# Polynomial Solvers for Saturated Ideals

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

## Abstract

*In this paper we present a new method for creating polynomial solvers for problems where a (possibly infinite) subset of the solutions are undesirable or uninteresting. These solutions typically arise from simplifications made during modeling, but can also come from degeneracies which are inherent to the geometry of the original problem.*

*The proposed approach extends the standard action matrix method to saturated ideals. This allows us to add constraints that some polynomials should be non-zero on the solutions. This does not only offer the possibility of improved performance by removing superfluous solutions, but makes a larger class of problems tractable. Previously, problems with infinitely many solutions could not be solved directly using the action matrix method as it requires a zero-dimensional ideal. In contrast we only require that after removing the unwanted solutions only finitely many remain. We evaluate our method on three applications, optimal triangulation, time-of-arrival self-calibration and optimal vanishing point estimation.*

## 1. Introduction

The success of geometric computer vision is largely built on the ability to efficiently and reliably solve systems of polynomial equations. Minimal solvers are used for robust model estimation using hypothesis and test frameworks such as RANSAC.

Naturally there has been a lot of research in computer vision devoted to constructing better polynomial solvers. The most common approach is based on the action matrix method, [29, 5], which reduces the problem to solving a linear system followed by an eigendecomposition. Early examples of applications in computer vision were based on highly problem specific derivations including semi-manual extraction of Gröbner bases [37, 36]. In [20], Kukelova *et al*. presented a system for automatically creating polynomial solvers based on the action matrix method . This generator has been used to solve many problem in geometric computer vision, *e.g*. [21, 39, 38, 32]. Recently an improvement to the automatic generator scheme was proposed [24], exploiting the inherent relations between the original equations. In [17] and [20, 30] the authors presented methods optimizing the so called elimination templates, with respect to stability and size respectively. With similar goals, Ask *et al*. [2] showed how to exploit certain symmetries available in some polynomial systems. This was later extended by both Kuang *et al*. [19] and Larsson *et al*. [23]. In [5], Byröd *et al*. proposed methods to improve numerical accuracy by selecting the quotient space basis at runtime using singular value decomposition (SVD) or by using QR factorization with column pivoting. In [3], Bujnak *et al*. proposed two different methods for extracting univariate polynomials directly from the action matrix. The roots of these polynomials can then be found efficiently using Sturm-sequences [14], instead of computing the full eigendecomposition of the action matrix. For an overview of other approaches to solving systems of polynomial equations see [5] and the references therein.

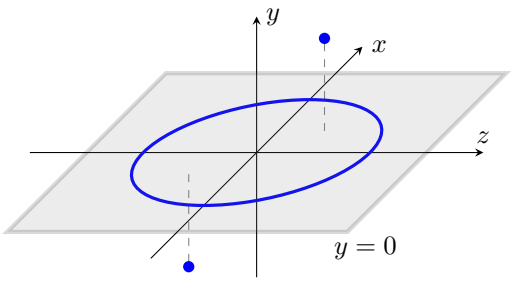


Figure 1. Solutions to the polynomial system in (23). The solution set consists of two points and a circle contained in the $xz$-plane. Saturating the ideal with $y$ removes any solution where $y = 0$.

In this paper we present an extension to the action matrix method which allows us to build polynomial solvers for saturated ideals. Saturation allows us to add constraints that some polynomials should be non-zero. The most interesting cases are when the original ideal contains infinitely many solutions while the saturated ideal is zero-dimensional. For these problems it is not possible to apply the action matrix

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

method directly.

State-of-the-art solvers for problems involving saturated ideals, *e.g.* [37, 18, 34, 4], were implemented by manually constructing equations from the saturated ideal as a pre-processing step and then applying the standard approach. These methods are essentially hand-crafted and problem specific. In contrast our approach is entirely generic and we have extended the automatic solver generator from [24] to also handle saturated ideals.

The contributions of this paper are two-fold:

- We provide a theoretical framework for using the action matrix method for saturated ideals.
- We show how to apply the theory in practice and provide an implementation in the automatic solver generator framework from [24].

## 2. Background

First let us fix some of the notation which we will use throughout the paper. We will (mostly) denote vectors with lower case and bold face, *e.g.* $\boldsymbol{x}, \boldsymbol{u}$ and $\boldsymbol{b}$. Matrices will be uppercase, *e.g.* $A, M$ and $H$. For a mapping $T : X \to Y$ we denote the restriction of $T$ to $S \subset X$ as $T\big|_S : S \to Y$. For equivalence classes we let brackets denote the map which takes any representative to its class, *i.e.* $[x]$ denotes the class which $x$ belongs to, *i.e.* $x \in [x]$.

### 2.1. Algebraic Geometry

We start with a brief review of some of the facts and definitions from algebraic geometry. For a more thorough review on algebraic geometry we recommend [7] and for how it has been applied in computer vision [5].

The set of all polynomials with $n$ variables and complex coefficients is denoted $\mathbb{C}[x_1, x_2, \ldots, x_n]$ or $\mathbb{C}[X]$, where $X = (x_1, x_2, \ldots, x_n)$ for short. This set forms a ring with the usual operations. For any polynomial system $F = \{f_i(\boldsymbol{x}) = 0\}_{i=1}^m$ there is a corresponding *ideal*

$$I(F) = \langle f_1, \ldots, f_m \rangle = \{\textstyle\sum_i h_i f_i \mid h_i \in \mathbb{C}[X]\}. \quad (1)$$

Closely connected is the set of shared zeros,

$$V(I) = \{\boldsymbol{x} \in \mathbb{C}^n \mid f(\boldsymbol{x}) = 0,\ \forall f \in I\}, \quad (2)$$

called an *affine variety*. The *quotient ring* $\mathbb{C}[X]/I$ consists of the equivalence classes over $I$, *i.e.*

$$[p] = [q] \iff p - q \in I. \quad (3)$$

This also means that $p(\boldsymbol{x}) = q(\boldsymbol{x})$ for all $\boldsymbol{x} \in V(I)$.

The quotient ring $\mathbb{C}[X]/I$ has the useful property that it is finite dimensional as a vector space when there are finitely many solutions (*i.e.* there are only finitely many points in the affine variety $V(I)$).

### 2.2. Solving using Action Matrices

In this section we give an overview of the action matrix method for solving polynomial systems. The goal is to convert the difficult problem of solving the polynomial system to an eigenvalue problem, for which there exist good numerical methods.

The idea is to consider the operator $T_\alpha^I : \mathbb{C}[X]/I \to \mathbb{C}[X]/I$, that multiplies with the polynomial $\alpha \in \mathbb{C}[X]$, *i.e.*

$$T_\alpha^I[p] = [\alpha p], \quad [p] \in \mathbb{C}[X]/I. \quad (4)$$

This operator is linear and if we fix a (linear-)basis $\{b_k\}_{i=1}^d$ for $\mathbb{C}[X]/I$ we can represent it with a matrix $M_\alpha = (m_{ij}) \in \mathbb{C}^{d\times d}$, *i.e.* $[\alpha b_i] = \left[\sum_j m_{ij} b_j\right]$. Since this equation holds for any $\boldsymbol{x} \in V(I)$ we have

$$M_\alpha \boldsymbol{b}(\boldsymbol{x}) = \alpha(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in V(I). \quad (5)$$

where $\boldsymbol{b} = (b_1, \ldots, b_d)^T$. This means that $\boldsymbol{b}$ and $\alpha$ evaluated at the solutions are eigenvectors and eigenvalues of the constant matrix $M_\alpha$. Thus if we can recover $M_\alpha$ we can find the solutions by simply computing its eigendecomposition. For a more in-depth review, especially on how to construct the polynomial solvers in practice, we recommend either [5], [20] or [24].

### 2.3. Saturations

The main focus of this paper is extending the action matrix method to saturated ideals. For an ideal $I \subset \mathbb{C}[X]$ we can define the *saturation w.r.t.* $f_s \in \mathbb{C}[X]$ as[1]

$$\mathrm{Sat}(I, f_s) = \{p \mid \exists N \geq 0, f_s^N p \in I\}. \quad (6)$$

The saturation allows us to essentially remove any solutions where $f_s(\boldsymbol{x}) = 0$ from the variety. In fact we have that

$$V(\mathrm{Sat}(I, f_s)) = \overline{V(I) \setminus V(\langle f_s \rangle)}, \quad (7)$$

where the closure is taken in the Zariski topology. For proof of the above property and more information about saturations we recommend Cox *et al*. [6].

A different approach for removing unwanted solutions is the so-called *Rabinowitsch trick*, where an additional variable $x_0$ is added alongside the new equation

$$1 - x_0 f_s(\boldsymbol{x}) = 0. \quad (8)$$

This will remove any solution where $f_s(\boldsymbol{x}) = 0$. However we will show examples in the experimental evaluation where this approach leads to much larger elimination templates compared to working directly with the saturated ideal.

[1] Note that while we only saturate with a single polynomial $f_s \in \mathbb{C}[X]$ here, the saturation can also be defined w.r.t. a polynomial ideal.

## 3. Action Matrices in Saturated Ideals

We are now ready to present our main theoretical contribution. Let $I = \langle f_1, \dots, f_m \rangle \subset \mathbb{C}[X]$ be an ideal such that

$$J = \mathrm{Sat}(I, f_s) = \{p \mid \exists N \geq 0,\ f_s^N p \in I\} \tag{9}$$

is zero-dimensional and let $\{b_k\}_{k=1}^d$ be a linear basis for the quotient ring $\mathbb{C}[X]/J$.

The goal is now to lift the properties we need from the saturated ideal $J$ into the original ideal $I$. This will allow us to create an elimination template directly from our original equations which can be used to find the action matrix from the saturated ideal.

**Lemma 1.** *For each $N \geq 0$ the set $\{f_s^N b_k\}_{k=1}^d$ is linearly independent in the quotient ring $\mathbb{C}[X]/I$.*

*Proof.* Assume otherwise. Then there exist some $c_i \in \mathbb{C}$ such that $\sum_i c_i f_s^N b_i \in I$. From the definition of the saturation we get

$$f_s^N \left(\textstyle\sum_i c_i b_i\right) \in I \implies \textstyle\sum_i c_i b_i \in J, \tag{10}$$

which is a contradiction. □

For $N \geq 0$ define

$$S_N = \left[\text{span } \{f_s^N b_k\}_{k=1}^d\right] \subset \mathbb{C}[X]/I. \tag{11}$$

From Lemma 1 we know that $S_N$ forms an $d$-dimensional subspace in $\mathbb{C}[X]/I$.

**Lemma 2.** *For each $\alpha \in \mathbb{C}[X]$ there exists $N \geq 0$ such that*

$$[\alpha p] \in S_N, \quad \forall p \in S_N, \tag{12}$$

i.e. *$S_N$ is stable under multiplication with $\alpha$.*

*Proof.* Let $\alpha \in \mathbb{C}[X]$ and consider $[\alpha b_k]$ in $\mathbb{C}[X]/J$. Since $\{b_k\}_{k=1}^d$ spans $\mathbb{C}[X]/J$ there exist $m_{ij} \in \mathbb{C}$ such that

$$[\alpha b_i] = \left[\textstyle\sum_j m_{ij} b_j\right] \ \Leftrightarrow\ p_i := \alpha b_i - \displaystyle\sum_j m_{ij} b_j \in J. \tag{13}$$

By definition there exist some $N_i \geq 0$ such that $f_s^{N_i} p_i \in I$. Take some $N \geq N_i$ for all $i$, then in $\mathbb{C}[X]/I$ we have

$$f_s^N p_i \in I \ \Leftrightarrow\ \left[\alpha f_s^N b_i\right] = \left[\textstyle\sum_j m_{ij} f_s^N b_j\right]. \tag{14}$$

Now for any $p \in S_N$ we have as $[p] = \left[\sum c_i f_s^N b_i\right]$. Applying (14) we get

$$[\alpha p] = \left[\textstyle\sum_i c_i \alpha f_s^N b_i\right] = \left[\textstyle\sum_i \sum_j c_i m_{ij} f_s^N b_j\right] \in S_N, \tag{15}$$

which proves the lemma. □

The following theorem will show a useful relationship between the two multiplication operators

$$T_\alpha^I : \mathbb{C}[X]/I \to \mathbb{C}[X]/I \quad \text{and} \quad T_\alpha^J : \mathbb{C}[X]/J \to \mathbb{C}[X]/J. \tag{16}$$

**Theorem 1.** *For large enough $N \geq 0$, the action matrices corresponding to $T_\alpha^I\big|_{S_N}$ and $T_\alpha^J$ are the same with respect to the basis $\{f_s^N b_k\}_{k=1}^d$ and $\{b_k\}_{k=1}^d$ respectively.*

*Proof.* While $\mathbb{C}[X]/I$ may be infinite dimensional as a vector space, we know from Lemma 2 that $\mathrm{Im}\, T_\alpha^I\big|_{S_N} \subset S_N$ for large enough $N$. The rest of the proof follows immediately by noting that $M_\alpha = (m_{ij})$ from (13) and (14) is indeed the action matrix for both mappings. □

To gain some intuition why this works consider the action matrix $M_\alpha = (m_{ij})$ corresponding to $T_\alpha^I\big|_{S_N}$, *i.e.*

$$\begin{bmatrix} \\ M_\alpha \\ \\ \end{bmatrix} \begin{pmatrix} \\ f_s^N \boldsymbol{b} \\ \\ \end{pmatrix} = \alpha \begin{pmatrix} \\ f_s^N \boldsymbol{b} \\ \\ \end{pmatrix}. \tag{17}$$

Note that while this equation is satisfied for all $\boldsymbol{x} \in V(I)$, any solution where $f_s(\boldsymbol{x}) = 0$ will correspond to a zero vector and not an eigenvector. Thus by computing the eigenvectors of $M_\alpha$ it is possible to recover only the solutions in the saturated ideal.

## 4. Implementation Details

To apply the theory presented in the previous section in practice, we extend the method for automatic solver generation from [24]. The steps taken are outlined below.

1. Generate an instance of the problem with coefficients in some prime field $\mathbb{Z}_p$. This allows for efficient and accurate computations.

2. Compute the saturated ideal

   $$J = \mathrm{Sat}(I, f_s) \tag{18}$$

   and find a linear basis $\{b_k\}_{k=1}^d$ for $\mathbb{Z}_p[X]/J$.

3. Form the polynomials used in the action matrix, *i.e.*

   $$p_i = \alpha b_i - \textstyle\sum_j m_{ij} b_j \in J. \tag{19}$$

4. By iteration find the smallest $N \geq 0$ such that

   $$f_s^N p_i \in I \quad \forall i. \tag{20}$$

5. Using the method described in [24], find $h_{ij} \in \mathbb{Z}_p[X]$ such that

   $$f_s^N p_i = \textstyle\sum_j h_{ij} f_j \tag{21}$$

   and create the elimination template from these.

Note that we essentially take the same steps as in [24], but we instead form the polynomials $p_i$ in the saturated ideal. Then we lift both the basis $\{b_k\}_{k=1}^d$ and the polynomials $p_i$ into the original ideal $I$ by multiplying with $f_s^N$. Each of the steps listed above can be efficiently carried out in algebraic geometry software such as Macaulay2 [10].

In the implementation we restrict ourselves to saturating a single monomial. This is however not very limiting, since if we instead wish to saturate some polynomial $f(\boldsymbol{x})$ we can introduce a new variable $x_0$ and the equation $x_0 - f(\boldsymbol{x}) = 0$. Saturating $x_0$ will then be equivalent to saturating $f(\boldsymbol{x})$ in the original formulation. Note that this equation is much simpler compared to (8) since the new variable $x_0$ appears linearly and is easy to eliminate.

In practice we found that for problems where we need to saturate a general polynomial $f(\boldsymbol{x})$, the elimination templates often become much smaller when the extra variable $x_0$ is not present in the quotient ring basis $\{b_k\}_{k=1}^d$. This can always be accomplished by choosing a monomial order where any monomial containing $x_0$ is greater than any monomial without $x_0$. If this is the case then clearly $\mathrm{LT}(x_0 - f(\boldsymbol{x})) = x_0$ and the normal set of the Gröbner basis will only contain monomials without $x_0$.

## 5. Toy Example

We will now show an overview of the steps taken to construct a polynomial solver for a toy example,

**Example 1.** Consider the following system of equations.

$$\begin{cases} f_1 = c_0x^2 + c_1y^2 + c_2z^2 + c_3 = 0, \\ f_2 = c_0x^2 + c_4xy + c_2z^2 + c_3 = 0, \\ f_3 = c_0x^2 + c_5yz + c_2z^2 + c_3 = 0, \end{cases} \tag{22}$$

where $c_0, c_1, \ldots, c_5 \in \mathbb{C}$ are constants. Note that while three quadratic equations in three variables in general have eight solutions, it is easy to see that this system becomes degenerate for $y = 0$.

To construct a polynomial solver for this problem we start by considering an instance of the system where the coefficients are integers, *e.g.*

$$\begin{cases} f_1 = x^2 + y^2 + z^2 - 1 = 0, \\ f_2 = x^2 + 2xy + z^2 - 1 = 0, \\ f_3 = x^2 + 2yz + z^2 - 1 = 0. \end{cases} \tag{23}$$

A depiction of the solution set for these equations is shown in Figure 1. For this particular instance we can use algebraic geometry software (*e.g.* [10]) to compute the saturation w.r.t. $y$.

$$J = \mathrm{Sat}(I, y) = \langle y - 2z,\ x - z,\ z^2 - \frac{1}{6} \rangle. \tag{24}$$

This also gives us basis for the quotient ring $\mathbb{C}[X]/J$,

$$b_0 = 1, \quad b_1 = z. \tag{25}$$

Taking the action polynomial as $x$ we get the action matrix

$$\begin{bmatrix} 0 & 1/6 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} z \\ 1 \end{pmatrix} = x \begin{pmatrix} z \\ 1 \end{pmatrix}, \quad (x, y, z) \in V(J). \tag{26}$$

Thus we have that the polynomials

$$p_1 = xz - 1/6, \quad p_2 = x - z, \tag{27}$$

both lie in $J$. They are however not in $I = \langle f_1, f_2, f_3 \rangle$, but when we multiply these by saturated variable $y$ we can express them in terms of the original equations,

$$yp_1 = -\frac{z}{3}f_1 + (\frac{5z}{12} - \frac{x}{12})f_2 + (\frac{x}{12} + \frac{y}{6} - \frac{z}{12})f_3, \tag{28}$$

$$yp_2 = \frac{1}{2}f_2 - \frac{1}{2}f_3. \tag{29}$$

Note that in this instance we only needed to multiply by $y$ to lift $p_1$ and $p_2$ into $I$, but in general higher powers might be required.

Now to solve a general instance of (22) we follow the approach in [24]. The elimination template is formed by multiplying each of the polynomials $f_i$ with the monomials occurring in the coefficients in (28)–(29), *i.e.*

$$\{zf_1,\ f_2,\ xf_2,\ zf_2,\ f_3,\ xf_3,\ yf_3,\ zf_3\}. \tag{30}$$

Using just linear combinations of these, it is then possible to recover the polynomials,

$$yp_1 = y(xz - m_{11}z - m_{12}), \tag{31}$$

$$yp_2 = y(x - m_{21}z - m_{22}), \tag{32}$$

from which we can extract the action matrix

$$M_x = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}. \tag{33}$$

We will in the following sections show how our method can be applied to a number of real world problems. These examples will show the benefits of our approach in terms of ease of construction of solvers without manual saturation (Section 6 and 7), significant speed-up (Section 7 and 8) and the benefit over introduction of auxiliary variables as in (8) (Section 8).

## 6. Triangulation

In this section we will investigate how our saturation framework can be used in multiple view triangulation. Geometrically triangulation seems easily done, by simply intersecting the back-projected image rays. However to find the

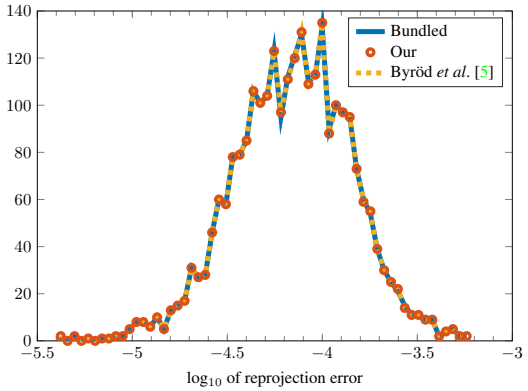


Figure 2. Error histograms for the dinosaur experiment for the different methods, on a log-scale.

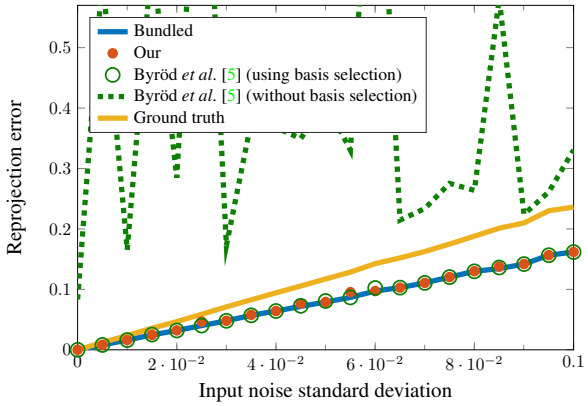


Figure 3. Synthetic triangulation test, comparing our method to non-linear optimization for different amounts of added image noise.

| | Template size | # Sol. | Exec. time |
|---|---|---|---|
| **Three view triang.** | | | |
| Stewénius *et al.* [37] | - | 47 | $> 1$ s |
| Byröd *et al.* [4] (relaxed ideal) | $225 \times 209^{\dagger}$ | 154 | 60 ms |
| Byröd *et al.* [5] (basis select.) | $225 \times 209^{\dagger}$ | 58 | 3 ms |
| Our | $571 \times 676$ | 47 | 10 ms |
| Our (new param.) | $209 \times 265$ | 50 | 3 ms |
| **Time-of-Arrival** | | | |
| Kuang *et al.* [18] (4,6) | $966 \times 925^{\dagger}$ | 38 | 0.6 s |
| Kuang *et al.* [18] (5,5) | $1,386 \times 1,539^{\dagger}$ | 42 | 1.4 s |
| Our (4,6) | $569 \times 692$ | 38 | 22 ms |
| Our (5,5) | $938 \times 1,301$ | 42 | 95 ms |
| **Vanishing point est.** | | | |
| Mirzaei *et al.* [28] | $2,860 \times 3,060$ | 40 | 1.8 s |
| Our | $246 \times 397$ | 40 | 5 ms |

Table 1. Overview over template sizes for the investigated applications in the paper. †: Several elimination steps are used, the largest of the elimination templates is reported.

global minimizer of the reprojection error for many views is an inherently difficult problem. For two views one can find the solution by solving a sixth-degree polynomial, [12], but for three views it becomes numerically and theoretically harder, and there have many papers dealing with this problem [37, 4, 5, 22]. There are also iterative methods, that do not guarantee a global optimum, [1, 16, 25, 13], methods that minimize the $L_\infty$-error [11, 27] and Branch-and-Bound methods whose worst case convergence is exponential [15, 26].

### 6.1. Optimal Three View Triangulation

We will initially model the triangulation problem in three views in the same way as described in [37]. To find the optimal solution we want to minimize the reprojection error

$$\underset{\mathbf{X}}{\text{minimize}} \sum_{i=1}^{3} \left(\frac{P_i^1 \mathbf{X}}{P_i^3 \mathbf{X}} - \mathbf{x}_i^1\right)^2 + \left(\frac{P_i^2 \mathbf{X}}{P_i^3 \mathbf{X}} - \mathbf{x}_i^2\right)^2, \qquad (34)$$

where $\mathbf{X} = [X\ Y\ Z\ 1]^T$ is the unknown 3D-point, and superscript denotes row-index. This is a non-convex problem and we will solve it by evaluating all local minima. In order to do this we find the points where the gradient vanishes. We have the freedom to make projective world coordinate changes without changing the error function. Similar to [37] we will use this to simplify our formulation, by choosing a coordinate system so that the third rows of the cameras are given by $[1\,0\,0\,0]$, $[0\,1\,0\,0]$ and $[0\,0\,1\,0]$ respectively. This will in turn change the denominators in (34) to $X$, $Y$ and $Z$ respectively. The gradient is easily calculated, but in order to get polynomial constraints we need to cross-multiply all the fractions with the denominators. In this way we end up with three equations, each of total degree 6 – giving rise to a one-dimensional ideal. In [37] they solved this by manually saturating the ideal, and after saturation with $X$, $Y$ and $Z$ ended up with 47 solutions. But the problem is very badly conditioned and the authors needed to use 128 bit arithmetic, making the solver impractically slow to use in practice. In [4] they used the same parametrization but showed that, by using a slightly different saturation approach and then considering a relaxed ideal, a practical solver could be developed. The solver still suffered from bad conditioning. In [5] the authors developed new methodology to handle poor conditioning by numerically choosing the linear basis during runtime in the minimal solver, using either SVD or QR factorization. We can use our automatic saturation process to avoid the cumbersome manual saturation steps. After saturation with $X$, $Y$ and $Z$ we end up with 47 solutions and get an elimination template of size $571 \times 676$. In order to compare our solver with the publicly available solver from [5] we ran a simple test on the well-known dinosaur sequence. Using 36 calibrated frames with a total of 2592 points, seen in at least three views, we extracted for each of these points the corresponding first, last and middle camera. We ran our solver and the solver from [5]. The

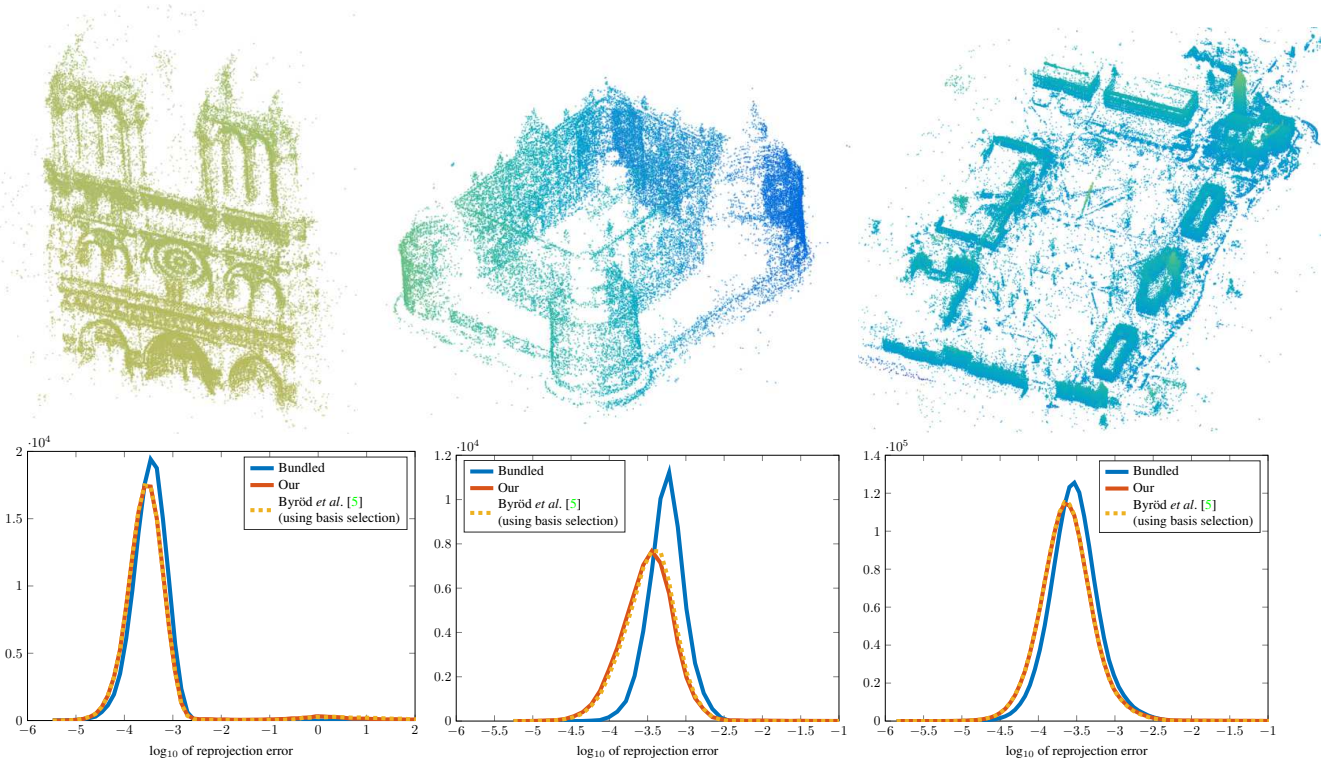


Figure 4. Triangulated points for three real problems using our triangulation method. Left to right: the Notre Dame dataset [35] with around 120,000 3D-points, the Orebro Castle dataset [31] (around 50,000 3D-points), and the Arts Quad dataset [8] (around 1,400,000 3D-points). Bottom shows histograms for reprojection errors for our method compared to the solutions provided in the original datasets (which are bundled over all available cameras, hence the slighly larger median error) and the solver from [5].

resulting mean of the reprojection error for all points was $8.90 \times 10^{-5}$ for both methods. Using the chosen cameras we can for each point triangulate its 3D position linearly and then perform non-linear optimization on the reprojection error. This gave the same error ($8.90 \times 10^{-5}$). In Figure 2 the error histogram is shown for all methods. One can see that they produce very similar results. Our developed solver has a runtime of around 10 ms, compared to around 3 ms for the handcrafted solver from [5].

#### 6.1.1 New Parameterization

We can actually further simplify our problem formulation by setting the last row of camera three to $[0\,0\,0\,1]$, instead of as previously $[0\,0\,1\,0]$ (*i.e.* the third camera's image plane to infinity). This means that we don't need to cross-multiply with factors of $Z$. This leads to three equations of total degree 6, 6 and 5 respectively. Furthermore, in this parameterization it turns out that it is sufficient to saturate only one of the variables. In this case the saturated ideal has 50 solutions. This approach gives a substantially smaller template of size $209 \times 265$, and in turn a much faster solver, with runtime under 3 ms. We have evaluated this new solver on both synthetic and real data. For a synthetic test, we randomly placed a 3D point and three cameras in a box with side-length 100. We then randomly chose an orientation for each camera with the constraint that the point would be visible, for a field of view of around 70°. We finally added normally distributed noise to the projection points with varying standard deviation. The result of running our algorithm, for varying degrees of noise, is shown in Figure 3. Also shown is the corresponding reprojection error for the randomly set ground truth 3D-point as well as the non-linearly optimized 3D-point position, using the ground truth position as initialization. The results for each noise level are evaluated on 1,000 random instances. We compare our results with the state-of-the-art solver from Byröd *et al.* [5]. Note that our solver achieves similar results without the special techniques for improving numerics from [5]. For comparison we have also included the results from the solver from [5] without basis selection. To test our triangulation method on real data, we ran it on three large scale problems where the camera matrices are available. In Figure 4 triangulated 3D points for the Notre Dame dataset [35] with around 120,000 3D-points, the Orebro Castle dataset [31]

with around 50,000 3D-points, and the Arts Quad dataset [8] with around 1,400,000 3D-points are shown.

## 7. Time-of-Arrival Self-Calibration

The term Time-of-Arrival (ToA) or alternatively Time-of-Flight (ToF), denotes the travel time of a signal from a transmitter to a receiver. If the speed of medium is known such measurements provide distance measurements from transmitters to receivers. The ToA self-calibration problem is the problem of estimating both transmitter positions $\mathbf{s}_j$ and receiver positions $\mathbf{r}_i$ given distance measurements,

$$d_{ij} = \|\mathbf{r}_i - \mathbf{s}_j\|_2. \tag{35}$$

Following [18] we rearrange the equations in (35) into four types,

$$\begin{aligned} d_{11}^2 &= (\mathbf{r}_1 - \mathbf{s}_1)^T(\mathbf{r}_1 - \mathbf{s}_1), && (36)\\ d_{1j}^2 - d_{11}^2 &= -2\mathbf{r}_1^T(\mathbf{s}_j - \mathbf{s}_1) + \mathbf{s}_j^T\mathbf{s}_j - \mathbf{s}_1^T\mathbf{s}_1, && (37)\\ d_{i1}^2 - d_{11}^2 &= -2(\mathbf{r}_i - \mathbf{r}_1)^T\mathbf{s}_1 + \mathbf{r}_i^T\mathbf{r}_i - \mathbf{r}_1^T\mathbf{r}_1, && (38)\\ \tfrac{d_{ij}^2 - d_{i1}^2 - d_{1j}^2 + d_{11}^2}{-2} &= (\mathbf{r}_i - \mathbf{r}_1)^T(\mathbf{s}_j - \mathbf{s}_1). && (39) \end{aligned}$$

Introducing the two matrices

$$R = \begin{bmatrix}(\mathbf{r}_2 - \mathbf{r}_1) \dots (\mathbf{r}_m - \mathbf{r}_1)\end{bmatrix} \tag{40}$$

$$S = \begin{bmatrix}(\mathbf{s}_2 - \mathbf{s}_1) \dots (\mathbf{s}_n - \mathbf{s}_1)\end{bmatrix} \tag{41}$$

the $(m-1)(n-1)$ constraints in (39) can then be written $B = R^T S$, where

$$B = \begin{pmatrix} \frac{d_{22}^2 - d_{21}^2 - d_{12}^2 + d_{11}^2}{-2} & \cdots & \frac{d_{2n}^2 - d_{21}^2 - d_{1n}^2 + d_{11}^2}{-2} \\ \vdots & \ddots & \vdots \\ \frac{d_{m2}^2 - d_{m1}^2 - d_{12}^2 + d_{11}^2}{-2} & \cdots & \frac{d_{mn}^2 - d_{m1}^2 - d_{1n}^2 + d_{11}^2}{-2} \end{pmatrix}. \tag{42}$$

By factorizing $B = \tilde{R}^T\tilde{S}$, we can almost solve the self-calibration problem, however the factorization is not unique. If $B = \tilde{R}^T\tilde{S}$, then $B = \tilde{R}^T A A^{-1}\tilde{S}$ is also a valid factorization. Furthermore both the sender position $\mathbf{s}_1$ and the receiver position $\mathbf{r}_1$ are unknown.

Since the choice of coordinate system is arbitrary, one may without loss of generality set $\mathbf{r}_1$ to the origin. Also since any matrix $A$ can be QR-factorized as a rotation matrix times a triangular matrix, one may assume that $A$ is triangular, *i.e.* $A = L$. These choices fixate most of the freedom in the coordinate system. Thus we parametrize the problem with $(L, \mathbf{b})$ so that

$$\begin{aligned} &\mathbf{r}_1 = \mathbf{0},\ \mathbf{s}_1 = L\mathbf{b}, \mathbf{r}_i = L^{-T}\tilde{R}_i,\ i = 2 \dots m, \\ &\mathbf{s}_j = L(\tilde{S}_j + \mathbf{b}),\ j = 2 \dots n. \end{aligned} \tag{43}$$

Using this parametrization the equations (36)–(38) become

$$\begin{aligned} d_{11}^2 &= \mathbf{b}^T H^{-1}\mathbf{b}, && (44)\\ d_{1j}^2 - d_{11}^2 &= \tilde{S}_j^T H^{-1}\tilde{S}_j + 2\mathbf{b}^T H^{-1}\tilde{S}_j, && (45)\\ d_{i1}^2 - d_{11}^2 &= \tilde{R}_i^T H\tilde{R}_i - 2\mathbf{b}^T\tilde{R}_i, && (46) \end{aligned}$$

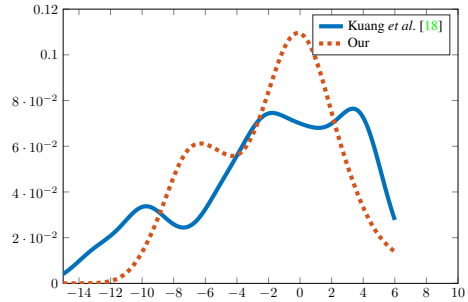


Figure 5. Histogram of $\log_{10}$ of residuals for solutions using the solver from Kuang *et al.* [18] and from proposed system.

using the symmetric matrix $H = (L^T L)^{-1}$. When both receivers and senders are in general 3D positions, there are two minimal problems to the time-of-arrival self-calibration problem – 6 receivers and 4 senders, and 5 receivers and 5 senders [18]. The solution strategy is now to first consider the constraints in (46), which are linear in the unknowns $(H, \mathbf{b})$. For the minimal problem with 6 receivers and 4 senders there are 5 such linear constraints on the 9 parameters in $(H, \mathbf{b})$. Thus it is possible to parametrize $(H, \mathbf{b})$ as polynomials of degree one in four unknowns $(x_1, x_2, x_3, x_4)$. The remaining equations (44) and (45) involve the inverse of $H$. By rewriting $H^{-1} = \operatorname{adj} H / \det H$ and multiplying with $\det H$, the remaining four equations

$$\begin{aligned} d_{11}^2 \det H - \mathbf{b}^T \operatorname{adj} H\mathbf{b} &= 0, && (47)\\ (d_{1j}^2 - d_{11}^2) \det H - \tilde{S}_j^T \operatorname{adj} H\tilde{S}_j - 2\mathbf{b}^T \operatorname{adj} H\tilde{S}_j &= 0 && (48) \end{aligned}$$

become polynomial equations in $(x_1, x_2, x_3, x_4)$. The problem has 38 solutions for which $\det H \neq 0$. However there is a one dimensional variety of solutions for which $\det H = 0$. The other minimal case with 5 receivers and 5 senders can in a similar manner be reduced to a system of five equations (47) and (48) in five unknowns. This problem has 42 solutions for which $\det H \neq 0$. For other cases see [33, 34].

Using our approach we generated solvers for the two minimal cases $(4, 6)$ and $(5, 5)$. A few different tests were made to experimentally verify the new solvers and to compare it with previous state-of-the-art as presented in [18]. Firstly, we generated 100 random instances of the problem. For each such instance we generated the system of polynomial equations and then ran different solvers to obtain all solutions. These were then compared in terms of template size, execution time and residuals (in terms of the log10 of the absolute values of the equations evaluated on the computed solutions). In none of the cases non-linear refinement on the obtained solutions was used. The residuals are illustrated in Figure 5. The underlying problem is numerically challenging, and one can see that both solvers suffer from bad conditioning. However, as will be shown next, the minimal solvers can still work well for initialization and outlier removal.

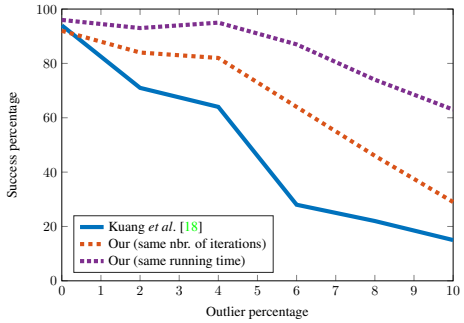


Figure 6. Comparison of our method and the solver from [18] in a RANSAC framework for different outlier ratios. Our system is run with the same running time or number of iterations respectively as [18].

Secondly, we used the solvers in a RANSAC scheme on overdetermined problems with 4 receivers and 25 senders. A few different scenarios were tried with increasing percentage of outliers. For this scenario an outlier among the 4 measurement from a sender, renders that whole sender useless. Since at least $4 \times 6$ outlier-free distance measurements are needed for a hypothesis and at least one other set of $4 \times 1$ outlier-free distance measurement from a 7th sender is needed to verify the solution, the problem can become impossible to solve already with 19 outliers among the 100 measurements. We thus only tried outlier percentages in the span 0-10 percent. The result is shown in Figure 6.

## 8. Vanishing Point Estimation

In [28] the authors present a method for estimating the vanishing points in a Manhattan world. It is based on solving the following minimization problem optimally,

$$\min_R J = \sum_{i=1}^{N} (\mathbf{n}_i^T R \mathbf{m}_i)^2, \;\; R^T R = I, \;\; \det(R) = 1, \quad (49)$$

where $\mathbf{n}_i$ is the known vanishing point in the canonical frame for each line $i$ given by its unit normalized representation $\mathbf{m}_i$. The sought $R$ is the rotation matrix that takes the camera to the canonical frame. The rotation is then parameterized using the Cayley-Gibbs-Rodrigues formulation using $\mathbf{s}^T = [s_1, \; s_2, \; s_3 \,]$, and

$$R(\mathbf{s}) = \frac{(1 - \mathbf{s}^T\mathbf{s})I + 2[\mathbf{s}]_\times + 2\mathbf{s}\mathbf{s}^T}{1 + \mathbf{s}^T\mathbf{s}}, \quad (50)$$

where $[\mathbf{s}]_\times$ is the cross-product matrix. The solution to (49) is found by enumerating all stationary points of $J$, by calculating the derivatives of (49) with respect to $\mathbf{s}$ and setting these to zero. The equations are given by

$$f_j(\mathbf{s}) = (1 + \mathbf{s}^T\mathbf{s})\frac{\partial J'}{\partial s_j} - 4 s_j J' = 0, \quad j = 1, 2, 3, \quad (51)$$

where $J'(\mathbf{s}) = (1+\mathbf{s}^T\mathbf{s})^2 J(\mathbf{s})$. The affine variety related to (51) contains a one-dimensional solution set corresponding

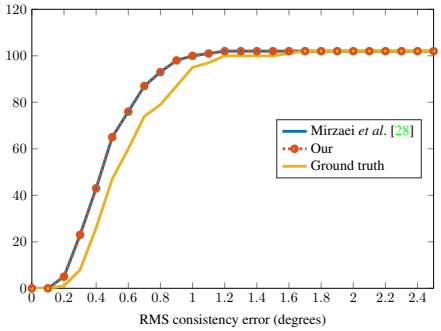


Figure 7. Vanishing point estimation RMS consistency on the York Urban dataset [9]. The figure shows that our method gives the same results as [28] on this dataset.

to the imaginary hypersphere $1 + \mathbf{s}^T\mathbf{s} = 0$. This means that the standard action matrix method cannot be used directly. In [28] the authors solve this by introducing an auxiliary variable $s_0$ and a new equation, $f_0 = s_0(1 + \mathbf{s}^T\mathbf{s}) - 1 = 0$, which results in a zero-dimensional solution set, containing at most 40 solutions. This results in a stable, but comparably slow solver, based on an elimination template of size $2,860 \times 3,060$.

We have tested our automatic saturation methodology on this problem, by directly solving (51) and saturate with $1 + \mathbf{s}^T\mathbf{s} = 0$. This results in an orders of magnitude faster solver with an elimination template of size $246 \times 397$. In order to compare the numerical properties of our solver we evaluated it on the same dataset as in [28], namely the York Urban Dataset (YUD), [9]. In Figure 7 the cumulative histogram of the consistency error is shown for our method compared to [28] and the ground truth estimate. The consistency error is defined as the RMS of $\sin^{-1}(\mathbf{v}_i^T\mathbf{m}_j)$ over all lines $j$ for each image, where $\mathbf{v}_i$ is the corresponding estimated vanishing point. One can note that we get the same distribution as [28] (see their paper for the discussion on the slighly better results compared to the ground truth estimate). The average running time of our solver on YUD is 5 ms, compared to 1.8 s for [28]. This corresponds to a speed-up of more than a factor of 300, which shows that in this case it is much more beneficial to saturate directly compared to introducing an auxiliary variable as in (8). Our fast solver is also more suited to be used in a RANSAC framework, to eliminate errors in the classification of the lines.

## 9. Conclusions

In this paper we have presented a new technique for building polynomial solvers for saturated ideals. In contrast to previous approaches the method avoids explicitly computing generators of the saturated ideal at runtime and instead lifts the problem into the original ideal. We believe that our method will help future research, enabling a whole new class of problems to be addressed in an efficient manner.

## References


[1] C. Aholt, S. Agarwal, and R. Thomas. A qcqp approach to triangulation. In *European Conference on Computer Vision (ECCV)*, 2012. 5

[2] E. Ask, Y. Kuang, and K. Åström. Exploiting p-fold symmetries for faster polynomial equation solving. In *International Conference on Pattern Recognition (ICPR)*, 2012. 1

[3] M. Bujnak, Z. Kukelova, and T. Pajdla. Making minimal solvers fast. In *Computer Vision and Pattern Recognition (CVPR)*, 2012. 1

[4] M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. In *Asian Conference on Computer Vision (ACCV)*, 2007. 2, 5

[5] M. Byröd, K. Josephson, and K. Åström. Fast and stable polynomial equation solving and its application to computer vision. *International Journal of Computer Vision (IJCV)*, 2009. 1, 2, 5, 6

[6] D. Cox, J. Little, and D. O'Shea. *Ideals, Varities and Algorithms*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, NY, USA, 2007. 2

[7] D. A. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 2005. 2

[8] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. SfM with MRFs: Discrete-continuous optimization for large-scale structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 35(12):2841–2853, December 2013. 6, 7

[9] P. Denis, J. H. Elder, and F. J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European Conference on Computer Vision (ECCV)*, 2008. 8

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

[11] R. Hartley and F. Schaffalitzky. $L_{inf}$ minimization in geometric reconstruction problems. In *Computer Vision and Pattern Recognition (CVPR)*, 2004. 5

[12] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding (CVIU)*, 68(2):146–157, 1997. 5

[13] J. Hedborg, A. Robinson, and M. Felsberg. Robust three-view triangulation done fast. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 152–157, 2014. 5

[14] D. Hook and P. McAree. Using sturm sequences to bracket real roots of polynomial equations. In *Graphics gems*, pages 416–422. Academic Press Professional, Inc., 1990. 1

[15] F. Kahl, S. Agarwal, M. K. Chandraker, D. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. *International Journal of Computer Vision (IJCV)*, 2008. 5

[16] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-sturm vs. optimal correction. In *British Machine Vision Conference (BMVC)*, 2008. 5

[17] Y. Kuang and K. Åström. Numerically stable optimization of polynomial solvers for minimal problems. In *European Conference on Computer Vision (ECCV)*, 2012. 1

[18] Y. Kuang, S. Burgess, A. Torstensson, and K. Åström. A complete characterization and solution to the microphone position self-calibration problem. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3875–3879. IEEE, 2013. 2, 5, 7, 8

[19] Y. Kuang, Y. Zheng, and K. Åström. Partial symmetry in polynomial systems and its applications in computer vision. In *Computer Vision and Pattern Recognition (CVPR)*, 2014. 1

[20] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision (ECCV)*, 2008. 1, 2

[21] Z. Kukelova, M. Bujnak, and T. Pajdla. Closed-form solutions to minimal absolute pose problems with known vertical direction. In *Asian Conference on Computer Vision (ACCV)*, 2010. 1

[22] Z. Kukelova, T. Pajdla, and M. Bujnak. Fast and stable algebraic solution to l2 three-view triangulation. In *International Conference on 3D Vision (3DV)*, 2013. 5

[23] V. Larsson and K. Åström. Uncovering symmetries in polynomial systems. In *European Conference on Computer Vision (ECCV)*, 2016. 1

[24] V. Larsson, K. Åström, and M. Oskarsson. Efficient solvers for minimal problems by syzygy-based reduction. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 4

[25] P. Lindstrom. Triangulation made easy. In *Computer Vision and Pattern Recognition (CVPR)*, 2010. 5

[26] F. Lu and R. Hartley. A fast optimal algorithm for l 2 triangulation. In *Asian Conference on Computer Vision (ACCV)*, 2007. 5

[27] Y. Min. L-infinity norm minimization in the multiview triangulation. In *International Conference on Artificial Intelligence and Computational Intelligence*, pages 488–494. Springer, 2010. 5

[28] F. M. Mirzaei and S. I. Roumeliotis. Optimal estimation of vanishing points in a manhattan world. In *International Conference on Computer Vision (ICCV)*, 2011. 5, 8

[29] H. M. Möller and H. J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numerische Mathematik*, 70(3):311–329, 1995. 1

[30] O. Naroditsky and K. Daniilidis. Optimizing polynomial solvers for minimal geometry problems. In *International Conference on Computer Vision (ICCV)*, 2011. 1

[31] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. In *Scandinavian Conference on Image Analysis (SCIA)*, 2011. 6

[32] 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. 1

[33] Z. Simayijiang, S. Burgess, Y. Kuang, and K. Åström. Minimal solutions for dual microphone rig self-calibration. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 2260–2264, Sept 2014. 7

[34] Z. Simayijiang, S. Burgess, Y. Kuang, and K. Åström. Toa-based self-calibration of dual-microphone array. *IEEE Journal on Selected Topics in Signal Processing*, 9(5):791–801, 2015. 2, 7

[35] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision (IJCV)*, 80(2):189–210, 2008. 6

[36] 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

[37] H. Stewenius, F. Schaffalitzky, and D. Nister. How hard is 3-view triangulation really? In *International Conference on Computer Vision (ICCV)*, 2005. 1, 2, 5

[38] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *Computer Vision and Pattern Recognition (CVPR)*, 2014. 1

[39] Y. Zheng, Y. Kuang, S. Sugimoto, K. Åström, and M. Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *International Conference on Computer Vision (ICCV)*, 2013. 1