The reason is that this often leads to fast and numerically stable algorithms. These methods were popularized in computer vision by Stewénius [46]. The earliest examples were methods that were very much handcrafted [24, 48], but since then much effort has been put into making the process of constructing the solvers more automatic. One of the main challenges is ways of automatically constructing the so-called *elimination template*, which is the main focus of this work.

Our contributions in this paper are:

(i) A non-iterative method for automatically finding the monomial expansion of the initial system of equations.

(ii) A non-iterative reduction step, that often gives a much more compact representation of the expanding set.

(iii) An efficient implementation of these ideas which produces stand-alone code for solving arbitrary instances of the problem.

(iv) Solvers for two previously unsolved minimal cases, based on our developed system.

1.1. Related work

The most closely related work is by Kukelova *et al*. [33], where the authors presented an automatic method for generating polynomial solvers. The automatic generator allows the user to specify a set of polynomial equations and then automatically generates stand-alone code for solving arbitrary problem instances. This automatic generator has been widely adopted in the computer vision community and it has been used to solve several problems in geometric computer vision (see *e.g*. Table 1 and the references therein.) The solvers generated using [33] are built on the action matrix method that reduces the polynomial equation system to an eigenvalue problem. Their automatic generator works by first computing a Gröbner basis in Macaulay2 [15] for a random problem instance. The Gröbner basis gives information on the number of solutions and provides a basis for the quotient space. The elimination template needed for computing the action matrix is then found by an iterative search process that alternates between expanding the equation system and performing Gaussian elimination. Once sufficiently many equations have been generated a pruning step is used to remove any unnecessary equations. The Gröbner basis computations and the Gaussian eliminations are performed in some prime field $\mathbb{Z}_p$ to avoid numerical problems. The main drawback of the approach in [33] is that as the number of variables and equations grows the iterative search and pruning step can quickly become intractable. In this work we propose a new method for finding the elimination template in place of the iterative search used

*This work has been financed by ELLIIT, MAPCI, eSSENCE and the Swedish Research Council (grant no. 2012-4213).

in [33]. We show on a large number of examples that our approach almost always produces smaller elimination templates and faster polynomial solvers.

While our focus has been on constructing smaller templates there has been a number of works that have addressed the problem of making solvers more numerically stable [9, 26] and faster [6, 32]. It is possible that these methods could be applied in conjunction with our method, but this is left as further work.

2. Background

In this section we remind the reader of some of the basic facts and definitions from algebraic geometry that we will use throughout the paper. For a more thorough introduction see [10].

Let $X = (x_1, \dots, x_n)$ be a number of variables, and $\mathbb{K}$ be some field. Then the set of all multivariate polynomials (with n variables) over $\mathbb{K}$ is denoted $\mathbb{K}[X]$, and this set with their natural operations forms a ring. In this paper we will only consider the case when $\mathbb{K} = \mathbb{C}$ or $\mathbb{K} = \mathbb{Z}_p$ for some prime number p. For a set of polynomials $F = \{f_i\}_{i=0}^m$ the set of shared zeros, *i.e.*

$$\mathcal{V}(F) = \{\boldsymbol{x} \in \mathbb{K}^n \mid f_i(\boldsymbol{x}) = 0,\ i = 1, 2, \dots, m\} \quad (1)$$

is called an *affine variety* and the set of all polynomial combinations of the elements in F, *i.e.*

$$I(F) = \{p \in \mathbb{K}[X] \mid p = \textstyle\sum_i h_i f_i, \quad h_i \in \mathbb{K}[X]\}, \quad (2)$$

forms an ideal in the polynomial ring $\mathbb{K}[X]$. When it is clear from the context which polynomials are meant we will omit F and simply write $\mathcal{V}$ and I.

Similar to the univariate case there exists a division algorithm for multivariate polynomials. Unfortunately the remainder depends on the order in which the polynomials are listed. Fortunately for any ideal I there exist special sets of generators $G = \{g_i\}_{i=0}^{\ell}$ called *Gröbner bases*, such that the remainder after division with G is uniquely defined, regardless of how the individual g_i are listed. This allows us to define the *normal form* of a polynomial $p \in \mathbb{K}[X]$ with respect to G as the unique remainder after division with G. This is denoted $\overline{p}^G$. Note that $p \in I$ if and only if $\overline{p}^G = 0$. For a Gröbner basis G the *normal set* is the set of all monomials not divisible by any element in G. It is easy to see that the normal form for any polynomial lies in the linear span of the normal set.

Another object of interest is the quotient space $\mathbb{K}[X]/I$, which is the set of equivalence classes over I (*i.e.* two elements are equivalent if their difference lies in I). If an affine variety $\mathcal{V}$ is zero dimensional (*i.e.* there are finitely many solutions) then the corresponding quotient space $\mathbb{K}[X]/I$ will be a finite dimensional vector space. For any Gröbner basis G of I we have that the normal set forms a vector space basis of the quotient space $\mathbb{K}[X]/I$.

2.1. The action matrix method

Next we give a brief overview of the action matrix method for solving polynomial systems. The main idea is to reduce the problem to an eigenvalue problem for which there exist good numerical methods. For a more thorough review of the action matrix method and how it has been applied in computer vision we recommend [37], [33] and [9].

Consider the operator $T_\alpha : \mathbb{K}[X]/I \to \mathbb{K}[X]/I$ which multiplies a polynomial with the fix monomial[1] $\alpha \in \mathbb{K}[X]$, *i.e.*

$$T_\alpha\left[p(\boldsymbol{x})\right] = \left[\alpha(\boldsymbol{x})p(\boldsymbol{x})\right], \quad p \in \mathbb{K}[X]. \quad (3)$$

The operator T_α is a linear map and thus if we choose a (linear) basis $\boldsymbol{b}$ for the quotient space we can express the operator with a matrix M, *i.e.*

$$[\alpha b_i] = \left[\textstyle\sum_j m_{ij} b_j\right] \quad \Leftrightarrow \quad [\alpha \boldsymbol{b}] = [M\boldsymbol{b}]\,. \quad (4)$$

For each $\boldsymbol{x} \in \mathcal{V}$ we must have $M\boldsymbol{b}(\boldsymbol{x}) = \alpha(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x})$. Thus if we evaluate α and $\boldsymbol{b}$ at the solutions we get eigenvalues and eigenvectors for the matrix M. So if we can find the action matrix M we can recover the solutions by solving an eigenvalue problem, and hence we have reduced the solving of the system of polynomial equations to a linear algebra problem.

The monomials $r_i = \alpha b_i$ are called the *reducible* monomials. If a Gröbner basis G is known and $\boldsymbol{b}$ is chosen as the normal set we can recover the action matrix by reducing the reducible monomials with the Gröbner basis, *i.e.* $\overline{r_i}^G = \sum_j m_{ij} b_j$. Due to roundoff error it is usually not possible to directly compute a Gröbner basis for a polynomial system corresponding to a real problem instance. Instead an alternative approach is taken to recover the action matrix. The idea is based on the observation that each

$$r_i - \textstyle\sum_j m_{ij} b_j \in I \quad (5)$$

and thus there exist some polynomials $h_{ij} \in \mathbb{K}[X]$ such that

$$r_i - \textstyle\sum_j m_{ij} b_j = \sum_j h_{ij} f_j. \quad (6)$$

To find the action matrix M the original set of equations $\{f_j(\boldsymbol{x}) = 0\}_{j=0}^m$ is expanded by adding new equations formed by multiplying each f_j by some monomials. If we have multiplied by sufficiently many monomials (*i.e.* all monomials in the unknown h_{ij}) we can express each polynomial (5) linearly in the expanded set of equations.

To do this in practice (see *e.g.* [26] for details) we write the expanded set of equations as $C\boldsymbol{X} = 0$, where the matrix C is called the *elimination template* and $\boldsymbol{X}$ is a vector of all the monomials occurring in the equations. By reordering

[1] For simplicity we take α as a monomial here but the theory holds for a general polynomial as well.

the monomials we can rewrite this as

$$C\boldsymbol{X} = \begin{bmatrix} C_E & C_R & C_B \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_E \\ \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{pmatrix} = 0, \tag{7}$$

where we have grouped the monomials into excessive monomials $\boldsymbol{x}_E$, reducible monomials $\boldsymbol{x}_R$ and basis monomials $\boldsymbol{x}_B$. The excessive monomials are simply the monomials which are neither reducible nor basis monomials. Now since we know $\boldsymbol{x}_R = M\boldsymbol{x}_B$ we simply perform Gaussian elimination on (7) and get the following form on the lower part of (7)

$$\begin{bmatrix} 0 & I & -M \end{bmatrix} \begin{pmatrix} \boldsymbol{x}_E \\ \boldsymbol{x}_R \\ \boldsymbol{x}_B \end{pmatrix} = 0 \tag{8}$$

from which we can extract the action matrix M.[2]

3. Finding elimination templates in $\mathbb{Z}_p[X]$

Now we will present our proposed approach for finding elimination templates for a given problem. Similarly to [33] we start by generating instances of the problem where the equations have coefficients in some prime field $\mathbb{Z}_p$. Due to the exact arithmetic available in these fields we can easily compute a Gröbner basis $G = \{g_k\}_{k=0}^{\ell}$ for the ideal. From the Gröbner basis we find a linear basis $\boldsymbol{b}$ for the quotient space by forming the normal set. As described in Section 2.1 we can then find the action matrix by reducing each of the reducible monomials,

$$\overline{r_i}^{G} = \textstyle\sum_j m_{ij} b_j. \tag{9}$$

Note that this only gives the action matrix for this particular integer instance, which we are not particularly interested in.

However, by keeping track of how the Gröbner basis elements are formed we can express each g_k in the generators f_j, *i.e.*[3]

$$g_k = \textstyle\sum_k c_{kj} f_j, \tag{10}$$

where the coefficients $c_{kj} \in \mathbb{Z}_p[X]$ are polynomials. Since each $r_i - \sum m_{ij} b_j \in I$ we can then find polynomials $h_{ij} \in \mathbb{Z}_p[X]$ such that

$$r_i - \textstyle\sum_j m_{ij} b_j = \sum_k a_{ik} g_k = \sum_k a_{ik} (\sum_j c_{kj} f_j) = \sum_j h_{ij} f_j \tag{11}$$

by dividing by $\{g_k\}$ and substituting each g_k with (10). While the polynomials h_{ij} are specific to this instance, they will typically have the same structure and only the coefficients (*i.e.* numbers) will be different for different problem instances. Thus to form our elimination template we expand our equation set by multiplying each equation $f_j(\boldsymbol{x}) = 0$ by the monomials in h_{ij} for each $i = 1, 2, \ldots, n_R$.

Typically the elimination templates found using this method will be quite large. In the next section we will show how we can find simpler polynomials h_{ij} that give more compact elimination templates.

3.1. Reducing the expansion

In the previous section we showed how to obtain polynomials $\boldsymbol{h}_i = (h_{i1}, \ldots, h_{in}) \in \mathbb{Z}_p[X]^n$ such that we could represent the polynomials needed for forming the action matrix, *i.e.*

$$p_i = r_i - \textstyle\sum_j m_{ij} b_j = \sum_j h_{ij} f_j. \tag{12}$$

These representations of p_i in $\{f_j\}$ are however not unique, since for any $\boldsymbol{s} = (s_1, \ldots, s_n) \in \mathbb{Z}_p[X]^n$ which satisfies

$$\textstyle\sum_j s_j f_j = 0, \tag{13}$$

we also have

$$p_i = \textstyle\sum_j (h_{ij} + s_j) f_j. \tag{14}$$

Consider the set of all $\boldsymbol{s} \in \mathbb{Z}_p[X]^n$ which satisfy this, *i.e.*

$$\mathcal{M} = \{\boldsymbol{s} \in \mathbb{Z}_p[X]^n \mid \textstyle\sum_j s_j f_j = 0\}. \tag{15}$$

This set forms a sub-module in $\mathbb{Z}_p[X]^n$ and is called the *first syzygy module* of $(f_1, \ldots, f_n)$ [11]. It captures all polynomial relations between the original equations $\{f_j(\boldsymbol{x}) = 0\}_{i=0}^{m}$ and it is clear that any representation of p_i in $\{f_j\}$ can be written as $\boldsymbol{h}_i + \boldsymbol{s}$ for some element $\boldsymbol{s} \in \mathcal{M}$.

Finding the $\boldsymbol{s} \in \mathcal{M}$ which yields that smallest template or the best numerics is a difficult problem. Instead we now present a simple heuristic that usually works well in practice. We start by computing a Gröbner basis $G_{\mathcal{M}}$ for the module $\mathcal{M}$. This is done on the prime field problem instance using Macaulay2. The Gröbner basis depends on the monomial order chosen for $\mathcal{M}$ and in this work we have used the Term-Over-Position-GRevLex which is the default order for modules in Macaulay2. Next to find a simpler representation of each p_i we compute the normal form w.r.t. $G_{\mathcal{M}}$ for each $\boldsymbol{h}_i$, *i.e.*

$$\tilde{\boldsymbol{h}}_i = \overline{\boldsymbol{h}_i}^{G_{\mathcal{M}}}. \tag{16}$$

This can be thought of as removing as much as possible of $\mathcal{M}$ from the representation. The following proposition shows that the new representations $\tilde{\boldsymbol{h}}_i$ are minimal in the sense that the maximum degree of the monomials is minimized. Note that there can be multiple representations with minimal degree and this approach only finds one of them.

Proposition 1. *If $\mathcal{M}$ is defined as above, $p = \sum_j h_j f_j$ and $G_{\mathcal{M}}$ is a Gröbner basis for $\mathcal{M}$ with respect to TOP-GRevLex (or any other degree first monomial order), then*

[2] Note that in practice some reducible monomials might be available from the basis monomials

[3] In Macaulay2 this can be accomplished using `ChangeMatrix`.

$\tilde{\boldsymbol{h}} = \overline{\boldsymbol{h}}^{G_{\mathcal{M}}}$ *satisfies*

$$\max_i \deg \tilde{h}_i \leq \max_i \deg \left(\tilde{h}_i + s_i \right) \quad \forall \boldsymbol{s} \in \mathcal{M} \tag{17}$$

i.e. *the representation* $\tilde{\boldsymbol{h}}$ *of* p *is of minimal degree.*

Proof. Assume that there exist some $\boldsymbol{s} \in \mathcal{M}$ such that

$$\max_i \deg \left(\tilde{h}_i + s_i \right) < \max_i \deg \tilde{h}_i. \tag{18}$$

Then since the monomial order is degree first we must have $\mathrm{LM}(\tilde{\boldsymbol{h}} + \boldsymbol{s}) < \mathrm{LM}(\tilde{\boldsymbol{h}})$ which implies $\mathrm{LM}(\tilde{\boldsymbol{h}}) = \mathrm{LM}(\boldsymbol{s})$. But then $\tilde{\boldsymbol{h}}$ is divisible by an element in $\mathcal{M}$ which is a contradiction. □

3.2. Implementation details

We have written an automatic generator in MATLAB, which uses the technique described above to find and reduce the elimination templates. The generator is similar to that of Kukelova *et al*. [33] in that it only requires the user to specify the problem equations and then generates stand-alone MATLAB code that can be used to solve arbitrary problem instances. The elimination template generation and reduction are performed in just a few lines of Macaulay2 [15]. The automatic generator allows the user to easily experiment with different parameters such as which action monomial to use and the monomial ordering for the ideal, which can greatly affect the size of the polynomial solver. The generator can also automatically identify and exploit any variable aligned symmetries as described in [35]. In the implementation we do not perform any refinement on the elimination templates (except for the reduction step described in Section 3.1). It is possible that by using template optimization techniques such as those described in [26, 39] the results could be further improved. We have made the code for generating solvers publicly available. For most of the problems that we have tried, the solver generation time is quite small. The median running time for all the problems described in Table 1 of our automatic generator (executed on a standard desktop computer) is 5.7s.

4. Evaluation of the reduction step

In this section we evaluate the reduction step proposed in Section 3.1. Note that while the reduction does not guarantee that the template will be smaller we will show that this is often the case in practice.

To perform the evaluation we applied the automatic generator to a wide variety of minimal problems from the computer vision literature. Table 1 shows both the original template sizes as reported by the authors and the resulting templates from our proposed automatic generator. We have marked the templates with the smallest number of elements in bold. It can be seen that in general the reduction step produces a smaller template. More interestingly is perhaps that the reduced template is often smaller than the template in the original paper. Note that many of the papers used the automatic generator from Kukelova *et al*. [33] (indicated with (*) in the table). For many of the tested problems we get a significant decrease in template size, and for some a very large decrease. For instance for the problem of estimating relative pose with a known rotation direction we go from a template of size 411×489, [45] to a template of size 40×57. If we assume that the time complexity of the solver is quadratic in the number of rows and linear in the number of columns this corresponds to a speed-up factor of 900. The problem contains a symmetry that was not used by the original authors and we have used the method in [35] to remove it. (Here the smaller template doesn't contain all the variables in the basis for the quotient space, but the remaining variables can be extracted linearly from the initial equations.)

5. Numerical accuracy of the solvers

The focus of the work in this paper has been on generating fast solvers in an automatic manner, and not on numerical accuracy. However, for the proposed solvers to be usable we need them to behave in an acceptable way in terms of accuracy as well. While a smaller template typically yields faster runtime, there is not always a clear correlation between accuracy and template size. In [39] it was reported that on a number of examples, smaller templates yielded better numerical accuracy as well. All of the generated solvers in Table 1 produce $\log_{10}$-residuals with a mode below -4.8 and most solvers have significantly better accuracy. The median of all of the $\log_{10}$-error modes is -10.9.

We will in this section give comparisons between our solvers and the original ones, for some specific problems. In terms of the underlying computer vision problem, there is often some meaningful statistical error that one can evaluate, *e.g*. the reprojection errors, but since our main contribution in this paper is an automatic way of constructing solvers to systems of polynomials equations, we have opted to evaluate the actual equation residuals instead.

We compare on three different problems, where the original solvers were publicly available, namely *image stitching with unknown focal length and radial distortion* [8, 39] , the *optimal PnP*-method of Hesch *et al*. [21] and the *optimal PnP*-method of Zheng *et al*. [54]. In Figure 1 the resulting error residual histograms are shown, for 5,000 runs of the solvers, with random input. The figure shows that for these problems we get similar accuracy as the original solvers while having smaller elimination templates. For the image stitching we have used the original solver presented in [8]. The smaller original template presented in Table 1 is from the paper of Naroditsky *et al*., but they reported almost identical numerics as the original solver [39].

Problem	Original		Proposed generator	
	Author	template size	no reduction step	with reduction step
Rel. pose 5pt	Stewénius *et al*. [47]	$\mathbf{10 \times 20}$	$\mathbf{10 \times 20}$	$\mathbf{10 \times 20}$
Rel. pose 8pt one-sided rad. dist.	Kuang *et al*. [30]	12×24	$\mathbf{11 \times 20}$	$\mathbf{11 \times 20}$
TDOA offset rank 2, 7,4 pts	Kuang *et al*. [28]	$\mathbf{20 \times 15}$	$\mathbf{20 \times 15}$	$\mathbf{20 \times 15}$
Rel. pose + one focal 6pt	Bujnak *et al*. [4] (*)	$\mathbf{21 \times 30}$	$\mathbf{21 \times 30}$	$\mathbf{21 \times 30}$
P3.5P + focal	Wu [52]	$\mathbf{20 \times 43}$	24×45	20×44
Rel. pose + const. focal 6pt	Kukelova *et al*. [33] (*)	$\mathbf{31 \times 46}$	31×50	31×50
Rel. pose + rad. dist. 8pt	Kukelova *et al*. [33] (*)	32×48	$\mathbf{31 \times 49}$	32×50
Rel. pose 6pt ones-sided rad. dist.	Kuang *et al*. [30]	48×70	$\mathbf{34 \times 60}$	$\mathbf{34 \times 60}$
TDOA offset rank 2, 5,6 pts	Kuang *et al*. [28]	105×83	105×83	$\mathbf{40 \times 42}$
Rolling shutter pose	Saurer *et al*. [44] (*)	48×56	50×55	$\mathbf{47 \times 55}$
Generalized P4P + scale	Ventura *et al*. [51] (*)	48×56	50×55	$\mathbf{47 \times 55}$
Stitching + const. focal + rad. dist. 3pt	Naroditsky *et al*. [39]	54×77	96×108	$\mathbf{48 \times 66}$
TDOA offset rank 3, 9,5 pts	Kuang *et al*. [28]	$\mathbf{70 \times 31}$	$\mathbf{70 \times 31}$	$\mathbf{70 \times 31}$
TDOA offset rank 3, 7,6 pts	Kuang *et al*. [28]	255×157	255×157	$\mathbf{75 \times 57}$
Generalized rel. pose 6pt	Stewénius *et al*. [48]	$\mathbf{60 \times 120}^{\ddagger}$	135×164	99×163
Optimal PnP	Hesch *et al*. [21]	120×120	93×116	$\mathbf{88 \times 115}$
Triangulation from satellite im.	Zheng *et al*. [53] (*)	93×120	93×116	$\mathbf{88 \times 115}$
Optimal PnP (Cayley)	Nakano [38] (*)	124×164	186×161	$\mathbf{118 \times 158}$
P4P + focal + rad. dist.	Bujnak *et al*. [5] (*)	136×152	$\mathbf{140 \times 144}$	140×156
Rel. pose + rad. dist. 6pt	Kukelova *et al*. [33] (*)	238×290	223×290	$\mathbf{154 \times 210}$
Rel. pose + 2 rad. dist. 9pt	Kukelova *et al*. [33] (*)	179×203	355×298	$\mathbf{165 \times 200}$
Rel. pose 7pt one-sided focal + rad. dist.	Kuang *et al*. [30]	200×231	249×214	$\mathbf{185 \times 204}$
Weak PnP	Larsson *et al*. [35]	234×276	$568 \times 498^{\dagger}$	$\mathbf{189 \times 232}^{\dagger}$
Weak PnP (2x2 sym)	Larsson *et al*. [35]	104×90	$83 \times 90^{\dagger}$	$\mathbf{49 \times 59}^{\dagger}$
Rolling shutter R6P	Albl *et al*. [2] (*)	$\mathbf{196 \times 216}$	222×230	204×224
Optimal pose w dir 4pt	Svärm *et al*. [49]	280×252	371×351	$\mathbf{203 \times 239}$
Rel. pose w dir. 3pt	Saurer *et al*. [45] (*)	411×489	287×324	$\mathbf{210 \times 255}$
Rel. pose w dir. 3pt (using sym.)	-	-	$94 \times 111^{\dagger}$	$\mathbf{40 \times 57}^{\dagger}$
Abs. pose quivers	Kuang *et al*. [27]	372×386	420×406	$\mathbf{217 \times 253}$
L_2 3 view triangulation (Relaxed)	Kukelova *et al*. [34] (*)	274×305	399×384	$\mathbf{239 \times 290}$
Rel. pose w angle 4pt	Li *et al*. [36] (*)	$\mathbf{270 \times 290}$	280×304	266×329
Refractive P5P	Haner *et al*. [16]	280×399	410×480	$\mathbf{240 \times 324}$
TDOA offset rank 3, 6,8 pts	Kuang *et al*. [28]	1359×754	1359×754	$\mathbf{356 \times 345}$
Optimal PnP	Zheng *et al*. [54] (*)	575×656	812×704	$\mathbf{521 \times 601}$
Optimal PnP (using sym.)	Zheng *et al*. [54] (*)	348×376	$484 \times 408^{\dagger}$	$\mathbf{302 \times 342}^{\dagger}$
Optimal pose w dir 3pt	Svärm *et al*. [49]	$1,260 \times 1,278$	918×726	$\mathbf{544 \times 592}$
Optimal PnP (quaternion)	Nakano [38] (*)	630×710	958×693	$\mathbf{604 \times 684}$
Refractive P6P + focal	Haner *et al*. [16]	648×917	$2,196 \times 1,913^{\dagger}$	$\mathbf{636 \times 851}^{\dagger}$
Rel. pose + const. focal + rad. dist. 7pt	Jiang *et al*. [23]	$886 \times 1,011$	$1,393 \times 1,237$	$\mathbf{581 \times 862}$
Dual-Receiver TDOA 5pt	Burgess *et al*. [7]	$2,625 \times 2,352$	$850 \times 1,167$	$\mathbf{455 \times 768}$
Optimal PnP (rot. matrix)	Nakano [38] (*)	$1,936 \times 1,976$	$1,698 \times 1,153$	$\mathbf{1,102 \times 1,135}$
L_2 3 view triangulation	Kukelova *et al*. [34] (*)	$1,866 \times 1,975$	$2,647 \times 2,584$	$\mathbf{1,759 \times 2,013}$

(*) Original template constructed using [33]. If several elimination templates are used, the largest of these templates is reported.
†: The problem contains variable-aligned symmetries [3, 31, 35] that was automatically found and removed by our generator.
‡: The original template doesn't generate the full Gröbner basis, and some additional operations on the template are performed.

Table 1. Comparison of elimination template sizes for some common minimal problems in computer vision. The template with the fewest elements, for each problem, is shown in bold.

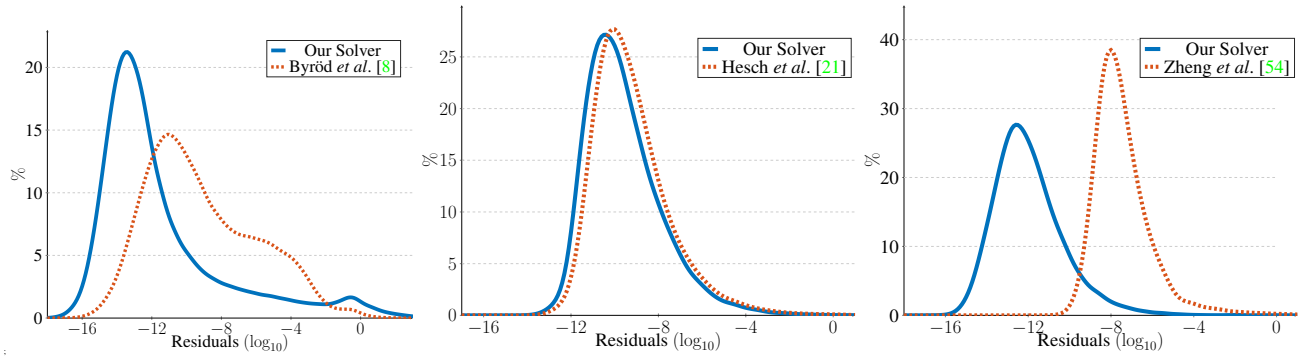

Figure 1. Kernel smoothed histograms of residual errors for 5,000 runs – of from left to right – *image stitching with unknown focal length and radial distortion* [8, 39] , the *optimal PnP*-method of Hesch *et al*. [21] and the *optimal PnP*-method of Zheng *et al*. [54].

6. Three views with known rotation axis

In order to test our framework on a novel case, we turn our attention to the problem of pose estimation in three views when the rotation axis is known. Known rotation axis can occur in practice when there is additional information from sensors such as accelerometers which are common in modern cell phones. The problem is minimal when we observe one point and two lines. The problem has previously been studied for two views, where the minimal case is three points [14, 40, 50], and for two generalized cameras, where the minimal case is four points [20, 50]. Let the cameras be chosen as

$$P_1 = \begin{bmatrix} I & \mathbf{0} \end{bmatrix}, P_2 = \begin{bmatrix} R(\theta_2, \boldsymbol{v}) & \boldsymbol{t}_2 \end{bmatrix}, P_3 = \begin{bmatrix} R(\theta_3, \boldsymbol{v}) & \boldsymbol{t}_3 \end{bmatrix} \tag{19}$$

where $R(\theta, \boldsymbol{v})$ denotes rotation of θ radians around the rotation axis $\boldsymbol{v}$. Since the rotation axis is assumed to be known we can w.l.o.g. assume that $\boldsymbol{v} = \begin{pmatrix} 0, 1, 0 \end{pmatrix}^T$ by rotating the image coordinate systems. The two line constraints can be formulated as

$$\text{rank} \begin{pmatrix} \begin{bmatrix} P_1^T \ell_{k1} & P_2^T \ell_{k2} & P_3^T \ell_{k3} \end{bmatrix} \end{pmatrix} = 2, \quad k = 1, 2 \tag{20}$$

where ℓ_{ij} denotes line i observed in image j. The rank constraint can be encoded as a polynomial constraint by requiring each of the 3×3 submatrices to have zero determinant. Finally the single point correspondence gives us three sets of equations

$$\lambda_k \boldsymbol{x}_k = P_k \boldsymbol{X}, \quad k = 1, 2, 3 \tag{21}$$

or equivalently by eliminating λ_k

$$\boldsymbol{x}_k \times P_k \boldsymbol{X} = \mathbf{0}, \quad k = 1, 2, 3 \tag{22}$$

where $\boldsymbol{x}_k$ denotes the image point in image k.

6.1. Building an efficient two-step solver

To build an efficient solver for this problem we start by noting that the top 3×3 submatrix in (20) only contains the rotation matrices. The determinant constraints are

$$\det \begin{pmatrix} \begin{bmatrix} \ell_{1k} & R_2^T \ell_{2k} & R_3^T \ell_{3k} \end{bmatrix} \end{pmatrix} = 0, \quad k = 1, 2 \tag{23}$$

which gives us two quadratic equations in the elements of R_2 and R_3. Since there are only two parameters in the rotations we can use these two equations to solve for the rotations independently from the rest of the variables.

Next we note that the constraint in (23) is invariant to the scale of the rotation matrices. We exploit this by parameterizing the rotations by non-unit quaternions (*i.e*. the Cayley transform), $\boldsymbol{q}_2 = \begin{pmatrix} 1, & s_2 \boldsymbol{v}^T \end{pmatrix}^T$, $\boldsymbol{q}_3 = \begin{pmatrix} 1, & s_3 \boldsymbol{v}^T \end{pmatrix}^T$. This parameterization gives us two quartic equations in the unknowns s_2 and s_3. Fixing the first element of the quaternion introduces a degeneracy for any 180-degree rotation.

Using the proposed automatic generator we construct a solver for this system. The resulting template size is 12×20 and the system has 8 solutions. The solutions include two false solutions $s_2 = s_3 = \pm i$ that were introduced by the non-unit quaternion parameterization. These can easily be discarded, and from the true solutions we can recover the correct rotations by rescaling each quaternion to unit length.

When the rotations are known we can use them to recover the translations $\boldsymbol{t}_2$ and $\boldsymbol{t}_3$. Using the point correspondence we can parameterize the two translations using the depths as

$$\boldsymbol{t}_k = \lambda_k \boldsymbol{x}_k - R_k \boldsymbol{X}, \quad k = 2, 3. \tag{24}$$

Since $P_1 = [I \ \mathbf{0}]$ we can select the scale such that $\boldsymbol{X} = \boldsymbol{x}_1$. Inserting (24) into the line constraints gives linear equations in the unknown λ_2 and λ_3, which allows us to solve for the translations. We evaluated the performance of the two-step solver on 10,000 random synthetic instances. Figure 2 shows the residuals for the rotation estimation and the distance from the recovered pose to the ground truth. In average solving for the rotations and finding all translations took less than one millisecond per instance. Figure 3 shows an example where we have used the minimal solver in a RANSAC framework to estimate the relative pose of three

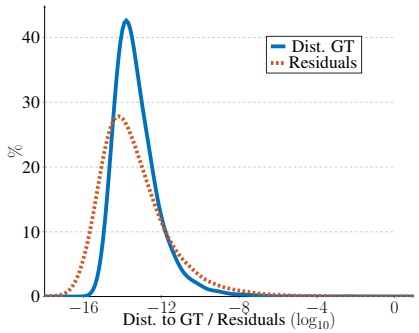

Figure 2. Results for 10,000 random synthetic instances for the rotation estimation in Section 6.1. The figure shows kernel smoothed distributions of both the residuals and the distance to the ground truth pose. The distance is defined as the maximum of $\|R_2 - R_2^{GT}\|_F$, $\|R_3 - R_3^{GT}\|_F$, $\|\boldsymbol{t}_2 - \boldsymbol{t}_2^{GT}\|_2$ and $\|\boldsymbol{t}_3 - \boldsymbol{t}_3^{GT}\|_2$.

cameras. The 3D structure is found using DLT [19]. Note that this is the result without any bundle adjustment.

6.2. Importance of good parameterization

In the previous section we developed an efficient solver by choosing a clever parameterization of the problem. Finding a good parameterization is often one of the main difficulties when building polynomial solvers for minimal problems. Knowing which parameterization will yield good solvers is non-trivial and usually a trial and error approach is taken. Automatic tools such as the generator proposed in this paper greatly speeds up this process and allows for much faster prototyping.

To illustrate the importance of good parameterization we show three alternative parameterizations for this problem.

1. First we fix the scale by setting $\boldsymbol{X} = \boldsymbol{x}_1$ and directly parameterize the two translations $\boldsymbol{t}_2 = \begin{pmatrix} t_{21}, t_{22}, t_{23} \end{pmatrix}^T$ and $\boldsymbol{t}_3 = \begin{pmatrix} t_{31}, t_{32}, t_{33} \end{pmatrix}^T$. The rotations are represented using unit quaternions, $\boldsymbol{q}_2 = (s_{21}, s_{22}\boldsymbol{v}^T)^T$ and $\boldsymbol{q}_3 = (s_{31}, s_{32}\boldsymbol{v}^T)^T$ with the additional constraints $\|\boldsymbol{q}_2\|^2 = \|\boldsymbol{q}_3\|^2 = 1$. This system has 24 solutions, however there exist two independent two-fold sign symmetries and by removing these we get the desired 6 solutions.

2. Next we remove the translations by parameterizing the depths λ_k in the 2nd and 3rd image, *i.e.* $\boldsymbol{t}_k = \lambda_k \boldsymbol{x}_k - R_k \boldsymbol{x}_1$. This reduces the number of unknowns to 6.

3. Finally we tried using unit-quaternions when solving for the rotations using only the line constraints. The parameterization avoids the two false solutions introduced by the non-unit quaternion parameterization. However there still remains a two-fold symmetry.

The different parameterizations are summarized in Table 2 which also shows the template sizes and average runtimes for the solvers. Note that there is about a three orders of magnitude difference in runtime between the fastest and slowest solvers, which shows the need of finding a good parameterization.

7. Nine lines in three views

In order to test the boundaries of our method, we have looked at a challenging unsolved minimal case. The classical problem of minimally estimating the geometry of three projective views of lines is an inherently difficult problem, [19, p. 413] and probably also not well numerically conditioned. Whereas algorithms for projective reconstruction from points in three views have existed for a long time, [22, 43], there exists no practical method for the corresponding minimal problem using lines. The problem is minimal with either six points or nine lines. There are also algorithms for different minimal combinations of lines and points, cf. [42]. Linear algorithms have been developed for over-constrained solutions of at least 13 lines [17], and for combinations of lines and points [18], and there exist non-linear methods for the over-constrained cases of 10-12 lines [29]. We assume that we have three unknown uncalibrated cameras, viewing nine unknown lines in space. Each camera has eleven parameters, and each line has four degrees of freedom. In addition to this, we can only determine a solution up to a global projective coordinate system, with 15 parameters. Each viewed line in each camera gives two constraints on our parameters. Since $3 \cdot 11 + 4 \cdot 9 - 15 = 2 \cdot 3 \cdot 9$, this gives a minimal system. A major reason that the line case is much more difficult than the point case, is that the projective coordinate system can be efficiently parameterized with five points. With lines we use fewer lines, and we need to be very careful in order to avoid specialized situations, *e.g.* lines intersecting. We have experimented with a large number of parameterizations of our problem, and in the end we found that the following gave the most tractable solution. First of all we make coordinate changes in the images so that the first two lines in each image are represented by $\ell_1 = (1, 0, 0)^T$ and $\ell_2 = (0, 1, 0)^T$. This corresponds to the lines $x = 0$ and $y = 0$ respectively. We then make a projective coordinate change so that the first camera is given by $P_1 = [I \ \mathbf{0}]$. This fixates 11 of the 15 degrees of freedom, and also gives that the first 3D-line must lie on the plane $\boldsymbol{\Pi}_1 = P_1^T \ell_1 = (1, 0, 0, 0)^T$ and the second line on the plane $\boldsymbol{\Pi}_2 = P_1^T \ell_1 = (0, 1, 0, 0)^T$. This in turn fixates the first two lines up to two parameters each. We can now fixate the final four degrees of freedom of our coordinate system by specifying two additional planes ($\boldsymbol{\Pi}_1'$ and $\boldsymbol{\Pi}_2'$) that the first two lines should lie on. (Assuming that a 3D-line is represented by two points $\boldsymbol{X}$ and $\boldsymbol{X}'$ this places two linear constraints on the coordinate change homography H, $(H\boldsymbol{X})^T\boldsymbol{\Pi}' = 0$ and $(H\boldsymbol{X}')^T\boldsymbol{\Pi}' = 0$ for each line). Choosing $\boldsymbol{\Pi}_1' = (0, 1, 0, -1)^T$ and $\boldsymbol{\Pi}_2' = (0, 0, 1, -1)^T$ gives

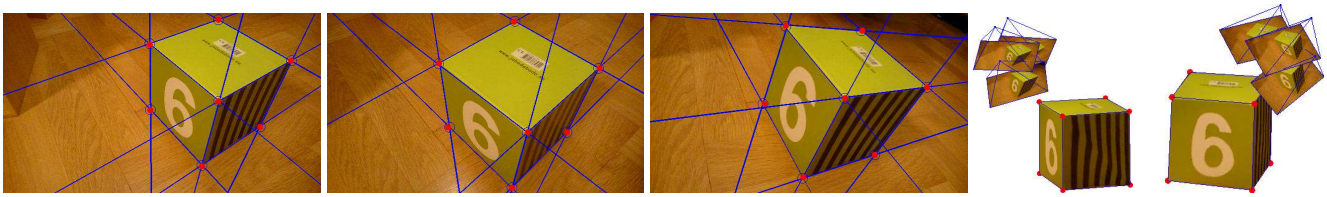

Figure 3. Example of three-view relative pose estimation with known rotation axis from one point and two lines. The image points and lines are shown in blue. The 3D-point reprojections are shown in red. The rightmost two images show the recovered 3D structure and poses.

Note	Unknowns	Additional equations	Template size	Runtime
Fixed scale by depth.	$s_{21}, s_{22}, s_{31}, s_{32}, \boldsymbol{t}_2, \boldsymbol{t}_3$	$\boldsymbol{X} = \boldsymbol{x}_1,\ \|\boldsymbol{q}_2\|^2 = \|\boldsymbol{q}_3\|^2 = 1$	$1,306 \times 1,261$	0.57s
Eliminated translations in the second and third image.	$s_{21}, s_{22}, s_{31}, s_{32}, \lambda_2, \lambda_3$	$\begin{cases} \boldsymbol{t}_k = \lambda_k \boldsymbol{x}_k - R_k \boldsymbol{X}, \ \ k = 2, 3 \\ \boldsymbol{X} = \boldsymbol{x}_1, \|\boldsymbol{q}_2\|^2 = \|\boldsymbol{q}_3\|^2 = 1 \end{cases}$	652×462	40ms
Unit quaternions.	$s_{21}, s_{22}, s_{31}, s_{32}$	$\|\boldsymbol{q}_2\|^2 = \|\boldsymbol{q}_3\|^2 = 1$	96×88	1.25ms
Non-unit quaternions.	s_2, s_3	$\boldsymbol{q}_k = \left(1,\ s_k \boldsymbol{v}^T\right)^T \ \ k = 2, 3$	12×20	0.3ms

Table 2. Comparison of different parameterizations for the one point two line problem from Section 6.

that the two additional cameras can be written

$$P_2 = \begin{bmatrix} x_1 & 1 & 0 & -1 \\ 0 & x_2 & x_3 & -x_3 \\ x_4 & x_5 & x_6 & x_7 \end{bmatrix}, \ P_3 = \begin{bmatrix} x_8 & x_9 & 0 & -x_9 \\ 0 & x_{10} & 1 & -1 \\ x_{11} & x_{12} & x_{13} & x_{14} \end{bmatrix}, \tag{25}$$

where $(x_1, \ldots, x_{14})$ are unknown parameters. We can now for each image-line triplet, $(\ell_{i1}, \ell_{i2}, \ell_{i3}), i = 3, \ldots, 9$, construct the matrix $M_i = [P_1^T \ell_{i1},\ P_2^T \ell_{i2},\ P_3^T \ell_{i3}]$. If the three image lines are views of the same line, the corresponding planes should intersect, and hence $\operatorname{rank} M_i = 2$, and all 3×3 sub-determinants of M_i should vanish. This gives three linearly independent second-degree polynomial constraints on $(x_1, \ldots, x_{14})$ for each i and in total 21 equations in the 14 parameters.

7.1. Building a solver

We have used our automatic generator on the formulation in (25). This leads to 36 solution[4], using nine lines, resulting in a template with size $20,273 \times 14,281$ without the reduction step. This problem has a large number of variables, and the resulting equations contain a large syzygy-set, and we have not been able to calculate this in Macaulay2 entirely. We have run a partial reduction step, based on taking all possible combinations of four equations, and this leads to a template of size $16,278 \times 13,735$. To evaluate our generated solver we ran the following test. We generated a large number of ground truth synthetic problems. We then ran our solver on the corresponding data, and compared the closest of our 36 solutions to the ground truth solution. The resulting histogram is shown in Figure 4. It also shows the difference to the ground truth solution after non-linear refinement of our solution. Our final elimination template is large, but very sparse. The complete solver has an average runtime of 17.8 s in Matlab on a standard desktop computer (Intel I7-3930K 64 GB ram).

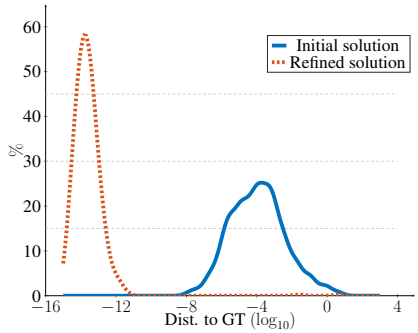

Figure 4. The distribution of the distance to the ground truth solution, for our initial nine-lines solver, and after non-linear refinement on the equation residuals.

8. Conclusions

In this paper we have presented a new method for finding elimination templates, using syzygy modules. The module encapsulates the ambiguity in representing the polynomials that are needed for constructing the action matrix. We have achieved state-of-the-art results by finding the normal form w.r.t. the module, but it is possible that a more advanced search over the syzygies would yield even better results, and this is an interesting venue for further work.

[4] In [1] the ideal of the trifocal tensor was investigated. Here they show that the corresponding variety has degree 297. However it turns out that most of these solutions do not correspond to a valid three view camera geometry, and these degeneracies are not present in our parameterization.

References

[1] C. Aholt and L. Oeding. The ideal of the trifocal variety. *Mathematics of Computation*, 83(289):2553–2574, 2014. 8

[2] C. Albl, Z. Kukelova, and T. Pajdla. R6p-rolling shutter absolute camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2292–2300, 2015. 5

[3] E. Ask, Y. Kuang, and K. Åström. Exploiting p-fold symmetries for faster polynomial equation solving. In *Int. Conf. Pattern Recognition*, Tsukuba, Japan, 2012. 5

[4] M. Bujnak, Z. Kukelova, and T. Pajdla. 3d reconstruction from image collections with a single known focal length. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1803–1810. IEEE, 2009. 5

[5] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Asian Conference on Computer Vision*, pages 11–24. Springer, 2010. 5

[6] M. Bujnak, Z. Kukelova, and T. Pajdla. Making minimal solvers fast. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1506–1513. IEEE, 2012. 2

[7] S. Burgess, Y. Kuang, and K. Åström. Pose estimation from minimal dual-receiver configurations. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2553–2556. IEEE, 2012. 5

[8] M. Byröd, M. Brown, and K. Åström. Minimal solutions for panoramic stitching with radial distortion. In *The 20th British Machine Vision Conference*. British Machine Vision Association (BMVA), 2009. 4, 6

[9] M. Byröd, K. Josephson, and K. Åström. Fast and stable polynomial equation solving and its application to computer vision. *Int. Journal of Computer Vision*, 2009. 2

[10] D. Cox, J. Little, and D. O'Shea. *Ideals, Varities and Algorithms*. Springer-Verlag, New York, NY, USA, 1992. 2

[11] D. A. Cox, J. Little, and D. O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006. 3

[12] M. Demazure. Sur deux problemes de reconstruction. Technical Report 882, INRIA, Rocquencourt, France, 1988. 1

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 1981. 1

[14] F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. *European Conf. on Computer Vision*, 2010. 6

[15] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at http://www.math.uiuc.edu/Macaulay2/. 1, 4

[16] S. Haner and K. Åström. Absolute pose for cameras under flat refractive interfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1428–1436, 2015. 5

[17] R. Hartley. Projective reconstruction from line correspondences. In *Conf. Computer Vision and Pattern Recognition*, pages 903–907. IEEE Computer Society Press, 1994. 7

[18] R. Hartley. Lines and points in three views and the trifocal tensor. *Int. Journal of Computer Vision*, 22(2):125–140, March 1997. 7

[19] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. 7

[20] G. Hee Lee, M. Pollefeys, and F. Fraundorfer. Relative pose estimation for a multi-camera system with known vertical direction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 540–547, 2014. 6

[21] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *2011 International Conference on Computer Vision*, pages 383–390. IEEE, 2011. 4, 5, 6

[22] A. Heyden. *Geometry and Algebra of Multipe Projective Transformations*. PhD thesis, Lund Institute of Technology, Sweden, 1995. 7

[23] F. Jiang, Y. Kuang, J. E. Solem, and K. Åström. A minimal solution to relative pose with unknown focal length and radial distortion. In *Asian Conference on Computer Vision*, pages 443–456. Springer, 2014. 5

[24] L. Kneip, R. Siegwart, and M. Pollefeys. Finding the exact rotation between two images independently of the translation. In *European Conference on Computer Vision*, pages 696–709. Springer, 2012. 1

[25] 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. 1

[26] Y. Kuang and K. Åström. Numerically stable optimization of polynomial solvers for minimal problems. In *European Conference on Computer Vision*, pages 100–113. Springer, 2012. 2, 4

[27] Y. Kuang and K. Åström. Pose estimation with unknown focal length using points, directions and lines. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 529–536, 2013. 5

[28] Y. Kuang and K. Åström. Stratified sensor network self-calibration from tdoa measurements. In *21st European Signal Processing Conference 2013*, 2013. 5

[29] Y. Kuang, M. Oskarsson, and K. Åström. Revisiting trifocal tensor estimation using lines. In *ICPR*, pages 2419–2423, 2014. 7

[30] Y. Kuang, J. E. Solem, F. Kahl, and K. Åström. Minimal solvers for relative pose with a single unknown radial distortion. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 33–40. IEEE, 2014. 5

[31] Y. Kuang, Y. Zheng, and K. Åström. Partial symmetry in polynomial systems and its applications in computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 438–445, 2014. 5

[32] Z. Kukelova, M. Bujnak, J. Heller, and T. Pajdla. Singly-bordered block-diagonal form for minimal problem solvers. In *Asian Conference on Computer Vision*, pages 488–502. Springer, 2014. 2

[33] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision*, pages 302–315. Springer, 2008. 1, 2, 3, 4, 5

[34] Z. Kukelova, T. Pajdla, and M. Bujnak. Fast and stable algebraic solution to l2 three-view triangulation. In *2013 International Conference on 3D Vision-3DV 2013*, pages 326–333. IEEE, 2013. 5

[35] V. Larsson and K. Åström. Uncovering symmetries in polynomial systems. In *European Conference on Computer Vision*, pages 252–267. Springer, 2016. 4, 5

[36] B. Li, L. Heng, G. H. Lee, and M. Pollefeys. A 4-point algorithm for relative pose estimation of a calibrated camera with a known relative rotation angle. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1595–1601. IEEE, 2013. 5

[37] H. M. Möller and H. J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numerische Mathematik*, 70(3):311–329, 1995. 2

[38] G. Nakano. Globally optimal dls method for pnp problem with cayley parameterization. In X. Xie, M. W. Jones, and G. K. L. Tam, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 78.1–78.11. BMVA Press, September 2015. 5

[39] O. Naroditsky and K. Daniilidis. Optimizing polynomial solvers for minimal geometry problems. In *2011 International Conference on Computer Vision*, pages 975–982. IEEE, 2011. 4, 5, 6

[40] O. Naroditsky, X. S. Zhou, J. Gallier, S. I. Roumeliotis, and K. Daniilidis. Two efficient solutions for visual odometry using directional correspondence. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):818–824, 2012. 6

[41] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2004. 1

[42] M. Oskarsson, A. Zisserman, and K. Åström. Minimal projective reconstruction for combinations of points and lines in three views. *Image and Vision Computing*, 22(10):777–785, 2004. 7

[43] L. Quan. Invariants of six points and projective reconstruction from three uncalibrated images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(1):34–46, 1995. 7

[44] O. Saurer, M. Pollefeys, and G. H. Lee. A minimal solution to the rolling shutter pose estimation problem. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1328–1334. IEEE, 2015. 5

[45] O. Saurer, P. Vasseur, C. Demonceaux, and F. Fraundorfer. A homography formulation to the 3pt plus a common direction relative pose problem. In *Asian Conference on Computer Vision*, pages 288–301. Springer, 2014. 4, 5

[46] H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, 2005. 1

[47] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006. 1, 5

[48] 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. 1, 5

[49] L. Svarm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 2016. 5

[50] C. Sweeney, J. Flynn, and M. Turk. Solving for relative pose with a partially known rotation is a quadratic eigenvalue problem. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 483–490. IEEE, 2014. 6

[51] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 422–429, 2014. 5

[52] C. Wu. P3. 5p: Pose estimation with unknown focal length. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2440–2448, 2015. 5

[53] E. Zheng, K. Wang, E. Dunn, and J.-M. Frahm. Minimal solvers for 3d geometry from satellite imagery. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 5

[54] Y. Zheng, Y. Kuang, S. Sugimoto, K. Åström, and M. Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2351, 2013. 4, 5, 6