How Hard is 3-view Triangulation Really?

Henrik Stewénius
Centre for Mathematical Sciences
Lund University, Sweden

Frederik Schaffalitzky
Robotics Research Group
University of Oxford, UK

David Nistér
Center for Visualization
University of Kentucky, USA

Abstract

We present a solution for optimal triangulation in three views. The solution is guaranteed to find the optimal solution because it computes all the stationary points of the (maximum likelihood) objective function.

Internally, the solution is found by computing roots of multi-variate polynomial equations, directly solving the conditions for stationarity.

The solver makes use of standard methods from computational commutative algebra to convert the root-finding problem into a 47×47 non-symmetric eigen-problem.

Although there are in general 47 roots, counting both real and complex ones, the number of real roots is usually much smaller. We also show experimentally that the number of stationary points that are local minima and lie in front of each camera is small but does depend on the scene geometry.

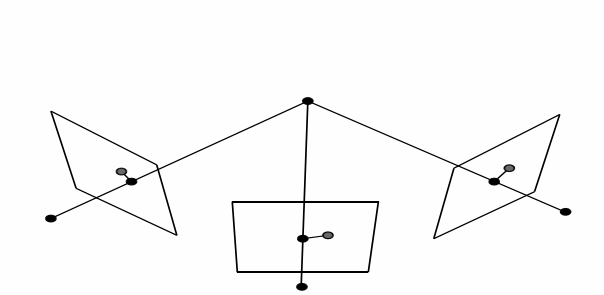

Figure 1. This paper presents an algorithm for optimal triangulation in three views. The algorithm is guaranteed to find the optimal solution, and can be used as a "black box" that accepts three image observations and camera poses and returns the optimal world point.

1 Introduction

Triangulation is a basic task in structure from motion, which is clearly defined and has been thoroughly studied. Finding the world point that minimizes the square-sum of image reprojection errors is widely accepted as the standard formulation. We refer to this as L_2-optimal triangulation, or just optimal triangulation.

Optimal triangulation in two views was solved by Hartley and Sturm [13], who gave a formulation that leads to six stationary points of the objective function. The six stationary points are extracted as the roots of a sixth-degree polynomial and the optimal solution is selected by evaluating the objective function.

It is natural to ask for the optimal solution for three or more views. In photogrammetry and computer vision it is accepted that three views often lead to greater stability and stronger disambiguation of the results [17, 10]. However, optimal triangulation for three views has not previously been solved in closed form (that is, without resorting to iterative methods which are not guaranteed to converge to the global optimum).

This paper presents a solution for optimal triangulation in three views. The solution is guaranteed to find the optimal solution. The solution can be used as a "black box" that accepts three image observations and camera poses and returns the optimal world point.

Internally, the solution is found by extracting the roots of a set of multi-variate polynomial equations. The roots correspond to stationary points in the objective function for triangulation. As in the two-view case, the optimal triangulation (among the finitely many stationary points) is found by straightforward evaluation of the objective function. There are 47 roots in total if we count real and complex solutions. They are found in three stages: firstly we compute a Gröbner basis for the polynomial equations, secondly we compute an *ideal quotient* to eliminate spurious solutions and thirdly we construct a so-called *action matrix* of size 47×47. The action matrix can be thought of as a *generalized companion matrix* and its eigen-decomposition gives the roots.

The paper is organized in three parts. The first part is introductory and explanatory. The next part (starting sec-

tion 4) is the "technical heart" with details for the interested reader. The last part (starting section 6) consists of an experimental section, discussion and conclusion. For a quick first reading, one could skip the central technical part.

2 Related Work

Triangulation has been used for a very long time in photogrammetry and is a basic tool in structure from motion. Traditionally it has been carried out by some simple ("linear") but sub-optimal procedure and is then refined to perfection by local optimization [25], where one typically minimizes the mean square errors of the image reprojections. A drawback of this is that convergence is not guaranteed. Hartley and Sturm [13] gave a procedure for optimal triangulation of a point seen in two views with known projection properties. Optimal here means a point that minimizes the mean square error, or L_2-norm of the reprojection errors. The procedure extracts the roots of a sixth-degree polynomial to find up to six stationary points, which are subsequently checked with reprojection to find the optimal point. One of the strengths of the above procedure is that it is projectively invariant, which is important when performing projective reconstruction.

A procedure for minimizing so-called directional error in two views was given in [22]. Although this procedure is not projectively invariant it can safely be used for calibrated cameras. It is also more computationally efficient since it only requires the solution of a quadratic polynomial. A balance between the two was given in [20], where optimal triangulation in two views according to L_∞-norm was solved. The procedure extracts the roots of a quartic polynomial and can therefore be carried out efficiently in closed form. It is projectively invariant and effectively keeps the reprojection errors equal in the two views. In [12] it was shown how to solve triangulation with L_∞-norm for any number of views as a convex optimization problem.

Although these triangulation problems are considered mainly when the camera positions are known, it is also of interest to know the triangulation error when solving for the cameras, since the triangulation error is used in scoring potential camera configurations. However, for efficiency reasons this is most often done with a sub-optimal approximation since the procedure has to be carried out a multitude of times and would otherwise become a bottleneck, see for example [27] for the case of two views and [24, 21] for three views.

Our solver is constructed using techniques from computational computational commutative algebra [4, 5, 16, 8, 9, 18]. Such techniques have been used to some extent to explore geometry problems in computer vision [6, 14, 15, 3, 23]. A common problem when applying these methods in floating point is that round-off errors quickly accumulate and it can be difficult to decide if a given coefficient should be zero or not. Our approach differs slightly from the naive approach in that we first determine the sequence of eliminations to be carried out, by doing the computations exactly over a finite field (using the computer algebra system Macaulay2 [16]). Once the sequence of eliminations has been determined, we implement them in floating point using matrix operations. This has the advantage that we can use numerically stabler orthogonal pivoting instead of simple Gaussian elimination.

3 Objective function

We are given three 3×4 projection matrices $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ and three image points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. Given a 3D world point $\mathbf{X}$ we project it to the images and compute the distances $d(\mathbf{P}_i\mathbf{X}, \mathbf{x}_i)$. The objective function that we want to minimize is the sum of squared distances:

$$C(\mathbf{X}) = \sum_{i=1}^{3} d(\mathbf{P}_i\mathbf{X}, \mathbf{x}_i)^2. \tag{1}$$

By translating image coordinates we may, without loss of generality, assume that each image point $\mathbf{x}_i$ is at the origin. Then the terms of the cost are:

$$d(\mathbf{P}_i\mathbf{X}, \mathbf{x}_i)^2 = \frac{|\mathbf{P}_i(1,:)\mathbf{X}|^2 + |\mathbf{P}_i(2,:)\mathbf{X}|^2}{|\mathbf{P}_i(3,:)\mathbf{X}|^2}, \tag{2}$$

where $\mathbf{P}_i(r,:)$ denotes the rth row of $\mathbf{P}_i$. For a stationary point of the cost C we require

$$\frac{\partial C}{\partial \mathbf{X}_j} = 0, \tag{3}$$

for each $j = 1, 2, 3, 4$. Multiplying these equations by suitable denominators gives polynomial equations in the four unknowns of $\mathbf{X}$ and these are the equations that we solve in this paper.

This approach may seem unwieldy and to simplify the algebra we choose a particular coordinate system in which the last rows of the camera projection matrices are as follows

$$\begin{array}{rcl} \mathbf{P}_1(3,:) & = & \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \end{array}\right], \\ \mathbf{P}_2(3,:) & = & \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \end{array}\right] \text{ and} \\ \mathbf{P}_3(3,:) & = & \left[\begin{array}{cccc} 0 & 0 & 1 & 0 \end{array}\right]. \end{array} \tag{4}$$

We call the homogeneous coordinates $\mathbf{X} = (x, y, z, t)$ and consider the equations

$$C_x = C_y = C_z = C_t = 0, \tag{5}$$

where subscripts denote partial derivatives. Multiplying these four functions by $x^3y^2z^2$, $x^2y^3z^2$, $x^2y^2z^3$ and $x^2y^2z^2$ (respectively) gives polynomial equations that are

homogeneous of degrees $6, 6, 6, 5$ (respectively). Because C is a homogeneous function (of degree 0) it follows (for example, by "Euler's Homogeneous Function Theorem") that

$$tC_t + xC_x + yC_y + zC_z = 0, \tag{6}$$

and so the polynomial equations are not independent: the quintic equation can be obtained by adding the three sextic equations and dividing by t. For this reason we ignore the quintic equation. Also, to remove the scale ambiguity of homogeneous coordinates we set $t = 1$ and work with non-homogeneous coordinates x, y, z from now on.

In summary, we have reduced the problem of finding stationary points of C to that of solving three simultaneous polynomial equations of degree six in the variables x, y, z. There is some discernible structure in the coefficients of the equations but we do not give them here for lack of space (they can in any case be computed easily with a symbolic computer algebra program such as Maple). We are only interested in solutions with $x, y, z \neq 0$ since the vanishing of a variable corresponds to a point on the principal plane of a camera.

The solutions to our three sextic equations can be thought of as forming a subset of $\mathbb{C}^3$. Exploration with the computer algebra system Macaulay2 [16] shows that (and this can be verified with pen and paper but we do not have space to do so) the subset consists of:

- 47 isolated points with $xyz \neq 0$,
- the three camera centers and
- the union of the three coordinate axes.

See Figure 2 for an illustration. Since only the 47 solutions are of interest to us, we need a way to suppress the unwanted solutions that have vanishing coefficients. This is addressed in the next section.

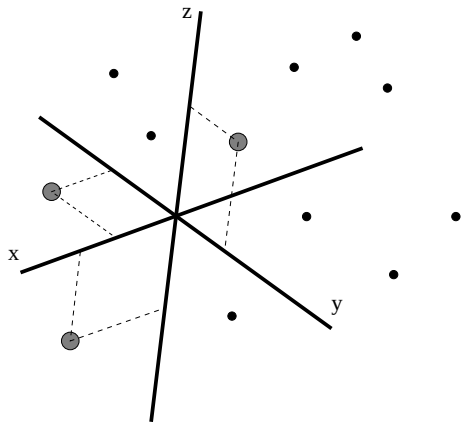

Figure 2. Diagrammatic picture of the zero-set, in 3D, of the three sextic equations. The solid lines are the three coordinate axes, the fat grey points are the camera centers and the small black points represent some (the real ones, say) of the 47 solutions that we want to find.

4 Algebraic background

Computational commutative algebra is a big subject and there is not space in this paper (or even in a much expanded version) to give a self-contained treatment. Luckily there are good sources that give complete treatments of the tools we use, for example [4, 5, 7].

The reader should know about (polynomial) rings and ideals, monomial orders, the notion of leading terms and Gröbner bases. The connections between polynomial equations, the ideal they generate and the zero-set (variety) of the ideal will also be important.

We generally work with graded reverse lexicographic order (GREVLEX) because it tends to require less computation than obvious alternatives such as simple lexicographic ordering and turns out to be convenient for the ideal saturation we carry out below.

An outline of the story in this section is as follows. We have three polynomial equations $f_1 = f_2 = f_3 = 0$ in three variables x, y, z. We replace these equations with the *ideal* $I = (f_1, f_2, f_3)$ that they generate. As mentioned in the previous section, this ideal admits (infinitely many) solutions that we are not interested in because they lie on the locus $\{xyz = 0\}$, which is the union of the principal planes of the cameras.

To remove these solutions we replace the ideal I with its *saturation*

$$J = \mathrm{sat}(I, f) = \left\{ p \mid f^k p \in I \text{ for some } k \geq 0 \right\} \tag{7}$$

with respect to xyz. This is an ideal that has the same solutions as I, except for those that lie on the locus $xyz = 0$ (technically, the saturation removes from I any primary component supported on the zero-set of the ideal (xyz); see [1] for material on this). Intuitively, the equations in the saturated ideal are those vanishing conditions p that *together with* f account for the vanishing conditions in I, and this explains why we get an ideal which does not vanish where f does.

Having arrived at an ideal with a finite number of solutions (47 to be precise) we follow a standard recipe [5] for computing the solutions via an eigenproblem: Consider the quotient ring $R = \mathbb{C}[x, y, z]/J$, which will be a vector space of dimension 47. A basis is furnished by the set of monomials $u_i(x, y, z)$ that are not divisible by any leading term from (a Gröbner basis of) J. These monomials are gathered in a vector $\mathbf{u}(x, y, z)$ of length 47. Any element of the quotient ring can now be represented by a linear combination of the elements of $\mathbf{u}(x, y, z)$. For any polyno-

mial f, multiplication by f gives a linear operator on R and so it has a 47×47 *action matrix* A_f with respect to the basis of monomials. The basic fact we need about this is that if (x, y, z) is a solution to our ideal, then $\mathbf{u}(x, y, z)$ is a left eigenvector of the action matrix A_f, with eigenvalue $f(x, y, z)$. This means that the 47 left eigenvectors of the action matrix A_f are (scaled) versions of the monomial vector $\mathbf{u}(x, y, z)$ evaluated at the 47 solutions.

Note that we have the freedom to choose a suitable polynomial f. Given a Gröbner basis for J, it is straightforward to compute the action matrix corresponding to any polynomial, but with GREVLEX order, the monomials x, y, z (the variables themselves) are typically present in the monomial basis and so it is convenient to simply choose $f = x$. We compute the action matrix A_x, and then read off the solutions directly from the left eigenvectors.

5 Numerical scheme

To implement our method numerically we represent systems of polynomials as matrices where each row corresponds to a polynomial and each column corresponds to a monomial. Gathering the monomials in a vector $\mathbf{X}$ and the coefficients in a matrix $\mathbf{M}$, the polynomial equations can be written $\mathbf{MX} = \mathbf{0}$. In this representation multiplication and division by monomials correspond to moving entries between columns.

For a given system of polynomials, we generally list the monomials in decreasing order (highest degrees first), so that leading monomials come first in each row. Using row eliminations we can put the matrix into upper echelon form, in the process discovering linear dependencies among generators and exposing leading terms of our ideal.

There are several figures in the paper showing the sparsity patterns of the matrices intermediate in the algorithm. The intricate sequence of sparsity structures is an important property of the problem (or ideal) that would be hard to determine in floating point arithmetic because of the difficulty of deciding whether a given coefficient should vanish, or is merely very small. To overcome this problem, we determined the sparsity patterns ahead of time using symbolic calculations over finite fields, where arithmetic is exact. Once this "template" of eliminations was determined, we ported it to a floating point implementation written in C++.

Standard methods for computing with ideals, such as Buchberger's algorithm [2, 4], correspond to certain fixed sequences of row operations that take into account only whether leading coefficients vanish or not. These methods generally assume exact arithmetic and in effect carry out eliminations in a fixed order and *without any pivoting strategy*. In contrast, we are working with floating point arithmetic and need to use a numerically sound pivoting strategy. We implemented both LU-like pivoting (where a pivot is sought to be maximal in its column) and QR-like orthogonal row operations (Householder reflections) to reach upper echelon forms. For most purposes it suffices to reduce the matrix to upper echelon form, but to make our action matrices we require a submatrix that is an identity matrix and for this we used LU / Gauss-Jordan elimination.

Initially, we have a system of three equations of degree six. After elimination we arrive at a matrix with the sparsity pattern shown in Figure 4, corresponding to degrees $6, 6, 5$ (the new quintic is actually the quintic that we ignored in section 3). This system corresponds to the ideal I whose saturation $J = \mathrm{sat}(I, xyz)$ we will compute. The saturation is computed in three steps at the end of which we get a Gröbner basis for the saturated ideal J.

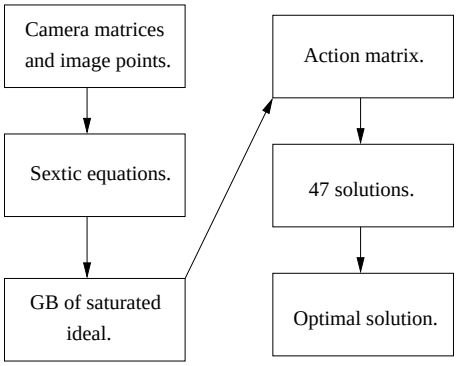

Figure 3. Data flow for the solver.

Partial saturation step

Each of these steps picks a permutation x_1, x_2, x_3 of the variables x, y, z and follows a recipe (to be given next) to produce new ideal generators. A full round of partial saturation consists of applying the partial saturation step for each permutation of x, y, z to the same input ideal. The recipe is described next.

We are given an ideal I_t with generators of degrees 5 and 6. We wish to compute and include some of the polynomials of $\mathrm{sat}(I_t, x_1)$ to form a new ideal I_{t+1}. It turns out that we can arrange for I_{t+1} to also have generators of degrees 5 and 6.

We multiply the generators of I_t with all possible monomials in x_2, x_3 such that the result has degree at most 7. These new generators are added to the old generators and monomials (matrix columns) are ordered in increasing degree of x_1. A row elimination step is carried out and any generator divisible by x_1 or x_1^2 is divided by this factor.

For illustration, the system before elimination is shown in Figure 8. The lines in the figures show where the degree if x_1 change.

The corresponding systems after elimination are shown in Figure 9. Careful scrutiny of these figures shows there are polynomials divisible by x_1 and x_1^2. The possible divisions are performed and this is how new polynomials are added to the generator sets.

Complete saturation process

After the first round of partial saturation the system is as shown in Figure 5. This system has full rank and we gain no additional equations of degree five by elimination.

This set of equations is taken into the second round of partial saturation, the result of this round have a large number of linearly dependent equations so we perform linear elimination on this system, the resulting system is shown in Figure 6.

These equations we take into a third round of partial saturation and the resulting equations are eliminated to get a system with a diagonal as shown in Figure 7. This matrix contains 37 polynomials which constitute a GREVLEX Gröbner basis.

From the last matrix the action matrix A_x for multiplication by x in the quotient ring $\mathbb{C}[x, y, z]/\operatorname{sat}(I, xyz)$ is extracted, simply by copying partial rows.

6 Experiments

6.1 Eigensolver and numerical precision

To compute the eigenvectors of the 47×47 action matrices from the previous section, we used the standard method [11] of reduction to Hessenberg form followed by implicit double-shift QR iterations to reach a real Schur decomposition. From this it is easy to extract the real eigenvalues and eigenvectors (complex solutions are of no interest). All arithmetic is carried out in high-precision floating point with 128 bits of mantissa, using the MPFR library [19]. The use of the high precision numerics is very costly in terms of computational time and our running time is $30s$.

6.2 Reality check: Oxford dinosaur data

Although we have validated the correctness of our method on synthetic data (where the right answer is known) it makes sense to try it on a standard real data set. We chose to use the well-known Oxford dinosaur [26] reconstruction with 36 camera matrices and 4983 point tracks. For each point track extending over three or more views we picked the first, middle and last frames and applied our triangulation solver to those three cameras to get a 3D point. (Note that this means that different points may be reconstructed using different cameras.) There are tracking errors in the

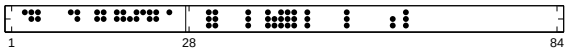

Figure 4. System after first elimination. The first 27 monomials (box) are of degree 6.

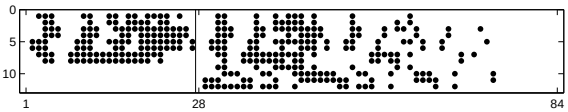

Figure 5. System after first round of partial saturation. The system has full rank so no elimination are used.

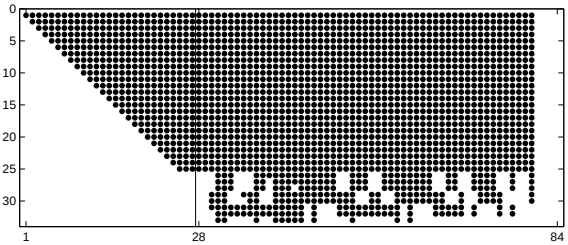

Figure 6. System after the second round of partial saturation and linear elimination, no linear elimination is applied to the 8 polynomials of degree 5.

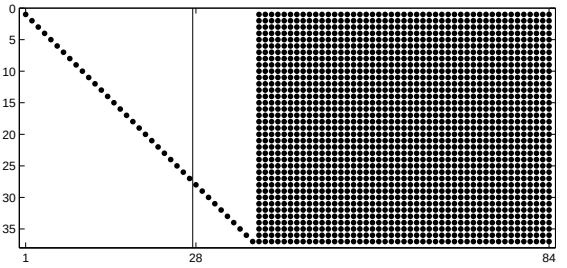

Figure 7. After the third round of partial saturation and an elimination step we have a Gröbner basis.

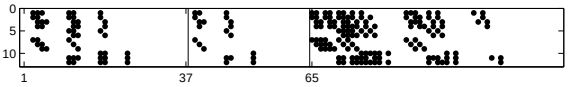

Figure 8. The expanded equations of one of the partial saturation steps of the first round. The first 37 monomials are in x_1^0, 38 to 65 in x_1^1 and the remaining in at least x_1^2.

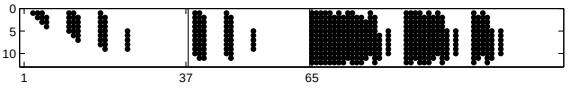

Figure 9. The reduced system during a step in the first round of partial saturation.

data and we suppressed any solution with a reprojection error greater than 2.0 pixels in any of the three images. The final result is 2577 points in 3D and a view of the point cloud reconstruction is shown in Figure 10.

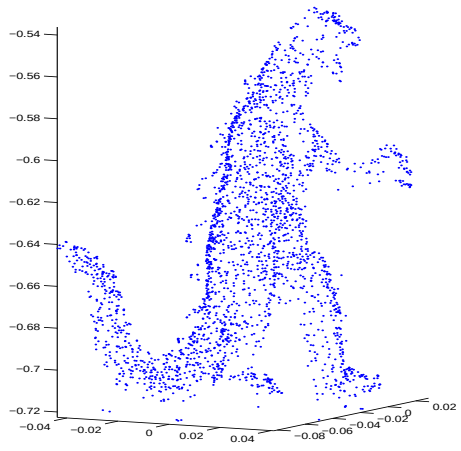

Figure 10. A view of 2577 points triangulated from the 36 cameras from the Oxford dinosaur reconstruction.

6.3 Synthetic data experiments

These experiments were designed to gather statistics on typical behaviors of the cost function and the solver. We randomly generated about 200,000 camera and 3D point configurations and added synthetic noise to the projected image points.

6.3.1 Camera geometries

The camera geometry we tested is a "turn-table" one, where the three cameras are approximately situated on a unit circle and view a 3D point about two units away, illustrated in Figure 11.

Images were taken to be 512×768 and focal length was taken to be 1000 pixels. The unperturbed cameras look directly at the origin, and a random perturbation of orientation with standard deviation of about 4 degrees was applied. The camera positions were also perturbated by normal noise with standard deviation 0.05 units. The 3D point was perturbed by an isotropic normal distribution with diagonal entries 0.1. Artificial normal image noise with standard deviation of 0.5 pixels was added.

We varied the "vergence angle" (the angular increment in the turn-table motion) from 5 to 10 to 20 to 30 degrees.

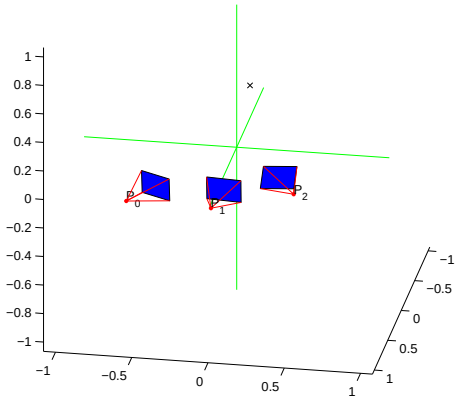

Figure 11. One example of the turn-table camera geometry used for experimental evaluation. The "vergence angle" (here 30 degrees) between consecutive views is varied in the experiments.

6.3.2 Numerical stability of the solver

To judge the numerical stability of the solver we started a local optimization at each real stationary point, and compared the smallest (over all real solutions) cost before and after. We found there was almost no drop in cost, which shows that the solver does accurately localize (local) minima. Statistics are shown in Figure 12.

6.3.3 Comparison with "linear" estimate + bundle

The natural competitor of our method is to initialize a 3D point using a "linear" method based on SVD of a 6×4 design matrix, followed by direct local optimization of the cost. We tested this approach on each random camera configuration generated, to see how often it yielded the global optimum. Figure 13 shows the difference that we get in cost with this method, compared to our global method.

6.3.4 Classification of real solution types

Every real solution $\mathbf{X}$ is further classified by two criteria. Firstly, as it is a stationary point, the cost has vanishing derivatives there but it might be a saddle point and not a local minimum. So we evaluate the Hessian of the cost function at $\mathbf{X}$, and record its signature (signs of eigenvalues). There are three cases: $+++$ which corresponds to a local minimum, $++-$ which is a saddle point, $+--$ which is also a saddle point and $---$ which is a local maximum (this would seem to be a rare case but there is no reason *a priori* why it could not happen). Secondly, it is determined whether the 3D world point lies in front of

each camera, or not. For a given camera setup this divides each real solutions into eight categories, summarized in the following table.

	vergence 5°		vergence 30°	
	in front	behind	in front	behind
minimum: + + +	67759	14056	45835	36691
saddle I: + + –	94	1207	120	1818
saddle II: + – –	0	0	0	0
maximum: – – –	0	0	0	0

The numbers show total numbers of solutions of each type, for runs of 200,000 camera geometries. Only the results for vergence angles of 5 and 30 degrees are shown, but other angles gave very similar results. For practical purposes, the most interesting statistic is the top left cell, which is the number of local minima in front of each camera. A method based on local optimization would have to correctly choose the right one so it is of interest to know how many such minima to expect. This is shown in Figure 14.

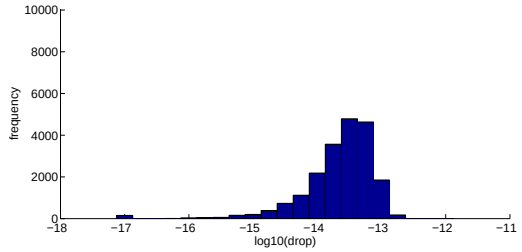

Figure 12. Histogram of absolute value of drop in cost as a result of local optimization, taken over about 200,000 camera configurations. The result is for a vergence angle of 5 degrees but is typical of other angles too.

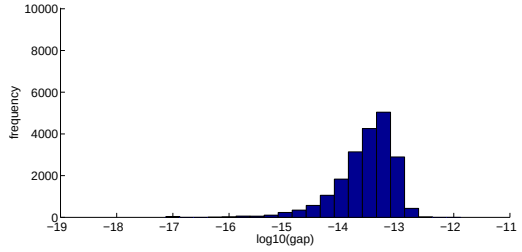

Figure 13. Histogram of absolute value of difference in final cost between our global method and the baseline method of "linear" estimate followed by local optimization. The result is for a vergence angle of 5 degrees but is typical of other angles too.

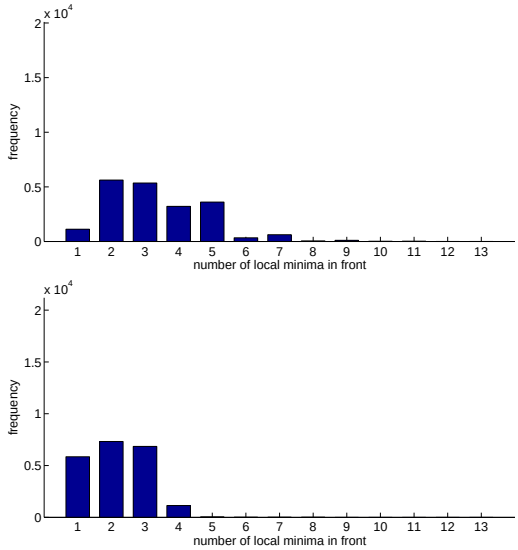

Figure 14. Histogram showing the number of local minima in front of each camera, collected over a run of about 200,000 camera geometries. The vergence angle was 5 degrees (top) and 30 degrees (bottom), respectively.

6.4 Discussion

We have shown experimentally that our method works well on both real and synthetic data. From the statistics gathered for turn-table geometries we make some observations: Firstly, we did not observe a single local maximum, or saddle point of type $+--$ in any of the 800,000 runs that we did. Secondly, the number of local minima in front of all three cameras is typically in the range 1–5 but it can be as large as 13. Thirdly, smaller vergence angles tend to have more local minima, as one might expect. Fourthly, linear initialization followed by local optimization found the global optimum in all cases that we tried.

7 Conclusions

We have given a closed-form solution for optimal triangulation in three views, a classical and previously unsolved problem. Previous solutions were based on local optimization and came with no guarantee of global optimality.

The solution was built using standard tools from computational commutative algebra, directly solving the conditions of stationarity. Previous uses of polynomial equation solvers in computer vision have tended to focus on "minimal cases" where the number of unknowns equal the number of constraints. In contrast, in this paper we solve an overconstrained (maximum likelihood) estimation problem in closed form.

A benefit of our approach is that it gives statistics about the problem that were never available before. For example, how hard is three-view triangulation really? In this paper we showed that even when cheirality is enforced, there can be multiple local minima of the cost function, and we have reported how many there can typically be. On the other hand we have shown experimentally that, in the turn-table setups we investigated, the classical approach of "linear" initialization + bundling gave the global optimum in all cases.

We have validated through experiments that our solution procedure is numerically sound. Future work will concentrate on improving the numerical stability and computational efficiency of the method. The goal would be an IEEE double-precision implementation that runs in a few milliseconds, obviating iterative methods based on local optimization.

A limitation of our method is the assumption that the three principal planes are distinct (this was necessary to make the simplifying change of coordinates). The case where two or all planes coincide is important as a model of the case where they nearly coincide, and it would be possible to explicitly treat that case by the same methods as used here.

References

[1] M. F. Atiyah and I. G. MacDonald, *Introduction to Commutative Algebra*, 1969.

[2] B. Buchberger, An algorithmic criterion for the solvability of algebraic systems of equations, *Aequationes Math.*, 4(3):374-383, 1970.

[3] S. Carlsson, D. Weinshall, Dual Computation of Projective Shape and Camera Positions from Multiple Images, *International Journal of Computer Vision*, 27(3):1-16, 1998.

[4] D. Cox and J. Little and D. O'Shea, *Ideals, Varieties, and Algorithms*, 1997.

[5] D. Cox and J. Little and D. O'Shea, *Using Algebraic Geometry*, 1998.

[6] M. Demazure, Sur Deux Problemes de Reconstruction, *Technical Report No 882, INRIA, Rocquencourt, France*, 1988.

[7] D. Eisenbud, *Commutative Algebra with a View Toward Algebraic Geometry*, Springer-Verlag, 1995.

[8] M. Elkadi and B. Mourrain, *Symbolic-numeric tools for solving polynomial equations and applications*, Springer, 2005.

[9] I. Emiris and J. Canny, Efficient incremental algorithms for the sparse resultant and the mixed volume, *Journal of Symbolic Computation*, 20(2):117-149, 1995.

[10] A. Fitzgibbon and A. Zisserman, Automatic Camera Recovery for Closed or Open Image Sequences, *Proc. European Conference on Computer Vision*, pp. 311-326, 1998.

[11] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1996.

[12] R. Hartley, and F. Schaffalitzky, L-∞ minimization in geometric reconstruction problems *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington DC, 2004.

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

[14] R. Holt, T. Huang and A. Netravali, Algebraic Methods for Image Processing and Computer Vision, *IEEE Transactions on Image Processing*, 5(6):976:986, 1996.

[15] R. Holt and A. Netravali, Number of Solutions for Motion and Structure from Multiple Frame Correspondence, International Journal of Computer Vision, 23:5-15, 1997.

[16] D. Grayson and M. Stillman, *Macaulay 2*, 1993-2002. `http://www.math.uiuc.edu/Macaulay2/`

[17] C. McGlone, Editor, E. Mikhail and J. Bethel, Associate Editors, *Manual of Photogrammetry*, 5th Edition, ISBN 1-57083-071-1, ASPRS, 2004.

[18] D. Manocha and J. Canny, *Multipolynomial resultant algorithms*, *Journal of Symbolic Computation*, 15(2):99-122, 1993.

[19] *The MPFR library*. `http://www.mpfr.org/`

[20] D. Nist´er. *Automatic dense reconstruction from uncalibrated video sequences*, PhD Thesis, Royal Institute of Technology KTH, ISBN 91-7283-053-0, March 2001.

[21] D. Nist´er, An Efficient Solution to the Five-Point Relative Pose Problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756-770, June 2004.

[22] J. Oliensis, Exact Two-Image Structure from Motion, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(12):1618-1633, 2002.

[23] L. Quan, B. Triggs, M. Ameller and B. Mourrain. Uniqueness of Minimal Euclidean Reconstruction from 4 Points, *obtained through personal communication*.

[24] P. Torr and A. Zisserman, Robust Parameterization and Computation of the Trifocal Tensor, *Image and Vision Computing*, 15:591-605, 1997.

[25] B. Triggs, P. McLauchlan, R. Hartley and A. Fitzgibbon, Bundle Adjustment - a Modern Synthesis, *Springer Lecture Notes on Computer Science*, Springer Verlag, 1883:298-375, 2000.

[26] Visual Geometry Group, University of Oxford. `http://www.robots.ox.ac.uk/~vgg/`

[27] Z. Zhang, Determining the Epipolar Geometry and its Uncertainty: a Review, *International Journal of Computer Vision*, 27(2):161-195, 1998.