# Fury of the Swarm: Efficient and Very Accurate Triangulation for Multi-View Scene Reconstruction

Shawn Recker, Mauricio Hess-Flores, Kenneth I. Joy
University of California Davis
One Shields Ave. Davis, CA 95616
Email: http://www.idav.ucdavis.edu/people/

## Abstract

*This paper presents a novel framework for practical and accurate $N$-view triangulation of scene points. The algorithm is based on applying swarm optimization inside a robustly-computed bounding box, using an angular error-based $L_1$ cost function which is more robust to outliers and less susceptible to local minima than cost functions such as $L_2$ on reprojection error. Extensive testing on synthetic data with ground-truth has determined an accurate position over $99.9\%$ of the time, on thousands of camera configurations with varying degrees of feature tracking errors. Opposed to existing polynomial methods developed for a small number of cameras, the proposed algorithm is at best linear in the number of cameras and does not suffer from inaccuracies inherent in solving high-order polynomials or Gröbner bases. In the specific case of three views, there is a two to three order of magnitude performance increase with respect to such methods. Results are provided to highlight performance for arbitrary camera configurations, numbers of cameras and under noise, which has not been previously achieved in the triangulation literature. Results on real data also prove that reprojection error is improved with respect to other methods.*

## 1. Introduction

The reconstruction of scenes from multiple images or video streams has become an essential component in a number of modern applications, such as robotics, surveillance and virtual reality. A number of systems [18, 3] are capable of accurately reconstructing scenes from thousands of images, obtained for instance from the Internet. However, increasing the accuracy, efficiency, and reconstruction density are still areas of ongoing research in the community.

Triangulation is an important step in the process of scene reconstruction. Triangulation determines the 3D location of a scene point $X$ from its imaged pixel location $x_i$ in two or more images. When $X$ reprojects exactly onto its $x_i$ coordinates, triangulation is trivial through simple, linear methods. However, in the presence of image noise, the reprojected coordinates of $X$ will not coincide with each respective $x_i$. In the general setting with an arbitrary number of cameras, possibly noisy camera parameters, and image measurements (*feature tracks*), the goal becomes finding the point $X$ that best fits a given track. There are a number of methods in the literature to perform triangulation. The midpoint method [9], though inaccurate in general, is by far the fastest method given two views. Another common, fast and somewhat accurate method that solves for 3D points based on linear least squares is *linear triangulation* [9]. Such methods are not very accurate, and the obtained solution is not necessarily the best from all feasible solutions. Therefore, given noisy inputs, the final obtained 3D position can be very inaccurate.

A number of optimal algorithms exist in the literature, but most solve for very small numbers of views. For two views, a closed-form solution was first proposed by Hartley and Sturm [8], made more practical by Kanatani *et al*. [10], and later made more efficient and simplified by Lindstrom [11]. These methods yield the set of all stationary points of their respective cost functions, which are then tested to obtain the optimal solution. These algorithms result in new image positions $x_i$, such that epipolar constraints are fulfilled. The final position can then be triangulated through any linear method. For three views, closed-form solutions were achieved by Stewénius *et al*. [19] and more recently by Byröd *et al*. [2]. Both algorithms make use of the *Gröbner basis* method for solving systems of polynomial equations, which is computationally expensive and susceptible to precision inaccuracies. More details on these methods are provided in Section 2.

A polynomial solver has not been achieved in the literature for more than three views. The approach that is traditionally used for multi-view optimization has been a two-phase method, where an initial linear method such as $N$-view linear triangulation [9] is applied to obtain an initial

point. This is proceeded by non-linear *bundle adjustment* optimization to reduce the sum-of-squares $L_2$-norm of reprojection error [12], a non-convex constrained optimization problem. This yields the maximum-likelihood estimate for the point assuming independent Gaussian image noise, which is geometrically meaningful. However, bundle adjustment is very expensive for high numbers of scene points and cameras, even though sparse implementations alleviate complexity [12]. Furthermore, it is very prone to miring in local minima, so it requires an accurate initialization to produce an accurate estimate. Also, several issues exist in $N$-view linear triangulation which affect the accuracy of the result, including numerical conditioning. sTherefore, while this two-phase procedure can be very accurate, there is always a risk of converging to a local minima of the cost function, especially when starting far from the solution. A recent triangulator by Recker *et al*. [16] uses a novel $L_1$ angular error cost function, which is optimized with adaptive gradient descent given an initial midpoint estimate. It shows a significant speed increase and better reprojection errors than other triangulators, such as $N$-view linear. However, the solution has not been proven to be very robust.

A few $N$-view optimal solvers have been proposed, explained in further detail in Section 2, but with very limited results. The work of Agarwal *et al*. [1] is based on fractional programming and branch-and-bound theory. However, their procedure is very expensive, and only short feature tracks were tested. There are also a few methods based on convex optimization on an $L_\infty$ cost function, such as Hartley and Kahl [7], Min [13], and most recently Dai *et al*. [4]. However, in general it is not clear how algorithms based on $L_\infty$ behave under noise and for arbitrary numbers of cameras. It is also not justified why $L_\infty$ was chosen over $L_1$, which behaves very well in Dai *et al*. [4].

In summary, though optimal solutions have been achieved for two and three views, a well-tested solver for an arbitrary number of views remains an open problem, which we seek to solve through thorough testing of an $L_1$-based solver. The main contribution of this paper is to introduce a practical and efficient algorithm for very accurate $L_1$-based $N$-view triangulation of scene points. Though not theoretically optimal, since a rigorous mathematical model is not provided here, in extensive simulation the triangulation algorithm converged to the global optimum in over $99.9\%$ of test cases with ground-truth data. The test cases varied across a wide number of camera configurations and feature tracking errors. There are two main novelties of this framework with respect to the previous literature. The first is a non-polynomial solution to the optimization problem, based on minimizing an $L_1$-based cost function [16]. The second is the introduction of a swarm optimization algorithm, applied on this cost function. The $L_1$ sum of angular errors requires fewer operations to compute than $L_2$ bundle adjustment with reprojection error [12], where expensive matrix operations are required to solve normal equations. It is also less susceptible to local minima and is robust in the presence of noise. There are no issues with numerical stability, such as those present in algorithms based on computing and solving Gröbner bases. Processing time is at best linear with the number of cameras, as discussed in Section 3.1. A summary of related work is provided in Section 2. The proposed triangulator is detailed in Section 3, followed by experimental results (Section 4), and conclusions (Section 5).

## 2. Related work

Multi-view scene reconstruction involves a number of stages applied sequentially, where the output of one stage directly affects accuracy in the following steps. There are many general scene reconstruction algorithms in the literature, several of which have been successful for certain applications. A comprehensive overview and comparison of different methods is given in Strecha *et al*. [20]. In this paper, our focus will be on triangulation and not on the other components of reconstruction, for which the reader is referred to surveys in the literature [20].

Triangulation is one of the final steps in multi-view scene reconstruction and its accuracy depends directly on the accuracy of previously-computed feature tracking, camera intrinsic calibration, and pose estimation [9]. Typically, $3 \times 4$ projection matrices are used to encapsulate all camera intrinsic and pose information. The most widely-used method in the literature, *linear triangulation* [9], involves solving for the best-fit 3D scene point in one algebraic, non-iterative step. A system of the form $AX = 0$ is solved, where the $A$ matrix is a function of feature track and camera projection matrix values. The obtained solution is a direct, best-fit and non-optimal solve. Numerical stability issues are possible, especially with near-parallel cameras.

To this end, a recent triangulator by Recker *et al*. [16] solves several of these issues. Their framework introduces an angular error-based $L_1$ cost function, which is robust to outliers and inexpensive to compute. After initializing through the simple midpoint method, adaptive gradient descent is applied on the cost function. A statistical sampling component, based upon confidence levels, reduces the number of rays used for triangulation of a given feature track. Results are demonstrated on real and synthetic data, where it achieves a significant speed increase and better reprojection errors than other triangulators, such as $N$-view linear. However, despite the great results, this triangulator has not been proven optimal. Furthermore, the initialization procedure based on the midpoint algorithm is not ideal due to the inaccuracy in this algorithm.

There exist a number of provably optimal triangulation algorithms in the literature, which also surpass linear triangulation in accuracy. The main drawback of these methods

is their computational efficiency and that the majority are designed specifically for two or three views. This class of algorithms is based on *polynomial methods* [7], where all stationary points of a cost function are computed and evaluated to find the global minimum. A solution can be achieved as long as the cost function can be expressed as a rational polynomial function in the parameters. The cost function's extrema are located where the derivatives with respect to the parameters become zero.

For two views, Hartley and Sturm's original optimal method [8] uses a given epipolar geometry estimation in order to apply a Sampson correction on original feature match positions $x$ and $x'$ over their respective image planes, so that they end up at the closest positions that lie on epipolar lines. This algorithm requires solving for the stationary points of a $6^{th}$-order polynomial and then evaluating each real root. The new computed positions for $x$ and $x'$ lie exactly on the same epipolar plane as the scene point they represent, such that any triangulation method can be applied. Most recently, Lindstrom's 'fast triangulation' algorithm [11] expresses optimal triangulation's equations in terms of Kronecker products, which allows for terms to cancel out and the cost function is reduced to a quadratic equation. The algorithm converges in two iterations and with one to four orders of magnitude performance increase over the previous technique. Unstable configurations such as near-parallel cameras are handled without degeneracies.

Existing methods for optimal three-view triangulation are also polynomial. The main difference is that feature track positions are left intact. The first of these algorithms was proposed by Stewénius *et al*. [19], who used techniques from computational commutative algebra such as the Gröbner basis method for solving polynomial equation systems. The algorithm computes the eigenvectors of $47 \times 47$ action matrices and evaluates the real solutions, where up to 24 minima exist. One major drawback is its costly processing time, since certain arithmetic operations must be performed in high-precision floating point, with 128 bits of mantissa, to avoid round-off error accumulation. The three-view method by Byröd *et al*. [2] alleviates numerical issues in the previous algorithm, but at the expense of a greater processing time. It is based on a modified version of the Gröbner basis method. In such polynomial approaches, the degree of the polynomial grows cubically with the number of views. The sequence of degrees is 6, 47, 148, 336, 638, 1081 for triangulation from two to seven cameras [19]. This implies a huge computational cost and potentially a large number of local minima.

We are aware of only a few attempts at optimal $N$-view triangulation for more than three views, which deviate from the polynomial method. The main issue is a lack of experimental results as far as error and processing time against different noise and camera configurations. The first is by Agarwal *et al*. [1], who demonstrated that several problems in multi-view geometry can be formulated using fractional programming, including triangulation. They propose a *branch and bound* algorithm that provably finds a solution arbitrarily close to the global optimum, which minimizes reprojection error under Gaussian noise. Hartley and Kahl's survey paper [7] as well as Min [13] perform convex optimization on an $L_\infty$ cost function making use of second-order cone programming (SOCP). Most recently, Dai *et al*. [4] described an $L_\infty$ optimization method, based on gradually contracting a region of convexity towards computing the optimum. With simulations consisting of 50 randomly-generated views of 100 random 3D points, it is claimed that convergence can be achieved from any initial point. The method outperforms previous $L_\infty$ methods in both time and iterations. In the next section, we propose a novel very accurate $N$-view triangulator, which outperforms the existing algorithms. It can be applied on an arbitrary number of cameras and is linear at best with the number of cameras. It is not plagued by numerical stability or precision issues. The algorithm is based on applying swarm optimization based on Recker *et al*.'s triangulation cost function [16], both of which will be discussed in detail.

## 3. Proposed triangulator

The polynomial method is intractable for an arbitrary number of views due to the cubic growth of the algorithm [19]. Instead, our focus will be on finding a more *practical* and efficient solution to the triangulation problem, where we do not seek to detect or to have knowledge of all local minima. We argue this is not necessarily important in the grand scheme of simply finding the global optimum. In fact, a potentially large number of polynomial solutions may violate cheirality and thus not be geometrically meaningful.

In general, our approach is based on searching within a bounding box where the probability of enclosing the global minimum of a cost function is extremely high. As we will demonstrate, our procedure works in $99.9\%$ of cases, with failures confined to very specific camera configurations. The cost function must be defined in terms of the $(X, Y, Z)$ position parameters for a given point, assuming fixed feature tracking and camera parameters. The task amounts to finding the 3D position which globally minimizes the cost.

**On the advantages of using the $L_1$ norm** Optimization involves finding the set of parameters that best fits a model, where a cost function is typically minimized over the set of parameters. However, there is a question of which cost functions should be referred to as very accurate. The $L_2$ least-squares solution is the maximum likelihood (ML) estimate under Gaussian image noise. However, it can be argued that image measurement noise

does not always follow such a distribution and ML is not necessarily equivalent to optimal [7]. The ML estimate may be biased, and as noise increases, the average estimate drifts away from the true value [7]. Furthermore, the ML cost function typically contains many local minima. The $L_\infty$ model assumes uniform bounded noise so that all measurements less than a threshold distance from the true value are equally likely, with zero probability beyond it. Many problems can be formulated in $L_\infty$ and give a single solution, an advantage over $L_2$ optimization. However, the $L_1$ norm, which essentially measures the median of noise and is more robust to outliers than $L_2$ [9] or $L_\infty$, is a more intuitive measurement. Unfortunately, it is not the ML estimate under Gaussian noise like $L_2$. Despite this shortcoming, it has desirable convergence properties and we have been surprised that $L_1$ was not investigated much for triangulation.

Recker *et al*. [16] proposed an $L_1$ triangulation cost function based on an angular error measure for a candidate 3D position, $p$, with respect to a feature track $t$. As input, a set of feature tracks across $N$ images and their respective $3 \times 4$ camera projection matrices $P_i$ are known. The error for a candidate 3D position $p$ is computed as follows. For a camera center $C_i$, a unit direction vector $v_i$ is first computed between it and the candidate position $p$. A second unit vector, $w_{ti}$, is computed as the ray from each $C_i$ through the 2D feature track $t$ in each image plane. Since $t$ generally does not coincide with the projection of $p$ in each image plane, there is frequently a non-zero angle between each possible $v_i$ and $w_{ti}$. Finally, the average of the dot products $v_i \cdot w_{ti}$ across all cameras is obtained. Each dot product can vary from $[-1, 1]$, but only points that lie in front of the cameras are taken into account, corresponding to the range $[0, 1]$. We use the same nomenclature as in Recker *et al*. [16] to define the cost function mathematically. Given $C$ the set of all cameras, $T$ the set of all feature tracks, and $p = (X, Y, Z)$ a 3D evaluation position, the cost function for $p$ with respect to a track $t \in T$ is displayed in Eq. 1. Here, $I = \{C_i \in C | t$ "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 $i$. This cost function has its lowest possible value at zero. Gradient values along the $X$, $Y$ and $Z$ directions are defined in Eqs. $5 - 7$ of Recker *et al*. [16].

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

It is important to determine the convexity properties of this function for an arbitrary number of cameras. A sum of convex functions is convex, but the same does not apply for quasi-convex functions, such as dot products. In order to verify convexity, the most common approach is to perform an analysis of the function's Hessian. However, this analysis becomes intractable for even small amounts of cameras. Instead, we take a different and much more simplified approach. Fig. 1(c) shows a scalar field, consisting of the $L_1$ cost function measured for a dense set of test positions near a known ground-truth position. The scalar field shows a very smooth variation in a large vicinity surrounding this position. This is key since there is a high chance of convergence to the global optimum even from large distances. Such scalar field renderings are not as mathematically rigorous as a direct convexity analysis, but in all of our testing we have not detected local minima in the near vicinity of the global minimum.

We chose Recker *et al*.'s cost function as a basis for our triangulator due to these attractive convergence properties. There are two important differences in the way the cost function is applied. Statistically-meaningful samples of rays are not used. By using sampling, the solution computed using our algorithm would be biased towards a particular subset of the cameras. Also, gradient descent optimization is replaced by *non-linear conjugate gradients*, since better results were obtained in practice. Fletcher and Reeves introduced the method of non-linear conjugate gradients was for locating an unconstrained local minimum of a non-linear function of several variables [5]. Since the introduction of this method, many additional modifications have been published [6], but we used their method, which works well due to the small number of parameters for optimization. As for the convergence criterion, we found that testing the magnitude of the gradient at each iteration and terminating when it was close to zero ($10^{-15}$) works well in practice.

### 3.1. Initialization

The key step in our algorithm is initialization, or where to start the swarm optimization. We initialize based on a procedure that is much more robust and exhaustive than the simple midpoint start of Recker *et al*. [16], which also uses an undesired fixed threshold. First, the midpoint algorithm is used to compute an initial point between every possible camera pair. Each of these points is then optimized by Recker *et al*.'s cost function, using non-linear conjugate gradients. The complexity of this step is $O(N^2)$ for $N$ cameras. Experiments have determined that for traditional feature track lengths and reasonable noise, our algorithm has linear performance. However, in the worst case the behavior is quadratic. The set of optimized midpoints (OMs) forms a bounding box, the green box depicted in Figs. 1(a,b). This box is the largest axis-aligned box that encloses all the OMs. We then compute an initial reference for swarm optimization inside this box. To compute this initial point, we must take into account the influence of outlier OMs. We use statistics on the OMs to compute an initial point inside the bounding box, defined as the centroid of all the OMs. Finally, this point is optimized. The main contri-

bution of our work with respect to Recker *et al*. is finding n very accurate position based on the initialization. For this, an exhaustive search based on swarm optimization is now performed, as discussed in Section 3.2.

### 3.2. Swarm optimization applied to triangulation

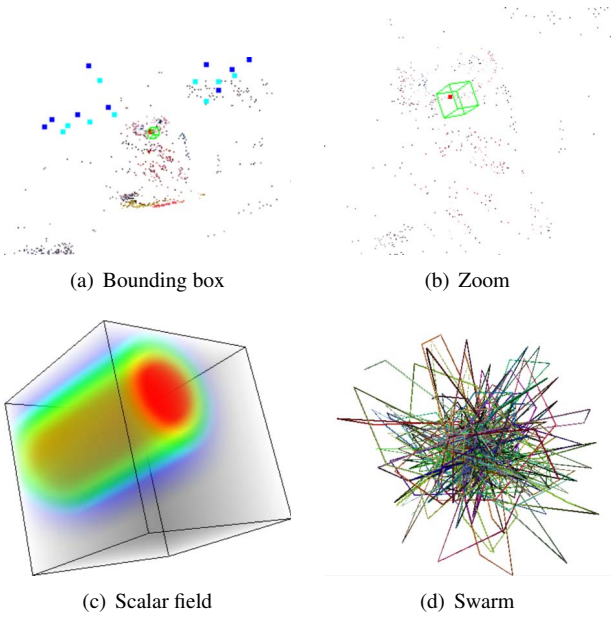
To understand the next step in the algorithm, a brief overview of particle swarm optimization is provided. Particle swarm optimization (PSO) is a stochastic approach for solving both continuous and discrete optimization problems [15]. In PSO, entities known as *particles*, or starting points, move in the search space of an optimization problem, where the position of a given particle represents a candidate solution. At each time step, each particle updates its position searching for a better solution, by changing its velocity according to rules originally inspired by behavioral models of bird flocking. Each particle's movement is a function of both the direction it has taken towards its best local solution and also the best global solution found so far by one or more particles in its vicinity, with some random perturbations. More specifically, the next iteration occurs after all particles have been moved and eventually, the swarm as a whole is likely moving closer towards the cost function optimum. There are many methods which define how to initialize the swarm and how to move each member in the search space [15].

Each particle consists of three $D$-dimensional vectors [15], where $D$ is the dimensionality of the search space: current position $x_i$, the previous best position $p_i$, and velocity $v_i$. In each iteration, the current position is evaluated as the solution. In our case, an evaluation of the $L_1$ cost function [16] is performed. If it is better than any previous result, $x_i$ is stored in $p_i$. The value of the best function result so far, $p_{best}$, is stored as well. As the algorithm keeps iterating, the goal is to keep finding better positions and updating $p_i$ and $p_{best}$. The values for $v_i$ are adjusted, and can be interpreted as step sizes for each particle. Finally, iterations end when some criterion is met, such as a sufficiently good fitness or a maximum number of iterations. In our implementation, 150 iterations were always used with great results, though future work includes adaptively estimating this value.

The swarm optimization process as described above is summarized in Fig. 1. We utilize 64 particles, since this has been proven to be a proper amount for our parameter space [15]. Notice in Fig. 1(d) that particles can end up far outside the box due to the random velocity initialization, but through communication with other particles march back into the box and converge at the global optimum.

## 4. Results

The proposed triangulator was tested exhaustively for its accuracy, processing time and general behavior, on both real and synthetic data. All implementation was done using



(a) Bounding box (b) Zoom (c) Scalar field (d) Swarm

Figure 1: Swarm optimization applied for a point of the *ET* dataset [18]. (a) Multi-view reconstruction, with cameras in dark blue. (b) Zoom of (a). (c) A volume view of a scalar field representing an $L_1$ angular cost function [16] evaluated at a dense grid of sample points inside the bounding box, with redder values closer to zero cost. (d) Particle path evolution in swarm optimization.

*C++*, 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 *Eigen* library (`http://eigen.tuxfamily.org`) was used.

### 4.1. Synthetic tests

Synthetic tests were performed to analyze the triangulator's performance in agreeing with ground-truth positions, given different levels of feature track noise and with varying track lengths. Processing time and reprojection error were the parameters of interest. It was not possible to compare performance directly with optimal two-view methods [8, 11] since in those methods tracks are modified. A direct comparison with optimal three-view triangulation is discussed in Section 4.1.1.

To this end, tests were performed on four different types of camera configurations, as displayed in Fig. 2. The first represents a set of cameras positioned in a circle above the scene. The second is similar to the first, except is uses only a semi-circle of cameras. The third involves a set of cameras in a line above the scene. The fourth involves a set of cameras that were placed randomly, representing an unstructured collection of images. In each case, all cameras

(a) Circle (b) Semi-circle (c) Line (d) Random

Figure 2: Synthetic camera configurations. Ground-truth positions are shown in black, with a selected one in red, the bounding box in green, and cameras in blue.

are looking towards the origin, $(0, 0, 0)$, of the scene.

The first experiment consisted of analyzing the triangulator under various levels of noise and with varying feature track lengths. The same camera intrinsics were used in all tests, with an image size of $1024 \times 1024$ and an identity camera calibration matrix (unity focal length). A total of $8316$ test datasets were created, each consisting of $100$ synthetic points with known ground-truth tracks. For each test configuration, $100$ points were randomly generated from the world origin, at different distances up to a radius of $0.4$ world units. It was ensured that none of the cameras were within the visual hull of the points. All $100$ points were triangulated for each set, and statistics were obtained. The individual datasets were generated for each of the four main types of camera setups. The number of cameras evaluated ranged from $2 - 100$, in increments of one. For each camera amount, feature tracking errors were simulated by adding image plane noise to ground-truth feature tracks, in random directions, up to $M\%$ of the image plane diagonal dimension. The value of $M$ was varied from integers $0\%$ to $20\%$. As for evaluation of results, a computed point was considered 'correct' when the median of the computed reprojection error was lower than the median of ground-truth reprojection error. Based on this criteria, over $99.9\%$ of all evaluated points were correct. The few failures were contained within an average of $5 \times 10^{-4}$ of the ground-truth position. These were mainly limited to unlikely cases where inaccurate midpoints from near-parallel cameras skewed bounding box computation.

Results for feature tracking error versus processing time are shown in Fig. 3, for feature tracks spanning $100$ cameras. In this simulation, it can be seen that a very accurate solution can be obtained even in the presence of high amounts of track noise. Surprisingly, processing time actually stabilizes at a near constant value after about $1\%$ image plane noise. Feature track length versus processing time is shown in Fig. 4, for $0\%$ and $1\%$ image plane noise. It shows that processing time is linear as the feature track length is increased, for lower errors. Keep in mind that a $1\%$ error is about $10$ pixels, which is a high tracking error in practice. For higher errors, behavior tends to be quadratic. As the number of cameras is increased, the time it takes to apply non-linear conjugate gradients on a number of quadratically-increasing midpoint pairs begins to dominate with respect to the time it takes for swarm optimization. For typical track lengths of $10$ and $20$ cameras, times are lower than $10$ms per point despite noise. For longer tracks and with noise, triangulating one point can take close to $100$ms. This is an excellent result which can be possibly sped up further through parallelization. In constrast, the polynomial method would be intractable for long tracks. In comparison with optimal $N$-view methods, for example the fractional programming method by Agarwal *et al*. [1] claims a near linear runtime with number of cameras. Results for triangulation were demonstrated on 3-6 cameras. The only indicative result for larger numbers is for camera resectioning, where processing times are approximately $40$-$700$ seconds on $6$-$100$ cameras using a $3GHz$ computer. This indicates an expensive branch and bounds framework. Based on these experiments, our triangulation may possibly be two to three orders of magnitude faster. With such speed and accuracy in near $100\%$ of cases, our method opens the door to accurate reconstructions even with very long feature tracks. Reconstruction from massive amounts of images may become more commonplace in the future due to the ubiquity of images, especially from mobile devices. In such cases, error propagation of mismatched tracks has a greater impact on accuracy, but as shown, very precise solutions can be obtained even under high amounts of noise.

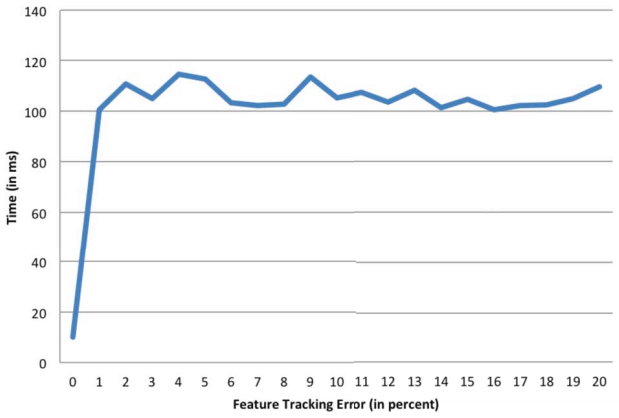



Figure 3: Feature tracking error vs. computation time (ms). Processing time stabilizes after $1\%$ error.

#### 4.1.1 Comparison with three-view triangulation

Byröd's optimal polynomial three-view method [2] evaluates $10$ points with exact ground-truth input. Our triangulator agrees with the ground-truth values up to at least $8$ decimal places, except in the third case, where the point actually lies behind the cameras. In our extensive

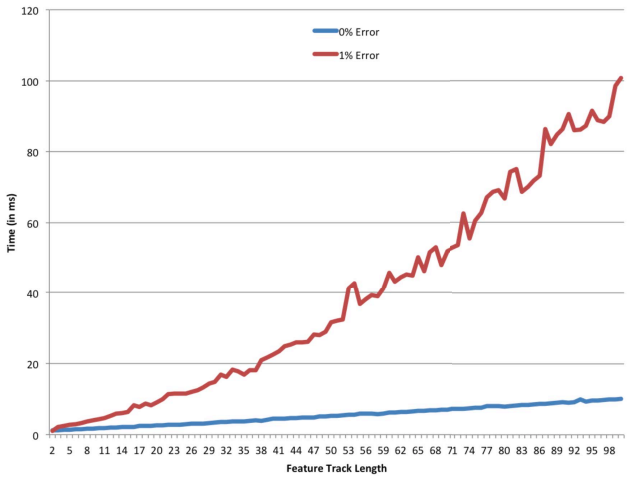



Figure 4: Feature track length (number of cameras) vs. computation time (ms), for $0\%$ and $1\%$ image-plane feature tracking error. Behavior is nearly linear for small errors.

testing, this is the only documented case of a large failure in the final result. This is due to the fact that our algorithm is designed to work only for points that meet the cheirality constraint. Polynomial methods can identify such points, but in practice this solution is not geometrically meaningful. As for processing time, Stewénius *et al*. [19] took 20 hours to triangulate the 2683 three-view points of the *Dinosaur* dataset [14] on 128-bit arithmetic, and Byröd *et al*. took 2.5 minutes, but our triangulator takes less than 6 seconds on a conventional laptop for all 4983 full tracks.

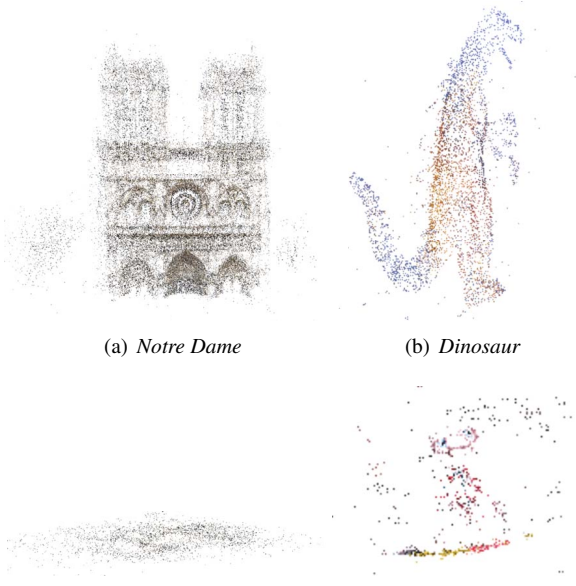
### 4.2. Evaluation on real datasets

For real scenes, processing time and reprojection error were evaluated, as displayed in Table 1. Results were excellent even in *Palmdale*, where the images present a strong radial distortion. Even though processing time is significantly slower with respect to Recker *et al*. [16], there were confirmed cases where positions computed with their algorithm were not optimal and were fixed by our triangulator. The added computational expense in our algorithm is necessary in order to claim accuracy and robustness. For example, in the *Brown06* [17] dataset the average reprojection error for Recker *et al*. was 1.453 and for our algorithm it was 1.322. This shows that certain points in their gradient descent lead to imprecise solutions, which our swarm optimization detected and corrected. Recker *et al*. outperformed bundle adjustment's final reprojection error in their experiments, so ours surpassing Recker *et al*. is very encouraging. Finally, some triangulations, generated by our algorithm, are shown in Fig. 5.

In summary, we present a very accurate triangulation

| Data set | $t_o$ | $t_f$ | $\epsilon_o$ | $\epsilon_f$ | $C$ |
|---|---|---|---|---|---|
| *Brown06* [17] | 10046 | 78 | 1.3224 | 1.4295 | 220 |
| *Brown12* [17] | 59727 | 172 | 1.3926 | 2.0891 | 337 |
| *castle19* [20] | 604 | 5 | 1.1985 | 1.1985 | 14 |
| *castle30* [20] | 819 | 7 | 1.3531 | 1.3531 | 19 |
| *Dinosaur* [14] | 4825 | 37 | 0.4677 | 0.4677 | 36 |
| *ET* [18] | 1017 | 7 | 0.4696 | 0.4696 | 9 |
| *fountain* [20] | 427 | 3 | 1.1610 | 1.1610 | 10 |
| *herzjesu* [20] | 864 | 7 | 1.2958 | 1.2958 | 22 |
| *Kermit* [18] | 674 | 4 | 0.4035 | 0.4035 | 11 |
| *Canyon* [16] | 2357 | 18 | 1.3818 | 1.3818 | 90 |
| *Palmdale* [16] | 163 | 3 | 4.2965 | 4.2965 | 66 |
| *Stockton* [16] | 2832 | 41 | 2.2052 | 4.4229 | 10 |
| *Campus* | 137 | 1 | 1.0000 | 1.0000 | 7 |
| *Walnut* [16] | 200 | 1 | 1.1240 | 1.1240 | 7 |

Table 1: Processing times $t_o$ and $t_f$ (ms) and average reprojection errors $\epsilon_o$ and $\epsilon_f$ (pixels), respectively for our method and fast triangulation [16], with number of cameras $C$, for real datasets. Due to the high accuracy of Recker et. al's method [16], in most datasets our technique only needs to fix a few of the final solutions, except in the Brown and Stockton datasets.



(a) *Notre Dame* (b) *Dinosaur*

(c) *Brown12* (d) *ET*

Figure 5: Scenes reconstructed with our triangulator. The triangulation of *Notre Dame* [18] in (a) took 10.34 minutes, with 73427 tracks spanning 290 cameras.

framework that presents excellent results on real and synthetic scenes. To our knowledge, this is the first near optimal L1-based triangulator, with excellent results for ar-

bitrary numbers of cameras and noise. However, there is room for improvement in the algorithm. Bounding box size is the main determinant of performance versus processing time. If greater speed is desired, without compromising accuracy, then a more efficient bounding box computation is required. We hope our work can be a contribution towards applications where very precise triangulation results are required.

## 5. Conclusions

This paper presents a framework for very accurate triangulation in multi-view scene reconstruction. This is the first very accurate N-view triangulation algorithm (optimal in 99.9% of cases) that has been proven succesful across arbitrary camera configurations, numbers of cameras and under noise. Though not theoretically optimal, convergence to the global optimum was achieved over 99.9% of the time over thousands of synthetic multi-view configurations with varying degrees of feature tracking noise. First, a bounding box is robustly computed based on an angular error-based $L_1$ cost function applied to initial pairwise optimized midpoint positions. An exhaustive search procedure based on particle swarm optimization is applied to obtain the final position inside the bounding box. Opposed to existing polynomial methods developed for a small number of cameras, the proposed algorithm is at best linear in the number of cameras and does not suffer from inaccuracies inherent in solving high-order polynomials or Gröbner bases. With real data, reprojection error is proven to be optimized with respect to other triangulation methods.

### Acknowledgement

This work was supported in part by Lawrence Livermore National Laboratory and the National Nuclear Security Agency through Contract No. DE-FG52-09NA29355. The authors thank their colleagues in the Institute for Data Analysis and Visualization (IDAV) at UC Davis for their support.

## 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, 3, 6

[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. 1, 3, 6

[3] Changchang Wu. VisualSfM: A visual structure from motion system. http://homes.cs.washington.edu/~ccwu/vsfm/, 2011. 1

[4] Z. Dai, Y. Wu, F. Zhang, and H. Wang. A novel fast method for $L_\infty$ problems in multiview geometry. In *Proceedings of the 12th European conference on Computer Vision - Volume Part V*, ECCV'12, pages 116–129, Berlin, Heidelberg, 2012. Springer-Verlag. 2, 3

[5] R. Fletcher and C. Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964. 4

[6] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006. 4

[7] R. Hartley and F. Kahl. Optimal algorithms in multiview geometry. In *Proceedings of the 8th Asian conference on Computer vision - Volume Part I*, ACCV'07, pages 13–34, Berlin, Heidelberg, 2007. Springer-Verlag. 2, 3, 4

[8] R. I. Hartley and P. Sturm. Triangulation. *Comput. Vis. Image Underst.*, 68(2):146–157, 1997. 1, 3, 5

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004. 1, 2, 4

[10] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-Sturm vs. optimal correction. In *Proceedings of the British Machine Vision Conference*, pages 18.1–18.10. BMVA Press, 2008. doi:10.5244/C.22.18. 1

[11] P. Lindstrom. Triangulation Made Easy. In *CVPR*, pages 1554–1561, 2010. 1, 3, 5

[12] 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. 2

[13] Y. Min. L-infinity norm minimization in the multiview triangulation. In *Proceedings of the 2010 international conference on Artificial intelligence and computational intelligence: Part I*, AICI'10, pages 488–494, Berlin, Heidelberg, 2010. Springer-Verlag. 2, 3

[14] Oxford Visual Geometry Group. Multi-view and Oxford Colleges building reconstruction. http://www.robots.ox.ac.uk/~vgg/, August 2009. 7

[15] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007. 5

[16] S. Recker, M. Hess-Flores, and K. I. Joy. Statistical angular error-based triangulation for efficient and accurate multiview scene reconstruction. In *Workshop on the Applications of Computer Vision (WACV)*, 2013. 2, 3, 4, 5, 7

[17] M. I. Restrepo, B. A. Mayer, A. O. Ulusoy, and J. L. Mundy. Characterization of 3-d volumetric probabilistic scenes for object recognition. *IEEE Journal of Selected Topics in Signal Processing*, 6:522–537, 09/2012 2012. 7

[18] 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. 1, 5, 7

[19] H. Stewénius, F. Schaffalitzky, and D. Nistér. How hard is 3-view triangulation really? *Computer Vision, IEEE International Conference on*, 1:686–693, 2005. 1, 3, 7

[20] 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. 2, 7