Fast and stable algebraic solution to L_2 three-view triangulation

Zuzana Kukelova, Tomas Pajdla
Czech Technical University, Faculty of Electrical Engineering,
Karlovo namesti 13, Prague, Czech Republic
{kukelova,pajdla}@cmp.felk.cvut.cz

Martin Bujnak
Bzovicka 24, 85107,
Bratislava, Slovakia
martin@solvergenerator.com

Abstract

In this paper we provide a new fast and stable algebraic solution to the problem of L_2 triangulation from three views. We use Lagrange multipliers to formulate the search for the minima of the L_2 objective function subject to equality constraints. Interestingly, we show that by relaxing the triangulation such that we do not require a single point in 3D, we get, after a linear correction, a solver that is faster, more stable and practically as accurate as the state-of-the-art L_2-optimal algebraic solvers [24, 7, 8, 9]. In our formulation, we obtain a system of eight polynomial equations in eight unknowns, which we solve using the Gröbner basis method. We get less (31) solutions than was the number (47-66) of solutions obtained in [24, 7, 8, 9] and our solver is more robust than [8, 9] w.r.t. critical configurations . We evaluate the precision and speed of our solver on both synthetic and real datasets. [1]

1. Introduction

The triangulation [13] is one of the fundamental problems in computer vision and it is an important part of all structure-from-motion systems [23, 21]. The problem can be formulated as follows:

Problem 1 *Given a set of n, $n \geq 2$, camera projection matrices $\{\mathtt{P}_i\}_{i=1}^n$, $\mathtt{P}_i \in \mathbb{R}^{3\times 4}$ and a set of image points $\{\mathbf{u}_i\}_{i=1}^n$, $\mathbf{u}_i = [u_i, v_i, 1]^\top$, find a point $\mathbf{X}$ in space, $\mathbf{X} = [X, Y, Z, 1]^\top$, such that*

$$\alpha_i \mathbf{u}_i = \mathtt{P}_i \mathbf{X},\ i = 1, \ldots, n, \alpha_i \in \mathbb{R}, \tag{1}$$

i.e., such that the points $\mathbf{u}_i$ are the projections of the point $\mathbf{X}$ using the projection matrices $\mathtt{P}_i$.

This problem requires to find the intersection of n known rays in space, and it is known as the triangulation in n views.

[1]This work has been supported by EC project FP7-SME-2011-285839 De-Montes and TA02011275 project ATOM.

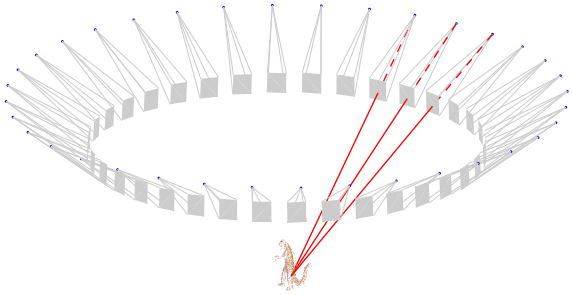

Figure 1. Illustrations of the 3-view triangulation problem.

For noise free image points $\{\mathbf{u}_i\}_{i=1}^n$, the triangulation problem is trivial. The 3D point $\mathbf{X}$ can be determined using the linear least square algorithm [14].

In the presence of noise, n rays from the camera centers through the image points $\mathbf{u}_i$ do not intersect in 3D, i.e. there doesn't exist a 3D point $\mathbf{X}$ that exactly satisfies (1) for all $\mathbf{u}_i$. Therefore, for noisy data, the triangulation problem becomes the problem of finding "the best intersection point X", see Figure 1.

Assuming independent Gaussian noise on the image measurements, the optimal, maximum likelihood solution to the triangulation problem is the solution that minimizes the L_2-norm of the reprojection error [13]. This leads to the following constrained optimization problem:

Problem 2 *Given a set of n, $n \geq 2$, camera projection matrices $\{\mathtt{P}_i\}_{i=1}^n$, $\mathtt{P}_i \in \mathbb{R}^{3\times 4}$ and a set of image points $\{\mathbf{u}_i\}_{i=1}^n$, $\mathbf{u}_i = [u_i, v_i, 1]^\top$,*

$$\textit{minimize} \qquad f(\hat{\mathbf{u}}) = \sum_{i=1}^{n} d(\mathbf{u}_i, \hat{\mathbf{u}}_i)^2, \tag{2}$$

$$\textit{subject to} \qquad \alpha_i \hat{\mathbf{u}}_i = \mathtt{P}_i \mathbf{X},\ i = 1, \ldots, n, \tag{3}$$

where $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_n)$, $\hat{\mathbf{u}}_i = [\hat{u}_i, \hat{v}_i, 1]^\top$, $i = 1, \ldots, n$ are corrected image points, $d(\cdot, \cdot)$ is the Euclidean metric,

$\alpha_i \in \mathbb{R}$ *are some constants, and* $\mathbf{X}$ *is the searched 3D point with homogeneous coordinates* $\mathbf{X} = [X, Y, Z, 1]^\top$.

2. Previous solutions

Many different methods for solving the triangulation problem have been proposed. In general, they differ in the function they minimize (L_2-norm [13, 7, 9, 8, 20, 19, 4], L_∞-norm [12, 22]), in the number of views they can handle (2 [13, 16, 19], 3 [24, 7, 9, 8], $n \geq 2$ [3, 20, 12, 4]), and in the method they use for the optimization (algebraic method [13, 7, 9, 8, 24], quadratic iterative method [19], branch-and-bound method [3, 20], QCQP method [4]).

One of the simplest solutions to the triangulation problem is the linear least square method [14]. This method is fast and can be applied to n views, however, it is not optimal and sometimes can yield very poor results.

Usually, the solution from the linear least square method [14] or from some method that minimizes L_∞-norm of the reprojection error [12, 22] is used as an initialization of a non-linear refinement method such as Levenberg-Marquardt [25]. Such a method is known as Bundle Adjustment and can be used not only to optimize the position of the 3D point but also to optimize the camera parameters. The main drawback of the Bundle Adjustment method is that it requires a good initial estimate of the 3D point and even for reasonable initialization it may fall into a local minima which can be very far from the global one.

Hartley and Sturm showed [13] that the L_2-optimal solution to the triangulation problem for two views, i.e. to Problem 2 for $n = 2$, can be found by solving a polynomial of degree six. This solution is quite simple for two views, but it can't be easily extended to more views.

Several fast iterative methods [16, 19] to the L_2-optimal two-view triangulation problem have been proposed recently. While the solution [16] doesn't satisfy the epipolar constraint [14] and may fall in a local minima, the solution [19] usually converges to the global optimum satisfying the epipolar constraint in two steps.

Recently, several solutions to the L_2-optimal n-view triangulation problem have been proposed [3, 20, 4]. The solution [20] uses the branch-and-bound technique and guarantees the global optimum, however it may sometimes take long time to get the optimum. On the other hand, the solution [4] uses semidefinite programming relaxations to formulate the L_2-optimal n-view triangulation problem as a quadratically constrained quadratic program and solve it in a polynomial time. However, this solution doesn't guarantee the globally optimal solution.

The most relevant alternatives to our solution are the algebraic solutions [24, 7, 8, 9] which formulate the L_2-optimal triangulation problem in three views as a system of polynomial equations. By computing all roots of this system they find stationary points of the L_2 objective function from which they extract the global minimum.

All solutions [24, 7, 8, 9] use the same formulation of the L_2-optimal three-view triangulation problem and the Gröbner basis method for solving the final system of polynomial equations. They differ in the method that they use for improving the numerical stability of the final Gröbner basis solver.

After placing the three image points $\mathbf{u}_i$ at the origin in their respective image coordinate systems, the L_2-objective function (2) can be replaced by the following cost function

$$\begin{aligned} \varphi(\mathbf{X}) &= \frac{(\mathtt{P}_1^1\mathbf{X})^2 + (\mathtt{P}_1^2\mathbf{X})^2}{(\mathtt{P}_1^3\mathbf{X})^2} + \frac{(\mathtt{P}_2^1\mathbf{X})^2 + (\mathtt{P}_2^2\mathbf{X})^2}{(\mathtt{P}_2^3\mathbf{X})^2} \\ &+ \frac{(\mathtt{P}_3^1\mathbf{X})^2 + (\mathtt{P}_3^2\mathbf{X})^2}{(\mathtt{P}_3^3\mathbf{X})^2}, \end{aligned} \tag{4}$$

which has to be minimized over $\mathbf{X}$ and in which $\mathtt{P}_j^i$ denotes the i^{th} row of the j^{th} projection matrix $\mathtt{P}_j$.

Finding all stationary points of $\varphi(\mathbf{X})$ (4), i.e. all solutions of $\nabla\varphi(\mathbf{X}) = \mathbf{0}$, leads to all local extrema from which the global optimum can be extracted. The function $\varphi(\mathbf{X})$ (4) is a rational function in three unknowns. Therefore, $\nabla\varphi(\mathbf{X}) = \mathbf{0}$ leads to three rational equations in three unknowns, coordinates of $\mathbf{X}$. Multiplying the partial derivatives by the denominators then produces three sixth degree polynomial equations in three unknown coordinates of $\mathbf{X}$.

The problem of this system of three polynomial equations in three unknowns is that the multiplication by the denominators introduces new stationary points, points where X, Y or Z are equal to zero. These "parasitic" solutions need to be eliminated by computing the saturation of the ideal [10]. In this case the saturation leads to quite a complicated system of nine fifth and sixth degree equations. Solution to this system using the standard Gröbner basis method is numerically unstable and therefore in [24] a 128 bit precision arithmetic was used to get a stable solver, which was, however, very slow (about 30s per point) and therefore impractical. The final solver resulted in 47 solutions.

In [7] authors solved the problem with the numerical instability of [24] by computing the zeros of a relaxed ideal, i.e. a smaller ideal with a larger solution set. This relaxation leads to quite stable solver which, however, requires to find eigenvalues of a 154×154 matrix and results in 154 candidate solutions. Therefore this solver is again impractical.

In [8] and [9] authors used special techniques for improving numerical stability of Gröbner basis solvers based on basis selection of relaxed ideals and on SVD and QR decomposition. Using these techniques they obtained more practical and numerically stable solutions to the L_2-optimal 3-view triangulation problem than the previous algebraic solutions [24, 7]. These solvers return from 50 to 66 solutions and require to perform QR [9] or SVD [8] decomposition of a 225×209 matrix and eigenvalue computations of

a matrix with as many rows as solutions of the system. The QR-based solver [9] runs about $6ms$ and the SVD-based solver [8] about $11ms$. The main drawback of these solvers is that they were crated mostly manually after careful studying of the structure of the input polynomials and they require special manipulations with these polynomials.

In this paper we propose a new algebraic solution to the problem of L_2 triangulation from three views. In contrast to the previous algebraic solutions [24, 7, 8, 9], which use the cost function (4), we use Lagrange multipliers to formulate the search for the global minima of the L_2-objective function (2) subject to a relaxed equality epipolar constraints [14]. We use only two epipolar constraints to obtain a system of eight polynomial equations in eight unknowns. This system leads to 31 solutions and can be solved using the standard Gröbner basis method [10] and the automatic generator of Gröbner basis solvers [17]. The resulting solver is faster, numerically more stable and has less critical configurations than the state-of-the-art L_2-optimal algebraic solvers [24, 7, 8, 9]. Moreover, the new solver doesn't require special manipulations with polynomials such as ideal saturation, basis selection or relaxations used in [7, 8, 9].

3. Problem formulation

Next we describe our formulation of the L_2 three-view triangulation problem.

Problem 3 *Given two essential matrices* $\mathtt{E}_{12}$ *and* $\mathtt{E}_{23}$*, between the first and the second view, and between the second and the third view, and given three corresponding image points* $\mathbf{u}_i = [u_i, v_i, 1]^\top,\ i = 1, \ldots, 3$

$$\textit{minimize} \qquad f(\hat{\mathbf{u}}) = \sum_{i=1}^{3} d(\mathbf{u}_i, \hat{\mathbf{u}}_i)^2, \tag{5}$$

$$\textit{subject to} \qquad \hat{\mathbf{u}}_1^\top \mathtt{E}_{12} \hat{\mathbf{u}}_2 = 0, \tag{6}$$

$$\hat{\mathbf{u}}_2^\top \mathtt{E}_{23} \hat{\mathbf{u}}_3 = 0, \tag{7}$$

where $\hat{\mathbf{u}} = (\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \hat{\mathbf{u}}_3)$*,* $\hat{\mathbf{u}}_i = [\hat{u}_i, \hat{v}_i, 1]^\top,\ i = 1, \ldots, 3$ *are corrected image points and* $d(\cdot, \cdot)$ *is the Euclidean metric.*

Constraints (6) and (7) are relaxations of (3) for $n = 3$. We only enforce ray $\hat{\mathbf{u}}_1$ to intersect ray $\hat{\mathbf{u}}_2$ and ray $\hat{\mathbf{u}}_2$ to intersect ray $\hat{\mathbf{u}}_3$. Therefore, rays $\hat{\mathbf{u}}_1$ and $\hat{\mathbf{u}}_3$ do not have to intersect in general. It is interesting to see that this relaxation still gives faster, more stable and practically as accurate solution as the L_2-optimal methods [24, 7, 8, 9], which requires all three rays to intersect.

When including the third constraint, i.e. $\hat{\mathbf{u}}_1^\top \mathtt{E}_{13} \hat{\mathbf{u}}_3 = 0$, the solver generator [17] produced a more complicated solver with 94 solutions, which required to compute eigenvalues of 94×94 matrix and is therefore much slower and impractical.

Problem 3 is the problem of finding the minima of a function subject to equality constraints. Such a problem can be solved by introducing Lagrange multipliers λ_1 and λ_2 and forming the Lagrange function $L(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \hat{\mathbf{u}}_3, \lambda_1, \lambda_2)$:

$$L = \sum_{i=1}^{3} d(\mathbf{u}_i, \hat{\mathbf{u}}_i)^2 + \lambda_1 \hat{\mathbf{u}}_1^\top \mathtt{E}_{12} \hat{\mathbf{u}}_2 + \lambda_2 \hat{\mathbf{u}}_2^\top \mathtt{E}_{23} \hat{\mathbf{u}}_3. \tag{8}$$

It can be shown that if $f(\hat{\mathbf{u}}_1^\star, \hat{\mathbf{u}}_2^\star, \hat{\mathbf{u}}_3^\star)$ is a minimum for the original constrained Problem 3, then there exist $\lambda_1^\star$ and $\lambda_2^\star$ such that $(\hat{\mathbf{u}}_1^\star, \hat{\mathbf{u}}_2^\star, \hat{\mathbf{u}}_3^\star, \lambda_1^\star, \lambda_2^\star)$ is a stationary point for the Lagrange function L (8), i.e. a point where all partial derivatives of L are zero.

In this case the Lagrange function $L(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \hat{\mathbf{u}}_3, \lambda_1, \lambda_2)$ (8) is the function of eight unknowns: six coordinates of image points $\hat{\mathbf{u}}_i$, $i = 1, \ldots, 3$ and two Lagrange multipliers λ_1, λ_2. Therefore, to find all stationary points of L (8) we need to solve the following system of eight quadratic polynomial equations in eight unknowns

$$\begin{aligned}
\hat{\mathbf{u}}_1^\top \mathtt{E}_{12} \hat{\mathbf{u}}_2 &= 0, && (9)\\
\hat{\mathbf{u}}_2^\top \mathtt{E}_{23} \hat{\mathbf{u}}_3 &= 0, && (10)\\
2\,\mathtt{S}(\hat{\mathbf{u}}_1 - \mathbf{u}_1) + \lambda_1 \mathtt{S}\,\mathtt{E}_{12} \hat{\mathbf{u}}_2 &= \mathbf{0}, && (11)\\
2\,\mathtt{S}(\hat{\mathbf{u}}_2 - \mathbf{u}_2) + \lambda_1 \mathtt{S}\,\mathtt{E}_{12}^\top \hat{\mathbf{u}}_1 + \lambda_2 \mathtt{S}\,\mathtt{E}_{23} \hat{\mathbf{u}}_3 &= \mathbf{0}, && (12)\\
2\,\mathtt{S}(\hat{\mathbf{u}}_3 - \mathbf{u}_3) + \lambda_2 \mathtt{S}\,\mathtt{E}_{23}^\top \hat{\mathbf{u}}_2 &= \mathbf{0}, && (13)
\end{aligned}$$

where $\mathtt{S}$ is the 2×3 matrix which returns first two coordinates of a three dimensional vector, i.e. the matrix

$$\mathtt{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{14}$$

Note that the equations (11)-(13) are vector equations obtained from partial derivatives of L (8) w.r.t. the coordinates of image points $\hat{\mathbf{u}}_i$.

After solving eight equations (9)-(13), we obtain all candidates $(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \hat{\mathbf{u}}_3)$ for the global minima of the function $f(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2, \hat{\mathbf{u}}_3)$ (5) subject to (6) and (7). In this case, equations (9)-(13) have 31 solutions. Finding the minimum of (5) from 31 values is then a trivial problem.

Once the image points $\hat{\mathbf{u}}_1^\star, \hat{\mathbf{u}}_2^\star, \hat{\mathbf{u}}_3^\star$ are computed, we use these points in the linear least square method [13] to obtain the 3D point $\mathbf{X}^\star$.

Next, we describe a solver which efficiently solves equations (9)-(13). This solver is based on the Gröbner basis method for solving systems of polynomial equations.

4. Efficient Gröbner basis solver

Our goal is to find all solutions of the system (9)-(13) of eight quadratic polynomial equations in eight unknowns. There exist several general algebraic methods [10] which can be used for this purpose and that are implemented, e.g.,

in Maple. However, to solve our problem we do not need to use general algorithms which can solve any system of polynomial equations, and hence are usually inefficient.

In our case, we always have a system of eight quadratic equations in eight unknowns of the form (9)-(13). For each instance of the three-view triangulation problem these eight equations differ only in coefficients which arise from input image points $\mathbf{u}_1$,$\mathbf{u}_2$ and $\mathbf{u}_3$, and from essential matrices $\mathtt{E}_{12}$ and $\mathtt{E}_{23}$.

For such a problem we can design a specific solver that solves only systems of polynomial equations of "our particular form", and that is faster than a general solver and therefore suitable for our application. Recently, the Gröbner basis method [10] was successfully used to create such efficient specific solvers to several computer vision problems [5, 6, 8, 9, 17, 18, 24]. There exists the automatic generator of such efficient specific Gröbner basis solvers [17]. More about the general Gröbner basis method can be found in [10] and about applications of this method for creating efficient specific solvers especially for computer vision problems can be found in [5, 6, 8, 9, 17, 18].

We used the automatic generator [17] to create an efficient specific Gröbner basis solver for systems of polynomial equations of the form (9)-(13).

From the generator we obtained an elimination template which encodes how to multiply the eight input polynomials (9)-(13) by the monomials and then how to eliminate the polynomials using the Gauss-Jordan (G-J) elimination process to obtain all polynomials necessary for constructing the "multiplication matrix", whose eigenvalues and eigenvectors give us the desired solutions. We used the automatic generator [17] to create the multiplication matrix $\mathtt{M}_{\hat{u}_2}$ for multiplication by the first coordinate of the second corrected image point $\hat{\mathbf{u}}_2$.

To get the elimination template, the generator first generated all monomial multiples of the initial eight polynomial equations up to the total degree of six. This resulted in 1320 polynomials in 1287 monomials. Then the generator removed all unnecessary polynomials and monomials, i.e., polynomials and monomials that do not affect the resulting multiplication matrix. This resulted in a 274×305 matrix $\mathtt{Q}$ representing the polynomials necessary for constructing the multiplication matrix $\mathtt{M}_{\hat{u}_2}$, i.e., the elimination template.

The final online solver of the eight equation (9)-(13) performs only one G-J elimination or QR decomposition of a single matrix $\mathtt{Q}$, which is constructed from the elimination template built by the automatic generator in the offline stage. This matrix contains coefficients which arise from specific measurements, i.e., image points $\mathbf{u}_1$,$\mathbf{u}_2$ and $\mathbf{u}_3$, and from the essential matrices $\mathtt{E}_{12}$ and $\mathtt{E}_{23}$. After G-J elimination of matrix $\mathtt{Q}$, multiplication matrix $\mathtt{M}_{\hat{u}_2}$ can be created from its rows. It is known [10] that eigenvalues of $\mathtt{M}_{\hat{u}_2}$ give the solutions to $\hat{u}_2$ The solutions to the remaining seven unknowns can be found from the eigenvectors of $\mathtt{M}_{\hat{u}_2}$ [10].

The computation of eigenvalues and eigenvectors of a multiplication matrix is a way of obtaining solutions. It is used in almost all Gröbner basis solvers to computer vision problems [5, 18] including the 3-view triangulation [7, 9, 8, 24] and that is also implemented in the automatic generator [17].

Recently, it was shown [6] that it is often better to replace the time-consuming eigenvalue computation with a computation of roots of the characteristic polynomial of a multiplication matrix. In [6] a method based on Danilevskii algorithm [11] for computing characteristic polynomials and on Sturm sequences [15] for computing roots of a single-variable polynomial was used to significantly speed-up several important minimal computer vision problems.

We have decided to use this method instead of computing eigenvalues and eigenvectors of the multiplication matrix $\mathtt{M}_{\hat{u}_2}$. Therefore, in our solver we first compute the characteristic polynomial of $\mathtt{M}_{\hat{u}_2}$ using Danilevskii [11] algorithm and then we compute the roots of this polynomial using efficient Sturm-sequences [15]. Moreover, we can work with normalized image coordinates and therefore we can calculate the roots of the characteristic polynomial only on a feasible interval. Using Danilevskii algorithm [11] and efficient Sturm-sequences [15] we have obtained around 50% speedup over eigenvalue computations.

5. Experiments

In this section we evaluate the precision and speed of the proposed solver on both synthetic and real datasets and compare it with the state-of-the-art algebraic three-view triangulation solvers based on QR decomposition [9] and based on SVD [8]. For real experiments we have used only the QR-based solver [9] since it is the fastest algebraic solver to the L_2-optimal three-view triangulation problem and with slightly better stability and accuracy than the SVD-based solver [8]. In some experiments we have compared our solver also with the standard linear least square method [14].

5.1. Synthetic data

We have studied the performance of the proposed solver on synthetically generated ground-truth 3D scenes. These scenes were generated using 3D points randomly distributed in cube $[-10, 10]^3$. Each 3D point was projected by three cameras with random or specific feasible orientation and position depending on the testing configuration. Finally, Gaussian noise with standard deviation σ was added to the image points assuming a 1000×1000 pixel image.

In the first experiment we have studied the behavior of the presented solver on noise free data to check its numerical stability and compared the results with the numerical stability of the QR-based solver [9] and the SVD-based

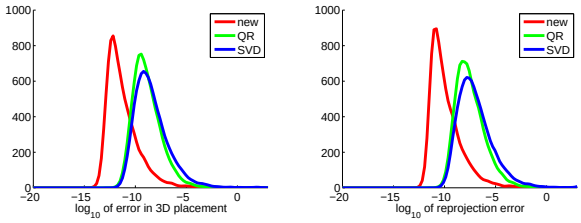

Figure 2. Log_{10} of the Euclidean distance of the estimated 3D point $\mathbf{X}^\star$ and the ground-truth 3D point $\mathbf{X}$ (left) and log_{10} of the reprojection error assuming $1000px \times 1000px$ image (right) for noise free synthetic scene.

solver [8]. In this experiment 10000 points randomly placed in a 3D cube $[-10, 10]^3$ were projected by 10000 random triplets of cameras with feasible position and orientation. The projected image points $\mathbf{u}_i$ were used to find the 3D point $\mathbf{X}^\star$ using the presented solver, the QR-based solver [9] and the SVD-based solver [8].

Figure 2 (left) shows the log_{10} of the error in 3D placement, i.e. log_{10} of the Euclidean distance of the estimated 3D point $\mathbf{X}^\star$ and the ground-truth 3D point $\mathbf{X}$. Figure 2 (right) shows the log_{10} of the reprojection error from the same experiment, i.e. log_{10} of the Euclidean distance of the estimated image points $\hat{\mathbf{u}}_i^\star$ and the ground-truth image points $\mathbf{u_i}$ assuming $1000px \times 1000px$ image. It can be seen that all tested solvers, the new proposed solver (red), the QR-based solver [9] (green) and the SVD-based solver [8] (blue) give very stable results but the stability and the accuracy of our new solver are slightly better.

This can be seen also from Table 1, which shows the number of errors in 3D placement (from 10000 measurements) larger than some level. It is visible that the new proposed solver gives fewer large errors than the QR-based solver [9] and the SVD-based solver [8].

3D error	> 1	$> 10^{-1}$	$> 10^{-2}$	$> 10^{-3}$	$> 10^{-5}$
New	4	6	9	18	59
QR	6	7	13	27	141
SVD	25	36	50	93	358

Table 1. The number of errors in 3D placement (from 10000 measurements) larger than a specified 3D error.

Figure 3 shows the histogram of the number of real solutions returned in this first experiment by all examined methods. Our new solver usually returns from 3 to 7 real candidate solutions for the global minima with the mean equal to 5.43. The mean of the number of real solution from the QR-based method [9] was 7.14 and from the SVD-based solver [8] 6.98.

In the next experiment we have studied the behavior of our new solver for cameras in the "turn-table configuration", i.e. for cameras with intersecting optical axes. We

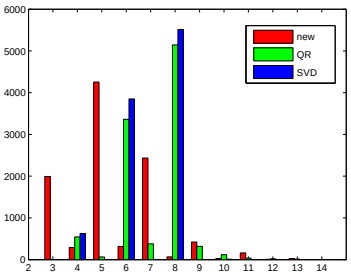

Figure 3. Number of real solutions.

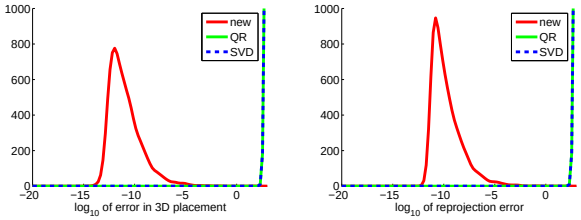

Figure 4. Log_{10} of the Euclidean distance of the estimated 3D point $\mathbf{X}^\star$ and the ground-truth 3D point $\mathbf{X}$ (left) and log_{10} of the reprojection error assuming $1000px \times 1000px$ image (right) for noise free synthetic scene and cameras in the "turn-table configuration".

have again checked the numerical stability of our new solver for noise free data and compared it with the results of the QR-based solver [9] and the SVD-based solver [8].

In this experiment 10000 points randomly placed in a 3D cube $[-10, 10]^3$ were projected by 10000 triplets of cameras with intersecting optical axes. The projected image points $\mathbf{u}_i$ were used to find the 3D point $\mathbf{X}^\star$ using our new solver, the QR-based solver [9] and the SVD-based solver [8].

Figure 4 (left) shows the log_{10} of the error in 3D placement and Figure 4 (right) the log_{10} of the reprojection error from this experiment. In this case the QR-based solver [9] (green) and the SVD-based solver [8] (dashed-blue) failed in all 10000 measurements and didn't deliver any result. This is displayed as a peak on the right hand sides of graphs in Figure 4. Failures of these two solvers are probably caused by the fact that in the "turn-table configuration" the projection matrices contain zeros on places were these solvers assume non-zero elements. On the other hand, our new solver (red) returns very stable and accurate results similar to the results for the general configuration.

Forward and sideways motions are critical configurations for all three considered solvers. However, for motions which are close to these configurations, i.e. they are not precisely forward or sideways, all solvers return sufficiently accurate results. This can be seen from the next experiment.

In this experiment 10000 points randomly placed in a 3D cube $[-10, 10]^3$ were projected by 10000 triplets of cameras. These cameras were first pointed in the same direction

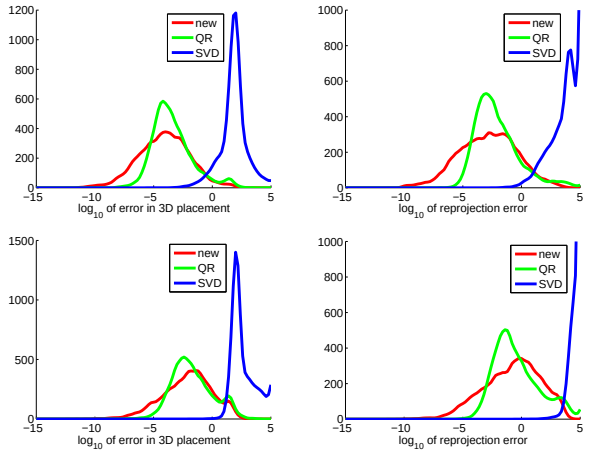

Figure 5. Log_{10} of the Euclidean distance of the estimated 3D point $\mathbf{X}^{\star}$ and the ground-truth 3D point $\mathbf{X}$ (left) and log_{10} of the reprojection error assuming $1000px \times 1000px$ image (right) for noise free synthetic scene and cameras in close-to sideway motions, which were "moved" from the pure sideways motion by right-multiplying projection matrices by rotations with random axes and the rotation angle $1/100$ degrees (top) and $1/500$ degrees (bottom).

(optical axes were intersecting at the infinity) and translated laterally to simulate the "sideways motion". Then projection matrices of these cameras were right-multiplied by different random rotations of a small angle to simulate a close-to critical configuration. This multiplication by random rotation matrices slightly rotate optical axes of cameras (not to intersect at the infinity) and simultaneously moves camera centres (not to lay on a line).

Figure 5 shows the log_{10} of the error in 3D placement (left) and the log_{10} of the reprojection error (right) for close-to sideway motions, which were "moved" from the pure sideway motion by right-multiplying projection matrices by rotations with random axes and the rotation angle $1/100$ degrees (top) and $1/500$ degrees (bottom).

It can be seen that even for configurations very close to the critical pure sideways motion our new solver gives reasonable results and slightly outperforms the QR-based solver [9]. The QR-based solver returns a higher number of larger errors, i.e. it has more failures than our new solver, Table 2. The median values of the errors in 3D placement and the reprojection errors for both these solvers and sideway motions "moved" by random rotations of given angles are in Tables 3 and 4. In this experiment the SVD-based solver [8] failed in most from 10000 measurements.

The final synthetic experiment shows the behavior of the new solver in the presence of noise in image measurements.

In this experiment we created 3D scenes as in the first experiment with camera triplets with random feasible position and orientation. In this case the Gaussian noise with standard deviation $\sigma = \frac{1}{3}px$, corresponding to $1px$ noise, was

3D error	> 10	$> 10^{0}$	$> 10^{-1}$
New	98	264	649
QR	276	481	829

Table 2. The number of errors in 3D placement (from 10000 measurements) larger than a specified 3D error for close-to sideways motions "moved" from the pure sideway motion by random rotations of $1/100$ degrees.

	3D error			
rotation	$1/10$ deg	$1/100$ deg	$1/500$ deg	$1/1000$ deg
New	7.23^{-8}	8.53^{-5}	9.81^{-3}	5.83^{-2}
QR	1.48^{-6}	1.69^{-4}	1.06^{-2}	6.51^{-2}

Table 3. The median of the errors in 3D placement (from 10000 measurements) for close-to sideways motions moved from the pure sideway motion by multiplying by random rotations of different given angles.

	Reprojection error			
rotation	$1/10$ deg	$1/100$ deg	$1/500$ deg	$1/1000$ deg
New	1.57^{-6}	1.82^{-3}	2.56^{-1}	1.62
QR	4.09^{-5}	3.54^{-3}	1.74^{-1}	0.98

Table 4. The median of the reprojection errors (from 10000 measurements) for close-to sideway motions moved from the pure sideway motion by multiplying by random rotations of different given angles.

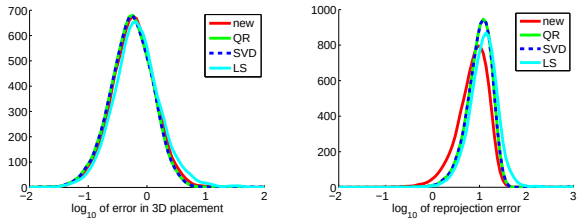

Figure 6. Log_{10} of the Euclidean distance of the estimated 3D point $\mathbf{X}^{\star}$ and the ground-truth 3D point $\mathbf{X}$ (left) and log_{10} of the reprojection error assuming $1000px \times 1000px$ image (right) for Gaussian noise $1px$.

added to projected image points assuming a 1000×1000 pixel image. It can be seen from Figure 6 that all considered solvers return very similar and good results and our new solver returns practically as accurate 3D points as the L_2-optimal QR-based and SVD-based solvers, see Figure 6 (left). In this case we have used for comparison also the LS solver (cyan), which directly takes input measurements $\mathbf{u}_i$ and uses least square method [14] to obtain the 3D point.

5.2. Real data

For experiments on real data we selected three popular datasets: Oxford Dino and Corridor datasets [2] and Notre Dame dataset [1]. The first one represents a "turntable configuration", the second one a forward motion and

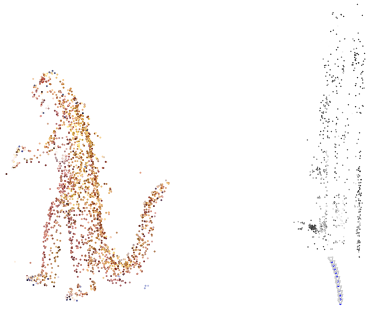

Figure 7. Resulting 3D reconstructions triangulated using our new algorithm for the Dino dataset (left) and the Corridor dataset (right).

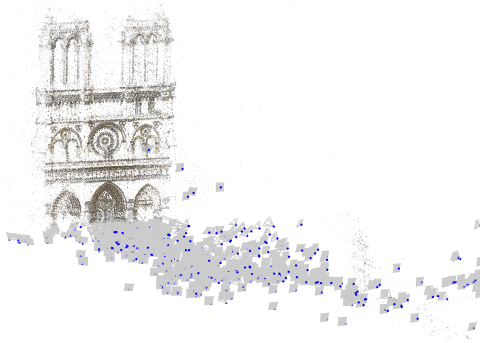

Figure 8. Resulting 3D reconstruction triangulated using our new algorithm for the Notre Dame dataset.

the last one a general scene.

In every experiment we used camera matrices and tracks with image measurements to create points in the 3D space. If camera lens distortion coefficients were available, we first undistorted the image measurements.

We dropped all tracks smaller than three. For tracks with length equal to three we used these three cameras to triangulate the 3D point using our new algorithm, the QR L_2-optimal algorithm [9] and the least squares solution [14]. For tracks with length greater than three we considered different strategies:

1. Three cams: We used the first, the middle and the last point of the track. This is a reasonable choice for tracked sequences like the Dino or the Corridor sequence. For the Notre Dame dataset it is equal to selecting three random points.

2. RANSAC: A triplet of points was randomly selected and a 3D point was calculated either using the linear least square solver [14] (further denoted "Ransac ls") or our new solver (further in the text "Ransac new"). We did not consider the QR-based solver [9] due to its low speed. Then the mean squared reprojection error w.r.t. to all cameras in the track was calculated. We repeated this procedure several times and selected the triplet with the smallest error (denoted triplet). Three points from the triplet were then used as a source for other triangulation methods (QR and new).

We also evaluated the least squares solution [14], which uses all points in the track (LS).

The Dino sequence has 4983 tracks from which 2661 have length greater than two after removing outliers. We removed the tracks which could not be triangulated, i.e. those triangulated 3D points that did not reproject to at least three cameras within three pixels reprojection error. It took 6.45 seconds to triangulate these 2661 points using our new algorithm and 35.14 seconds using the QR algorithm [7]. This time includes the time needed for extracting essential matrices from given camera matrices, calculation of optimal image measurements and calculation of the 3D point using the least squares method. Table 5 summarizes triangulation results. We can see that the least square solution obtained using all points from the track (LS) provides a better estimate compared to our new solver and the L_2-optimal QR solver [9] when only the first, the middle and the last point of the track were considered (Three cams strategy). Our new solver and also the QR-based solver [9] become better when a more reasonable triplet of cameras is selected (Ransac strategy).

MSE	LS	triplet	QR	new
All cams	0.2490	-	-	-
Three cams	-	-	0.2638	0.2523
Ransac ls	-	0.1716	0.1730	0.1706
Ransac new	-	0.1723	0.1723	0.1723

Table 5. Dino sequence reconstruction errors in pixels.

The Oxford corridor sequence contains 737 tracks with length greater than two after removing outliers. It took 1.57 seconds with our new solver and 9.3 seconds with the QR algorithm [9] to triangulate these points. The QR algorithm failed several times due to numerical instability. The mean squared reprojection errors are presented in Table 6.

MSE	LS	triplet	QR	new
All cams	0.1445	-	-	-
Three cams	-	-	0.1555	0.1554
Ransac ls	-	0.1281	0.1573	0.1316
Ransac new	-	0.1279	0.1440	0.1279

Table 6. Corridor sequence reconstruction errors in pixels.

The Notre Dame dataset consists of 715 cameras and 127431 tracks. About 121980 of them are longer than two and not contaminated by outliers. It took 275.7 seconds to triangulate these tracks using our new solver. The mean

squared reprojection error of our new solver is 0.337 pixels and 0.4102 pixels for points triangulated using the linear least square method [14] using all points in the track.

It is interesting to observe that our sub-optimal solver produces a more optimal solution with respect to the whole track, than the L_2-optimal QR algorithm [9].

6. Conclusion

We have proposed a new fast and stable algebraic solution to the problem of L_2 triangulation from three views. We have formulate the search for the minima of the L_2-objective function subject to relaxed equality epipolar constraints as a system of eight polynomial equations in eight unknowns using Lagrange multipliers. We have efficiently solved the system using the Gröbner basis method [10, 17] combined with the Danilevskii [6, 11] method for computing characteristic polynomials and efficient Sturm-sequences [15] for finding roots of a single-variable polynomial. Our formulation leads to less (31) solutions than was the number (47-66) of solutions obtained by the state-of-the-art L_2-optimal solvers [24, 7, 8, 9].

The final solver can be generated using the automatic generator of Gröbner basis solvers [17] without employing special techniques for improving numerical stability used in [7, 8, 9] and its optimized version [6] runs about $1ms$. The new relaxed solver is faster, more robust w.r.t. critical configurations, numerically more stable and practically as accurate as the state-of-the-art L_2-optimal solutions [24, 7, 8, 9], what we have demonstrated in experiments on synthetic and real datasets.

References

[1] http://phototour.cs.washington.edu/datasets.

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

[3] S. Agarwal, M. Chandraker, F. Kahl, D. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. In *ECCV'06*, volume 1, pages 592–605, 2006.

[4] C. Aholt, S. Agarwal, and R. R. Thomas. A qcqp approach to triangulation. In *ECCV'12*, pages 654–667, 2012.

[5] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *CVPR'08*, 2008.

[6] M. Bujnak, Z. Kukelova, and T. Pajdla. Making Minimal Solvers Fast. In *CVPR'12*, 2012.

[7] M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. In *ACCV'07*, 2007.

[8] M. Byröd, K. Josephson, and K. Åström. Improving numerical accuracy of gröbner basis polynomial equation solvers. In *ICCV'07*, 2007.

[9] M. Byröd, K. Josephson, and K. Åström. A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations. In *ECCV'08*, 2008.

[10] D. Cox, J. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer, 2nd edition, 2005.

[11] A. M. Danilevskii. The numerical solution of the secular equation (russian). *Matem. sbornik*, 44:169–171, 1937. In Russian.

[12] R. I. Hartley and F. Schaffalitzky. l_∞ minimization in geometric reconstruction problems. In *CVPR'04*, 2004.

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

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[15] D. G. Hook and P. R. McAree. Graphics gems. In A. S. Glassner, editor, *Graphics gems*, chapter Using Sturm sequences to bracket real roots of polynomial equations, pages 416–422. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[16] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-sturm vs. optimal correction. In *BMVC'08*, 2008.

[17] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *ECCV'08, Part III*, volume 5304 of *Lecture Notes in Computer Science*, 2008.

[18] Z. Kukelova and T. Pajdla. A minimal solution to radial distortion autocalibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2410–2422, December 2011.

[19] P. Lindstrom. Triangulation made easy. In *CVPR'10*, pages 1554–1561. IEEE, 2010.

[20] F. Lu and R. Hartley. A fast optimal algorithm for l2 triangulation. In *ACCV'07*, 2007.

[21] Microsoft. Photosynth - http://www.photosynth.net.

[22] Y. Min. L-infinity norm minimization in the multiview triangulation. In *AICI'10*, 2010.

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

[24] H. Stéwenius, F. Schaffalitzky, and D. Nistér. How hard is 3-view triangulation really? In *ICCV'05*, volume 1, pages 686–693, 2005.

[25] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment a modern synthesis. In *Vision Algorithms: Theory and Practice, LNCS*, pages 298–375. Springer Verlag, 2000.