Triangulation Made Easy

Conference Paper *in* Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition · June 2010

DOI: 10.1109/CVPR.2010.5539785 · Source: DBLP

1 author:

Peter Lindstrom
Lawrence Livermore National Laboratory
97 PUBLICATIONS **5,183** CITATIONS

SEE PROFILE

Triangulation Made Easy

Peter Lindstrom
Lawrence Livermore National Laboratory
pl@llnl.gov

Abstract

We describe a simple and efficient algorithm for two-view triangulation of 3D points from approximate 2D matches based on minimizing the L_2 reprojection error. Our iterative algorithm improves on the one by Kanatani et al. *[5] by ensuring that in each iteration the epipolar constraint is satisfied. In the case where the two cameras are pointed in the same direction, the method provably converges to an optimal solution in exactly two iterations. For more general camera poses, two iterations are sufficient to achieve convergence to machine precision, which we exploit to devise a fast, non-iterative method. The resulting algorithm amounts to little more than solving a quadratic equation, and involves a fixed, small number of simple matrix-vector operations and no conditional branches. We demonstrate that the method computes solutions that agree to very high precision with those of Hartley and Sturm's original polynomial method [2], though achieves higher numerical stability and 1–4 orders of magnitude greater speed.*

1. Introduction

Triangulation is one of the most fundamental problems in computer vision. The problem can be stated as follows: Given a 3D point X projected to $x_i = P_i X$ in two or more cameras, recover the 3D position of X from its 2D projections. When X is consistent with the matched points x_i, this is a trivial linear problem. In practice, however, the measured and reprojected points do not exactly coincide, which causes the rays from the camera centers through the imaged points not to intersect in 3D. This may be due to uncertainties in relative camera poses or intrinsics (*i.e.* errors in P_i), or to the inherent difficulties in designing automated methods that perfectly match points to within subpixel accuracy across images (*i.e.* errors in x_i). In the presence of noise, the triangulation problem becomes one of finding the 3D point that best describes the observed image points.

Several criteria and error functionals for triangulation have been proposed in the literature [1, 3, 9, 10]. Under the assumption that the imaged points are perturbed by Gaussian noise, the optimal, *maximum likelihood* solution [2] minimizes the L_2 reprojection error

$$d(x, \hat{x})^2 + d(x', \hat{x}')^2 \tag{1}$$

where d is the Euclidean image-plane distance, x and x' are the observed points, and $\hat{x}$ and $\hat{x}'$ are corrected points that satisfy the epipolar constraint [7]

$$\hat{x}^\mathsf{T} F \hat{x}' = 0 \tag{2}$$

Here F denotes the rank-2 *fundamental matrix* defined for a pair of cameras. When satisfied, the epipolar constraint implies that the corresponding rays intersect at a point in 3D. A globally optimal solution to this non-convex constrained optimization problem is due to Hartley and Sturm [2] (often referred to as *optimal triangulation* or the *polynomial method*), and amounts to finding all roots of a degree-six polynomial. Though theoretically optimal, the task of reliably finding polynomial roots using finite precision arithmetic is nontrivial. Moreover, the relative difficulty of implementing this method and its computational cost can be considerable. Henceforth we will refer to this method as `hs`.

Iterative schemes are possible alternatives to Hartley and Sturm's "direct" method. The *bundle adjustment* approach [12] optimizes the position of the 3D point explicitly (and possibly the camera parameters as well), and thus ensures that the epipolar constraint is satisfied. Its main drawback is a reliance on a good initial guess of the *3D* position of the point, and even with reasonable estimates such a method often fails [8]. On the other hand, iteration over the positions of the *2D* points $\langle \hat{x}, \hat{x}' \rangle$ has the benefit of a good initializer $\langle x, x' \rangle$, and here the main concern becomes one of ensuring that the epipolar constraint is met upon convergence. Kanazawa and Kanatani [6] proposed such a method that converges quickly but generally to a solution that does not satisfy Eq. (2). Kanatani *et al.* [5] more recently presented an improved higher-order method that does satisfy the epipolar constraint, and which converges to a local extremum of the reprojection error. Like [6], Kanatani *et al.*'s method, which we will refer to as `ksn`, solves in each iteration a linear equation, but improves on [6] by incorporating information from the previous iteration. Aside from

no guarantees on global optimality, the chief drawback of this method is the number of iterations required—especially in unstable camera configurations—and the need to test for convergence. Moreover, early termination of this iterative scheme is not possible as the epipolar constraint is generally not satisfied until convergence.

In this paper we propose a new image-space iterative method for two-view triangulation that takes optimal steps $\langle \Delta x_k, \Delta x'_k \rangle$ in each iteration k by solving a quadratic rather than linear equation. In particular, each intermediate iterate $\langle x_k, x'_k \rangle$ is guaranteed to satisfy the epipolar constraint, and hence forms a valid (though possibly suboptimal) solution, allowing for early termination. We show that as Kanatani *et al*.'s higher-order method converges, its steps approach those computed by our method. We further prove that when the two cameras' principal axes are parallel (*i.e*. the cameras face in the same direction), the method converges in exactly two iterations. This is an important use case that occurs, for instance, in (zero toe in) stereo photography and structure from motion from a single forward-facing camera. In practice, convergence to very high precision (more than 10 digits) is achieved in at most two iterations for more general camera poses as well; a result that we use to simplify the second iteration for a fully non-iterative method. In addition to being very simple and efficient, we show using extensive experiments that the solutions delivered by our method are in excellent agreement with those of Hartley and Sturm's, but at only a tiny fraction of computational cost. In fact, due to the polynomial method's susceptibility to ill-conditioning, our method often yields lower reprojection errors in practice. Thus, although not theoretically optimal, our method is a practical and sometimes preferred alternative to the polynomial method.

We begin by presenting an iterative version of our algorithm and discuss its similarities with the one by Kanatani *et al*. We later prove that when the cameras point in the same direction two iterations suffice, and that the solution is indeed an extremum of the reprojection error. We then rewrite parts of the algorithm to make it more efficient, and also discuss how to make it numerically robust. We conclude with experimental results and a discussion of future work.

2. Preliminaries

For simplicity of presentation, we will assume that the cameras are calibrated and that the point matches $\langle x, x' \rangle$ are normalized, *i.e*. they have been premultiplied by the inverse camera calibration matrix K^{-1}. This allows us to work with *essential matrices* $E = [t]_\times R$, where R and t are the relative orientation and translation of the two cameras, and where $[\cdot]_\times$ represents the cross product operator (see also [3]). The uncalibrated case can be treated similarly, with F replacing E, as long as pixels are square; a prerequisite for the Euclidean reprojection error to be meaningful. We represent points in the image plane using homogeneous coordinates, *e.g*. $x = \begin{pmatrix} x_1 & x_2 & 1 \end{pmatrix}^\mathsf{T}$. Using this notation, the epipolar constraint becomes

$$x^\mathsf{T} E x' = 0 \tag{3}$$

Vectors in the image plane have only two components, and to combine 2-component vectors and homogeneous points we define the matrix

$$S = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{4}$$

Thus $S^\mathsf{T} \Delta x$ are the 3D coordinates of the 2D displacement Δx. We will make frequent use of

$$\tilde{E} = S E S^\mathsf{T} \tag{5}$$

which is the upper left 2×2 submatrix of E.

The *epipoles* $e = t$ and $e' = R^\mathsf{T} t$ are the projections of the camera centers c and c', and are the left and right null vectors of E, respectively. The line through e' and an image point x' in the second camera projects to an *epipolar line* $\ell = E x'$ in the first camera. Thus a point x on ℓ satisfies $\ell^\mathsf{T} x = 0$. Similarly, $\ell' = E^\mathsf{T} x$. We use $\langle x, x' \rangle$ to denote the measured points, and $\langle \hat{x}, \hat{x}' \rangle$ the corrected points.

3. Optimality conditions

Optimal triangulation, as formulated by Hartley and Sturm [2], can be expressed as the following minimization problem:

$$\begin{array}{ll} \text{minimize} & \Delta x^\mathsf{T} \Delta x + \Delta x'^\mathsf{T} \Delta x' \\ \text{subject to} & (x - S^\mathsf{T} \Delta x)^\mathsf{T} E (x' - S^\mathsf{T} \Delta x') = 0 \end{array} \tag{6}$$

with $\Delta x = S(x - \hat{x})$, $\Delta x' = S(x' - \hat{x}')$. Equation (6) is a quadratically constrained quadratic optimization problem. We introduce a Lagrange multiplier -2λ for the scalar constraint, resulting in the Lagrangian

$$\begin{aligned} f(\Delta x, \Delta x', \lambda) = {} & \Delta x^\mathsf{T} \Delta x + \Delta x'^\mathsf{T} \Delta x' \\ & - 2\lambda (x - S^\mathsf{T} \Delta x)^\mathsf{T} E (x' - S^\mathsf{T} \Delta x') \end{aligned} \tag{7}$$

Setting the gradient of f to zero, we obtain the following quadratic equations:

$$\hat{x}^\mathsf{T} E \hat{x}' = (x - S^\mathsf{T} \Delta x)^\mathsf{T} E (x' - S^\mathsf{T} \Delta x') = 0 \tag{8}$$

$$\Delta x = \lambda S E (x' - S^\mathsf{T} \Delta x') = \lambda S E \hat{x}' = \lambda n \tag{9}$$

$$\Delta x' = \lambda S E^\mathsf{T} (x - S^\mathsf{T} \Delta x) = \lambda S E^\mathsf{T} \hat{x} = \lambda n' \tag{10}$$

These are one scalar and two vector equations for a total of five constraints involving five unknowns (Δx, $\Delta x'$, and λ).

Equation (8) implies that $\hat{x}$ and $\hat{x}'$ must lie on corresponding epipolar lines $\ell = E\hat{x}'$ and $\ell' = E^{\mathsf{T}}\hat{x}$. Equations (9) and (10), on the other hand, constrain $\hat{x}$ and $\hat{x}'$ to lie on lines orthogonal to ℓ and ℓ' that pass through x and x', respectively (see Fig. 1). In other words, Δx and $\Delta x'$ are parallel to the normal vectors n and n' of the epipolar lines.

The geometric consequences of these line intersection constraints are well known: $\angle x\hat{x}e$ must be a right angle, and hence the optimum in the first camera must lie somewhere on the smallest circle in the image plane that contains x and the epipole e (and similarly for the second camera). The approach taken by Hartley and Sturm is to parameterize this circle in the first camera via a stereographic projection from the epipole e onto the line $\begin{pmatrix}0 & 0 & 1\end{pmatrix}^{\mathsf{T}} \times (e - x)$ (the vertical line through $x = x_0$ in Fig. 1). Each point on this line can then be expressed using a single parameter t, and can later be mapped back to the corresponding point on the circle.

In addition to the directional constraints of Eqs. (9) and (10), these equations impose a magnitude constraint on an optimal solution, *i.e.* that the Lagrange multiplier λ take on the same value in both equations. We can thus express Eqs. (8) to (10) intuitively as a set of conditions that any optimal solution $\langle \hat{x}, \hat{x}' \rangle$ must satisfy:

(i) $\hat{x}$ and $\hat{x}'$ lie on corresponding epipolar lines.

(ii) $\hat{x}$ and $\hat{x}'$ are the projections of x and x' onto these epipolar lines.

(iii) The corrections Δx and $\Delta x'$ are linearly related by a single parameter λ.

Note that these three requirements are necessary for a pair $\langle \hat{x}, \hat{x}' \rangle$ to satisfy the epipolar constraint and be an *extremum* of the reprojection functional f. They are, in general, not sufficient for global optimality. As Hartley and Sturm point out, as many as three minima and three maxima may exist. (Since f is parameterized over a circle, it is periodic, and hence there are as many minima as there are maxima.) In practice, however, f has with very high probability ($> 99.6\%$ in our experiments) a single minimum, and hence our primary goal is to satisfy the three constraints above.

4. Iterative algorithm

Our iterative approach to satisfying the optimality constraints is to linearize Eqs. (9) and (10) by replacing on the right hand side the optimal $\langle \hat{x}, \hat{x}' \rangle$ points with the current best estimate $\langle x_k, x'_k \rangle$. The measured points $x = x_0$ and $x' = x'_0$ serve as initial estimates. Substituting the current estimates Δx_k and $\Delta x'_k$ into Eq. (8) results in a single quadratic equation in the unknown λ_k:

$$\lambda_k^2 n_k^{\mathsf{T}} \tilde{E} n'_k - \lambda_k (n_1^{\mathsf{T}} n_k + {n'_1}^{\mathsf{T}} n'_k) + x_0^{\mathsf{T}} E x'_0 = 0 \quad (11)$$

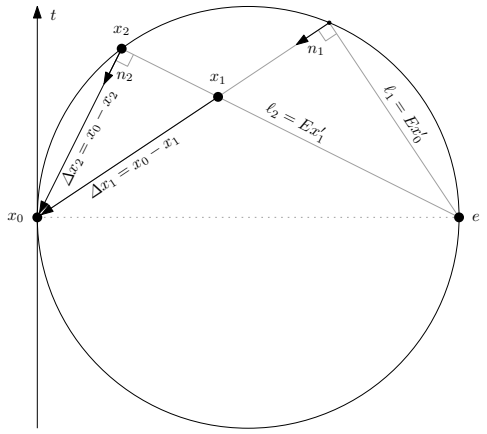

Figure 1: Iterative refinement of x. The optimum, x_2, is the orthogonal projection of the measured point x_0 onto the epipolar line $e = Ex'_1 = Ex'_2$, and lies on the smallest circle through x_0 and the epipole e.

with $n_k = SEx'_k$, $n'_k = SE^{\mathsf{T}}x_k$. Of the two possible roots, we choose the smaller one (in magnitude) as λ governs how far to step from the measured points. Given this value of λ we update the displacements Δx and $\Delta x'$ using Eqs. (9) and (10), and consequently the positions $\langle x, x' \rangle$, and continue with the next iteration. The resulting algorithm is presented in Listing 2. The issue of convergence is not addressed here, and will be discussed in more detail below.

Note that by solving Eq. (8), we explicitly enforce the epipolar constraint in each iteration. Moreover, the displacements Δx and $\Delta x'$ are based on a single step size, λ, and hence two of the three optimality conditions are met. This is in contrast to the method by Kanatani *et al.* [5], which enforces the epipolar constraint only upon convergence. In fact, the only difference between our iterative method and Kanatani's is the step size λ. The two methods employ the same step *directions* n_k and n'_k (as does the original method [6]), *i.e.* from the measured points $\langle x_0, x'_0 \rangle$ in a direction orthogonal to the current epipolar lines associated with $\langle x_k, x'_k \rangle$. These directions are in some sense the best possible choice, as upon convergence they point to the optimum. By also satisfying the epipolar constraint, we additionally take the best possible length step.

Whereas our step size λ_k is the root of a quadratic equation, one can show that the λ_k computed by Kanatani's method is a solution to

$$\lambda_k^2 n_k^{\mathsf{T}} \tilde{E} n'_k - \lambda_k (n_1^{\mathsf{T}} n_k + {n'_1}^{\mathsf{T}} n'_k) + x_0^{\mathsf{T}} E x'_0 = (\Delta x_k - \Delta x_{k-1})^{\mathsf{T}} \tilde{E} (\Delta x'_k - \Delta x'_{k-1}) \quad (12)$$

Since $\Delta x_k = \lambda_k n_k$ and $\Delta x'_k = \lambda_k n'_k$, the quadratic term λ_k^2 cancels, resulting in a linear equation in λ_k (see Listing 1). As Kanatani's method converges, the right-hand-side of Eq. (12) approaches zero, and hence the λ_k com-

`ksn` (x_0, x'_0, E)

1. $\Delta x_0 \leftarrow 0$
2. $\Delta x'_0 \leftarrow 0$
3. **for** $k = 1, \ldots$
4. $\quad n_k \leftarrow S E x'_{k-1}$
5. $\quad n'_k \leftarrow S E^\mathsf{T} x_{k-1}$
6. $\quad \lambda_k \leftarrow \frac{x_0^\mathsf{T} E x'_0 - \Delta x_{k-1}^\mathsf{T} \tilde{E} \Delta x'_{k-1}}{n_k^\mathsf{T} n_k + {n'_k}^\mathsf{T} n'_k}$
7. $\quad \Delta x_k \leftarrow \lambda_k n_k$
8. $\quad \Delta x'_k \leftarrow \lambda_k n'_k$
9. $\quad x_k \leftarrow x_0 - S^\mathsf{T} \Delta x_k$
10. $\quad x'_k \leftarrow x'_0 - S^\mathsf{T} \Delta x'_k$

Listing 1: The higher-order method by Kanatani *et al.* [5]. Differences with respect to Listing 2 are highlighted.

puted by Kanatani's method approaches the quadratic solution obtained directly by our method (*c.f.* Eq. (11)).

We remark that our implementation uses a careful choice in the computation of λ. Since we are interested only in the smaller of the two roots, we compute $\lambda = \frac{c}{b + \mathrm{sgn}(b) d}$ instead of the equivalent $\lambda = \frac{b - \mathrm{sgn}(b) d}{a}$ to guarantee effective addition instead of subtraction. This avoids the potential for catastrophic cancellation (*c.f.* [11, §5.6]).

When the error functional has only one minimum, as we found to be the case in more than 99.6% of our experiments, our iterative method quickly converges to that minimum. Choosing the larger of the two roots for λ in each iteration takes us instead to the single maximum.

Once the corrected points $\langle \hat{x}, \hat{x}' \rangle$ have been computed, any method can be used to infer the 3D point X, as the two rays must intersect. When pose and intrinsics are known, the method outlined in [6] may be used:

$$z = \hat{x} \times R \hat{x}' \qquad X = \frac{z^\mathsf{T} E \hat{x}'}{z^\mathsf{T} z} \hat{x} \tag{13}$$

4.1. Convergence criterion

To assess convergence, it is important to ask what we wish to converge to. A "small enough" change in $\langle x_k, x'_k \rangle$, though normally an indicator of convergence, does not imply that the optimality conditions are met. Since in our method and as well [5, 6] condition (iii)—single λ—is guaranteed, only the remaining two need examination. That is, the solution $\langle \hat{x}, \hat{x}' \rangle$ must lie at the intersection of corresponding epipolar lines and on the orthogonal lines through $\langle x, x' \rangle$. In other words, $\langle \hat{x}, \hat{x}' \rangle$ must be the projection of $\langle x, x' \rangle$ onto the epipolar lines defined by $\langle \hat{x}, \hat{x}' \rangle$. As a general convergence criterion, we measure the distance of $\langle \hat{x}, \hat{x}' \rangle$ to this intersection, which accounts for both epipolar constraint violation (by methods like `ksn`, for instance) and optimality along the epipolar lines.

`iter` (x_0, x'_0, E)

1. **for** $k = 1, \ldots$
2. $\quad n_k \leftarrow S E x'_{k-1}$
3. $\quad n'_k \leftarrow S E^\mathsf{T} x_{k-1}$
4. $\quad a_k \leftarrow n_k^\mathsf{T} \tilde{E} n'_k$
5. $\quad b_k \leftarrow \frac{1}{2}(n_1^\mathsf{T} n_k + {n'_1}^\mathsf{T} n'_k)$
6. $\quad c_k \leftarrow x_0^\mathsf{T} E x'_0$
7. $\quad d_k \leftarrow \sqrt{b_k^2 - a_k c_k}$
8. $\quad \lambda_k \leftarrow \frac{c_k}{b_k + \mathrm{sgn}(b_k) d_k}$
9. $\quad \Delta x_k \leftarrow \lambda_k n_k$
10. $\quad \Delta x'_k \leftarrow \lambda_k n'_k$
11. $\quad x_k \leftarrow x_0 - S^\mathsf{T} \Delta x_k$
12. $\quad x'_k \leftarrow x'_0 - S^\mathsf{T} \Delta x'_k$

Listing 2: Our iterative method.

5. Non-iterative algorithm

The advantage of our method over Kanatani's, aside from faster convergence, is that any intermediate iterate satisfies the epipolar constraint and therefore constitutes a valid (albeit possibly suboptimal) solution, thus allowing for execution of only a small, fixed number of iterations. In fact, a remarkable result is that our iterative algorithm can be shown to attain a minimum in exactly two iterations when the cameras face in the same direction, regardless of translation and the remaining rotational degree of freedom (see Appendix A). In this case the first iteration finds the correct epipolar lines, and the second iteration simply moves the points along these lines to the projections of $\langle x, x' \rangle$. For more general camera poses, we observed that our method, with extremely high likelihood, converges in two iterations to 12 digits of precision or more. This suggests the possibility of running only the first iteration to identify the epipolar lines, followed by a projection step.

In the general setting, this projection step may violate condition (iii), since then the resulting step sizes in the two views may require different values $\frac{\Delta x_1^\mathsf{T} n_2}{n_2^\mathsf{T} n_2}$ and $\frac{\Delta {x'_1}^\mathsf{T} n'_2}{{n'_2}^\mathsf{T} n'_2}$ for λ. An alternative solution is to fix λ by computing

$$\lambda_2 = \frac{\Delta x_1^\mathsf{T} n_2 + \Delta {x'_1}^\mathsf{T} n'_2}{n_2^\mathsf{T} n_2 + {n'_2}^\mathsf{T} n'_2} = \lambda_1 \frac{2 d_1}{n_2^\mathsf{T} n_2 + {n'_2}^\mathsf{T} n'_2} \tag{14}$$

This is equivalent to applying one step of `iter` followed by one step of `ksn`. In the general case this linearization of λ_2 comes at the expense of violating the epipolar constraint (i). As we shall see, the discrepancy is usually on the order of 10^{-15}, and is for all practical intents inconsequential.

Our non-iterative algorithm, implemented both ways, is presented in Listing 3. A C++ implementation of the faster `niter2` version compiles to 36 additions/subtractions, 49 multiplications, 2 divisions, and 1 square root, for a total of 88 floating-point scalar arithmetic operations.

`niter1` (x, x', E)

1. $n \leftarrow SEx'$
2. $n' \leftarrow SE^\mathsf{T}x$
3. $a \leftarrow n^\mathsf{T}\tilde{E}n'$
4. $b \leftarrow \frac{1}{2}(n^\mathsf{T}n + n'^\mathsf{T}n')$
5. $c \leftarrow x^\mathsf{T}Ex'$
6. $d \leftarrow \sqrt{b^2 - ac}$
7. $\lambda \leftarrow \frac{c}{b+d}$
8. $\Delta x \leftarrow \lambda n$
9. $\Delta x' \leftarrow \lambda n'$
10. $n \leftarrow n - \tilde{E}\Delta x'$
11. $n' \leftarrow n' - \tilde{E}^\mathsf{T}\Delta x$
12. $\Delta x \leftarrow \frac{\Delta x^\mathsf{T} n}{n^\mathsf{T} n} n$
13. $\Delta x' \leftarrow \frac{\Delta x'^\mathsf{T} n'}{n'^\mathsf{T} n'} n'$
14. $x \leftarrow x - S^\mathsf{T}\Delta x$
15. $x' \leftarrow x' - S^\mathsf{T}\Delta x'$

`niter2` (x, x', E)

1. $n \leftarrow SEx'$
2. $n' \leftarrow SE^\mathsf{T}x$
3. $a \leftarrow n^\mathsf{T}\tilde{E}n'$
4. $b \leftarrow \frac{1}{2}(n^\mathsf{T}n + n'^\mathsf{T}n')$
5. $c \leftarrow x^\mathsf{T}Ex'$
6. $d \leftarrow \sqrt{b^2 - ac}$
7. $\lambda \leftarrow \frac{c}{b+d}$
8. $\Delta x \leftarrow \lambda n$
9. $\Delta x' \leftarrow \lambda n'$
10. $n \leftarrow n - \tilde{E}\Delta x'$
11. $n' \leftarrow n' - \tilde{E}^\mathsf{T}\Delta x$
12. $\lambda \leftarrow \lambda \frac{2d}{n^\mathsf{T} n + n'^\mathsf{T} n'}$
13. $\Delta x \leftarrow \lambda n$
14. $\Delta x' \leftarrow \lambda n'$
15. $x \leftarrow x - S^\mathsf{T}\Delta x$
16. $x' \leftarrow x' - S^\mathsf{T}\Delta x'$

Listing 3: Our two non-iterative methods with differences highlighted. `niter1` guarantees conditions (i) and (ii), whereas `niter2` guarantees condition (iii).

6. Results

We evaluated our method on both synthetic and real data and made comparisons with our C++ implementations of Hartley and Sturm's method [2] (`hs`) and the one by Kanatani *et al.* [5] (`ksn`). Our data sets exhibit a wide variety of more than 100,000 relative camera poses. Statistics on these data sets are reported in Table 1.

6.1. Synthetic data

We begin by examining how our method performs on the synthetic data. Each of these data sets is comprised of 10×10 random point clouds of 10,000 points each, projected onto images of size $1,024^2$ pixels, with a focal length of 512 pixels. These point clouds have a Gaussian radial distribution with a standard deviation of $\frac{1}{4}$ relative to the baseline. Each member of this 10×10 ensemble corresponds to a certain image-space Gaussian noise level $\sigma = 2^n$ in pixels, with $n = -5, -4, \ldots, +4$, and distance from center of baseline $\delta = 2^d$ for $d = -1, 0, \ldots, +8$. In "orbital" the cameras point at the point cloud centroid; in "lateral" the cameras point in the same direction and are translated laterally (a stable configuration simulating a stereo camera); while in "forward" the cameras also point in the same direction but one is translated directly forward (an unstable configuration simulating a forward-facing camera attached to a vehicle or robot).

Table 1 lists the number of real polynomial roots found by `hs`. For root finding we first implemented the popular eigenvalue method [11]. In spite of the polynomial being of low degree, we found this method unreliable due to poor conditioning. With relative coefficient magnitudes often spanning more than the 16 digits of precision available in a double, we observed catastrophic loss of precision in the eigensolver. Instead we used Bond's implementation of the well-known Jenkins-Traub root finding method [4] from `http://www.crbond.com/download/misc/rpoly.cpp`. In addition to complexity of implementation, the need for a sophisticated root finding technique in `hs` are two arguments in favor of our much simpler method.

	Orbital	Lateral	Forward	Corridor	Dinosaur	Notre Dame
Cameras	20	2	2	11	36	715
Poses	10	1	1	55	322	118,467
Points	10^6	10^6	10^6	737	4,983	127,431
Matches	10^6	10^6	10^6	12,003	25,841	4,579,786
1 root		10^6				
2 roots	10^6		10^6	12,003	25,841	4,553,033
4 roots						26,753
Agreement	8	16	8	6	7	3

Table 1: Data sets used. "Poses" denotes the number of camera pairs in which at least one point is visible. A point may partake in multiple pairwise "matches." "Roots" refers to the number found in the C++ implementation of `hs`. "Agreement" is the minimum number of digits to which the `hs` and `niter1` reprojection errors agree.

Returning to Table 1, with the epipoles at infinity in the "lateral" case the polynomial is reduced to linear, and hence only one root exists. The other two synthetic data sets result in exactly two roots in all two million cases. This table also lists the degree of precision at which the resulting reprojection error agreed between `hs` and our non-iterative method `niter1`. This agreement is always at least three digits, and in over 99.99% of all cases six digits or more. For each of these three data sets, our non-iterative method converged to at least twelve digits of precision in two iterations or less.

Histograms of relative reprojection errors between `hs` and our iterative and non-iterative methods are shown in Fig. 3. We compute the relative error as $\Delta E = \frac{E - E'}{\min\{E, E'\}}$, where E and E' are the reprojection errors found by one of our methods and `hs`, respectively. Thus $\Delta E = \pm 10^{-15}$ means that our method agreed with `hs` to 15 digits of precision. The histogram bins span $\pm[10^i, 10^{i+1})$; blue columns show cases where our method found a better solution, green indicates the range $(-10^{-17}, +10^{-17})$ around machine epsilon, while red columns indicate that `hs` did better.

Evidently our method compares well with `hs` in practice—in spite of `hs` being globally optimal in theory—with relative errors more or less exhibiting a normal distribution around zero, and with our three methods producing qualitatively similar histograms. In the worst case, `niter1` exceeded the reprojection error reported by `hs` by one part in 10^8, while ensuring that the epipolar constraint was met.

6.2. Real data

We also evaluated our method on the multi-view data sets "dinosaur" and "corridor" from `http://www.robots.ox.ac.uk/~vgg/data/data-mview.html`, and "Notre Dame" from `http://phototour.cs.washington.edu/datasets/`. Exactly two roots were found for all point matches in the first two data sets, while `hs` sometimes reported four roots for the third. Indeed, this is the only data set for which we found more than two roots (but not more than four). We separated out these cases to test whether our method would get stuck in a local minimum. The dark columns in the bottom row of Fig. 3 suggest no particular problems for any of our methods in these situations. We also measured the square distance of the `niter2` solutions to the epipolar lines, and found on both the synthetic and real data this distance to be at most 10^{-9} (in normalized coordinates); in 99.999% of all cases this discrepancy was less than 10^{-15}.

6.3. Convergence

To demonstrate the superiority in convergence of our method over `ksn`, we counted the number of iterations required to converge to 12 digits of precision for the "forward" unstable camera pose, where the imaged points are close to the epipoles. Our method is guaranteed to converge in (at most) two iterations regardless of noise and scene distance, as was also verified experimentally. Fig. 2 plots the average number of iterations required by `ksn` for different levels of noise and distance. As expected, the number of iterations increases both with noise level and distance. In the worst case, `ksn` required 11 iterations to converge.

6.4. Timings

Using the synthetic data we evaluated the speed of our `niter2` method relative to `hs` and `ksn` on a dual 3.2 GHz Intel Xeon PC. In Table 2 we also include the results of two simple methods that do not minimize reprojection error: the midpoint (`mid`) and linear homogeneous method (`dlt`); see [3]. We note that the timings for both of these methods, which compute 3D points directly, include the time to project the solutions back to 2D. Nevertheless, even when measuring 3D extraction, our method is more than twice as fast as `mid`, which might be considered the simplest triangulation method known. We also note that on the "orbital" and "forward" data sets our method is 50 times faster than Hartley and Sturm's, and about 20 times faster than Kanatani's own publically available C++ implementation of `ksn`. In case of the "lateral" data set, the Jenkins-Traub root finder used by `hs` spent considerably more time isolating the single root, and here our method was 8,000 times faster.

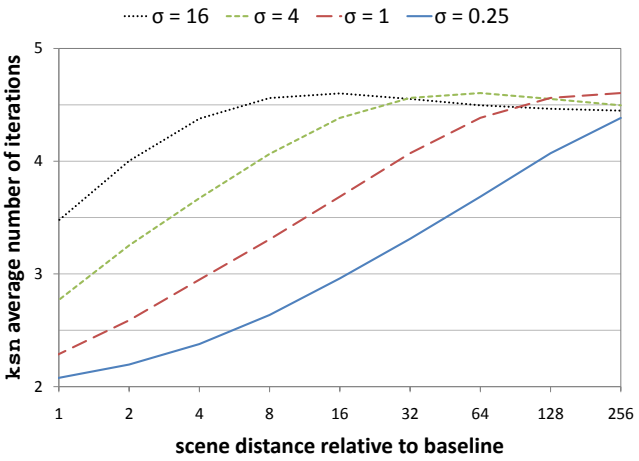

Figure 2: Average number of iterations required by `ksn` to converge as a function of scene distance and noise level σ.

	`hs`	`dlt`	`ksn`	`mid`	`niter2`
Points/sec	130 K	140 K	1.7 M	1.7 M	6.5 M
Speedup	1.0	1.1	13	13	50

Table 2: Speed of computing optimal image points $\langle \hat{x}, \hat{x}' \rangle$.

7. Conclusions

We presented a fast iterative method for uncalibrated two-view triangulation that takes optimal step sizes so as to enforce the epipolar constraint in each iteration. In practice, the optimal epipolar lines are found in the first iteration, allowing the second iteration to be cast as a simple projection step that renders the method non-iterative. The problem case of unstable camera configurations—where triangulation methods exhibit their greatest variation in quality and speed—is chiefly where our method excels over the one by Kanatani *et al*. [5]. Our simple method is furthermore non-iterative and several times faster than theirs. In spite of no theoretical optimality guarantees in the general setting, we showed our method to generally be more reliable and accurate in practice than Hartley and Sturm's, in addition to being fifty times faster.

Future work will investigate the convergence properties of our new method in order to gain an understanding of whether it may fail, and why the method converges to such great precision in only two iterations. The problem treated here is limited to two views. We envision possible extensions to three-view triangulation, which likely will involve the solution of cubic equations.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

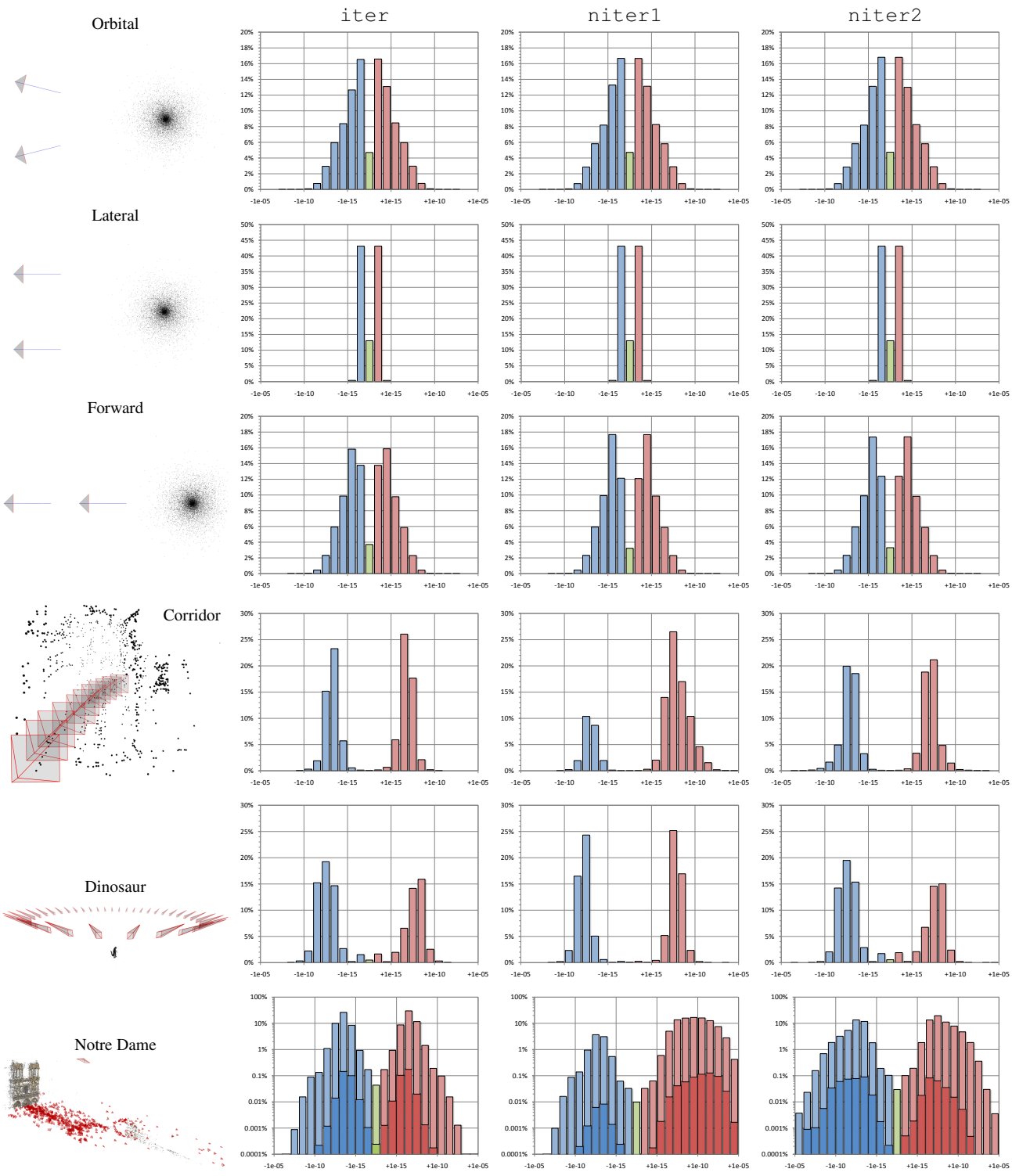

Figure 3: Histograms of reprojection error relative to Hartley and Sturm's method for our iterative and two non-iterative methods. Due to numerical issues, our method often finds better solutions (blue columns). Note that the vertical axis for Notre Dame is logarithmic to highlight (using darker columns) the cases where `hs` found four roots and where multiple minima may exist.

A. Proof of optimality

We here prove that our method is optimal when the two cameras point in the same direction (that is, when their principal axes are parallel). Let R and t be the relative orientation and translation of the two cameras, and let E be the associated essential matrix, with

$$R = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} \tag{15}$$

$$t = \begin{pmatrix} x & y & z \end{pmatrix}^\mathsf{T} \tag{16}$$

$$\tilde{E} = SES^\mathsf{T} = \begin{pmatrix} 0 & -z \\ z & 0 \end{pmatrix} Q \tag{17}$$

for any orthogonal 2×2 matrix Q and unit vector t. Thus, the cameras both point in the direction $\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\mathsf{T}$. One may easily verify that

$$\tilde{E}^\mathsf{T}\tilde{E} = \tilde{E}\tilde{E}^\mathsf{T} = z^2 I \tag{18}$$

$$ES^\mathsf{T}SE^\mathsf{T}S^\mathsf{T}SE = z^2 E \tag{19}$$

We have in the first iteration

$$\begin{aligned} a_1 &= n_1^\mathsf{T}\tilde{E}n_1' = {x_0'}^\mathsf{T}E^\mathsf{T}S^\mathsf{T}SES^\mathsf{T}SE^\mathsf{T}x_0 \\ &= x_0^\mathsf{T}ES^\mathsf{T}SE^\mathsf{T}S^\mathsf{T}SEx_0' = z^2 x_0^\mathsf{T}Ex_0' \\ &= z^2 c_1 \end{aligned} \tag{20}$$

$$a_1\lambda_1 = 2b_1 - \frac{c_1}{\lambda_1} = 2b_1 - (b_1 + d_1) = b_1 - d_1 \tag{21}$$

In the second iteration, $c_2 = c_1$ and

$$\begin{aligned} a_2 &= n_2^\mathsf{T}\tilde{E}n_2' = (n_1 - \lambda_1\tilde{E}n_1')^\mathsf{T}\tilde{E}(n_1' - \lambda_1\tilde{E}^\mathsf{T}n_1) \\ &= n_1^\mathsf{T}\tilde{E}n_1' - \lambda_1 n_1^\mathsf{T}\tilde{E}\tilde{E}^\mathsf{T}n_1 - \lambda_1 {n_1'}^\mathsf{T}\tilde{E}^\mathsf{T}\tilde{E}n_1' \\ &\quad + \lambda_1^2 {n_1'}^\mathsf{T}\tilde{E}^\mathsf{T}\tilde{E}\tilde{E}^\mathsf{T}n_1 \\ &= a_1 - 2z^2 b_1\lambda_1 + z^2 a_1\lambda_1^2 \\ &= a_1 + z^2(a_1\lambda_1^2 - 2b_1\lambda_1) = a_1 - z^2 c_1 \\ &= 0 \end{aligned} \tag{22}$$

$$\begin{aligned} b_2 &= \tfrac{1}{2}(n_1^\mathsf{T}n_2 + {n_1'}^\mathsf{T}n_2') \\ &= \tfrac{1}{2}(n_1^\mathsf{T}(n_1 - \lambda_1\tilde{E}n_1') + {n_1'}^\mathsf{T}(n_1' - \lambda_1\tilde{E}^\mathsf{T}n_1)) \\ &= \tfrac{1}{2}(n_1^\mathsf{T}n_1 - a_1\lambda_1 + {n_1'}^\mathsf{T}n_1' - a_1\lambda_1) \\ &= b_1 - a_1\lambda_1 = b_1 - (b_1 - d_1) \\ &= d_1 \end{aligned} \tag{23}$$

From Listing 2 and $a_2 = 0$, we obtain a linear expression

$$\lambda_2 = \frac{c_2}{2b_2} = \frac{c_1}{2d_1} \tag{24}$$

For the method to converge in the second iteration, we need Δx_2 to be the projection of Δx_1 onto n_2 (see Fig. 1), and similarly in the second camera. In other words, we need to show that

$$\lambda_2 = \frac{\Delta x_1^\mathsf{T}n_2}{n_2^\mathsf{T}n_2} = \frac{{\Delta x_1'}^\mathsf{T}n_2'}{{n_2'}^\mathsf{T}n_2'} = \frac{\Delta x_1^\mathsf{T}n_2 + {\Delta x_1'}^\mathsf{T}n_2'}{n_2^\mathsf{T}n_2 + {n_2'}^\mathsf{T}n_2'} \tag{25}$$

We limit the proof to the first camera, as the same holds in the second camera due to symmetry:

$$\begin{aligned} \frac{\Delta x_1^\mathsf{T}n_2}{n_2^\mathsf{T}n_2} &= \lambda_1\frac{n_1^\mathsf{T}n_2}{n_2^\mathsf{T}n_2} = \frac{\lambda_1(n_1^\mathsf{T}n_1 - a_1\lambda_1)}{n_1^\mathsf{T}n_1 - 2a_1\lambda_1 + z^2\lambda_1^2{n_1'}^\mathsf{T}n_1'} \\ &= \frac{\lambda_1(n_1^\mathsf{T}n_1 - b_1 + d_1)}{n_1^\mathsf{T}n_1 - 2(b_1 - d_1) + \frac{b_1 - d_1}{b_1 + d_1}{n_1'}^\mathsf{T}n'} \\ &= \frac{\lambda_1(b_1+d_1)(n_1^\mathsf{T}n_1 - b_1 + d_1)}{(b_1+d_1)(n_1^\mathsf{T}n_1 - 2(b_1-d_1)) + (b_1-d_1)(2b_1 - n_1^\mathsf{T}n_1)} \\ &= \frac{c_1(n_1^\mathsf{T}n_1 - b_1 + d_1)}{2d_1 n_1^\mathsf{T}n_1 - 2b_1 d_1 + 2d_1^2} \\ &= \frac{c_1(n_1^\mathsf{T}n_1 - b_1 + d_1)}{2d_1(n_1^\mathsf{T}n_1 - b_1 + d_1)} \\ &= \frac{c_1}{2d_1} = \frac{c_2}{2b_2} = \lambda_2 \end{aligned} \tag{26}$$

References

[1] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Computer Vision and Pattern Recognition*, volume 1, pages 504–509, 2004.

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

[3] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2nd edition, 2004.

[4] M. A. Jenkins. Algorithm 493: Zeros of a real polynomial [C2]. *ACM Transactions on Mathematical Software*, 1(2):178–189, 1975.

[5] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-Sturm vs. optimal correction. In *British Machine Vision Conference*, page 55, 2008.

[6] Y. Kanazawa and K. Kanatani. Reliability of 3-D reconstruction by stereo vision. *IEICE Transactions on Information and Systems*, E78-D(10):1301–1306, 1995.

[7] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[8] F. Lu and R. Hartley. A fast optimal algorithm for L_2 triangulation. In *Asian Conference on Computer Vision*, pages 279–288, 2007.

[9] K. Nordberg. The triangulation tensor. *Computer Vision and Image Understanding*, 113(9):935–945, 2009.

[10] J. Oliensis. Exact two-image structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1618–1633, 2002.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C*. Cambridge University Press, 2nd edition, 1992.

[12] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–372, 1999.