Triangulation of Points, Lines and Conics

Josephson, Klas; Kahl, Fredrik

Published in:
Image Analysis (Lecture Notes in Computer Science)

DOI:
10.1007/978-3-540-73040-8_17

2007

Triangulation of Points, Lines and Conics

Klas Josephson and Fredrik Kahl

Centre for Mathematical Sciences,
Lund University, Lund, Sweden
{klasj, fredrik}@maths.lth.se

Abstract. The problem of reconstructing 3D scene features from multiple views with known camera motion and given image correspondences is considered. This is a classical and one of the most basic geometric problems in computer vision and photogrammetry. Yet, previous methods fail to guarantee optimal reconstructions - they are either plagued by local minima or rely on a non-optimal cost-function. A common framework for the triangulation problem of points, lines and conics is presented. We define what is meant by an optimal triangulation based on statistical principles and then derive an algorithm for computing the globally optimal solution. The method for achieving the global minimum is based on convex and concave relaxations for both fractionals and monomials. The performance of the method is evaluated on real image data.

1 Introduction

Triangulation is the problem of reconstructing 3D scene features from their projections. Naturally, since it is such a basic problem in computer vision and photogrammetry, there is a huge literature on the topic, in particular, for point features, see [1,2]. The standard approach for estimating point features is:

(i) Use a linear least-squares algorithm to get an initial estimate.
(ii) Refine the estimate (so called *bundle adjustment*) by minimizing the sum of squares of reprojection errors in the images.

This methodology works fine in most cases. However, it is well-known that the cost-function is non-convex and one may occasionally get trapped in local minima [3]. The goal of this paper is to develop an algorithm which computes the globally optimal solution for a cost-function based on statistical principles [4].

In [3], the two-view triangulation problem for points was treated. The solution to the optimal problem was obtained by solving a sixth degree polynomial. This was generalized for three views in [5], but the resulting polynomial system turns out to be of very high degree and their solution method based on Gröbner bases becomes numerically unstable. In [6] convex linear matrix inequalities (LMI) relaxations are used to approximate the non-convex cost-function (again, in the point case), but no guarantee of actually obtaining the global minimum is provided. For line and conic features, the literature is limited to closed-form solutions with algebraic cost-functions and to local optimization methods, see [1] and the references therein.

B.K. Ersbøll and K.S. Pedersen (Eds.): SCIA 2007, LNCS 4522, pp. 162–172, 2007.

In this paper, we present a common framework for the triangulation problem for any number of views and for three different feature types, namely, points, lines and conics. An algorithm is presented which yields the global minimum of the statistically optimal cost-function. Our approach is most closely related to the work in [7], where fractional programming is used to solve a number of geometric reconstruction problems including triangulation for points. Our main contributions are the following. First, we show how a covariance-weighted cost-function - which is the statistically correct thing to consider - can be minimized using similar techniques as in [7] for the point case. For many point and corner detectors, e.g., [8,9], it is possible to obtain information of position uncertainty of the estimated features. Second, we present a unified framework for the triangulation problem of points, lines and conics and the corresponding optimal algorithms. Finally, from an algorithmic point of view, we introduce convex and concave relaxations of monomials in the optimization framework in order to handle Plücker constraints appearing in the line case.

2 Projective Geometry

In the triangulation of points, lines and conics, it is essential to have the formulation of the projection from the three dimensional space to the two dimensional image space in the same way as the standard projection formulation used in the point case. For that reason we begin with a short recapitulation of the projection of points with a standard pinhole camera. After that the methods to reformulate the projection of lines and quadrics into similar equations are considered. For more reading on projective geometry see [1].

2.1 Points

A perspective/pinhole camera is modeled by,

$$\lambda x = PX, \quad \lambda > 0, \tag{1}$$

where P denotes the camera matrix of size 3×4. Here X denotes the homogeneous coordinates for the point in the 3D space, $X = [U\ V\ W\ 1]^T$, and x denote the coordinates in the image plane, $x = [u\ v\ 1]$. The scalar λ can be interpreted as the depth, hence $\lambda > 0$ if the point appears in the image.

2.2 Lines

Lines in three dimensions have four degrees of freedom - a line is determined by the intersection of the line with two predefined planes. The two intersection points on the two planes encode two degrees of freedom. Even if lines only have four degrees of freedom, there is no universal way of representing every line in $\mathbb{P}^4$. One alternative way to represent a line is to use Plücker coordinates. With Plücker coordinates, the line is represented in an even higher dimensional space $\mathbb{P}^5$. The over parameterization is hold back by a quadratic constraint that has to

be fulfilled for every line. In [1] definitions and properties for Plücker coordinates are described. The big benefit with Plücker coordinates is the Plücker camera that makes it possible to write the projection of lines as $\lambda l = P_{\mathcal{L}}\mathcal{L}$. A drawback on the other hand is that they have to fulfill the quadratic constraint

$$l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0, \tag{2}$$

otherwise projection of lines can be formulated in the same manner as point projections, but now it is a projection from projective space of dimension 5 to the image. Hence the line camera matrices are of dimension 3×6.

2.3 Conics

As for lines, we are interested in writing the projection of a quadric to an image conic in the form of the projection formula for points. To do that we use the projection formula of the duals to the quadric conics. These duals are the envelopes of the structures. For conics, the envelope consists of lines and for quadrics, the envelope consist of all planes tangent to the quadric locus. Provided the quadrics and conics are non-degenerate, one can show that the equations for the duals are, $U^T L U = 0$, where U are homogeneous plane coordinates and $L = C^{-1}$. Similar for conics, one gets, $u^T l u = 0$, where u are homogeneous line coordinates and $l = c^{-1}$. The projection for the envelope forms are,

$$\lambda l = PLP^T \quad \lambda \neq 0. \tag{3}$$

Now we want to reformulate (3) so it appears in a similar way as the point projection formula. This can be done in the form, $\lambda \tilde{l} = P_C \tilde{L}$, where $\tilde{l}$ and $\tilde{L}$ are column vectors of length 6 and 10 obtained from stacking the elements in l and L. P_C is an 6×10 matrix. The entries in P_C are quadratic expressions in P.

As for the line case, it is not possible to make the interpretation that the scalar λ of the projected conic corresponds to the depth.

3 Triangulation

In triangulation the goal is to reconstruct a three dimensional scene from measurements in N images, $N \geq 2$. The camera matrices P_i, $i = 1 \ldots N$, are considered to be known for each image. In the point case, the camera matrix can be written $P = (p_1, p_2, p_3)^T$, where p_j is a 4–vector. Let $(u, v)^T$ denote the image coordinates and $X = (U, V, W, 1)^T$ the extended coordinates of the point in 3D. This gives the reprojection error

$$r = \left(\frac{u\ p_3^T X - p_1^T X}{p_3^T X}, \frac{v\ p_3^T X - p_2^T X}{p_3^T X} \right), \tag{4}$$

Further, $\sum_{i=1}^{N} \|r_i\|_2^2$ is the objective function to minimize if the smallest reprojection error is to be achieved in L_2-norm. To use the optimization algorithm

proposed in this paper (see next section), it is necessary to write the error in each image as a rational function f/g where f is convex and g concave.

It is easy to see that the L_2-norm of the residual in (4) can be written as $\|r\|^2 = ((a^T X)^2 + (b^T X)^2)/(p_3^T X)^2$, where a, b are 4-vectors determined by the image coordinates and the elements of the camera matrix. By choosing $f = ((a^T X)^2 + (b^T X)^2)/(p_3^T X)$ (with the domain $p_3^T X > 0$) and $g = p_3^T X$, one can show that f is indeed convex and g concave. It is straight forward to form the same residual vectors in the line and conic cases - the only difference is that the dimension is different.

3.1 Incorporation of Covariance

The optimal cost-function is to weight the residual vector by its covariance [4] (at least to a first order approximation). Incorporating covariance weighted error transforms to,

$$\|Lr\| = \left\| L \left(\frac{x_1 p_n^T X - p_1^T X}{p_n^T X}, \dots \right) \right\|, \tag{5}$$

where L is the cholesky factorization of the inverse covariance matrix to the structure in each image. Notice that we have chosen to normalize by the last coordinate and in that case the covariance becomes a 2×2 symmetric matrix in the point and line cases and a 5×5 matrix in the conic case. The reason why the covariance matrix is one dimension lower than the image vector is that there is no uncertainty in the last element of the extended image coordinates.

3.2 Problem Formulation

In all of the above cases, the optimization problem to solve is the following:

$$\min \sum_{i=1}^{n} \|L_i r_i\|^2. \tag{6}$$

The only thing which differs (except for dimensions) in the different cases is that in the line case it is necessary to fulfill the quadratic Plücker constraint (2) for the coordinates of the three dimensional structure.

4 Branch and Bound Optimization

Branch and bound algorithms are used to find the global optimum for non-convex optimization problems. The algorithm gives a provable upper and lower bound of the optimum and it is possible to get arbitrary close to the optimum.

On a non-convex, scalar-valued objective function Φ at the domain Q_0 the branch and bound algorithm works by finding a lower bound to the function Φ on the domain Q_0. If the difference between the optimum for the bounding functions and the lowest value of the function Φ - calculated so far - is small

enough, then the optimum is considered to be found. Otherwise the domain Q_0 is splitted into subdomains and the procedure is repeated in these domains.

If the lower bound on a subdomain has its optimum higher than a known value of the objective function in another subdomain it is possible to neglect the first subdomain since we know that the optimum in that region is greater than the lowest value obtained so far.

4.1 Bounding

The goal of the bounding function Φ_{lb} is that it should be (i) a close underestimator to the objective function Φ and (ii) easy to compute the lowest value Φ_{lb} in given domain. Further, as the domain of the bounding functions is partitioned into smaller regions, the approximation gap to the objective function must converge (uniformly). A good choice of Φ_{lb} is the convex envelope [7].

Fractional Relaxation. Fractional programming is used to minimize/maximize a sum of $p \geq 1$ fractions subject to convex constraints. In this paper we are interested of minimizing

$$\begin{aligned} \min_x \ & \sum_{i=1}^{p} \frac{f_i(x)}{g_i(x)} \\ \text{subject to } \ & x \in D, \end{aligned} \tag{7}$$

where f_i and g_i are convex and concave, respectively, functions from $\mathbb{R}^n \to \mathbb{R}$, and the domain $D \subset \mathbb{R}^n$ is a convex set. On top of this it is assumed that f_i and g_i are positive and that one can compute a priori bounds on f_i and g_i. Even under these assumptions it can be shown that the problem is $\mathcal{NP}$-complete [10].

It is showed in [7] that if you have bounds on the domain D it is possible to rewrite (7) to a problem that is possible to find the convex envelope to for every single fraction by a *Second Order Cone Program* (SOCP) [11].

When Φ is a sum of ratios as in (7) a bound for the function can be calculated as the sum of the convex envelopes of the individual fractions. The summarized function will be a lower bound and it fulfills the requirements of a bounding function. This way of calculating Φ_{lb} by solving a SOCP problem can be done efficiently [12].

A more exhaustive description on fractional programming in multiple view geometry can be found in [7] where point triangulation (without covariance weighting) is treated.

Monomial Relaxation. In the line case the Plücker coordinates have to fulfill the Plücker constraint (2). This gives extra constraints in the problem to find lower bounds.

If we make the choice in the construction of the Plücker coordinates that the first point lies on the plane $z = 1$ and the second on the plane $z = 0$, remember

that the Plücker coordinates are independent of the construction points, the two points $X = (x_1,\ x_2,\ 1,\ 1)^T$ and $Y = (y_1,\ y_2,\ 0,\ 1)^T$ gives the following Plücker coordinates for the line (2.2),

$$\mathcal{L} = (x_1y_2 - x_2y_1,\ -y_1,\ x_1 - y_1,\ -y_2,\ y_2 - x_2,\ 1)^T. \tag{8}$$

This parameterization involves that the last coordinate is one and that only the first one is nonlinear to the points of intersection. Hence it is only necessary to make a relaxation of the first coordinate (in addition to the fractional terms).

In [13] the convex and the concave envelopes are derived for a monomial y_1y_2. The convex and the concave envelopes are given by,

$$\mathbf{convenv}(y_1y_2) = \max \left\{ \begin{array}{l} y_1y_2^U + y_1^U y_2 - y_1^U y_2^U \\ y_1y_2^L + y_1^L y_2 - y_1^L y_2^L \end{array} \right\}, \tag{9}$$

$$\mathbf{concenv}(y_1y_2) = \min \left\{ \begin{array}{l} y_1y_2^L + y_1^U y_2 - y_1^U y_2^L \\ y_1y_2^U + y_1^L y_2 - y_1^L y_2^U \end{array} \right\}. \tag{10}$$

Given bounds on x_1, x_2, y_1 and y_2 in the parameterization of a line, it is possible to propagate the bounds to the monomials x_1y_2 and x_2y_1.

4.2 Branching

It is necessarily to have a good strategy when branching. If a bad strategy is chosen the complexity can be exponentially but if a good choice is made it is possible to achieve a lower complexity - at least in practice.

A standard branching strategy for fractional programming [14] is to branch on the denominator s_i of each fractional term t_i/s_i. This limits the practical use of branch and bound optimization to at most about 10 dimensions [15] but in the case of triangulation the number of branching dimensions can be limited to a fixed number (at most the degree of freedom of the geometric primitive). Hence, in the point case is it enough to branch on three dimensions, in the line case four and in the cases of conics nine dimensions maximally.

In the line case, we choose not to branch on the denominators. Instead the coordinates of the points where the line intersect with the planes $z = 0$ and $z = 1$ are used for the parameterization (4.1). This gives four dimensions to split at, independent of the number of images. It is also important to choose a coordinate system such that the numerical values of these parameters are kept at a reasonable magnitude.

For strategies for branching and more on fractional programming see [15].

4.3 Initialization

It is necessary to have an initial domain Q_0 for the branch and bound algorithm. The method used for this is similar in the point and conic case but different in the line case due to the Plücker constraint.

Points and Conics. In order to get a bound on the denominators, we assume a bound on the maximal reprojection error. Ideally, with correctly weighted covariance, each such residual $L_i r_i$ should approximately be i.i.d. with unit variance. Thus the bounds are constructed from a user given maximal reprojection error. The bounds on the denominators $g_i(x)$ can then be calculated by the following optimization problem,

$$\begin{aligned} \text{for } i = 1, \ldots, p, \ \min/\max \quad & g_i(x) \\ \text{subject to } \ & \frac{f_j(x)}{g_j(x)} \leq \gamma \quad j = 1, \ldots, p, \end{aligned} \tag{11}$$

where γ is the user given bound on the reprojection error. This is a quadratic convex programming problem. In the experiments, γ is set to 3 pixels.

Lines. In the case of lines, the Plücker constraint makes things a bit more problematic. Instead we choose a more geometric way of getting bounds on the coordinates of the two points defining the line.

For each image line l, two parallel lines are constructed with γ pixels apart (one on each side of l). Then, we make the hypothesis that the two points defining the optimal 3D line (with our choice of coordinate systems) are located on the planes $z = 0$ and $z = 1$, respectively. Now, finding bounds on x_1, x_2, y_1, y_2, see equation (8), becomes a simple linear programming problem. Again, it is important to choose the coordinate system such that the planes $z = 0$ and $z = 1$ are located appropriately. In addition, to avoid getting an unbounded feasible region, the maximum depth is limited to the order of the 3D point furthest away. In the experiments, we set γ to 5 pixels.

5 Experiments

The implementation is made in Matlab using a toolbox called SeDuMi [12] for the convex optimization steps.

The splitting of dimensions has been made by taking advantage of the information where the minimum of the bounding function is located.

While testing the various cases, we have found that the relaxation performed in the line case - a combination of fractions and monomials - the bounds on the denominators is a critical factor for the speed of convergence. To increase the convergence speed, a local gradient descent step is performed on the computed solution in order to quickly obtain a good solution which can be employed to discard uninteresting domains.

Two public sets of real data[1] were used for the experiments with points and lines. One of a model house with a circular motion and one of a corridor with a mostly forward moving motion. The model house has 10 frames and the corridor 11. In these two sequences there were no conics. A third real data sequence was used for conic triangulation. In Fig. 1 samples of the data sets are given.

[1] http://www.robots.ox.ac.uk/∼vgg/data.html

Fig. 1. Image sets used for the experiments

Table 1. Reprojection errors for points and lines with three different methods on two data sets

	Points					
Data set	Optimal		Bundle		Linear	
	Mean	Std	Mean	Std	Mean	Std
House	0.15	0.14	0.15	0.14	0.16	0.15
Corridor	0.13	0.11	0.13	0.11	0.13	0.11

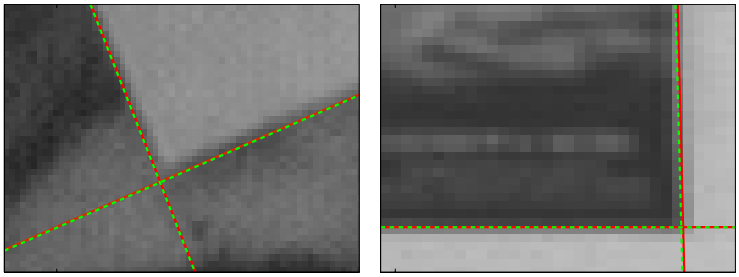
Lines					
Optimal		Bundle		Linear	
Mean	Std	Mean	Std	Mean	Std
1.40	0.92	1.41	0.93	1.62	1.03
3.42	4.29	3.30	4.34	4.02	5.45

Points and lines were reconstructed and then the reprojection errors between different methods were compared. The other methods compared are bundle adjustment and a linear method [1]. The covariance structure for the lines was computed by fitting a line to measured image points. In the reconstruction only the four first frames were used. In the house scene, there are 460 points and in the corridor 490. The optimum was considered to be found if the gap between the global optimum and the under-estimator was less than 10 %. The results are presented in Table 1.

In the house scene, the termination criterion was reached already in the first iteration for all points and for most of them the bounding functions was very close to the global minimum (less than the 10 % required). In the corridor scene, the average number of iterations were 3 and all minima were reached within at most 23 iterations.

In the line case, the under-estimators works not as well as in the point case. This is due to the extra complexity of the Plücker coordinates. Thus more iterations are needed. For the house scene with the circular motion the breakpoint is reached within at most 120 iterations for all the tested 12 lines. However, for the corridor sequence with a weaker camera geometry (at least for triangulation purposes) it is not even enough with 500 iterations for 6 of the tested 12 lines. Even if a lot of iterations are needed to certify the global minimum, the location of the optimum in most cases is reached within less than a handful of iterations.

It can be seen in Table 1 that both a linear method and bundle adjustment works fine for these problems. However, in some cases the bundle adjustment reprojection errors get higher than the errors for the optimal method. This shows that bundle adjustment (which is based on local gradient descent) sometimes gets stuck in a local minimum.



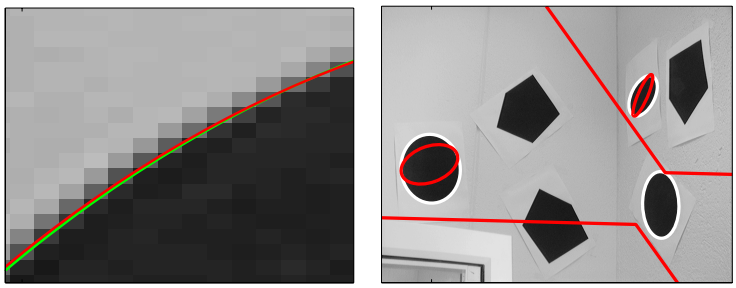
(a) (b)

Fig. 2. The result from reprojection of lines. The green dashed line is the original and the red solid line is the reprojected. Image (a) is from the house scene and (b) is from the corridor.

The result can also be seen in Fig. 2 where two lines from each data set are compared with reprojected line.

5.1 Conics

For conics, an example images can be found in Fig. 1. The covariance structure was estimated by fitting a conic curve to measured image points. The corresponding 3D quadrics were computed with the optimal and a linear method. The result of the reprojected conics from these two methods are imaged in Fig. 3.



(a) (b)

Fig. 3. The result of the reprojected conics of the data set in Fig. 1. In image (a) a part of the reconstruction with optimal method is viewed. The light green is the reprojection and the dark red the original conic. In (b) the red lines are the reprojection after linear method and the white when the optimal method were used.

The number of iterations performed to reach the global minimum with a gap less than 5 % of the bounding function for the three conics were 3, 6 and 14. As can be seen from the images, the quadrics in the data set are planar and hence the condition number of the corresponding 4×4 matrix should be zero. For the three estimated quadrics with the optimal method, the condition numbers are

$1.2 \cdot 10^{-3}$, $3.7 \cdot 10^{-7}$ and $8.8 \cdot 10^{-6}$. This can be compared with the result for the linear estimate with condition numbers of $3.7 \cdot 10^{-4}$, $4.1 \cdot 10^{-5}$ and $1.1 \cdot 10^{-4}$.

Fig. 3 (a) shows the reprojected conic compared with the original for one of the conics. The fitting is very good and it is obvious from Fig. 3 (b) that the linear result is far from acceptable.

6 Discussion

A unified treatment of the triangulation problem has been described using covariance propagation. In addition to traditional local algorithms and algorithms based on algebraic objective functions, globally optimal algorithms have been presented for the triangulation of points, lines and conics. For most cases, local methods work fine (except for conics) and they are generally faster in performance. However, none of the competing methods have a guarantee of globality.

A future line of research is to include more constraints in the estimation process, for example, planar quadric constraints. This opens up the possibility to perform optimal auto-calibration using the image of the absolute conic.

Acknowledgments

The authors thanks Manmohan Chandraker and Sameer Agarwal at Department of Computer Science and Engineering, University of California, San Diego.

This work has been funded by the Swedish Research Council through grant no. 2004-4579 'Image-Based Localisation and Recognition of Scenes', grant no. 2005-3230 'Geometry of multi-camera systems' and the European Commission's Sixth Framework Programme under grant no. 011838 as part of the Integrated Project SMErobot.

References

1. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004)
2. Slama, C.C.: Manual of Photogrammetry. American Society of Photogrammetry, Falls Church, VA (1980)
3. Hartley, R., Sturm, P.: Triangulation. Computer Vision and Image Understanding 68(2), 146–157 (1997)
4. Kanatani, K.: Statistical Optimization for Geometric Computation: Theory and Practice. In: Elsevier Science, Elsevier, North-Holland, Amsterdam (1996)
5. Stewénius, H., Schaffalitzky, F., Nistér, D.: How hard is three-view triangulation really? In: Int. Conf. Computer Vision, Beijing, China (2005)
6. Kahl, F., Henrion, D.: Globally optimal estimates for geometric reconstruction problems. In: Int. Conf. Computer Vision, Beijing, China (2005)
7. Agarwal, S., Chandraker, M., Kahl, F., Kriegman, D., Belongie, S.: Practical global optimization for multiview geometry. In: ECCV, Graz, Austria (2006)
8. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference. pp. 147–151 (1988)

9. Triggs, B.: Detecting keypoints with stable position, orientation, and scale under illumination changes. In: ECCV, Prague, Czech (2004)
10. Freund, R.W., Jarre, F.: Solving the sum-of-ratios problem by an interior-point method. Journal of Global Optimization 19, 83–102 (2001)
11. Tawarmalani, M., Sahinidis, N.V.: Semidefinite relaxations of fractional programs via novel convexification techniques. Journal of Global Optimization 20, 133–154 (2001)
12. Sturm, J.: Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Optimization Methods and Software 11(12), 625–653 (1999)
13. Ryoo, H.S., Sahinidis, N.V.: Analysis of bounds for multilinear functions. Journal of Global Optimization 19(4), 403–424 (2001)
14. Benson, H.P.: Using concave envelopes to globally solve the nonlinear sum of ratios problem. Journal of Global Optimization 22, 343–364 (2002)
15. Schaible, S., Shi, J.: Fractional programming: the sum-of-ratios case. Optimization Methods and Software 18, 219–229 (2003)