Triangulation of Points, Lines and Conics

Josephson, Klas; Kahl, Fredrik

Published in:
Journal of Mathematical Imaging and Vision

DOI:
10.1007/s10851-008-0097-y

2008

Triangulation of Points, Lines and Conics

Klas Josephson and Fredrik Kahl

May 25, 2008

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 [7, 13]. The standard approach for estimating point features is a two-step approach:

(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 re-projection errors in the images.

This methodology works fine in most cases. However, it is well-known that the cost-function is indeed non-convex and one may occasionally get trapped in local minima [6]. The goal of this

paper is to derive the statistically optimal cost-function, cf. [12], and an algorithm which gives the globally optimal solution with a guarantee. The algorithm may serve as a benchmarking tool for other methods which are non-optimal. On the other hand, as we shall see, the developed algorithm shows good empirical performance on real data and it can be used for reliably computing optimal estimates in a structure from motion system.

In [6], 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 [16], but the resulting polynomial system turns out to be of very high degree and the solution method based on Gröbner bases becomes numerically unstable. Even though the state-of-the-art methods have improved in this area [2], this approach is not generalizable for an arbitrary number of views. In [11] convex linear matrix inequalities 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 [7] and the references therein. Global optimization techniques have also been applied to other problems in multiple view geometry, see [10] as well as the survey paper [5].

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 [9], 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 globally. For many feature detectors, e.g., [4, 19], 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 apply convex and concave relaxations of monomials in the optimization framework in order to handle the Plücker constraints for line features. To our knowledge, global optimization over the Plücker manifold has not been done previously in multiple view geometry.

The paper is organized as follows. Section 2 contains a recapitulation on projective geometry and how points, lines and conics projection can be written in a similar manner. In Section 3 we give expressions for the geometrical reprojection error and state the problem we solve. Further on in Section 4 the optimization using a branch and bound algorithm is described, including calculation of a lower bound for the objective function. In Section 5 evaluation of the method is presented

with experiments on three different data set.

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 [7].

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, $x = [u\ v\ 1]^T$. The scalar λ can be interpreted as the depth, hence $\lambda > 0$ if the point appears in the image. This property is useful in the optimization performed in this paper.

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 are 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.

The Plücker coordinates to a line in three dimensional space can be obtained from the Plücker matrix,

$$L = AB^T - BA^T, \tag{2}$$

where A and B are two arbitrary 3D points in homogeneous coordinates on the line. It is easy to see that L is a skew-symmetric 4×4 matrix. Further on are the six Plücker coordinates defined as

the elements[1],

$$\mathcal{L} = \{l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}\} \tag{3}$$

of L.

From the fact that L has rank 2 and hence $\det L = 0$ it follows that the coordinates satisfy the constraint

$$l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0. \tag{4}$$

The Plücker coordinates are also easy to deal with when changing coordinate systems. If a points X transforms according to $X' = HX$, the construction of the Plücker matrix gives,

$$\begin{aligned} L' &= (HA)(HB)^T - (HB)(HA)^T = \\ &= H(AB^T)H^T - H(BA^T)H^T = \\ &= H(AB^T - BA^T)H^T = HLH^T. \end{aligned} \tag{5}$$

Thus the Plücker matrix transforms as $L' = HLH^T$. Further, it is easy to show that the coordinates are independent of the choice of points defining the line.

One advantage with the Plücker representation of lines is that it is possible to construct a Plücker camera, $P_{\mathcal{L}}$, that makes it possible to write the projection formula in a similar way as in the point case. The Plücker camera equation looks like,

$$\lambda l = P_{\mathcal{L}}\mathcal{L}, \tag{6}$$

where $P_{\mathcal{L}}$ is the so called Plücker camera. The Plücker camera is given by

$$P_{\mathcal{L}} = \begin{pmatrix} p_2 \wedge p_3 \\ p_3 \wedge p_1 \\ p_1 \wedge p_2 \end{pmatrix}, \tag{7}$$

where p_i^T denote the rows of the point camera matrix and $p_i \wedge p_j$ denotes the Plücker line constructed by p_i and p_j. The Plücker camera is hence a 3×6 matrix with elements that are quadratic in the elements of the standard camera matrix. Further, l denotes the projected image line, represented by a 3×1 vector.

One difference, however, in the projection equation for lines compared with that of points is that it is not possible to interpret the scale λ as depth.

[1] Alternative definitions are also used.

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. Before that we first make a short recapitulation on conics and quadrics.

A general conic in the plane is defined by the quadratic form

$$x^T c x = 0. \tag{8}$$

Here x are points in homogeneous coordinates on the edge of the conic and c is a 3×3 symmetric matrix. In a similar manner a quadric in space is defined by the quadratic form,

$$X^T C X = 0, \tag{9}$$

where X is a point in homogeneous coordinates and C a 4×4 symmetric matrix.

When quadrics are projected through a pinhole camera (1), is it a lot easier to work with the duals to the conics and quadrics. 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

$$U^T L U = 0, \tag{10}$$

where U are homogeneous plane coordinates and $L = C^{-1}$. Similar for conics, one gets

$$u^T l u = 0, \tag{11}$$

where u are homogeneous line coordinates and $l = c^{-1}$. The projection for the envelope forms are

$$\lambda l = P L P^T \quad \lambda \neq 0. \tag{12}$$

Now we want to reformulate (12) so it appears in a similar form as the point projection formula. This is indeed possible.

Lemma 2.1. *The projection of a quadric,*

$$\lambda l = P L P^T, \tag{13}$$

can be written

$$\lambda \tilde{l} = P_C \tilde{L}. \tag{14}$$

where $\tilde{l}$ and $\tilde{L}$ denote 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.

Proof. Equation (13) can be written as a tensor product, $\lambda\hat{l} = (P \otimes P)\hat{L}$, where $\hat{l}$ and $\hat{L}$ denote the stacked columns of l and L. Due to the symmetry in l and L it can be rewritten as (14). □

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

$$r = \left(u - \frac{p_1^T X}{p_3^T X},\ v - \frac{p_2^T X}{p_3^T X} \right). \tag{15}$$

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.

The residual r in (15) can be rewritten as

$$r = \left(\frac{u\, p_3^T X - p_1^T X}{p_3^T X}, \frac{v\, p_3^T X - p_2^T X}{p_3^T X} \right), \tag{16}$$

is it easy to see that the L_2-norm of the residual 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 Error Distribution

In the remainder of this paper, it will be assumed that the errors on the residual elements are Gaussian. This is true when triangulating points given that the errors that appear in the image plane are normally distributed. For lines and conics this will not be entirely true [8] but we argue that it is an good approximation when the magnitude of the errors is small.

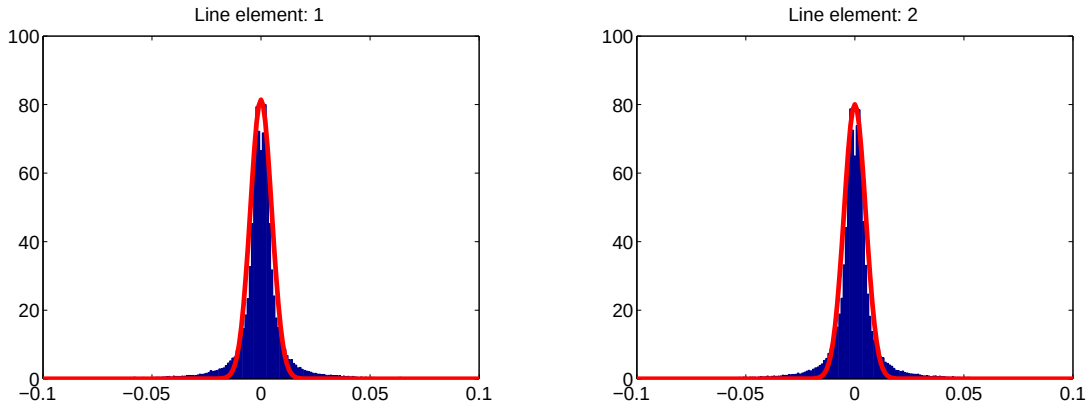


Figure 1: The distributions of the errors in the two residuals for line case when Gaussian noise is added to 21 sample points of the true line. A Gaussian distribution is overlaid to the histogram to compare with.

In order to validate this claim, the following experiments were performed: 100,000 random images were created consisting of one conic and one line. These structures were than sampled with 21 and 63 points for the line and the conic, respectively. Gaussian noise was then added to these points with a standard deviation corresponding to one pixel in an image with a resolution 1000×1000 pixels. In the next step, a line and a conic were fitted to these points using least-squares. In the line case, the scale was adjusted such that the first two coordinates have a norm of one and translated such that the distance to the origin was one in order to avoid passing through the origin (since this would mean that the third coordinate would be zero).

As can be seen in Figure 1 the elements in the line vector residual is not perfectly normally distributed but it is an good approximation. In the conic case, Figure 2 shows that two of the elements are closer to be perfect normally distributed. These two elements correspond to the center of the conic divided by a scale factor. The other three elements are close to normal distributions even though they are not as close as the other two.

3.2 Incorporation of Covariance

The residual vector (16) may in dimension independent notation be written

$$r = \left(\frac{x_1 p_n^T X - p_1^T X}{p_n^T X}, \ldots, \frac{x_{n-1} p_n^T X - p_{n-1}^T X}{p_n^T X} \right). \tag{17}$$

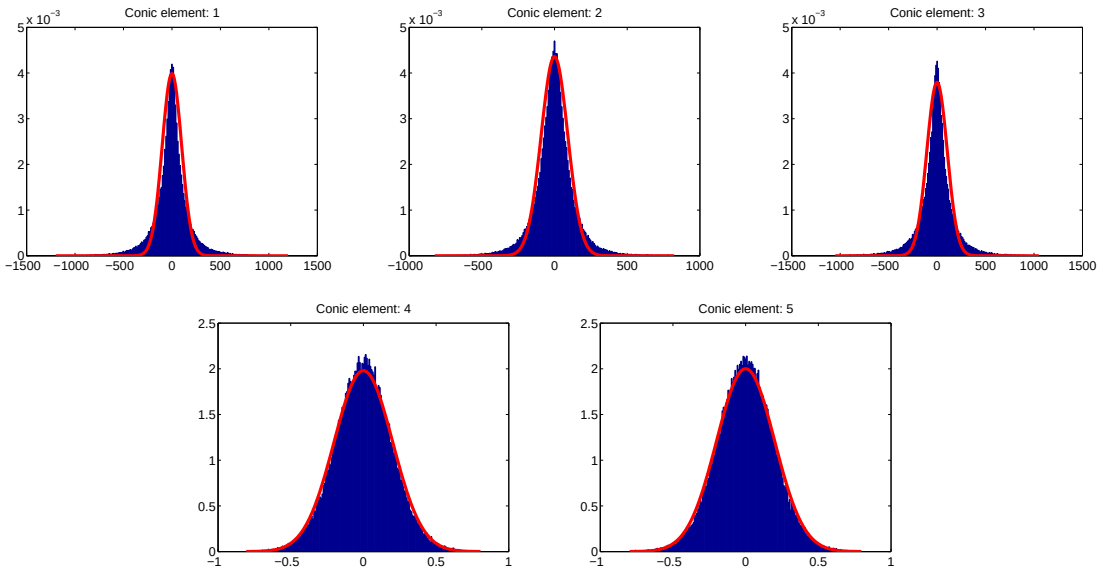


Figure 2: The distributions of the errors in the five elements of the conic residual vector when Gaussian noise is added to 63 sample points. A Gaussian distribution is overlaid to the histogram to compare with.

The statistically optimal cost-function is to weight the residual vector by its covariance [12] (at least to a first order approximation). Incorporating covariance weighted error transforms to,

$$\|Lr\| = \left\| L \left(\frac{x_1 p_n^T X - p_1^T X}{p_n^T X}, \dots \right) \right\|, \tag{18}$$

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 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.

3.3 Problem Formulation

For the triangulation of points, lines or conics, the optimization problem to solve can be formulated as follows:

$$\min \sum_{i=1}^{n} \|L_i r_i\|^2. \tag{19}$$

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 (4) 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 result of the algorithm is 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. The lower bounding function should ideally be a convex function to make it possible to find a minimum reliably. If the difference between the optimum for the bounding function and the lowest value of the function Φ - calculated so far - is small enough, say ϵ, 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. The algorithm repeats until the gap between the bounding function and the value of the objective function is smaller than the predefined value ϵ. Hence, the global optimum is attained within a preset $\epsilon > 0$ which can be chosen arbitrarily small.

If the lower bounding function on a subdomain has its minimum 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 under-estimator to the original objective function Φ and (ii) easy to compute the lowest value Φ_{lb} in a given domain. For practical reasons, convex functions are good candidates as bounding functions. 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, defined as follows.

Definition 1 (Convex Envelope)**.** Let $f : S \to \mathbb{R}$, where $S \subset \mathbb{R}^n$ is a nonempty convex set. The convex envelope of f over S (denoted **convenv** f) is a convex function such that (*i*) **convenv** $f(x) \leq f(x)\ \forall\ x \in S$ and (*ii*) for any other convex function u, satisfying $u(x) \leq f(x)\ \forall\ x \in S$, we have **convenv** $f(x) \geq u(x)\ \forall\ x \in S$.

In Figure 3, an example of a convex envelope of a one dimensional function is plotted.

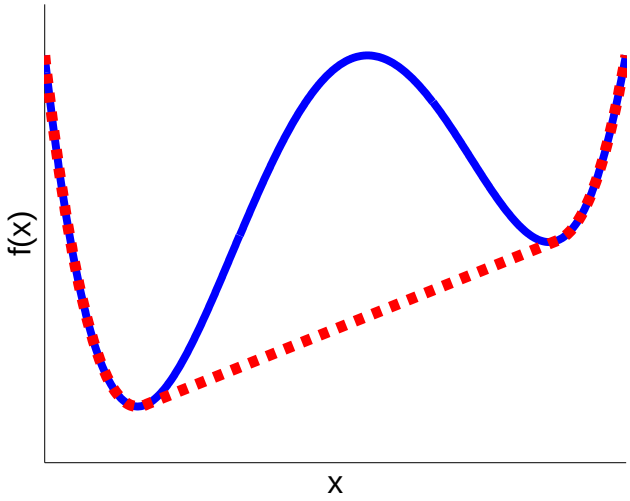


Figure 3: Illustration of the convex envelope; the blue solid line is a function $f(x)$ and the dashed red line is its convex envelope **convenv** $f(x)$.

The concave envelope is defined in a similar manner. It can often be hard to compute the convex envelope (or concave envelope), in fact, it may be as hard as computing the global optimum itself.

4.1.1 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

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{p} \frac{f_i(x)}{g_i(x)} \\ \text{subject to} \quad & x \in D, \end{aligned} \tag{20}$$

where f_i and g_i are convex and concave, respectively, functions from $\mathbb{R}^n \rightarrow \mathbb{R}$, 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 [3].

Instead of studying equation (20) more let us rewrite it. Assume that we have bounds on the functions $f_i(x)$ and $g_i(x)$, where the intervals are $[l_i,\ u_i]$ and $[L_i,\ U_i]$, respectively. Let the $2p$-dimensional rectangle $[l_1,\ u_1] \times \cdots \times [l_p,\ u_p] \times [L_1,\ U_1] \times \cdots \times [L_p,\ U_p]$ be denoted Q_0.

With auxiliary variables $t = (t_1, \ldots, t_p)^T$ and $s = (s_1, \ldots, s_p)^T$ it can be showed that the global optimum to (20) is identic as to the following optimization problem,

$$\min_{x,t,s} \quad \sum_{i=1}^{p} \frac{t_i}{s_i} \tag{21}$$

$$\begin{aligned} \text{subject to} \quad & f_i(x) \leq t_i \quad g_i(x) \geq s_i \\ & x \in D \qquad (t,s) \in Q_0. \end{aligned}$$

In [18] it has been shown that the problem of finding the convex envelope to a single fraction t/s as the parts in the sum in (21), where $t \in [l,u]$ and $s \in [L,U]$, is given as the solution of the following *Second Order Cone Program* (SOCP):

$$\begin{aligned} \text{convenv}\left[\frac{t}{s}\right] \quad = \quad & \text{minimize} \;\; r \\ \text{subject to} \quad & \left\| \begin{array}{c} 2\lambda\sqrt{l} \\ r' - s' \end{array} \right\| \leq r' + s' \\ & \left\| \begin{array}{c} 2(1-\lambda)\sqrt{u} \\ r - r' - s + s' \end{array} \right\| \leq r - r' + s - s' \\ \lambda L \leq s \leq \lambda U \qquad & (1-\lambda)L \leq s - s' \leq (1-\lambda)U \\ r' \geq 0 \qquad & r - r' \geq 0 \\ l \leq t \leq u \qquad & L \leq s \leq U \end{aligned} \tag{22}$$

where $\lambda = \frac{u-t}{u-l}$ for ease the notation and r, r', s' are auxiliary scalar variables.

When Φ is a sum of ratios as in (21) a bound for the function can be calculated as the sum of the convex envelopes of the individual fractions. When summarizing the convex envelopes the sum will, however, generally not be the convex envelope to the function. Still the function will be a lower bound and it fulfills the requirements of a bounding function. This way of calculating Φ_{lb} by solving p SOCP problems can be done efficiently [17].

A more exhaustive description on fractional programming in multiple view geometry can be found in [9] where point triangulation (without covariance weighting) is treated.

4.1.2 Monomial Relaxation

In the line case the Plücker coordinates have to fulfill the Plücker constraint (4). 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),

$$\mathcal{L} = (x_1y_2 - x_2y_1,\ -y_1,\ x_1 - y_1,\ -y_2,\ y_2 - x_2,\ 1)^T. \tag{23}$$

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

The choosen parameterization does not work if the line is parallel to the plane $z = 0$. If this is the case we change the coordinate system.

In [14] the convex and the concave envelopes are derived for a monomial y_1y_2. The convex envelope is given by

$$\mathbf{convenv}(y_1y_2) = \max \left\{ \begin{array}{c} y_1y_2^U + y_1^Uy_2 - y_1^Uy_2^U \\ y_1y_2^L + y_1^Ly_2 - y_1^Ly_2^L \end{array} \right\}. \tag{24}$$

Similarly the concave envelope is

$$\mathbf{concenv}(y_1y_2) = \min \left\{ \begin{array}{c} y_1y_2^L + y_1^Uy_2 - y_1^Uy_2^L \\ y_1y_2^U + y_1^Ly_2 - y_1^Ly_2^U \end{array} \right\}. \tag{25}$$

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 [1] 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 (4.1.2). 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. When the decision of which rectangle to split is taken there are two choices to make. The first is on which dimension to split and the second where to split in the chosen dimension. A reasonable pick of dimension is the one with the largest interval. The decision to split can be made in several different ways. One alternative is to make use of the bounding function. The hypothesis is that the domain with the minimum bounding function is the best candidate for the minimum of the objective function. To get as good estimation as possible close to that point the splitting location can be chosen close to the minimum of the bounding function. Another way to split is to split the dimension into two equal parts. Both these splitting procedure can be shown to be convergent [1].

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.

4.3.1 Points and Conics

In order to get a bound on the denominators, we assume a bound on the maximal reprojection error. 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,

$$\begin{aligned} \text{for } i = 1, \ldots, p, \qquad & \min/\max \;\; g_i(x) \\ \text{subject to} \qquad & \frac{f_j(x)}{g_j(x)} \leq \gamma \quad j = 1, \ldots, p, \end{aligned} \tag{26}$$

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.

4.3.2 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 we first set the scale such that the norm of the first two coordinates equals one and then we make a translation to avoid the line to pass through the origin. Two parallel lines (l_1 and l_2) are then 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, in order to find bounds on x_1, x_2, y_1, y_2, see equation (23), we need to solve the optimization problems,

$$\begin{aligned} \text{max/min} \quad & x_i \\ \text{subject to} \quad & l_{1i}^T P_i X \leq 0 \\ & l_{2i}^T P_i X \geq 0, \end{aligned} \tag{27}$$

where $X = [x_1, x_2, 0, 1]^T$ and the corresponding for y_i with X substituted to $Y = [y_1, y_2, 1, 1]^T$. 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 [17] 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, as described in Section 4.2.

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[2] 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 camera motion. The model house has 10 frames and the corridor 11. In these two sequences there were no conics. In Figure 4 samples of the data sets for points and lines are given. A third real data sequence was used for conic triangulation, see Figure 6.

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

Figure 4: Image sets used for the experiments.

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 [7]. 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 Tables 1 and 2.

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

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

Data set	Optimal		Bundle		Linear	
	Mean	Std	Mean	Std	Mean	Std
House	1.40	0.92	1.41	0.93	1.62	1.03
Corridor	3.42	4.29	3.30	4.34	4.02	5.45

Table 2: Reprojection errors for lines with three different methods on two data sets.

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 minimas 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. This makes it necessary to perform more iterations before it is possible to state that the global optimum is reached within a given certainty. In the house scene with the circular camera 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.

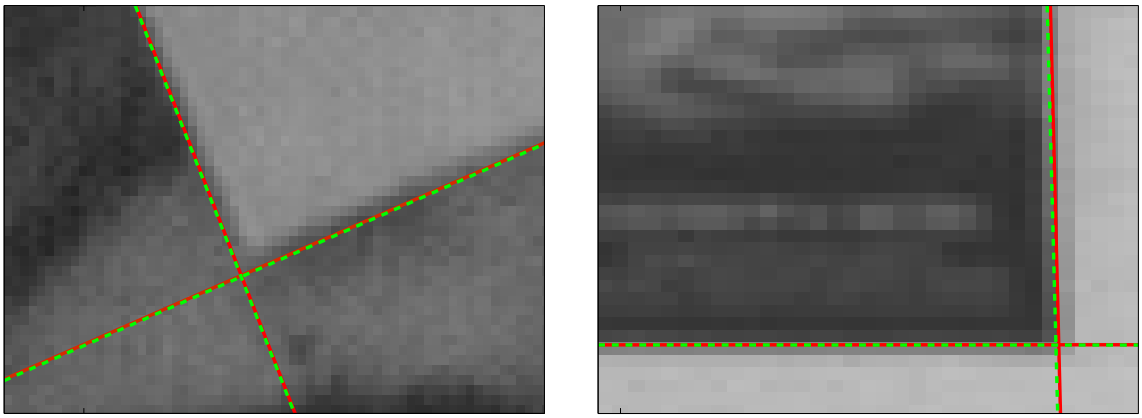
It can be seen in Tables 1 and 2 that both a linear method and bundle adjustment work fine for these problems. However, in some cases the bundle adjustment reprojection errors get higher than the errors for the optimal method. This shows that bundle adjustment (which is based on local gradient descent) sometimes gets stuck in a local minimum. The opposite also occurs in some cases: bundle adjustment gets better results than the optimal method. The reason for this is twofold. The first reason is that a threshold is used in the optimal method for the gap between the optimal value and the found value of the objective function. This gap is fixed to 10% in our experiments which leaves a margin. The second reason is that this margin of error is not attained in all line experiments where we instead reach the maximum iteration number.

The result can also be seen in Figure 5 where two lines from each data set are compared with reprojected line.

5.1 Conics

For conics, some example images can be found in Figure 6 with marked image conics. The covariance structure was estimated by fitting a conic curve to measured image points. The corresponding 3D quadrics were first computed with the optimal and a linear method. The result of the reprojected conics from these two methods are imaged in Figure 7.

The number of iterations performed to reach the global minimum with a gap less then 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 $1.1 \cdot 10^{-4}$.



(a) (b)

Figure 5: 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 of the optimal method was also compared with bundle adjustment. In the local optimization the result of the linear method were used as initialization. The norm of the residual from the three estimated quadrics are shown in Table 3. As can be seen there the result of the optimal method and bundle adjustment is identical. This shows that bundle adjustment reached the global optimum in all the experiments.

Conic #	Optimal	Bundle	Linear
1	39.5	39.5	1190
2	5.00	5.00	160
3	15.3	15.3	2660

Table 3: The norm of the residuals when quadrics were reconstructed. It is obvious that the linear method isn't good enough and that bundle adjustment reaches the global optimum.

Figure 7 (a) shows the reprojected conic compared with the original for one of the conics. The fitting is very good and it is obvious from Figure 7 (b) that the linear result is far from acceptable. In part (c) of Figure 7 a comparison of the optimal method and bundle adjustment is shown. As can be seen there the result is almost identical.

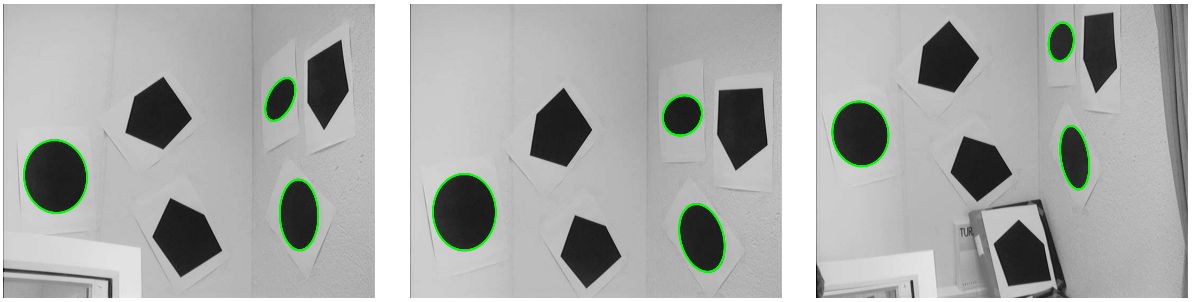
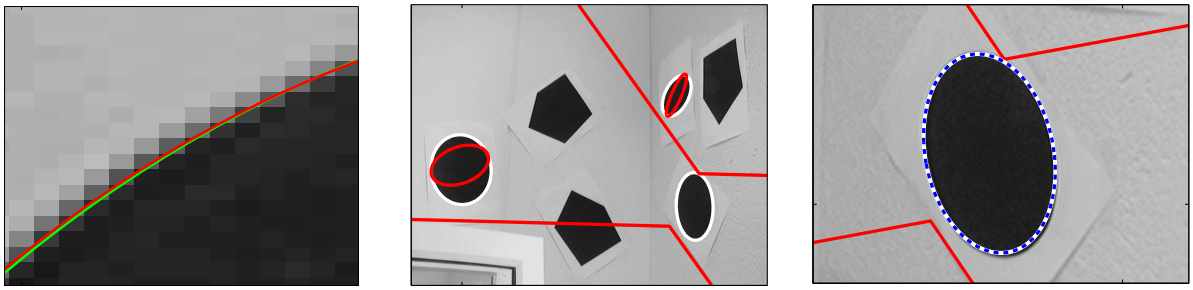


Figure 6: The images with the data used for conics.



(a) (b) (c)

Figure 7: The result of the reprojected conics of the data set in Figure 6. In image (a) a part of the reconstruction with optimal method is viewed. The light green line 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. In (c) the image in (b) is zoomed in. The added blue dashed line is the result from bundle adjustment initialized with the result from the linear method (red), as can be seen this line almosst perfectly coincide with the result from the optimal method.

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 and they are generally faster in performance. However, none of the competing methods have a guarantee of globality.

The performed experiments show that bundle adjustment works well. This conclusion may come as no surprise. It has already been observed in the two-view case for points [6]. Now it is shown that this is true for any number of views for point features. Perhaps the main contribution

of this paper is to serve as a benchmarking algorithm of other algorithms since it gives a way to evaluate the performance of other methods.

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 absolute conic. An other line of research is to improve computational efficiency for global optimal triangulation problems.

7 Acknowledgments

This work has been funded by the Swedish Research Council (grants no. 2004-4579, no. 2005-3230 and no. 2007-6476), by the Swedish Foundation for Strategic Research (SSF) through the programme Future Research Leaders, and by the European Research Council (GlobalVision grant no. 209480).

References

[1] Harold P. Benson. Using concave envelopes to globally solve the nonlinear sum of ratios problem. *Journal of Global Optimization*, 22:343–364, 2002. 12, 13

[2] Martin Byröd, Klas Josephson, and Kalle Åström. Improving numerical accuracy of gröbner basis polynomial equation solvers. In *International Conference on Computer Vision*, 2007. 2

[3] R. W. Freund and F. Jarre. Solving the sum-of-ratios problem by an interior-point method. *J. Glob. Opt.*, 19(1):83–102, 2001. 10

[4] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988. 2

[5] R. Hartley and F. Kahl. Optimal algorithms in multiview geometry. In *Asian Conf. Computer Vision*, Tokyo, Japan, 2007. 2

[6] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997. 1, 2, 18

[7] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition. 1, 2, 3, 15

[8] Stephan Heuel. *Uncertain Projective Geometry: Statistical Reasoning for Polyhedral Object Reconstruction*, volume 3008 of *Lecture Notes in Computer Science*. Springer, 2004. 6

[9] F. Kahl, S. Agarwal, M.K. Chandraker, D.J. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. *International Journal of Computer Vision*, 2007. 2, 11

[10] F. Kahl and R. Hartley. Multiple view geometry under the L_∞-norm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2007. To appear. 2

[11] F. Kahl and D. Henrion. Globally optimal estimates for geometric reconstruction problems. *Int. Journal Computer Vision*, 74(1):3–15, 2007. 2

[12] K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science, Amsterdam, The Netherlands, 1996. 2, 8

[13] J.C. McGlone, E.M. Mikhail, J. Bethel, and R. Muellen. *Manual of Photogrammetry*. American Society of Photogrammetry and Remote Sensing , Falls Church, VA, 2004. 1

[14] Hong Seo Ryoo and Nikolaos V. Sahinidis. Analysis of bounds for multilinear functions. *Journal of Global Optimization*, 19(4):403–424, 2001. 12

[15] Siegfried Schaible and Jianming Shi. Fractional programming: the sum-of-ratios case. *Optimization Methods and Software*, 18:219–229, 2003. 12

[16] H. Stewénius, F. Schaffalitzky, and D. Nistér. How hard is three-view triangulation really? In *Int. Conf. Computer Vision*, pages 686–693, Beijing, China, 2005. 2

[17] J.F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999. 11, 14

[18] Mohit Tawarmalani and Nikolaos V. Sahinidis. Semidefinite relaxations of fractional programs via novel convexification techniques. *Journal of Global Optimization*, 20:133–154, 2001. 11

[19] Bill Triggs. Detecting keypoints with stable position, orientation, and scale under illumination changes. In *European Conf. Computer Vision*, volume 4, pages 100–113, Prague, Czech, 2004. 2