# Triangulation for Points on Lines

Adrien Bartoli, Jean-Thierry Lapresté

ECCV'06 – In Proceedings of the European Conference on Computer Vision, Graz, Austria, May 2006.

# Triangulation for Points on Lines

Adrien Bartoli and Jean-Thierry Lapresté

LASMEA – CNRS / Université Blaise Pascal
Clermont-Ferrand, France
Adrien.Bartoli@gmail.com

**Abstract.** Triangulation consists in finding a 3D point reprojecting the best as possible onto corresponding image points. It is classical to minimize the reprojection error, which, in the pinhole camera model case, is nonlinear in the 3D point coordinates. We study the triangulation of points lying on a 3D line, which is a typical problem for Structure-From-Motion in man-made environments. We show that the reprojection error can be minimized by finding the real roots of a polynomial in a single variable, which degree depends on the number of images. We use a set of transformations in 3D and in the images to make the degree of this polynomial as low as possible, and derive a practical reconstruction algorithm. Experimental comparisons with an algebraic approximation algorithm and minimization of the reprojection error using Gauss-Newton are reported for simulated and real data. Our algorithm finds the optimal solution with high accuracy in all cases, showing that the polynomial equation is very stable. It only computes the roots corresponding to feasible points, and can thus deal with a very large number of views – triangulation from hundreds of views is performed in a few seconds. Reconstruction accuracy is shown to be greatly improved compared to standard triangulation methods that do not take the line constraint into account.

## 1 Introduction

Triangulation is one of the main building blocks of Structure-From-Motion algorithms. Given image feature correspondences and camera matrices, it consists in finding the position of the underlying 3D feature, by minimizing some error criterion. This criterion is often chosen as the reprojection error – the Maximum Likelihood criterion for a Gaussian, centred and *i.i.d.* noise model on the image point positions - though other criteria are possible [5, 9, 10].

Traditionally, triangulation is carried out by some sub-optimal procedure and is then refined by local optimization, see *e.g.* [7]. A drawback of this is that convergence to the optimal solution is not guaranteed. Optimal procedures for triangulating points from two and three views were proposed in [6, 13].

We address the problem of triangulating points lying on a line, that is, given image point correspondences, camera matrices and a 3D line, finding the 3D point lying on the 3D line, such that the reprojection error is minimized.

Our main contribution is to show that the problem can be solved by computing the real roots of a degree-($3n$-2) polynomial, where $n$ is the number of views. Extensive experiments on simulated data show that the polynomial is very well balanced since large number of views and large level of noise are handled. The method is valid whatever the calibration level of the cameras is – projective, affine, metric or Euclidean.

One may argue that triangulating points on a line only has a theoretical interest since in practice, triangulating a line from multiple views is done by minimizing the reprojection error over its supporting points which 3D positions are hence reconstructed along with the 3D line. Indeed, most work consider the case where the supporting points do *not* match accross the images, see *e.g.* [3]. When one identifies correspondences of supporting points accross the images, it is fruitful to incorporate these constraints into the bundle adjustment, as is demonstrated by our experiments. This is typically the case in man-made environments, where one identifies *e.g.* matching corners at the meet of planar facades or around windows. Bartoli *et al.* [2] dubbed Pencil-of-Points or 'POP' this type of features. In order to find an initial 3D reconstruction, a natural way is to compute the 3D line by some means (*e.g.* by ignoring the matching constraints of the supporting points, from 3D primitives such as the intersection of two planes, or from a registered wireframe CAD model) and then to triangulate the supporting point correspondences using point on line triangulation. The result can then be plugged into a bundle adjustment incorporating the constraints.

Our triangulation method is derived in §2. A linear least squares method minimizing an algebraic distance is provided in §3. Gauss-Newton refinement is summarized in §4. Experimental results are reported in §5 and our conclusions in §6.

*Notation.* Vectors are written using bold fonts, *e.g.* $\mathbf{q}$, and matrices using sans-serif fonts, *e.g.* $\mathsf{P}$. Almost everything is homogeneous, *i.e.* defined up to scale. Equality up to scale is denoted $\sim$. The inhomogenous part of a vector is denoted using a bar, *e.g.* $\mathbf{q}^\mathsf{T} \sim (\bar{\mathbf{q}}^\mathsf{T} \ 1)$ where $^\mathsf{T}$ is transposition. Index $i = 1, \ldots, n$, and sometime $j$ are used for the images. The point in the $i$-th image is $\mathbf{q}_i$. Its elements are $\mathbf{q}_i^\mathsf{T} \sim (q_{i,1} \ q_{i,2} \ 1)$. The 3D line joining points $\mathbf{M}$ and $\mathbf{N}$ is denoted $(\mathbf{M}, \mathbf{N})$. The $\mathcal{L}_2$-norm of a vector is denoted as in $\|\mathbf{x}\|^2 = \mathbf{x}^\mathsf{T}\mathbf{x}$. The Euclidean distance measure $d_e$ is defined by:

$$d_e^2(\mathbf{x}, \mathbf{y}) = \left\| \frac{\mathbf{x}}{x_3} - \frac{\mathbf{y}}{y_3} \right\|^2 = \left( \frac{x_1}{x_3} - \frac{y_1}{y_3} \right)^2 + \left( \frac{x_2}{x_3} - \frac{y_2}{y_3} \right)^2 . \tag{1}$$

*Related work.* Optimal procedures for triangulating points in 3D space, and points lying on a plane were previously studied. Hartley and Sturm [6] showed that triangulating points in 3D space from two views, in other words finding a pair of points satisfying the epipolar geometry and lying as close as possible

to the measured points, can be solved by finding the real roots of a degree-6 polynomial. The optimal solution is then selected by straightforward evaluation of the reprojection error. Stewénius *et al.* [13] extended the method to three views. The optimal solution is one of the real roots of a system of 3 degree-6 polynomials in the 3 coordinates of the point. Chum *et al.* [4] show that triangulating points lying on a plane, in other words finding a pair of points satisfying an homography and lying as close as possible to the measured points, can be solved by finding the real roots of a degree-8 polynomial.

## 2 Minimizing the Reprojection Error

We derive our optimal triangulation algorithm for point on line, dubbed 'POLY'.

### 2.1 Problem Statement and Parameterization

We want to compute a 3D point $\mathbf{Q}$, lying on a 3D line $(\mathbf{M}, \mathbf{N})$, represented by two 3D points $\mathbf{M}$ and $\mathbf{N}$. The $(3 \times 4)$ perspective camera matrices are denoted $\mathsf{P}_i$ with $i = 1, \ldots, n$ the image index. The problem is to find the point $\hat{\mathbf{Q}}$ such that:

$$\hat{\mathbf{Q}} \sim \arg \min_{\mathbf{Q} \in (\mathbf{M},\mathbf{N})} \mathcal{C}_n^2(\mathbf{Q}),$$

where $\mathcal{C}_n$ is the $n$-view *reprojection error*:

$$\mathcal{C}_n^2(\mathbf{Q}) = \sum_{i=1}^{n} d_e^2(\mathbf{q}_i, \mathsf{P}_i\mathbf{Q}). \tag{2}$$

We parameterize the point $\mathbf{Q} \in (\mathbf{M}, \mathbf{N})$ using a single parameter $\lambda \in \mathbb{R}$ as:

$$\mathbf{Q} \sim \lambda\mathbf{M} + (1-\lambda)\mathbf{N} \sim \lambda(\mathbf{M} - \mathbf{N}) + \mathbf{N}. \tag{3}$$

Introducing this parameterization into the reprojection error (2) yields:

$$\mathcal{C}_n^2(\lambda) = \sum_{i=1}^{n} d_e^2(\mathbf{q}_i, \mathsf{P}_i(\lambda(\mathbf{M} - \mathbf{N}) + \mathbf{N})).$$

Defining $\mathbf{b}_i = \mathsf{P}_i(\mathbf{M} - \mathbf{N})$ and $\mathbf{d}_i = \mathsf{P}_i\mathbf{N}$, we get:

$$\mathcal{C}_n^2(\lambda) = \sum_{i=1}^{n} d_e^2(\mathbf{q}_i, \lambda\mathbf{b}_i + \mathbf{d}_i). \tag{4}$$

Note that a similar parameterization can be derived by considering the inter-image homographies induced by the 3D line [12].

### 2.2 Simplification

We simplify the expression (4) of the reprojection error by changing the 3D coordinate frame and the image coordinate frames. This is intended to lower the degree of the polynomial equation that will ultimately have to be solved. Since the reprojection error is based on Euclidean distances measured in the images, only rigid image transformations are allowed to keep invariant the error function, while full projective homographies can be used in 3D. We thus setup a standard canonical 3D coordinate frame, see *e.g.* [8], such that the first camera matrix becomes $\mathsf{P}_1 \sim (\ \mathrm{I} \quad \mathbf{0})$. Note that using a projective basis does not harm Euclidean triangulation since the normalization is undone once the point is triangulated. The canonical basis is setup by the following simple operations:

$$\mathsf{H} \leftarrow \begin{pmatrix} \mathsf{P}_1 \\ 0\ 0\ 0\ 1 \end{pmatrix} \qquad \mathsf{P}_i \leftarrow \mathsf{P}_i\mathsf{H}^{-1} \qquad \mathbf{M} \leftarrow \mathsf{H}\mathbf{M} \qquad \mathbf{N} \leftarrow \mathsf{H}\mathbf{N}.$$

Within this coordinate frame, we can write $\mathbf{M}^\mathsf{T} = (\bullet\ \bullet\ 1\ \bullet)$ and $\mathbf{N}^\mathsf{T} = (\bullet\ \bullet\ 1\ \bullet)$ without loss of generality, as pointed out in [7, §A6], from which we get:

$$\begin{aligned} \mathbf{b}_1 &= \mathsf{P}_1(\mathbf{M} - \mathbf{N}) = (b_{1,1}\ \ b_{1,2}\ \ 0)^\mathsf{T} \\ \mathbf{d}_1 &= \mathsf{P}_1\mathbf{N} = (d_{1,1}\ \ d_{1,2}\ \ 1)^\mathsf{T}. \end{aligned}$$

We then apply a rigid transformation $\mathsf{T}_i$ in each image defined such that $\mathsf{T}_i\mathbf{b}_i$ lies on the $y$-axis and such that $\mathsf{T}_i\mathbf{d}_i = \mathsf{T}_i\mathsf{P}_i\mathbf{N}$ lies at the origin. This requires that the point $\mathbf{N}$ does not project at infinity is any of the images. We ensure this by constraining $\mathbf{N}$ to project as close as possible to one of the image points[1], say $\mathbf{q}_1$. The reprojection error (4) for the first view is $\mathcal{C}_1^2(\lambda) = d_e^2(\mathbf{q}_1, \lambda\mathbf{b}_1 + \mathbf{d}_1) = \|\lambda\bar{\mathbf{b}}_1 + \bar{\mathbf{d}}_1 - \bar{\mathbf{q}}_1\|^2$. We compute $\lambda$ as the solution of $\frac{\partial\mathcal{C}_1^2}{\partial\lambda} = 0$, which gives, after some minor calculations, $\lambda = (\bar{\mathbf{q}}_1 - \bar{\mathbf{d}}_1)^\mathsf{T}\bar{\mathbf{b}}_1 / \|\bar{\mathbf{b}}_1\|^2$. Substituting in equation (3) yields the following operations:

$$\mathbf{N} \leftarrow \frac{(\mathsf{P}_1\mathbf{N} - \mathbf{q}_1)^\mathsf{T}\mathsf{P}_1(\mathbf{M} - \mathbf{N})}{\|\mathsf{P}_1(\mathbf{M} - \mathbf{N})\|^2}(\mathbf{M} - \mathbf{N}) + \mathbf{N}.$$

Obviously, the $\mathbf{d}_i = \mathsf{P}_i\mathbf{N}$ must be recomputed. These simplications lead to:

$$\mathbf{b}_1 = (0\ \ b_{1,2}\ \ 0)^\mathsf{T} \qquad \mathbf{d}_1 = (0\ \ 0\ \ 1)^\mathsf{T} \qquad \mathbf{b}_{i>1} = (0\ \ b_{i,2}\ \ b_{i,3})^\mathsf{T} \qquad \mathbf{d}_{i>1} = (0\ \ 0\ \ d_{i,3})^\mathsf{T}.$$

The rigid transformations $\mathsf{T}_i$ are quickly derived below. For each image $i$, we look for $\mathsf{T}_i$ mapping $\mathbf{d}_i$ to the origin, and $\mathbf{b}_i$ to a point on the $y$-axis. We decompose $\mathsf{T}_i$ as a rotation around the origin and a translation:

$$\mathsf{T}_i = \begin{pmatrix} \mathsf{R}_i & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix} \begin{pmatrix} \mathrm{I} & -\mathbf{t}_i \\ \mathbf{0}^\mathsf{T} & 1 \end{pmatrix}.$$

[1] Note that this is equivalent to solving the single view triangulation problem.

The translation is directly given from $\mathsf{T}_i\mathbf{d}_i \sim (0\ 0\ 1)^\mathsf{T}$ as $\mathbf{t}_i = \bar{\mathbf{d}}_i/d_{i,3}$. For the rotation, we consider $\mathsf{T}_i\mathbf{b}_i \sim (0\ \bullet\ \bullet)^\mathsf{T}$, from which, setting $\mathbf{r}_i = \bar{\mathbf{b}}_i - b_{i,3}\mathbf{t}_i$, we obtain $\mathsf{R}_i = \begin{pmatrix} r_{i,2} & -r_{i,1} \\ r_{i,1} & r_{i,2} \end{pmatrix} / \|\bar{\mathbf{r}}_i\|$.

This leads to the following expression for the reprojection error (4) where we separated the leading term:

$$\mathcal{C}_n^2(\lambda) = q_{1,1}^2 + (\lambda b_{1,2} - q_{1,2})^2 + \sum_{i=2}^{n} \left( q_{i,1}^2 + \left( \frac{\lambda b_{i,2}}{\lambda b_{i,3} - d_{i,3}} - q_{i,2} \right)^2 \right).$$

The constant terms $q_{1,1}^2$ and $q_{i,1}^2$ represent the vertical counterparts of the point to line distance in the images. This means that only the errors along the lines are to be minimized.

### 2.3 Solving the Polynomial Equation

Looking for the minima of the reprojection error $\mathcal{C}_n^2$ is equivalent to finding the roots of its derivative, *i.e.* solving $\frac{\partial \mathcal{C}_n^2}{\partial \lambda} = 0$. Define $\mathcal{D}_n = \frac{1}{2}\frac{\partial \mathcal{C}^2}{\partial \lambda}$:

$$\mathcal{D}_n(\lambda) = (\lambda b_{1,2} - q_{1,2}) b_{1,2} + \sum_{i=2}^{n} \left( \frac{\lambda b_{i,2}}{\lambda b_{i,3} + d_{i,3}} - q_{i,2} \right) \left( \frac{b_{i,2} d_{i,3}}{(\lambda b_{i,3} + d_{i,3})^2} \right).$$

This is a nonlinear function. Directly solving $\mathcal{D}_n(\lambda) = 0$ is therefore very difficult in general. We thus define $\tilde{\mathcal{D}}_n(\lambda) = \mathcal{D}_n(\lambda)\mathcal{K}_n(\lambda)$, where we choose $\mathcal{K}_n$ in order to cancel out the denominators including $\lambda$ in $\mathcal{D}_n$. Finding the zeros of $\tilde{\mathcal{D}}_n$ is thus equivalent to finding the zeros of $\mathcal{D}_n$. Inspecting the expression of $\mathcal{D}_n$ reveals that $\mathcal{K}_n(\lambda) = \prod_{i=2}^{n} (\lambda b_{i,3} + d_{i,3})^3$ does the trick:

$$\begin{aligned} \tilde{\mathcal{D}}_n(\lambda) = {} & (\lambda b_{1,2} - q_{1,2}) b_{1,2} \prod_{i=2}^{n} (\lambda b_{i,3} + d_{i,3})^3 \\ & + \sum_{i=2}^{n} \left( b_{i,2} d_{i,3} \left( \lambda b_{i,2} - q_{i,2} (\lambda b_{i,3} + d_{i,3}) \right) \prod_{j=2, j \neq i}^{n} (\lambda b_{j,3} + d_{j,3})^3 \right). \end{aligned} \tag{5}$$

As expected, $\tilde{\mathcal{D}}_n$ is a polynomial function, whose degree depends on the number of images $n$. We observe that cancelling the denominator out for the contribution of each $(i > 1)$-image requires to multiply $\mathcal{D}_n$ by a cubic, namely $(\lambda b_{i,3} + d_{i,3})^3$. Since the polynomial required for image $i = 1$ is linear, the degree of the polynomial to solve is $3(n-1) + 1 = 3n - 2$.

Given the real roots $\lambda_k$ of $\tilde{\mathcal{D}}_n(\lambda)$, that we compute as detailled below for different number of images, we simply select the one for which the reprojection error is minimized, *i.e.* $\hat{\lambda} = \arg\min_k \mathcal{C}_n^2(\lambda_k)$, substitute it in equation (3) and transfer the recovered point back to the original coordinate frame:

$$\hat{\mathbf{Q}} \sim \mathsf{H}^{-1} \left( \hat{\lambda}\mathbf{M} + \left(1 - \hat{\lambda}\right)\mathbf{N} \right).$$

*A single image.* For $n = 1$ image, the point is triangulated by projecting its image onto the image projection of the line. The intersection of the associated viewing ray with the 3D line gives the 3D point. In our framework, equation (5) is indeed linear in $\lambda$ for $n = 1$: $\tilde{\mathcal{D}}_1(\lambda) = (\lambda b_{1,2} - q_{1,2})b_{1,2} = b_{1,2}^2\lambda - q_{1,2}b_{1,2}$.

*A pair of images.* For $n = 2$ images, equation (5) gives:

$$\tilde{\mathcal{D}}_2(\lambda) = (\lambda b_{1,2} - q_{1,2})b_{1,2}(\lambda b_{2,3} + d_{2,3})^3 + b_{2,2}d_{2,3}(\lambda b_{2,2} - q_{2,2}(\lambda b_{2,3} + d_{2,3})),$$

which is a quartic in $\lambda$ that can be solved in closed-form using Cardano's formulas: $\tilde{\mathcal{D}}_2(\lambda) \sim \sum_{d=1}^{4} c_d\lambda^d$, with:

$$\begin{cases} c_0 = -q_{2,2}d_{2,3}^2b_{2,2} - b_{1,2}q_{1,2}d_{2,3}^3 \\ c_1 = d_{2,3}(b_{2,2}^2 - 3b_{1,2}q_{1,2}b_{2,3}d_{2,3} + b_{1,2}^2d_{2,3}^2 - q_{2,2}b_{2,3}b_{2,2}) \\ c_2 = 3b_{1,2}b_{2,3}d_{2,3}(b_{1,2}d_{2,3} - q_{1,2}b_{2,3}) \\ c_3 = b_{1,2}b_{2,3}^2(3b_{1,2}d_{2,3} - q_{1,2}b_{2,3}) \\ c_4 = b_{1,2}^2b_{2,3}^3. \end{cases}$$

*Multiple images.* Solving the $n \geq 3$ view case is done in two steps. The first step is to compute the coefficients $c_j$, $j = 0, \ldots, 3n$-2 of a polynomial. The second step is to compute its real roots. Computing the coefficients in closed-form from equation (5), as is done above for the single- and the two-view cases, lead to very large, awkward formulas, which may lead to roundoff errors. We thus perform a numerical computation.

A standard root-finding technique is to compute the eigenvalues of the ((3$n$-2)×(3$n$-2)) companion matrix of the polynomial, see *e.g.* [1]. Computing all the roots ensures the optimal solution to be found. This can be done if the number of images is not too large, *i.e.* lower than 100, and if computation time is not an issue. However, for large numbers of images, or if real-time computation must be achieved, it is not possible to compute and try all roots. In that case, we propose to compute only the roots corresponding to feasible points.

Let $\lambda_0$ be an approximation of the sought-after root. For example, one can take the result of the algebraic method of §3, or even $\lambda_0 = 0$ since our parameterization takes the sought-after root very close to 0. Obviously, we could launch an iterative root-finding procedure such as Newton-Raphson from $\lambda_0$ but this would not guarantee that the optimal solution is found.

One solution to efficiently compute only the feasible roots is to reparameterize the polynomial such that those lie close to 0, and use an iterative algorithm for computing the eigenvalues of the companion matrix on turn. For example, Arnoldi or Lanczos' methods, compute the eigenvalues with increasing magnitude starting from the smallest one. Let $\lambda_c$ be the last computed eigenvalue, and $\mathbf{Q}_1$ and $\mathbf{Q}_2$ the reconstructed points corresponding to $\lambda_c$ and $-\lambda_c$. If both $\mathbf{Q}_1$ and $\mathbf{Q}_2$ reproject outside the images, the computation is stopped. Indeed, the next root that would be computed would have greater magnitude than $\lambda_c$, and would obviously lead to a point reprojecting further away than the previous one outside the images.

The reparameterization is done by computing a polynomial $\mathcal{P}_n(\lambda) = \tilde{\mathcal{D}}_n(\lambda + \lambda_0)$. A simple way to achieve this reparameterization is to estimate the coefficients $c_j$, $j = 1, \ldots, 3n\text{-}1$, of $\mathcal{P}_n$, as follows. We evaluate $z \geq 3n\text{-}1$ values $v_k = \tilde{\mathcal{D}}_n(\lambda_k + \lambda_0)$ from equation (5) for $\lambda_k \in [-\delta, \delta]$, and solve the associated Vandermonde system: $\sum_{j=0}^{3n-2} c_j \lambda_k^j = v_k \quad$ for $k = 1, \ldots, z$. We typically use $z = 10(3n\text{-}1)$. The parameter $\delta \in \mathbb{R}^{*+}$ reflects the size of the sampling interval around $\lambda_0$. We noticed that this parameter does not influence the results, and typically chose $\delta = 1$. Obviously, in theory, using $z = 3n\text{-}1$, *i.e.* the minimum number of samples, at distinct points, is equivalent for finding the coefficients. However we experimentally found that using extra samples evenly spread around the expected root $\lambda_0$ has the benefit of 'averaging' the roundoff error, and stabilizes the computation.

One could argue that with this method for estimating the coefficients, the simplifying transformations of §2.2 are not necessary. A short calculation shows that this is partly true since if the canonical 3D projective basis were not used along with the normalization of the third entries of $\mathbf{M}$ and $\mathbf{N}$ to unity, then the degree of the polynomial would be $3n$ instead of $3n$-2.

## 3 An Algebraic Criterion

We give a linear algorithm, dubbed 'ALGEBRAIC', based on approximating the reprojection error (2) by replacing the Euclidean distance measure $d_e$ by the algebraic distance measure $d_a$ defined by $d_a^2(\mathbf{x}, \mathbf{y}) = \mathsf{S}[\mathbf{x}]_\times \mathbf{y}$ with $\mathsf{S} = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{smallmatrix}\right)$, and where $[\mathbf{x}]_\times$ is the $(3 \times 3)$ skew-symmetric matrix associated to cross-product, *i.e.* $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$. This gives an algebraic error function:

$$\mathcal{E}_n^2(\lambda) = \sum_{i=1}^{n} d_a^2(\lambda \mathbf{b}_i + \mathbf{d}_i, \mathbf{q}_i) = \sum_{i=1}^{n} \|\lambda \mathsf{S}[\mathbf{q}_i]_\times \mathbf{b}_i + \mathsf{S}[\mathbf{q}_i]_\times \mathbf{d}_i\|^2.$$

A closed-form solution is obtained, giving $\lambda_a$ in the least squares sense:

$$\lambda_a = -\frac{\sum_{i=1}^{n} \mathbf{b}_i^\mathsf{T} [\mathbf{q}_i]_\times \tilde{\mathsf{I}} [\mathbf{q}_i]_\times \mathbf{d}_i}{\sum_{i=1}^{n} \mathbf{b}_i^\mathsf{T} [\mathbf{q}_i]_\times \tilde{\mathsf{I}} [\mathbf{q}_i]_\times \mathbf{b}_i} \qquad \text{with} \qquad \tilde{\mathsf{I}} \sim \mathsf{S}^\mathsf{T} \mathsf{S} \sim \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{smallmatrix}\right).$$

## 4 Gauss-Newton Refinement

As is usual for triangulation and bundle adjustment [7], we use the Gauss-Newton algorithm for refining an estimate of $\hat{\lambda}$ by minimizing the nonlinear least squares reprojection error (2). The algorithm, that we do not derived in details, is dubbed 'GAUSS-NEWTON'. We use the best solution amongst POLY and ALGEBRAIC as the initial solution.

## 5 Experimental Results

### 5.1 Simulated Data

We simulated a 3D line observed by $n$ cameras $\mathsf{P}_i$. In order to simulate realistic data, we reconstructed the 3D line as follows. We projected the line onto the images, and regularly sampled points on it, that were offset orthogonally to the image line with a Gaussian centred noise with variance $\sigma_l$. The 3D line was then reconstructed from the noisy points using the Maximum Likelihood triangulation method in [3], which provided $\mathbf{M}$ and $\mathbf{N}$. Finally, a point lying on the true 3D line was projected onto the images, and corrupted with a Gaussian centred noise with variance $\sigma_p$, which gave the $\mathbf{q}_i$. We varied some parameters of this setup, namely $n$ and $\sigma_p$, and the spatial configuration of the cameras, in order to compare the algorithms under different conditions. We compared two cases for the cameras: a stable one, in which they were evenly spread around the 3D line, and an unstable one, in which they were very close to each other. The default parameters of the setup are $\sigma_l = 0.1$ pixels, $\sigma_p = 3$ pixels, $n = 10$ views and stable cameras.

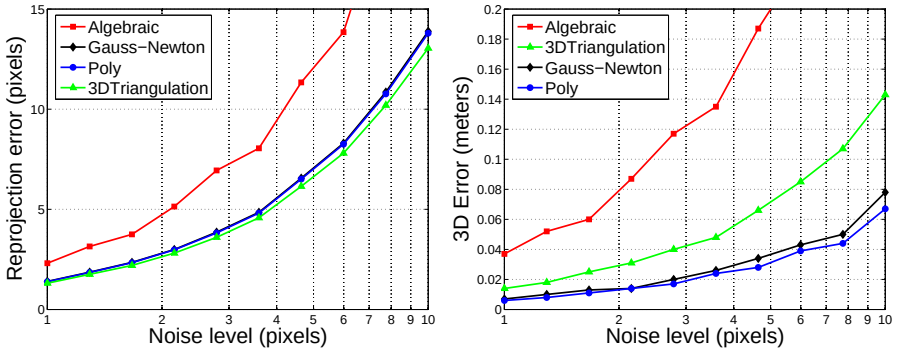



**Fig. 1.** Reprojection error (left) and 3D error (right) versus the level of noise.

We had two main goals in these experiments. First, we wanted to determine what in practice is the maximum number of views and noise that the proposed triangulation method can deal with, for stable and unstable camera configurations. Second, we wanted to determine to which extent the line constraint improves the accuracy of the reconstructed 3D point, compared to standard unconstrained triangulation. We measured two kinds of error: the reprojection error, quantifying the ability of the methods to fit the measurements, and a 3D error, quantifying the accuracy of the reconstruction.

We compared the three algorithms, described in the paper (POLY, §2 ; ALGEBRAIC, §3 ; GAUSS-NEWTON, §4) and 3DTRIANGULATION, which is a standard Maximum Likelihood triangulation, ignoring the line constraint, *e.g.* [7].

Figure 1 shows the results for varying noise level on the image points ($\sigma_p = 1, \ldots, 10$ pixels), and figure 2 for varying number of views ($n = 2, \ldots, 200$). Note the logarithmic scaling on the abscissa. General comments can be made about these results:

- 3DTriangulation always gives the lowest reprojection error.
- Algebraic always gives the highest reprojection error and 3D error.
- Poly and Gauss-Newton always give the lowest 3D error.

Small differences in the reprojection error may lead to large discrepancies in the 3D error. For example, Poly and Gauss-Newton are undistinguisable on figures 1 (left) and 2 (left), showing the reprojection error, while they can clearly be distinguished on figures 1 (right) and 2 (right), showing the 3D error. This is due to the fact that Gauss-Newton converges when some standard precision is reached on the reprojection error. Increasing the precision may improve the results, but would make convergence slower.

For $n = 10$ views, figure 1 shows that the accuracy of the 3D reconstruction is clearly better for the optimal methods Poly and Gauss-Newton using the line constraint, compared to 3DTriangulation that does not use this constraint. The difference in 3D accuracy is getting larger as the noise level increases. For a $\sigma_p = 1$ pixel noise, which is what one can expect in practice, the difference in accuracy is 1 cm, corresponding to 1% of the simulated scene scale. This is an important difference.

However, for $\sigma_p = 3$ pixels, beyond 20 views, figure 2 (left) shows that the reprojection error for 3DTriangulation and Poly/Gauss-Newton are hardly distinguishable, while we expect from figure 2 (right) the difference in 3D error to be negligible beyond 200 views.

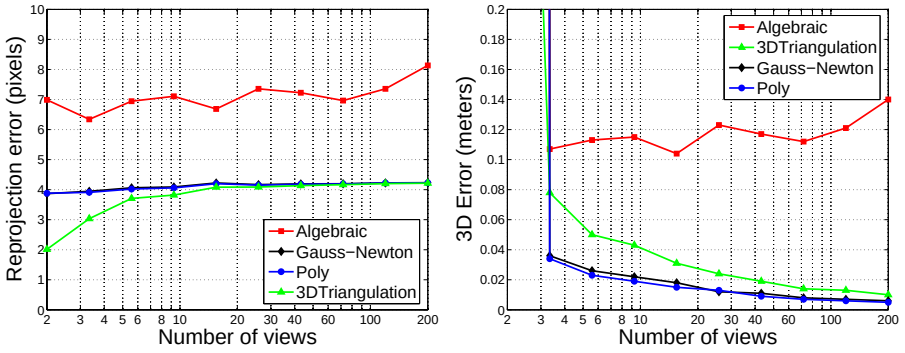



**Fig. 2.** Reprojection error (left) and 3D error (right) versus the number of views.

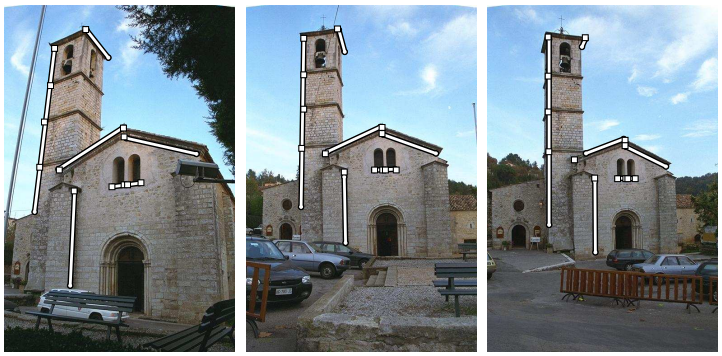
The results presented above concern the stable camera setup. For the unstable case, we obtained slightly lower reprojection errors, which is due to the fact that

the 3D model is less constrained, making the observations easier to "explain". However, as was expected, the 3D errors are higher by a factor of around 2. The order of the different methods remains the same as in the stable case. We noticed that incorporating the line constraint improves the accuracy compared to 3DTriangulation to a much higher extent than in the stable case.

### 5.2 Real Data

We tested the four reconstruction algorithms on several real data sets. For two of them, we show results. We used a Canny detector to retrieve salient edgels in the images, and adjusted segments using robust least squares. Finally, we matched the segments by hand between the images, except for the 387 frame 'building' sequence where automatic traking was used. The point on line correspondences were manually given, again besides for the 'building' sequence for which correlation based tracking was used. We reconstructed the 3D lines from the edgels by the Maximum Likelihood method in [3].

*The 'Valbonne church' sequence.* We used 6 views from the popular 'Valbonne church' image set. Some of them are shown on figure 3, together with the 6 input segments and 13 inputs points. The cameras were obtained by Euclidean bundle adjustment over a set of points [11]. The reprojection errors we obtained were: Algebraic → 1.37 pixels ; Poly → 0.77 pixels ; Gauss-Newton → 0.77 pixels. Figure 4 (a) shows lines and points reprojected from the 3D reconstruction. The



Frame 1 Frame 3 Frame 6

**Fig. 3.** 3 out of the 6 images taken from the 'Valbonne church' sequence, overlaid with 6 matching segments and 13 corresponding points.

reprojection errors we obtained for the points shown on figure 4 (b) were:

| Point | Algebraic | Poly | Gauss-Newton |
|---|---|---|---|
| 1 | 4.03 pixels | 2.14 pixels | 2.14 pixels |
| 2 | 6.97 pixels | 1.95 pixels | 1.95 pixels |
| 3 | 2.84 pixels | 2.21 pixels | 2.21 pixels |
| 4 | 4.65 pixels | 2.14 pixels | 2.14 pixels |

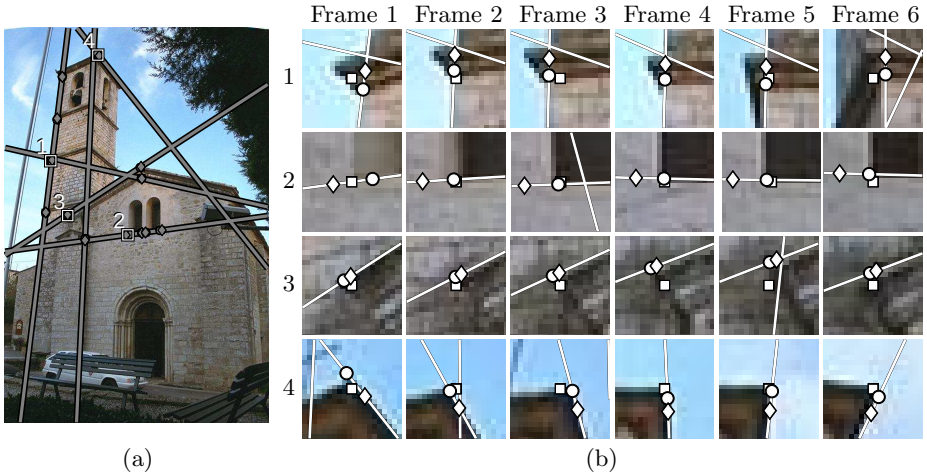



**Fig. 4.** Reprojected 3D lines and 3D points. (a) shows 4 different numbered points, for which (b) shows a close up for all the 6 images. The squares are the original points, the diamonds are the points reconstructed by Algebraic, and the circles are the points reconstructed from Poly and Gauss-Newton (they are undistinguishable).

*The 'Building' sequence.* This sequence is a continuous video stream consisting of 387 frames, showing a building imaged by a hand-held camera, see figure 5. We reconstructed calibrated cameras by bundle adjustment from interest points that were tracked using a correlation based tracker.

The segment we tracked is almost the only one that is visible throughout the sequence, and thus allows to test our triangulation methods for a very large number of views, namely 387. For the 7 points we selected, we obtained a mean reprojection error of 4.57 pixels for Algebraic, of 3.45 pixels for Poly and Gauss-Newton. Unconstrained triangulation gave a 2.90 pixels reprojection error. These errors which are higher than for the two previous data sets, are explained by the fact that there is non negligible radial distortion in the images, as can be seen on figure 5.


Frame 1 Frame 387

**Fig. 5.** 2 out of the 387 images of the 'building' sequence, overlaid with the matching segments and 7 corresponding points.

## 6 Conclusions

We proposed an algorithm for the optimal triangulation, in the Maximum Likelihood sense, of a point lying on a given 3D line. Several transformations of 3D space and in the images lead to a degree-($3n$-2) polynomial equation. An efficient algorithm computes the real roots leading to feasible points only. Experimental evaluation on simulated and real data show that the method can be applied to large numbers of images, up to 387 in our experiments. The experiments were done for many different real data sets, indoor and outdoor, small, medium and large number of images, calibrated and uncalibrated reconstructions. Comparison of triangulated points with ground truth for the case of simulated data show that using the line constraint greatly improves the accuracy of the reconstruction.

*Acknowledgements.* The first author thanks F. Schaffalitzky and A. Zisserman for having provided the projection matrices of the 'Valbonne church' sequence.

## References

1. F. S. Acton. *Numerical Methods That Work*. Washington: Mathematical Association of America, 1990. Corrected edition.
2. A. Bartoli, M. Coquerelle, and P. Sturm. A framework for pencil-of-points structure-from-motion. *European Conference on Computer Vision*, 2004.
3. A. Bartoli and P. Sturm. Multiple-view structure and motion from line correspondences. *International Conference on Computer Vision*, 2003.
4. O. Chum, T. Pajdla, and P. Sturm. The geometric error for homographies. *Computer Vision and Image Understanding*, 97(1):86–102, January 2005.
5. R. Hartley and F. Schaffalitzky. $L_\infty$ minimization in geometric reconstruction problems. *Conference on Computer Vision and Pattern Recognition*, 2004.
6. R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

7. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. Second Edition.
8. Q.T. Luong and T. Vieville. Canonic representations for the geometries of multiple projective views. *Computer Vision and Image Understanding*, 64(2):193–229, 1996.
9. D. Nistèr. *Automatic Dense Reconstruction From Uncalibrated Video Sequences*. PhD thesis, Royal Institute of Technology, KTH, March 2001.
10. J. Oliensis. Exact two-image structure from motion. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1618–1633, 2002.
11. F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets. In *Proceedings of the European Conference on Computer Vision*, 2002.
12. C. Schmid and A. Zisserman. The geometry and matching of lines and curves over multiple views. *International Journal of Computer Vision*, 40(3):199–234, 2000.
13. H. Stewénius, F. Schaffalitzky, and D. Nistér. How hard is 3-view triangulation really? In *Proceedings of the International Conference on Computer Vision*, 2005.