Title

Statistical Angular Error-Based Triangulation for Efficient and Accurate Multi-View Scene Reconstruction

Authors

Recker, Shawn
Hess-Flores, Mauricio
Joy, Kenneth I.

Publication Date

2013

Peer reviewed

Statistical Angular Error-Based Triangulation for Efficient and Accurate Multi-View Scene Reconstruction

Shawn Recker[1], Mauricio Hess-Flores[2], Kenneth I. Joy[3]
University of California, Davis
[1]strecker@ucdavis.edu, [2]mhessf@ucdavis.edu, [3]kenneth.i.joy@gmail.com

Abstract

This paper presents a framework for N-view triangulation of scene points, which improves processing time and final reprojection error with respect to standard methods, such as linear triangulation. The framework introduces an angular error-based cost function, which is robust to outliers and inexpensive to compute, and designed such that simple adaptive gradient descent can be applied for convergence. Our method also presents a statistical sampling component based on confidence levels, that reduces the number of rays to be used for triangulation of a given feature track. It is shown how the statistical component yields a meaningful yet much reduced set of representative rays for triangulation, and how the application of the cost function on the reduced sample can efficiently yield faster and more accurate solutions. Results are demonstrated on real and synthetic data, where it is proven to significantly increase the speed of triangulation and optimize reprojection error in most cases. This makes it especially attractive for efficient triangulation of large scenes given the speed and low memory requirements.

1. Introduction

During the past years, there has been an escalation in the amount of work dealing with multi-view reconstruction of scenes, for example in applications such as robotics, surveillance and virtual reality. State-of-the-art systems [13] are already capable of reconstructing large scenes with high accuracy, with imagery obtained from the Internet, but higher accuracy and efficiency are still areas of ongoing work in the research community.

A fundamental component of reconstruction is *triangulation*, which refers to determining the 3D location of a scene point X from its location x_i in two or more images. When X reprojects exactly onto its imaged x_i coordinates, implying that all epipolar constraints [6] are satisfied, triangulation is trivial through any linear method. Due to noise, however, reprojected coordinates will not coincide with each respective x_i, which dually implies that rays through each x_i from each camera center will not intersect perfectly in 3D space. Therefore, triangulation is a difficult task given an arbitrary number of views and noisy image measurements, or *feature tracks*, and the goal becomes finding the point that best fits a given track. To this end, there are fast and somewhat accurate methods that solve for 3D points based on linear least squares, such as linear triangulation [6]. The midpoint method [6], though inaccurate in general, is by far the fastest method given two views. Such methods are not optimal, which means that given noisy inputs, the final 3D position can potentially be very inaccurate. More recent methods, which have tried to achieve higher accuracy, have focused on minimizing the L_2-norm of reprojection error. This yields the geometrically-meaningful maximum-likelihood estimate for the point assuming independent Gaussian image noise. This is a non-convex constrained optimization problem.

In the specific case of two views, a closed-form solution for optimal triangulation was first achieved by Hartley and Sturm [5], and then made more efficient by Lindstrom [7]. Both methods yield the set of all stationary points of their respective objective functions, which are then tested to obtain the optimal solution. Their objective functions actually alter the image positions x_i themselves, such that epipolar constraints are fulfilled and the final position can be triangulated through any linear method. For three views, a closed-form solution was first achieved by Stewénius *et al*. [14] and more recently by Byröd *et al*. [2]. Both of these methods make use of the *Gröbner basis* method for solving polynomial equation systems, which is computationally expensive and susceptible to precision inaccuracies. More details on these methods are provided in Section 2.2.

Currently, for more than three views, an optimal solver has not yet been achieved. The approach that has traditionally been used has been a two-phase method, where an initial linear method such as N-view linear triangulation [6] is applied to obtain an initial point, followed by non-linear *bundle adjustment* optimization to reduce the sum-of-

squares reprojection error [8]. However, this can be an expensive element in a reconstruction pipeline for high numbers of scene points and cameras, despite efficient sparse implementations [8]. Several issues exist in N-view linear triangulation, including numerical conditioning of the solution. It is also not clear how N-view linear triangulation would scale with high numbers of cameras viewing the same scene point. In general, while this two-phase procedure can give very accurate results, there is always a risk that the optimization will converge to a local and not global optimum of the cost function. The closest solution to an optimal solver for the N-view case has been the work of Agarwal *et al*. [1]. Unfortunately, their procedure is very expensive, timing results for triangulation are not provided, and only short feature tracks were tested. Though optimal solutions have been achieved for two and three views, increasing the efficiency and accuracy for general N-view triangulation is still an open problem.

The main contribution of this paper is to introduce a new framework for triangulation, aimed particularly at improving efficiency and accuracy in general N-view triangulation. There are two changes with respect to the literature. The first is to present a new cost function for triangulation, for which standard optimization algorithms such as adaptive gradient descent can be applied. The cost function is computed as the L_1 sum of angular differences between rays produced from each camera center to each candidate 3D position, which requires less operations to compute than L_2 on reprojection error [8]. This function is smoothly-varying in a large vicinity near the global optimum, and we can show that even somewhat inaccurate starting positions, such as with the midpoint method, can still lead to the optimum after optimization with simple adaptive gradient descent. This optimizer does not require the matrix operations of solving normal equations, such as in bundle adjustment with Levenberg-Marquardt [8]. The second is the use of a statistical sampling method based on confidence levels, where only a subset of a feature track's rays are used for triangulation. Overall, the framework provides a combination of speed and accuracy which can be considered an improvement to the current methods in the literature for N-view triangulation. While we do not guarantee optimality, our framework is very practical and efficient, and through extensive experiments we have shown that convergence to the global optimum is possible given a reasonable starting position. With respect to existing triangulation methods, ours possesses faster processing times and lower final reprojection errors. Also, there are no issues with numerical stability and we avoid the computation and solving of Gröbner bases. A summary of related work is provided in Section 2. The framework is detailed in Section 3, followed by experimental results (Section 4), and conclusions (Section 5).

2. Related work

Triangulation is an essential component in multi-view scene reconstruction. The following sequential stages are necessary for performing multi-view reconstruction, keeping in mind that there are many different algorithms and that these are just the most common steps. Feature matching and tracking can be sparse (for example with SIFT [9] or dense [6]), and consist on computing and linking the pixel coordinates in all images for each particular scene point, wherever it is visible. Dense approaches particularly suffer from issues such as occlusions, repetitive patterns, textureless regions and illumination changes. With tracks, the cameras' intrinsic parameters, such as focal length, can be recovered through self-calibration. Camera projection matrices can then be computed from the intrinsic and extrinsic parameters (pose) of translation and rotation [6]. Extrinsic calibration usually requires an estimation of the 'epipolar geometry' between image pairs or triplets [6], which mathematically encapsulates the intrinsic projective geometry between the images. There are a number of methods for pose estimation, and an overview is given in Rodehorst *et al*. [12]. Given camera projection matrices and feature tracks, triangulation is applied to compute the scene's 3D structure. This can be achieved by methods such as 'linear triangulation'. Finally, bundle adjustment can be performed to minimize the L_2 reprojection error of all computed structure points across all cameras with respect to the computed feature tracks [8]. Based on these basic steps, there are a number of successful general reconstruction algorithms in the literature, and comprehensive overviews and comparisons of different methods are given in Strecha *et al*. [15]. Software packages such as *Bundler* [13] are capable of estimating all calibration and structure parameters from a set of images. The rest of this section will provide a more in-depth analysis of triangulation, specifically.

2.1. Standard linear triangulation

In standard linear triangulation [6], solving for the best-fit 3D scene point involves setting up a $2N \times 4$ data matrix and performing SVD or eigen analysis to obtain a solution to a system of the form $AX = 0$ for each 3D position. In the simplest case of two views, the input is a set of 2D matches (x_i, x_i') between two images and the corresponding 3×4 projection matrices for each camera, respectively P and P'. Let $x_i = PX_i$ be the image plane coordinates of a 3D scene point X_i, and $x_i' = P'X_i'$ the coordinates of its match x_i' given the scene point X_i'. Linear triangulation makes use of the fact that X_i and X_i' should be equal since the feature matches x_i and x_i' should view the exact same 3D point. However, with noise, this is usually not the case. With X_i and X_i' being forced to be equal, this leads respectively to $K^{-1}x_i = P_{cam}X_i$ and $K^{-1}x_i' = P'_{cam}X_i$, where $P = KP_{cam}$ and $P' = K'P'_{cam}$, with K and K' be-

ing respective 3×3 intrinsic calibration matrices. Using normalized image coordinates, the expressions $x_{cam,i} = (x, y) \equiv P_{cam} X_i$ and $x'_{cam,i} = (x', y') \equiv P'_{cam} X_i$ are obtained. It follows that 2D image positions up to a scale factor w can be obtained in terms of the rows $P_{cam,i}$, as shown in Eq. 1 and expanded in Eq. 2. If the same equations are also set up for $x'_{cam,i}$ using P'_{cam}, a 4×4 system of the form $AX = 0$ is obtained. For better numerical conditioning, A is normalized by rows A_i such that $A_{norm} = (\frac{A_1}{||A_1||} \frac{A_2}{||A_2||} \frac{A_3}{||A_3||} \frac{A_4}{||A_4||})^T$. Extension to N views is trivial, following the same procedure such that each camera introduces two rows to an A matrix of size $2N \times 4$.

$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} P_{cam,1} \\ P_{cam,2} \\ P_{cam,3} \end{bmatrix} X_i \tag{1}$$

$$\begin{bmatrix} xP_{cam,3} - P_{cam,1} \\ yP_{cam,3} - P_{cam,2} \end{bmatrix} X_i = 0 \tag{2}$$

Finally, X_i is obtained as the eigenvector corresponding to the smallest eigenvalue of ${A_{norm}}^T A_{norm}$, or equivalently the last column of V in the singular value decomposition $A_{norm} = UDV^T$. The obtained 3D point is in homogeneous coordinates $\tilde{X}_i = (X_i Y_i Z_i W_i)^T$, such that the final 3D position is $X_i = (\frac{X_i}{W_i} \frac{Y_i}{W_i} \frac{Z_i}{W_i})^T$.

There are a number of issues with the solution provided by linear triangulation. First, the obtained solution is a simple, direct solve which is a 'best fit' to the input feature tracks, regardless of how noisy these are. It does not seek an optimal solution, or even one which minimizes reprojection error. Additionally, there is the potential for numerical stability issues, especially with near-parallel cameras in the two-view setup, where bad condition numbers for A can lead to very poor triangulations. Also, it has not been clear how performance and processing time scale with feature track length, and we analyze this in Section 4.1.

2.2. Optimal triangulation methods

Given the short-comings of linear triangulation, there are a number of recent *optimal* methods, which outperform linear triangulation in accuracy, but at the expense of computational efficiency. One such method is Hartley and Sturm's optimal method for two views [5]. This algorithm uses a given epipolar geometry estimation in order to alter original feature match positions x and x' over their respective image planes, such that they end up at the closest positions that lie on epipolar lines. This results in minimizing a 6^{th}-order polynomial to obtain the stationary points which minimize the L_2 reprojection error cost function, and then choose the correct solution by evaluation. The cost function includes parameters from the epipolar lines and fundamental matrix. Since the new positions for x and x' are ensured to lie on the same epipolar plane as the scene point they represent, any method can be used for direct triangulation. Lindstrom's 'fast triangulation' algorithm [7] re-writes optimal triangulation's equations in terms of Kronecker products, which allows for terms to cancel out so that the cost function is reduced to a quadratic equation. This eliminates the difficulties in solving a 6^{th}-order polynomial. Convergence occurs in exactly two iterations, and matches to very high precision with the original optimal triangulation result [5], but with higher stability and 1-4 orders of magnitude greater speed. Additionally, unstable camera configurations are handled with great results, including near-parallel cameras. Both of these algorithms are designed only for the two-view case, and alter the feature matches themselves.

The concept of solving for the stationary points of a cost function and evaluating to obtain the global optimal solution was first extended to the three-view case by Stewénius *et al*. [14], whose solver is constructed using techniques from computational commutative algebra. The solution amounts to computing the eigenvectors of 47×47 action matrices and evaluating the real solutions for the global optimum. One drawback of this method is that certain arithmetic operations are performed in high-precision floating point, with 128 bits of mantissa, in order to avoid round-off error accumulation. This makes the algorithm costly in terms of processing time. The three-view method by Byröd *et al*. [2] improves on the numerical issues by using a modified version of the Gröbner basis method for solving polynomial equation systems, the relaxed ideal method, which then trades speed for an increase in numerical stability. For optimal N-view triangulation, the only method we're aware of is by Agarwal *et al*. [1]. They demonstrate that several problems in multi-view geometry can be formulated using fractional programming. They also propose a *branch and bound* triangulation algorithm that provably finds a solution arbitrarily close to the global optimum, which minimizes reprojection error under Gaussian noise. While the method does provide a way to perform optimal N-view triangulation, no timing results are given, though they do claim almost linear runtime scaling with number of cameras. Results were demonstrated on just 3-6 cameras, and do not discuss how this scales and performs for larger numbers. For resectioning, on a $3GHz$ computer times are approximately 40-700 seconds on 6-100 cameras, which indicates that their brach and bound framework is generally expensive. As will now be discussed, we provide a non-optimal yet very accurate and efficient alternative to the discussed algorithms.

3. Triangulation cost function

We propose both a new cost function for triangulation, as well as introduce a statistical sampling component to make it more efficient. As input, we assume a set of feature tracks across N images and the respective 3×4 camera projection matrices P_i are known.

The cost function for our triangulation scheme is based on an angular error measure for a candidate 3D position, p, with respect to a feature track t. Recker *et al.* [11] first introduced a similar angular-based cost function for scalar field analysis, and we create a variation which allows for minima and fully develop the algorithm mathematically for the specific purpose of triangulation. To calculate the error for a candidate 3D position p, as shown in Fig. 1, for each camera center C_i, a unit direction vector v_i is first computed between it and the 3D position. A second unit vector, w_{ti}, is obtained by casting a ray from each C_i through the 2D projection of the evaluation point on each image plane (blue image plane dot in Fig. 1). Take into account that this projection generally does not coincide with the projection of vector v_i on each image plane (purple image plane dot in Fig. 1), and hence there is typically a non-zero dot product between each possible v_i and w_{ti}. Finally, the average of the errors (dot products) across all cameras is obtained. Notice that if this value is zero, the set of rays perfectly intersect at this evaluation position, making it the global optimum.

Mathematically, with C_i cameras, let T be the set of all

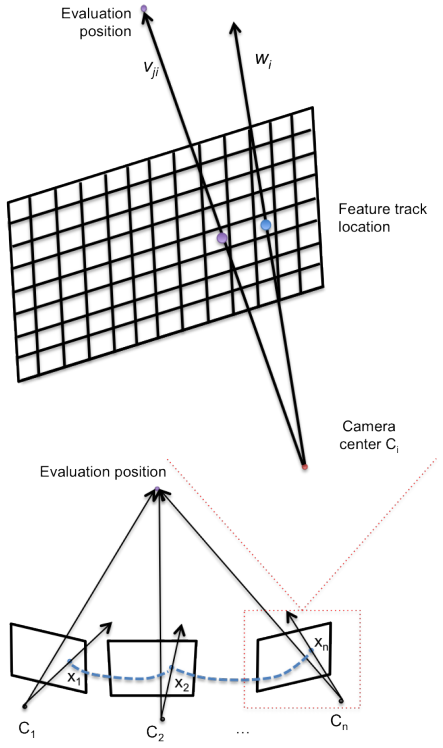


Figure 1: Angular error calculation for a candidate 3D position p, with respect to a feature track t.

feature tracks, and $p = (X, Y, Z)$ is the evaluation point under consideration. The cost function for p with respect to a track $t \in T$ is shown in Eq. 3, where $I = \{C_i | t$ "appears in " $C_i\}$, $\vec{v_i} = (p - C_i)$, and $\vec{w_{ti}} = P_i^+ t_i$. The right pseudo-inverse of P_i is given by P_i^+, and t_i is the homogeneous coordinate of track t in camera i. The expanded equation is shown in Eq. 4, with vector $v_i = (v_{i,X}, v_{i,Y}, v_{i,Z}) = (X - C_{i,X}, Y - C_{i,Y}, Z - C_{i,Z})$, its normalized version, $\hat{v_i} = \frac{v_i}{||v_i||}$, and similarly for $\hat{w_{ti}} = \frac{w_{ti}}{||w_{ti}||}$.

$$f_{t \in T}(p) = \frac{\sum_{i \in I}(1 - \hat{v_i} \circ \hat{w_{ti}})}{||I||} \tag{3}$$

$$f_{t \in T}(p) = \frac{\sum_{i \in I} 1 - (\hat{v_{i,X}} \hat{w_{ti,X}} + \hat{v_{i,Y}} \hat{w_{ti,Y}} + \hat{v_{i,Z}} \hat{w_{ti,Z}})}{||I||} \tag{4}$$

The cost function defined in Eq. 3 has its lowest possible value at zero but is non-convex, and thus a direct linear solution is not possible. Non-linear optimization is needed, for which a gradient calculation is required. Gradients along the X, Y and Z directions are defined by $\nabla f_{t \in T} = \left(\frac{\sum_{i \in I} a}{||I||}, \frac{\sum_{i \in I} b}{||I||}, \frac{\sum_{i \in I} c}{||I||} \right)$, with a, b and c defined as in Eqs. 5-7, with the denominator $d = ((C_{i,X} - X)^2 + (C_{i,Y} - Y)^2 + (C_{i,Z} - Z)^2)^{(3/2)}$ in each expression.

$$\begin{aligned} a = (&-C_{i,Y}^2 \hat{w_{ti,X}} - C_{i,Z}^2 \hat{w_{ti,X}} - C_{i,X} \hat{w_{ti,Y}} Y \\ &+ \hat{w_{ti,Y}} XY - \hat{w_{ti,X}} Y^2 + C_{i,Y}(C_{i,X} \hat{w_{ti,Y}} - \hat{w_{ti,Y}} X \\ &+ 2\hat{w_{ti,X}} Y) - C_{i,X} \hat{w_{ti,Z}} Z + \hat{w_{ti,Z}} XZ - \hat{w_{ti,X}} Z^2 \\ &+ C_{i,Z}(C_{i,X} \hat{w_{ti,Z}} - \hat{w_{ti,Z}} X + 2\hat{w_{ti,X}} Z))/d \end{aligned} \tag{5}$$

$$\begin{aligned} b = (&-C_{i,X}^2 \hat{w_{ti,Y}} - C_{i,Z}^2 \hat{w_{ti,Y}} - C_{i,Y} \hat{w_{ti,X}} X \\ &- \hat{w_{ti,Y}} X^2 + \hat{w_{ti,X}} XY + C_{i,X}(C_{i,Y} \hat{w_{ti,X}} + 2\hat{w_{ti,Y}} X \\ &- \hat{w_{ti,X}} Y) - C_{i,Y} \hat{w_{ti,Z}} Z + \hat{w_{ti,Z}} YZ - \hat{w_{ti,Y}} Z^2 \\ &+ C_{i,Z}(C_{i,Y} \hat{w_{ti,Z}} - \hat{w_{ti,Z}} Y + 2\hat{w_{ti,Y}} Z))/d \end{aligned} \tag{6}$$

$$\begin{aligned} c = (&-C_{i,X}^2 \hat{w_{ti,Z}} - C_{i,Y}^2 \hat{w_{ti,Z}} - C_{i,Z} \hat{w_{ti,X}} X \\ &- \hat{w_{ti,Z}} X^2 - C_{i,Z} \hat{w_{ti,Y}} Y - \hat{w_{ti,Z}} Y^2 + \hat{w_{ti,X}} XZ + \\ &\hat{w_{ti,Y}} YZ + C_{i,X}(C_{i,Z} \hat{w_{ti,X}} + 2\hat{w_{ti,Z}} X - \hat{w_{ti,X}} Z) \\ &+ C_{i,Y}(C_{i,Z} \hat{w_{ti,Y}} + 2\hat{w_{ti,Z}} Y - \hat{w_{ti,Y}} Z))/d \end{aligned} \tag{7}$$

3.1. Cost function properties

Our cost function is an L_1 sum of error terms derived from dot products between rays in 3D space. Each dot product can vary from $[-1, 1]$, but in practice we only deal with points that lie in front of the cameras, and hence the range $[0, 1]$. The function is positive-definite, with an absolute minimum at zero. Intuitively, minimizing our cost function to zero means that the triangulation rays are perfect. In this perfect condition, reprojection error would also be zero, but we do not minimize reprojection error directly by measuring 2D pixel errors as in the L_2 reprojection error cost function. This cost function is not mathematically equivalent to ours, and has different local minima. Ours is not optimal under Gaussian noise like L_2, but L_1 is more robust to

outliers [6], and essentially determines the median angular error. It is also significantly cheaper to compute ours: a single dot product and a subtraction for each term compared to a matrix multiply, subtraction, and then distance computation for L_2 on reprojection error. The only way to guarantee optimality would be by solving for all of its stationary points and evaluating, which we have not yet achieved. However, in practice we have been able to show that nonlinear optimization methods, even simple ones such as gradient descent, can lead to the optimum value from a wide range of starting locations. This is true even when starting with the generally-inaccurate midpoint method [6], as discussed in the Results section. Fig. 2 shows a scalar field, consisting of our cost function measured for a dense set of test positions near a known ground-truth position. Notice the smooth variation in a large vicinity surrounding this position, which is key to the success of our cost function. The advantage of being able to use gradient descent is to avoid matrix storage and operations, foregoing the matrix operations of solving normal equations, such as in bundle adjustment with Levenberg-Marquardt [8].

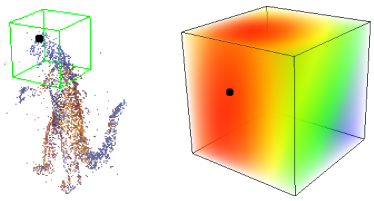


Figure 2: Multi-view reconstruction (left), where the camera follows a circular path above the scene. A volume view of a scalar field representing our angular cost function evaluated at a dense grid of sample points around a ground-truth position (black dot in both images) is also shown (right). Redder values are closer to zero cost.

3.2. Statistical sampling

A second component to our triangulation framework is the use of statistically-meaningful samples of rays as opposed to the entire available set, N. For this, we make use of confidence levels in selecting a random set of rays from the full set. For instance, a 95% confidence level with a 5% margin of error implies that a point computed from a reduced sample will have a 95% probability of being within a 5% margin of error of the position obtained with the entire set of rays. The central limit theorem states that when the sample size is large, approximately 95% of the sample means will fall within $t = 1.96$ standard errors σ of the population mean μ, $\mu \pm \frac{\sigma}{\sqrt{N}}$, assuming a normal distribution of the data. To compute sample size, n_0, we use Cochran's formula [3], shown in Eq. 8. The value σ^2 is an estimate of the variance, and we used $\sigma = 0.5$ as the fixed value. The value for 'd' corresponds to the maximum error of estimate for a sample mean, which we fix at 5%, or 0.05. In case the obtained sample size exceeds 5% of N, Cochran's correction formula [3] should be used to calculate the final sample size, n, as shown in Eq. 8. For example, if a feature track consists of 10000 cameras, a 95% confidence level yields a much-reduced sample size of 370. Though sampling implies a known risk that a randomly-chosen set of rays may not be representative of the entire population for a given feature track, we found that in practice the sampling procedure works surprisingly well. Final reprojection errors are on par with those obtained when using full sets of rays, and a significant speed-up is achieved, as will be discussed in the Results section. This sampling technique may be reminiscent of RANSAC [4], but there are some key differences. First, we measure a simple geometrical error in search for the first which meets the criteria, and do not seek to fit a model. Second, we are not concerned with inliers and outliers, as we have a fixed threshold. Finally, we seek an approximate starting position at the cheapest possible price, and RANSAC would be significantly more expensive.

$$n_0 = \frac{t^2\sigma^2}{d^2} \qquad n = \frac{n_0}{1 + \frac{n_0}{N}} \tag{8}$$

3.3. Implemented triangulators

The proposed cost function and sampling component were used to design a number of triangulators, which differ in how initial starting positions are computed, the confidence level used, and in the way optimization is performed. Sampling is only applied for feature track lengths greater than 30; otherwise the full sample is used. For confidence levels, $c\%$, 90% and 95% are standard and were found to produce the best results, though we also tested 75% for added speed and 99% for improved confidence. Let M denote a starting point obtained through the midpoint method, and L for linear triangulation. Initial starting positions based on the midpoint algorithm were computed as follows for the multi-view case ($N > 2$). For a feature track, after obtaining a sample of n cameras based on $c\%$ confidence, we pick an initial random pair from the sample and obtain the ratio between the pair's baseline in 3D and the distance between the shortest-distance locations on the two rays through corresponding feature track locations. If the ratio is more than 0.1, another pair is chosen until one meets the criteria, for which the midpoint is computed as the initial 3D position for optimization. While this is not as robust as analyzing all possible combinations for the smallest ratio, in practice we found this to be much faster and to perform surprisingly well. If no ray pair meets the criteria, we assume a very inaccurate feature track and do not attempt triangulation, discarding the track. Such an initial condition analysis is not performed in standard linear triangulation.

The following triangulators were evaluated: standard linear triangulation (**LT**), our multi-view midpoint triangulation (**MP**), midpoint followed by cost function optimization (**GTM**) without sampling, the previous triangulator starting with linear (**GTL**), midpoint/linear triangulation with sampling at $c\%$ and cost function optimization with a 'full finish' (**SFGTM(75,90,95,99)** and **SFGTL(95,99)**, respectively), midpoint/linear triangulation with sampling at $c\%$ and cost function optimization using just the sample (**SGTM(75,90,95,99)** and **SGTL(75,90,95,99)**, respectively), and standard linear triangulation with sampling at $c\%$ (**SLT95** and **SLT99**). For the 'full finish' (F) methods, the difference is that near convergence, as determined by the relative step sizes between consecutive iterations of gradient descent, the sample n is replaced by the full set of rays N, in order to evaluate if this made any difference.

4. Results

A number of tests were designed to test the accuracy, processing time and general behavior of the proposed triangulators on both real and synthetic data. Code for our methods, and also for linear and midpoint triangulations [6], was written using *C++*. All tests were conducted on a MacBook Pro with an *Intel Core i7* processor at 2.66 GHz with 4 GB of RAM, running Mac OS X Lion 10.7.3. For matrix operations such as eigen-analysis and SVD, the efficient *Eigen* library (`http://eigen.tuxfamily.org`) was used.

4.1. Synthetic tests

Synthetic tests were designed to observe the behavior of the proposed triangulators versus ground-truth structure in the presence of feature track noise and with very long tracks. The main parameters evaluated in all tests were processing time and reprojection error. Our triangulators were compared against *LT* and our midpoint method, *MP*, but not against the optimal methods. In two-view methods [5, 7] the tracks are corrected so we cannot compare directly, and for three views we discuss further in Section 4.1.1.

The first experiment consisted of analyzing our proposed triangulators under noise. Results for processing time versus feature tracking error are shown in Fig. 3. For this test, 100 synthetic points were rendered, with ground-truth tracks created over 1000 cameras, placed in an evenly-spaced circular configuration. For each run, feature tracking errors were simulated synthetically by adding image plane noise to the ground-truth tracks, in random directions, up to $M\%$ of the image plane diagonal dimension, from integers 0% to 6%. All 100 points were triangulated at each noise level, and the average was obtained. The bottleneck of our algorithm is the gradient descent step. However, it can be seen that triangulators *SGTL**, *SGTM** and *SLT** are faster than *LT*, except when using $c = 99\%$, across all feature tracking error values. Results for average and standard deviation of reprojection error versus feature tracking error are shown in Fig. 4 and Fig. 5. The behavior is, in general, linear for all methods, showing stability in the presence of noise. However, the values are slightly better for our 'full finish' methods *SGTFM** and *SGTFL** than for *LT*. Processing time versus feature track length is shown in Fig. 6. It can be seen that our triangulators show a much slower increase in processing time than *LT*, as the feature track length is increased. For example, *SGTM95* is more than 150 times faster than *LT* for 100000 cameras. This is key towards triangulation in large scenes with long feature tracks.

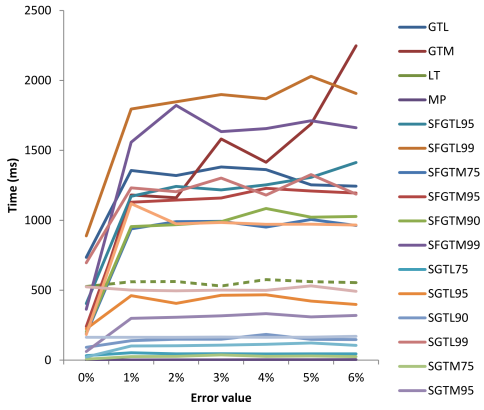
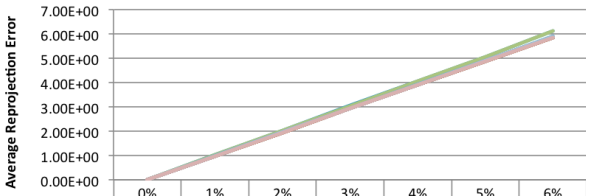


Figure 3: Feature tracking error versus computation time (ms), for all tested triangulation methods. For comparison, a dotted line is used for standard linear triangulation, *LT*.

4.1.1 Three-view triangulation and bundle adjustment

In the implementation of Byröd's optimal three-view method [2], projection matrices and feature tracks are provided from which they claim to have obtained the 10 exact ground-truth 3D values. Our triangulators agree with the ground-truth values up to at least 8 decimal places, except in the third case. With very noisy initial inputs, our method can yield an inaccurate result, such as the unlikely occurrence of this case. As for processing time, Stewénius *et al.* [14] took 20 hours to triangulate the *Dinosaur* dataset [10] on 128-bit arithmetic, and Byröd *et al.* took 2.5 minutes, whereas for example *SGTM95* takes 0.5 seconds on hardware that is not much newer.

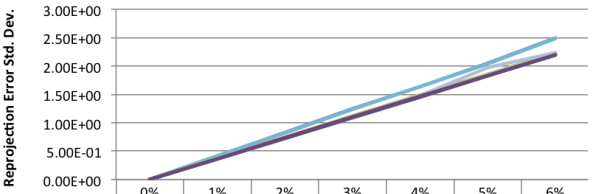
We also compared our *SGTFM90* to *LT* when followed by bundle adjustment. In *fountain-P11* [15], *SGTFM90* was 9% faster than *LT* in obtaining 455 initial 3D positions, and both point clouds were optimized using SBA [8]. Our points converged to a final total reprojection error of 1.479 pixels in 21 iterations, whereas the *LT* points took 74 iterations and five times longer, converging to an error of 16.57 pixels. This shows that SBA likely converged to a local minimum when starting with *LT*. Our triangulation, though not opti-



	0%	1%	2%	3%	4%	5%	6%
GTL	9.23E-14	0.978502	1.95416	2.93101	3.90816	4.88837	5.86434
LT	9.01E-14	0.978503	1.95416	2.93102	3.9082	4.88839	5.86441
SFGTL95	4.42E-12	0.978502	1.95416	2.93101	3.90816	4.88837	5.86434
SFGTL99	9.00E-13	0.978502	1.95416	2.93101	3.90816	4.88837	5.86434
SFGTM75	4.66E-13	0.978502	1.95416	2.93101	3.90816	4.88837	5.86434
SFGTM95	1.01E-12	0.978502	1.95416	2.93101	3.90816	4.88837	5.86434
SFGTM90	8.70E-13	0.978502	1.95416	2.93101	3.90816	4.88837	5.86434
SFGTM99	4.65E-13	0.978502	1.95416	2.93101	3.90816	4.88837	5.86434
SGTL75	1.48E-11	1.02391	2.04297	3.06135	4.06353	5.06791	6.11549
SGTL95	2.05E-12	0.979756	1.95713	2.93569	3.91376	4.89625	5.87375
SGTL90	2.08E-12	0.98442	1.96477	2.94883	3.93259	4.91748	5.89589
SGTL99	3.27E-13	0.978533	1.95433	2.93119	3.90842	4.88871	5.86472
SGTM75	2.53E-13	1.01938	2.02914	3.04286	4.05377	5.07013	6.11563
SGTM95	4.26E-13	0.979957	1.95744	2.93524	3.91383	4.89582	5.87331
SGTM90	3.07E-13	0.984822	1.96618	2.94885	3.92851	4.89765	5.89956
SGTM99	5.30E-13	0.978548	1.95429	2.93119	3.90836	4.88864	5.8647
SLT95	1.45E-12	0.979869	1.95732	2.93591	3.91454	4.89558	5.87274
SLT99	7.86E-13	0.978566	1.95426	2.93118	3.90844	4.88874	5.86481

Error value

Figure 4: Feature tracking error versus average reprojection error (pixels), for all tested triangulators.

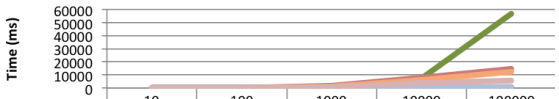


	0%	1%	2%	3%	4%	5%	6%
GTL	4.84E-14	0.364801	0.729582	1.09711	1.45801	1.82569	2.18916
LT	4.76E-14	0.3648	0.729595	1.09713	1.45801	1.82583	2.18932
SFGTL95	1.61E-11	0.364801	0.729582	1.09711	1.45801	1.82569	2.18916
SFGTL99	3.69E-12	0.364801	0.729582	1.09711	1.45801	1.82569	2.18916
SFGTM75	1.04E-12	0.364801	0.729582	1.09711	1.45801	1.82569	2.18916
SFGTM95	7.27E-12	0.364801	0.729582	1.09711	1.45801	1.82569	2.18916
SFGTM90	4.60E-12	0.364801	0.729582	1.09711	1.45801	1.82569	2.18916
SFGTM99	9.96E-13	0.364801	0.729582	1.09711	1.45801	1.82569	2.18916
SGTL75	6.57E-11	0.417627	0.833346	1.24944	1.64266	2.0482	2.48955
SGTL95	9.76E-12	0.366486	0.73335	1.10286	1.46614	1.83517	2.20119
SGTL90	7.34E-12	0.372722	0.744276	1.12084	1.49165	1.86396	2.23118
SGTL99	1.13E-12	0.364945	0.72968	1.09741	1.4584	1.82598	2.18947
SGTM75	3.40E-13	0.41348	0.81742	1.23159	1.63125	2.03678	2.4874
SGTM95	1.32E-12	0.366484	0.734192	1.10275	1.46493	1.8345	2.20455
SGTM90	4.68E-13	0.373204	0.743604	1.1205	1.48682	1.97831	2.23344
SGTM99	2.84E-12	0.364908	0.729702	1.0973	1.45827	1.82605	2.18963
SLT95	3.83E-12	0.366575	0.733828	1.10271	1.46679	1.83534	2.1995
SLT99	2.98E-12	0.364877	0.72983	1.09738	1.45826	1.82603	2.18959

Error value

Figure 5: Feature tracking error versus standard deviation of reprojection error (pixels), for all tested triangulators.

mal, yields more-accurate points in less time, and this also



	10	100	1000	10000	100000
LT	30	114	744	7262	56900
MP	2	1	2	6	29
SGTL75	48	52	57	63	241
SGTL95	50	184	544	607	794
SGTL90	48	147	203	198	370
SGTL99	41	219	1666	7649	14632
SGTM75	15	31	34	54	225
SGTM95	20	126	430	496	651
SGTM90	21	97	161	140	315
SGTM99	24	151	1263	6026	12204
SLT95	28	79	226	333	433
SLT99	28	111	669	3795	5479

Feature Track Length

Figure 6: Feature track length (number of cameras) versus computation time (ms), for all tested triangulators.

Data set	t_l	t_p	ϵ_l	ϵ_p	C
Canyon	22354	14442	0.231	0.220	90
Canyon-dense	192426	83916	1.841	1.839	3
Dinosaur	1170	627	0.468	0.468	36
Palmdale-dist	9518	5110	1.713	1.138	66
Palmdale	2723	1475	1.645	1.159	66
Stockton	3187	2333	0.982	0.726	10

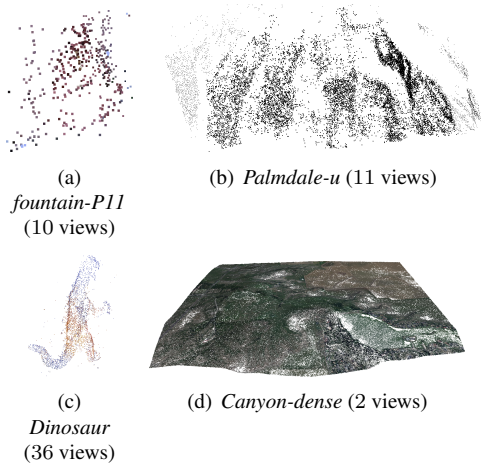
Table 1: Processing times t_l and t_p (ms) and average reprojection errors ϵ_l and ϵ_p (pixels) respectively for *LT* and *SGTM95*, with number of cameras C, for real datasets.

aids in the convergence of bundle adjustment.

4.2. Evaluation on real data

For real scenes, processing time and reprojection error were evaluated. These included *castle-P30*, *fountain-P11* and *Herz-Jesu-P25* [15], *Dinosaur* [10], *Notre Dame* [13], and our own datasets *Palmdale*, *Stockton*, *Walnut Creek* and *Canyon*. Our results demonstrated *SGTFM90* and *SGTM95* to be two of the overall best-performing triangulators. Table 1 displays processing time and reprojection error results for *LT* versus *SGTM95*. Our triangulators always outperformed *LT*, and especially in *Palmdale-dist*, where the images present a strong radial distortion. However, standard deviation of reprojection errors is slightly lower when using a linear triangulation start, perhaps indicating that certain initial midpoints misguide optimization. Finally, reconstructions obtained with *SGTFM90* are shown in Fig. 7.

In summary, we present a triangulation framework that presents excellent behavior both in real and synthetic testing, and based on these results we can claim that it outperforms the only proven and standard method for more than three views, N-view linear triangulation, while also



(a) *fountain-P11* (10 views)

(b) *Palmdale-u* (11 views)

(c) *Dinosaur* (36 views)

(d) *Canyon-dense* (2 views)

Figure 7: Scenes reconstructed with triangulator *SGTFM90*.

behaving excellently for two and three views. If guaranteed optimality is required, optimal algorithms [5, 7, 14, 2] for those cases may give slightly more accurate results, but at potentially much higher processing times. Ultimately, our framework is very flexible and its use depends on what the end user wants. For fast but not very accurate results, use a lower confidence and a midpoint start. For better accuracy but a slower result, use a linear start and a higher confidence level. We do not guarantee optimality, but present our work as a very competitive alternative to other triangulators, and especially for general N-view triangulation.

5. Conclusions

This paper presented a framework for triangulation in multi-view scene reconstruction, which outperforms the only standard tool for N views, linear triangulation, in processing time and final reprojection error. We provide two main contributions. First, a robust, inexpensive and smoothly-varying cost function that allows for reliable convergence to the global optimal solution with simple adaptive gradient descent. Additionally, we present a statistical sampling strategy designed to decrease the processing time by using a small sample from the total set of rays for a feature track. In general, the implemented triangulators outperform N-view linear triangulation in real and synthetic testing, and are especially attractive for large numbers of cameras. Despite no guaranteed optimality, our triangulators are also very competitive in practice for two and three-view triangulation, and up to several orders of magnitude faster than optimal solvers.

Acknowledgments

This work was supported in part by the Department of Energy, National Nuclear Security Agency through Contract No. DE-GG52-09NA29355. This work was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

[1] S. Agarwal, M. K. Ch, F. Kahl, and S. Belongie. Practical global optimization for multiview geometry. In *ECCV*, pages 592–605, 2006.

[2] M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. In *Proceedings of the 8th Asian Conference on Computer Vision*, ACCV'07, pages 549–559, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] W. G. Cochran. *Sampling Techniques, 3rd Edition*. John Wiley, 1977.

[4] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Readings in computer vision: issues, problems, principles, and paradigms*, pages 726–740, 1987.

[5] R. I. Hartley and P. Sturm. Triangulation. *Comput. Vis. Image Underst.*, 68(2):146–157, 1997.

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

[7] P. Lindstrom. Triangulation Made Easy. In *CVPR*, pages 1554–1561, 2010.

[8] M. Lourakis and A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg-Marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, August 2000.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal On Computer Vision*, 60(2):91–110, 2004.

[10] Oxford Visual Geometry Group. Multi-view and Oxford Colleges building reconstruction. `http://www.robots.ox.ac.uk/~vgg/`, August 2009.

[11] S. Recker, M. Hess-Flores, M. A. Duchaineau, and K. I. Joy. Visualization of scene structure uncertainty in a multi-view reconstruction pipeline. In *VMV 2012*, in press.

[12] V. Rodehorst, M. Heinrichs, and O. Hellwich. Evaluation of relative pose estimation methods for multi-camera setups. In *International Archives of Photogrammetry and Remote Sensing (ISPRS '08)*, pages 135–140, Beijing, China, 2008.

[13] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 835–846, New York, NY, USA, 2006. ACM.

[14] H. Stewénius, F. Schaffalitzky, and D. Nistér. How hard is 3-view triangulation really? *Computer Vision, IEEE International Conference on*, 1:686–693, 2005.

[15] C. Strecha, W. von Hansen, L. J. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR'08*, 2008.