Depth Data Assisted Structure-from-Motion Parameter Optimization and Feature Track Correction

Shawn Recker*†, Christiaan Gribble†, Mikhail M. Shashkov*, Mario Yepez*, Mauricio Hess-Flores*, and Kenneth I. Joy*
*Institute of Data Analysis and Visualization
University of California Davis
Davis, California
Email: http://idav.ucdavis.edu/people/
†Applied Technology Operation
SURVICE Engineering
Belcamp, Maryland
Email: {shawn.recker, christiaan.gribble}@survice.com

***Abstract*—Structure-from-Motion (SfM) applications attempt to reconstruct the three-dimensional (3D) geometry of an underlying scene from a collection of images, taken from various camera viewpoints. Traditional optimization techniques in SfM, which compute and refine camera poses and 3D structure, rely only on feature tracks, or sets of corresponding pixels, generated from color (RGB) images. With the abundance of reliable depth sensor information, these optimization procedures can be augmented to increase the accuracy of reconstruction. This paper presents a general cost function, which evaluates the quality of a reconstruction based upon a previously established angular cost function and depth data estimates. The cost function takes into account two error measures: first, the angular error between each computed 3D scene point and its corresponding feature track location, and second, the difference between the sensor depth value and its computed estimate. A bundle adjustment parameter optimization is implemented using the proposed cost function and evaluated for accuracy and performance. As opposed to traditional bundle adjustment, in the event of feature tracking errors, a corrective routine is also present to detect and correct inaccurate feature tracks. The filtering algorithm involves clustering depth estimates of the same scene point and observing the difference between the depth point estimates and the triangulated 3D point. Results on both real and synthetic data are presented and show that reconstruction accuracy is improved.**

I. Introduction

Many modern applications require the reconstruction of three-dimensional (3D) geometry from imagery of some object or scene from different viewpoints. Often, a point cloud representing the 3D structure is produced. This process requires a number of stages, such as feature tracking, frame decimation, self-calibration, camera pose estimation, structure computation, and parameter optimization. State-of-the-art applications [1], [2] are already capable of producing accurate large scale reconstructions, but increasing the accuracy and performance of these systems is an ongoing problem in the reconstruction community.

A variety of algorithms for multi-view reconstruction, also known as structure-from-motion (SfM), are available, but the great majority contain a final non-linear optimization stage for all computed parameters, known as *bundle adjustment* [3], which has been proven to be necessary but also very expensive. There are a number of issues with this approach. First, traditional SfM techniques are usually based on computing *feature tracks*, or sets of corresponding pixels, across a set of RGB images of a given scene. Although these tracks may be noisy, they are then used to compute and refine the parameters of the camera(s) viewing the scene, along with its 3D structure. Finally, bundle adjustment uses these potentially noisy feature tracks as ‘ground truth’ to perform camera parameter and structure optimization. Given the noisy estimates, it is possible for the optimization to converge to a local instead of global minimum, providing incorrect final parameter estimates. Furthermore, the quantity which is typically minimized, the *reprojection error* [4] between the computed scene structure and corresponding feature tracks, is the best estimate of error in the absence of ground-truth information, but its minimization does not ensure that a physically-correct set of parameters is obtained in a ground-truth sense.

At the same time, RGB-D cameras, also commonly known as depth cameras, have become very popular during the past few years, particularly with the advent of the *Microsoft Kinect* [5]. The reliable (though possibly noisy) depth sensor information provided by such cameras introduces more information about the scene than measurements acquired solely with RGB cameras, such that multi-view reconstruction should benefit from the extra information. Optimization procedures can be augmented to increase the accuracy of the reconstruction, which is the approach we investigate in this work.

Given the issues that arise with traditional bundle adjustment, and the recent ubiquity of depth sensors such as the Microsoft Kinect, this paper presents a general cost function, which evaluates the quality of a reconstruction based upon a previously established angular cost function and includes depth data estimates. The cost function takes into account two error measures: first, the angular error between each computed

3D scene point and its corresponding feature track location, and second, the difference between the sensor depth value and its computed estimate. A bundle adjustment parameter optimization is implemented using the proposed cost function and evaluated for accuracy and performance. As opposed to traditional bundle adjustment, in the event of feature tracking errors, a corrective routine is also present to detect and correct inaccurate feature tracks. The algorithm involves clustering depth estimates of the same scene point and observing the difference between the depth point estimates and the triangulated 3D point. Results on both real and synthetic data are presented and show that reconstruction accuracy is improved. Related Work is presented in Section II, followed by the algorithms in Section III. Experimental results are presented in Section IV, conclusions in Section V, and future work in Section VI.

II. Background

There exist many algorithms for multi-view reconstruction, but the following sequential stages are common to most systems. Typically, feature matches (between consecutive pairwise views) and tracks (concatenating across multiple views) are generated. Such tracking can be sparse [6], [7] or dense [8] and consists of computing and linking the pixel coordinates in all images for each scene point, whenever it is visible. Frame decimation [9] is often applied at this point, particularly for sequential image sets, to remove images with very small or very large baselines. A baseline is the relative separation of images in world space. Small baselines lead to bad numerical conditioning in pose and structure estimation, whereas large baselines introduce problems in feature tracking. Next, camera intrinsic parameters are estimated through a process called self-calibration [4]. In most cases, much of this information is already known. Also, epipolar geometry can be estimated from pairs or triplets of views. Epipolar geometry encapsulates the intrinsic projective geometry between groups of views and is used in the process of determining camera pose (position and orientation). Only relative positions and orientations can be obtained between views. Once camera parameters are estimated, computation of scene structure is achieved through triangulation methods, such as linear triangulation [4]. In this method, the 3D position of a scene point, given a set of cameras and pixel feature track positions corresponding to the point, is computed as the best-fit intersection position in space for the set of rays from each camera center and through the feature track positions. Finally, because errors in all of the above steps influence accuracy of the computed structure, *bundle adjustment* [3] is performed to optimize some, or even all, of the camera and structure parameters. This section will describe the process of bundle adjustment in-depth and highlight recent techniques involving RGB-D (depth) cameras to provide context for the proposed bundle adjustment + depth algorithm.

A. Bundle Adjustment

The result of pose estimation and triangulation are respectively the projection matrices P and P' and a set of 3D points, one corresponding to each feature track. If all estimates were perfect, a set of rays starting from each camera center would go through each corresponding pixel in each image plane, finally intersecting at an exact 3D position is space. Since in general this situation will not occur, the objective of *bundle adjustment* is to adjust these rays in such a way that the 'total reprojection error' of the 3D points with respect to their corresponding 2D feature tracks in each camera is minimized. The end result of this minimization is a change in both the positions of the original 3D points as well as in the cameras' projection matrices $P_1.....P_M$, where intrinsic and radial distortion parameters may be allowed to vary in the minimization along with the pose parameters, which are typically optimized. The cost function which is traditionally minimized can be expressed as the sum of squares of the reprojection error between each 3D point and the feature matches which yielded it, as shown in Eq. 1 for the general case of N 3D points seen in M cameras.

$$min(a_i, b_j) \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} (d(Q(a_i, b_j), x_{ij}))^2 \tag{1}$$

Reprojection error is a non-linear, real-valued function. The number of terms in the sum can potentially be very large. Here, x_{ij} is the position of the i_{th} feature on image j. The binary variable v_{ij} equals '1' if point i is visible in image j ('0' otherwise). The vectors a_j and b_i parameterize each camera j and 3D point i, respectively, with $Q(a_j, b_i)$ as the reprojection of point i on image j. Finally, d is the Euclidean distance in each image between each original correspondence and its associated reprojection. This minimization involves a total of $3N + 11M$ parameters, and can be achieved using the *Levenberg-Marquardt* (LM) algorithm [3].

The minimization is achieved using non-linear least-squares algorithms, from which LM has proven to be one of the most successful, due mainly to its use of an effective damping strategy that lends it the ability to converge quickly from a wide range of initial guesses. By iteratively linearizing the function to be minimized in the neighborhood of the current estimate, the LM algorithm involves the solution of linear systems known as the 'normal equations'. The solution of such linear systems determines an increment to the current estimate. In the particular case of bundle adjustment, these equations have a sparse block structure due to the lack of interaction between the parameters.

This sparse block structure can be exploited to greatly speed up the algorithm, which is inherently time-consuming and computationally expensive from the minimization involving perhaps millions of parameters.

A sparse bundle adjustment C/C++ package known as *SBA*, written by Lourakis and Argyros [3], is widely used to implement bundle adjustment given initial structure and pose estimates. In this sparse variant of the LM algorithm, the zeros pattern is explicitly taken into account to avoid storing and operating on such elements.

Weighted Bundle Adjustment. As mentioned, the Levenberg-Marquardt algorithm is based on solving the normal equations at each iteration. In *weighted* bundle adjustment, each input feature is weighted differently with the objective of improving convergence by giving less weight to those features that are more likely to be inaccurate. Such weights are implemented as covariances. The normal equations have the form shown in Eq. 2, but when using weighted bundle adjustment, the equations change to the form shown in Eq. 3, where Σ corresponds to a block-diagonal matrix consisting of 2×2 covariance matrices for each input feature, J is the parameter Jacobian matrix, δ_p the parameter update step, μ the damping term and ϵ the error vector.

$$(J^T J + \mu I)\delta_p = J^T \epsilon \tag{2}$$

$$(J^T \Sigma_x^{-1} J + \mu I)\delta_p = J^T \Sigma_x^{-1} \epsilon \tag{3}$$

B. RGB-D Depth Cameras and Algorithms

RGB-D cameras, also commonly known as depth cameras, have become very popular during the past few years, particularly with the advent of the *Microsoft Kinect*. This device has brought quality, low-cost and real-time depth sensing to the masses, including researchers and enthusiasts. The Kinect uses structured light to generate real-time depth maps containing discrete range measurements of the physical scene [10]. This data can be reprojected as a set of discrete 3D points (or point cloud). The main issue with the acquired depth data is noise. Depth measurements often fluctuate and depth maps contain 'holes' where no readings were obtained. To generate 3D models for use in applications such as gaming, physics, or CAD, higher-level surface geometry needs to be inferred from the noisy data [10].

Perhaps the most popular application of the Kinect, and of RGB-D cameras overall, has been the highly-successful *Kinect Fusion* algorithm by Izadi et al. [10], which enables a user holding and moving a standard Kinect camera to rapidly create detailed 3D reconstructions of an indoor scene. Only the depth data from Kinect is used to track the 3D pose of the sensor and reconstruct a single and accurate 3D model of the physical scene in real-time, such that a user can move the Kinect within any indoor space and reconstruct a 3D model of the scene in seconds. The system continuously tracks the 6 degrees-of-freedom (DOF) pose of the camera and fuses new viewpoints of the scene into a global surface-based representation. A novel GPU pipeline allows for accurate camera tracking and surface reconstruction at real-time rates.

One issue with the original Kinect Fusion algorithm is that it is expensive in memory when constructing and updating the global model. To this end, there are a number of recent algorithms dealing with efficient model creation. Quiroga et al. [11] propose a scene flow approach that exploits the local and piecewise rigidity of real world scenes. By modeling the motion as a field of twists, the method encourages piece-wise smooth solutions of rigid body motions. A general formulation is provided to solve for local and global rigid motions by jointly using intensity and depth data. Another approach to computing dense scene flow between a pair of consecutive RGB-D frames is presented by Hornacek et al. [12]. The availability of depth data is exploited by seeking correspondences with respect to patches specified not as the pixels inside square windows, but as the 3D points that are the inliers of spheres in world space. Steinbrücker et al. [13] propose a method to generate highly detailed, textured 3D models of large environments from RGB-D sequences. The system runs in real-time on a standard desktop PC. To reduce memory consumption, the acquired depth maps and colors are fused in a multi-scale octree representation of a signed distance function. To estimate the camera poses, a pose graph is constructed and dense image alignment is used to determine the relative pose between pairs of frames. Thomas and Sugimoto [14] describe a new 3D scene representation using a set of planes that is cheap in memory use and, nevertheless, achieves accurate reconstruction of indoor scenes from RGB-D image sequences. Projecting the scene onto different planes reduces significantly the size of the scene representation and allows generation of a global textured 3D model with lower memory requirements while keeping accuracy and easiness to update with live RGB-D measurements. Raposo et al. [15] present a novel approach for estimating the relative motion between successive RGB-D frames using plane-primitives instead of point features. The planes in the scene are extracted and the motion estimation is cast as a plane-to-plane registration problem with a closed-form solution. The algorithm by Bylow et al. [16] can also reconstruct large scale 3D scenes despite many planar surfaces.

Besides the seminal work of Izadi et al. [10] and others on dense modeling, a great number of recent works explore different applications of depth cameras. To give some concrete examples, Stückler and Behnke [17] propose an expectation-maximization (EM) framework for dense 3D segmentation of moving rigid parts in RGB-D video, which segments two images into pixel regions that undergo coherent 3D rigid-body motion. Boom et al. [18] use the intensity image and depth information from the RGB-D camera to estimate the point light source position in a scene. Assuming the Lambertian reflectance model, the RGB-D camera provides the image and the surface normals, and the remaining unknowns are the albedo and light parameters (light intensity and direction). The algorithm of Saygili et al. [19] provides dense depth estimations of transparent objects and specular surfaces with high accuracy. A fully-connected CRF based hybrid refinement algorithm is proposed, incorporating stereo cues from cross-modal stereo between IR and RGB cameras of the Kinect and Kinect's depth map. The work by Zeisl et al. [20] addresses the problem of wide-baseline registration of RGB-D data. They utilize the principle of salient directions present in the geometry and propose to extract (several) directions from the distribution of surface normals or other cues such as observable symmetries. For geometric pose estimation from tentative matches, a fast and robust two-point sample consensus scheme integrating an early rejection phase is proposed. Finally, Song and Xiao [21] provide a unified benchmark dataset of 100 RGB-D videos with high diversity. Additionally, different kinds of RGB-D tracking algorithms using 2D or 3D models are proposed, and a quantitative comparison of various algorithms with RGB or RGB-D input are presented.

Despite the great number of recent algorithms which make use of depth cameras, very few algorithms have dealt with the incorporation of depth data into bundle adjustment optimization [3] in scene reconstruction, which is the main topic of this paper. The work by Afzal et al. [22] is, to the best of our knowledge, one of the only algorithms closely related to our work, but differs fundamentally on its use of the Iterative Closest Point (ICP) algorithm to help guide the optimization. They propose *BAICP+*, which combines the bundle adjustment and ICP algorithms to take into account both 2D visual and 3D shape information in one minimization formulation to estimate relative pose parameters of each camera.

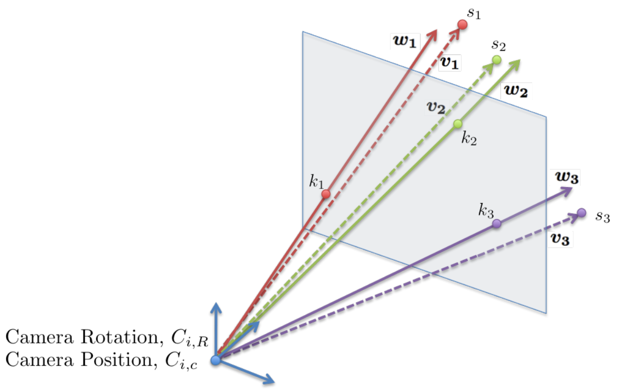


Fig. 1: The cost function takes into account two error metrics. The first is the angular difference of the unit vectors $\hat{\vec{v}}$ and $\hat{\vec{w}}$, where $\hat{\vec{v}}$ is the unit vector from the camera center to the 3D scene point s_i and $\hat{\vec{w}}$ is the unit vector from the camera center that passes through the feature track location k_i for s_i. This portion of the cost function has a global minimum of zero, which implies the camera parameters and 3D structure match perfectly. The second portion is a difference between the sensor depth value, $D(k_i)$, and its computed estimate $||\vec{v}||$, the distance between the camera center and 3D structure point. Similarly, the second portion has a global minimum at zero.

III. Methodology

Two contributions to SfM applications are presented in this paper. The first is a general cost function for parameter optimization. The cost function evaluates the quality of a reconstruction based upon an angular error metric, presented by Recker et al. [23], and depth data estimates. The second, as opposed to traditional bundle adjustment, in the event of feature tracking errors, is a corrective routine to detect and correct inaccurate feature tracks, based upon depth data as opposed to traditional epipolar constraints.

A. Cost Function

The cost function takes into account two error measures: first, the angular error between each computed 3D scene point and its corresponding feature track location [23], and second, the difference between the sensor depth value and its computed estimate. Figure 1 contains a visual depiction of the cost function. Mathematically, the cost function can be written as the sum of two error terms, $E_{angular}$ and E_{depth}, as in Eq. 4.

$$E(C_i, s_j) = E_{angular}(C_i, s_j)^2 + E_{depth}(C_i, s_j)^2 \tag{4}$$

Here C_i is a single camera (and its associated data) and s_j is a 3D scene point. $E_{angular}$ is the angular error used by Recker et al. [23] and is rewritten in Eq. 5 in its singular form (for one camera and scene point).

$$E_{angular}(C_i, s_j) = \qquad 1.0 - \hat{\vec{v}} \cdot \hat{\vec{w}} \tag{5a}$$

$$E_{angular}(C_i, s_j) = \quad 1.0 - (s_j \,\hat{-}\, C_{i,c}) \cdot (C_{i,P+}\hat{k_j} - C_{i,c}) \tag{5b}$$

Note that $C_{i,c}$ is the camera center of projection (camera position), $C_{i,P+}$ is the right pseudo-inverse of the camera projection matrix, and k_j is the feature track location of s_j in C_i. Also, note that $\vec{v}$ and $\vec{w}$ are normalized to have unit length, as in the original formulation [23].

E_{depth} measures the difference between the sensor depth value and its computed estimate. The equation is presented in Eq. 6.

$$E_{depth}(C_i, s_j) = ||(s_j - C_{i,c})|| - D(k_j) \tag{6}$$

Here, $D(k_j)$ is the depth sensor value at the feature track location for s_j in camera C_i. In the current form, Eq. 4 only evaluates a single camera and 3D scene point. To evaluate an entire reconstruction, the cost function can be extended to its final form in Eq. 7.

$$E(C, P) = \sum_{C_i \in C} \sum_{s_j \in P_{C_i}} E(C_i, s_j) \tag{7}$$

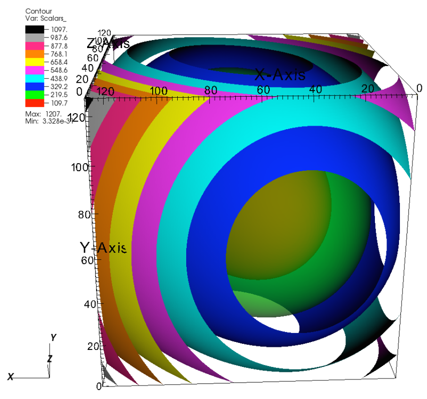
In this equation, C is the set of all cameras and P_{C_i} is the set of all visible scene structure in camera C_i.

B. Cost Function Analysis

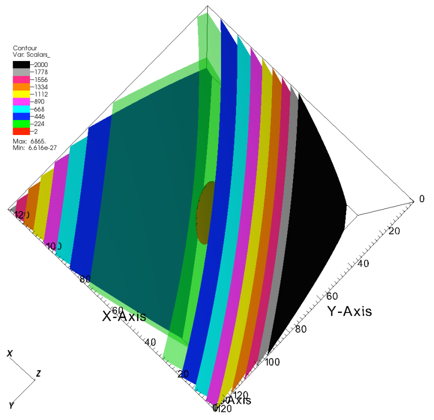
As with any cost function, examining its properties and behavior is important for determining appropriate use. One of the most important properties to determine is convexity. Proving a function is convex implies that there exists a single, global optimum that can be easily obtained using standard convex optimization algorithms. However, a convexity analysis of the function's Hessian is not tractable due to the dimensionality of the problem.

Despite these problems, a simple yet effective approach to obtaining insight into a function's topology is to perform a scalar field analysis similar to that employed by Recker et al. [23], as shown in Fig. 2. In order to obtain the fields, Eq. 7 was densely sampled holding certain parameters fixed while varying others. The renderings show the change in function value when one of the parameters is changed. In Fig. 2a, a single 3D scene position was varied given fixed camera parameters. In Fig. 2b, a single camera position was varied while holding the rotation and 3D scene structure fixed, and in Fig. 2c, the camera rotation was varied while holding the position and 3D scene structure fixed.

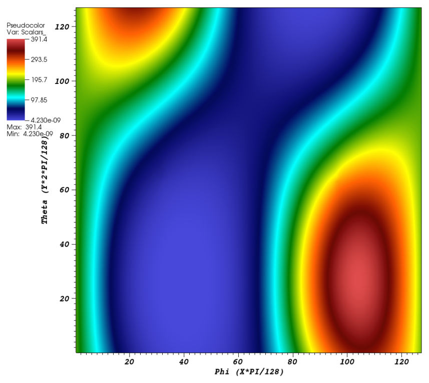
Upon initial inspection, the cost function tended toward a single global optimum when holding certain parameters fixed and varying others. Unfortunately, the true topology of the function cannot be understood from visualization alone. First, the function was sampled in order to generate the scalar fields and therefore if a local minimum is not sampled it would not be included in the analysis. In addition, color blending might 'hide' local minima. Despite not fully understanding the topology, the cost function still yielded good results when used for SfM parameter optimization (bundle adjustment, see Section IV).



(a) 3D structure variation



(b) Camera position variation



(c) Camera rotation variation

Fig. 2: This figure contains the results of the scalar field analysis for the presented cost function. The fields were generated from synthetic data and are $128 \times 128 \times 128$ in dimension. In part (a), a single 3D structure point was projected into 36 cameras with known parameters. The scalar field was centered over the 3D structure point and the function value at each point in the field was computed with the synthetic data. The dimensions extended five units in each direction for each dimension. For (b) and (c), a single synthetic camera was generated with 100 known 2D/3D correspondences. In (b), the scalar field was centered over the camera position and a test camera was generated with the scalar field position and original camera rotation. The test camera previously generated was used to generate each cost function value in the scalar field. In (c), spherical coordinates ($\theta \in [0, 2\pi]$ and $\phi \in [0, \pi]$) were used to define the view direction, and along with the general up vector, $\vec{up} = (0, 1, 0)^T$, provided a camera rotation matrix. The test camera in this scenario was generated with the computed rotation matrix and original camera position. In all three figures, the cost function tends toward a single global optimum, namely the original data, which indicates good cost function behavior.

C. *Feature Track Filtering*

Feature tracking correction and refinement with depth estimates involved grouping each 3D triangulated point with its corresponding 3D depth points. The 3D depth points are obtained by unprojecting depth estimates from each feature track location on each image back into the scene. Then each cluster of triangulated point and depth points are analyzed to determine whether to refine the feature track estimate or to eliminate it.

While there are many metrics that can be applied to make the refine/reject determination, this paper examines a few simple yet effective metrics that were evaluated experimentally. In the first metric, *NormalSP*, the average distance of the depth points from the triangulated structure point is computed. The standard deviation is computed and any depth point that exceeds a user defined multiple of the standard deviation is counted. If the count exceeds the expected population for the distribution in this range of the total, then the cluster is rejected. Similarly, *NormalCoM* evaluates a cluster based upon an estimated normal distribution; however, instead of computing the distance of the depth points from the structure point, it evaluates the distance between each point in the cluster and the cluster's center of mass.

After all clusters have been evaluated, all rejected clusters (and their associated feature tracks) are eliminated, leaving only the clusters to be refined. Several refinement procedures are presented in this paper, but more complicated algorithms could be used in their place. The first refinement technique, named *CenterOfMass*, simply takes the cluster's center of mass as the updated 3D position for the triangulated point. This updated position is used to correct the feature track location in each image. While *CenterOfMass* performs an evenly weighted average across the entire cluster, *WeightedAverage* allows uneven weighting to be applied to the triangulated point and depth points within the cluster. Finally, *DistanceWeighted* weights each point in the cluster based on its distance to the cluster's centers of mass, where the most distant point has the lowest weight and the closest has the highest.

IV. Results

The proposed algorithms were tested extensively for their accuracy, processing time, and general behavior, on both real and synthetic data. All implementation was done using C++ on a MacBook Pro with and *Intel Core i7* processor at 2.66 GHz with 4GB of RAM running Mac OS X Mavericks 10.9.5.

A. *Bundle Adjustment Synthetic Testing*

The initial set of synthetic tests evaluated the performance of the bundle adjustment routine using the proposed cost function. The routine, called ADBA (angular depth bundle adjustment), was implemented using Google's Ceres-Solver [24] and was compared against a Ceres implementation of the traditional bundle adjustment cost function [3]. The data consisted of 25 randomly placed cameras (varying orientations and positions) looking at 100 scene points. The data was perturbed in four specific scenarios. In the first, the 3D scene structure was moved according to a normal distribution with mean $\mu = 0$ and standard deviation $\sigma \in [1, 20]$ world space units. The scene was crafted such that the largest σ value corresponds to a change in position of about 20% of total scene structure size. The second scenario varied the original camera positions according to a normal distribution with $\mu = 0$ and $\sigma \in [1, 20]$, and the third varied the camera rotation according to a normal distribution with $\mu = 0$ and $\sigma \in [\frac{\pi}{180}, \frac{\pi}{9}]$. The final scenario varied these parameters simultaneously according the distributions stated previously.

To compute the results, the scene configuration was generated 50 times for each σ value. Both optimization procedures were applied to all the scene configuration runs. Six different metrics were recorded and averaged across the 50 runs. A few of the metrics were based upon an L_1 norm, which effectively can be though of as sum of error terms. The first metric, *L_1 3D point distance to ground truth* computed the total distance between each original 3D scene point and its final position after optimization for all 100 scene points. Similarly, *L_1 camera distance to ground truth* computed the total distance between each original camera position and its final optimized position for all cameras. In addition, a rotation error metric, *L_1 camera rotation to ground truth*, computed the relative rotation error between each original camera and its final rotation. The relative rotation error was based on the formulation given by Lepetit et al. [25]. Average execution time, average number of optimization iterations, and total cost function value were also computed. The results of this experiment are shown in Figure 3.

From Figure 3, it can be seen that the parameter optimization utilizing the proposed cost function outperforms the traditional bundle adjustment routine. In the scenarios that only changed the 3D point and camera positions, the resulting L_1 3D point distance to ground truth (Figure 3a) and L_1 camera distance to ground truth (Figure 3b) produce similar results, with ADBA resulting in slightly more accurate values. Varying the rotation resulted in far more accurate camera orientations for ADBA, as shown in Figure 3c. Despite the compound errors in the final scenario, ADBA is able to obtain values that are far closer to the original data, as shown in Figures 3d–3f. It should be noted that this additional accuracy comes with some additional processing time. In the final scenario, ADBA required an additional 5 seconds of processing time (about 14% of total execution time) compared to the original bundle adjustment. This trade-off might be acceptable for applications in which accuracy is favored over speed.

B. Feature Track Filtering Synthetic Testing

The next set of synthetic tests evaluated the performance of the feature tracking correction routine. The correction routine consists of a filtering metric and a refinement scheme. For this experiment, *NormalSP* (NSP) and two versions of the *NormalCoM* (depth only, NCOM, and both depth and structure, NCOM2) metrics were used. Three refinement techniques, *CenterOfMass* (COM), *Weighted Average* (WA) with a 50% weight to the structure point and 50% to the depth point center of mass, and *DistanceWeighted* (DIS), were also evaluated such that each metric was paired with each refinement scheme. Similar to the previous experiment, the reconstruction data consisted of 25 cameras with random position and orientation viewing 100 scene points. However, the data was only varied by introducing simulated feature tracking noise. The feature tracks were modified according to a normal distribution with mean $\mu = 0$ and standard deviation $\sigma \in [1, 10]$ pixels. Again, the scene was crafted such that the largest σ value corresponds to a change in position of about 10% of total scene structure size.

Each scene configuration was generated 50 times for each σ value in the experiment. Two different metrics were recorded and averaged across the 50 runs. The first metric, *Average 3D point distance to ground truth*, is similar to L_1 3D point distance to ground truth metric, except that the result is averaged across the number of points in the collection. The average 3D point distance to ground truth was recorded for both the refined structure points and feature tracks and the data prior to refinement but after the metric was applied. Similarly, the *Average feature track distance to ground truth* metric, which measures the average distance between given keypoints and their corresponding ground truth keypoints across a collection of feature tracks, was recorded for both refined and pre-refined data. For this specific experiment, the metrics need to be averages because the number of remaining feature tracks can differ between the applied filtering metrics. The results of this experiment are displayed in Figure 4.

Results from Figure 4 indicate that reconstruction accuracy can be improved by utilizing the proposed feature track filtering and refinement mechanism. Analysis indicates that utilizing the NCOM2 metric filters tracks that do not closely fit the ground truth data as shown in Figures 4b and 4d. Interestingly, NCOM2 actually retained 89% of the feature tracks, with NSP retaining 88% and NCOM retaining 81%. NCOM2 performed best when paired with the COM and DIS refinement techniques. With limited noise, the DIS refinement technique performed better than the COM but as error increases, the COM refinement surpasses DIS.

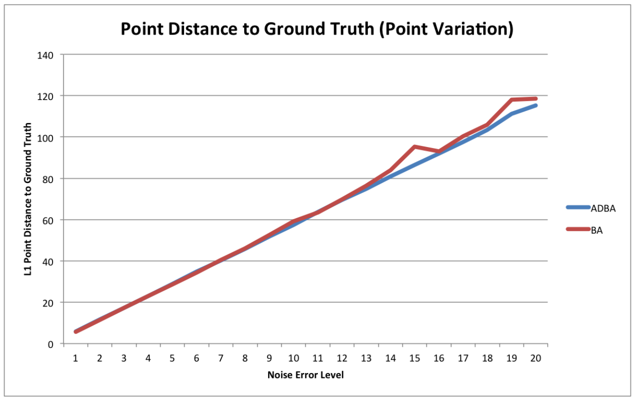
C. Real Datasets

In addition to the set of synthetic experiments, the proposed algorithms were integrated into a depth SfM pipeline, which was run on real images. The resulting reconstructions were more accurate than with traditional bundle adjustment, though the proposed procedures increase execution time. Figure 5 displays the results from the real datasets.

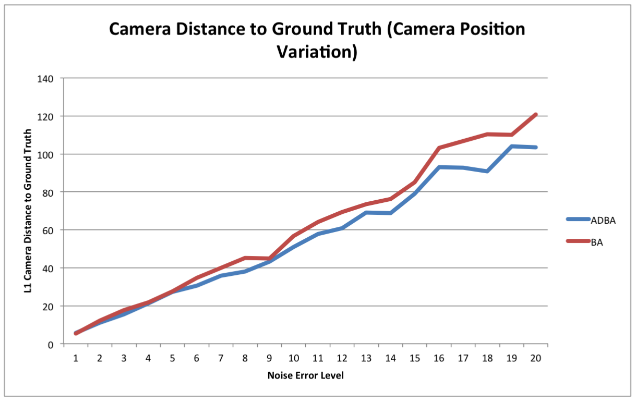
V. Conclusion

In conclusion, this paper presents two improvements to the traditional structure-from-motion reconstruction pipeline, both of which use depth data. The first is a general cost function that evaluates the quality of reconstruction based on a previously established angular cost function along with depth data estimates. This cost function is used to implement a bundle adjustment parameter optimization to increase reconstruction accuracy. Second, in the event of feature tracking errors, a corrective routine is present to detect and correct inaccurate feature tracks. The algorithm involves clustering depth estimates of the same reconstruction point and observing differences between them.

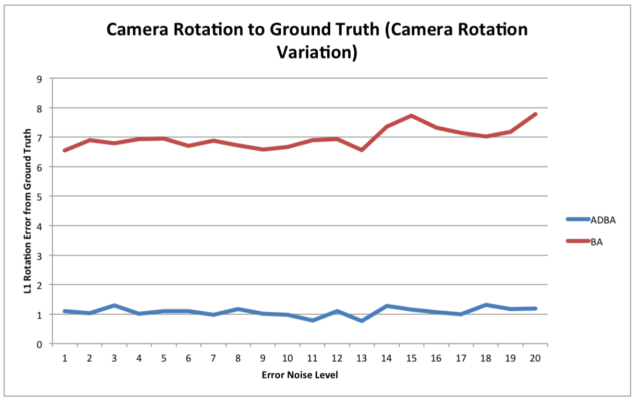
Analysis showed that the cost function has a good topology and behaved well in parameter optimization. Experiments were performed that show increased reconstruction accuracy in the presence of camera noise and imprecise 3D structure computation, when compared to traditional bundle adjustment. In addition, inaccurate feature tracks have been corrected



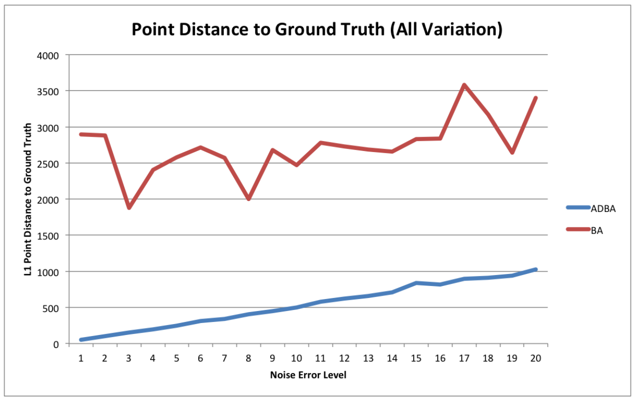
(a) 3D point distance versus ground truth, varying 3D points



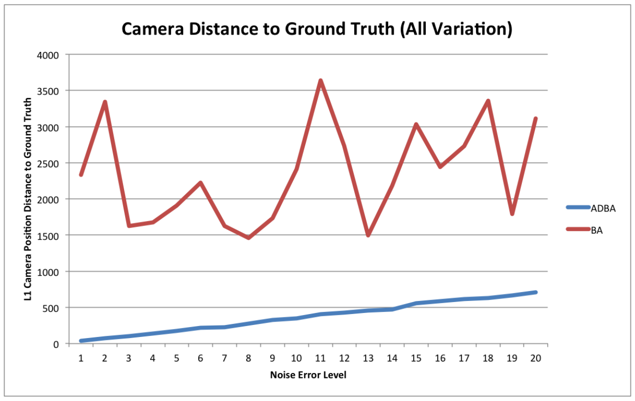
(b) Camera position distance versus ground truth, varying camera positions



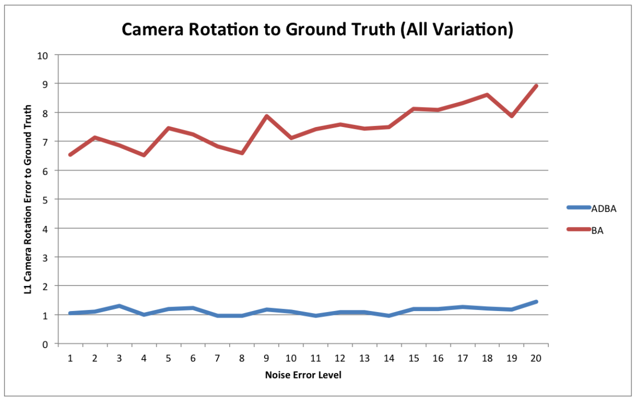
(c) Camera rotation distance versus ground truth, varying camera rotations



(d) 3D point distance versus ground truth, varying everything

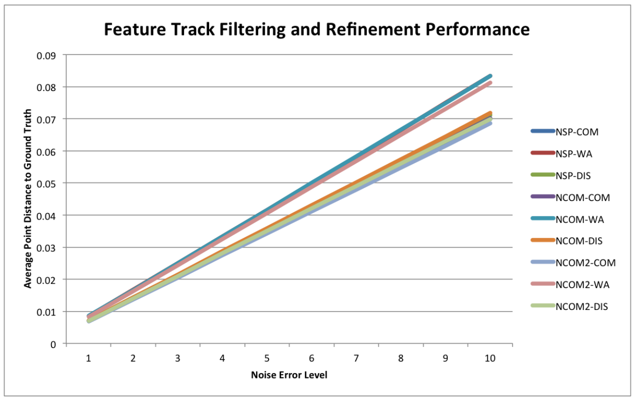


(e) Camera position distance versus ground truth, varying everything

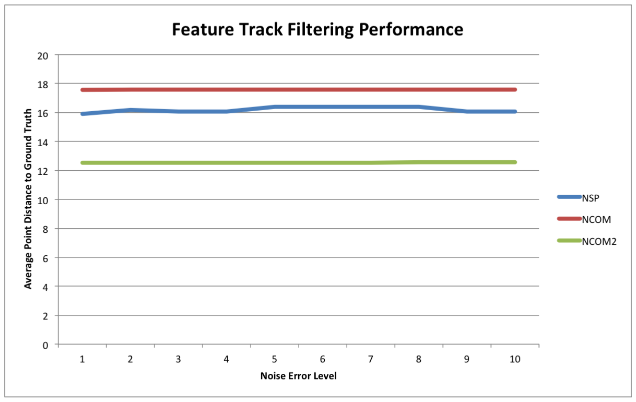


(f) Camera rotation distance versus ground truth, varying everything

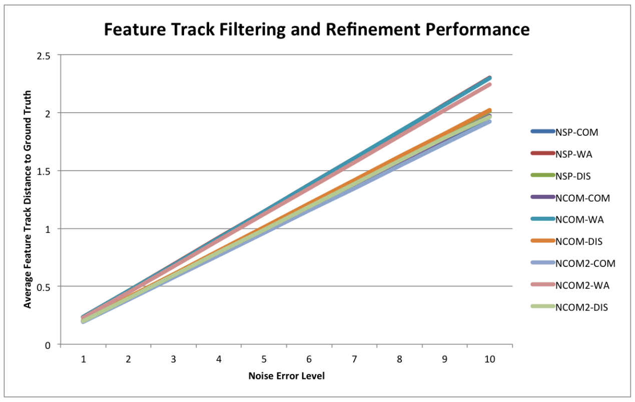
Fig. 3: This figure compares the performance of the proposed bundle adjustment cost function with the traditional one. Synthetic data, consisting of 25 randomly posed cameras and 100 scene points, was perturbed in four specific scenarios. In the first, the 3D scene structure was moved according to a normal distribution with mean $\mu = 0$ and standard deviation $\sigma \in [1, 20]$ world space units. The second scenario varied the original camera positions according to a normal distribution with $\mu = 0$ and $\sigma \in [1, 20]$, and the third varied the camera rotation according to a normal distribution with $\mu = 0$ and $\sigma \in [\frac{\pi}{180}, \frac{\pi}{9}]$. The final scenario simultaneously varied these parameters according the distributions stated previously.



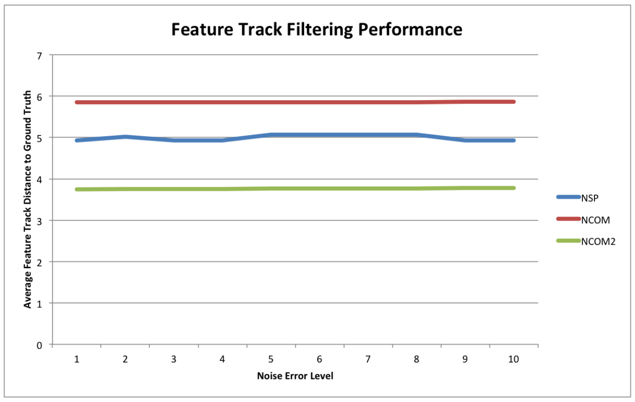
(a) L_1 point distance to ground truth for all metrics and refinements



(b) L_1 point distance to ground truth for all metrics prior to refinement

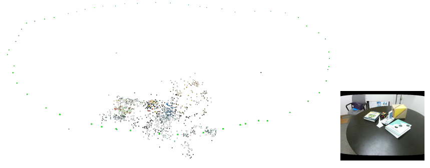


(c) L_1 feature track distance to ground truth for all metrics and refinements

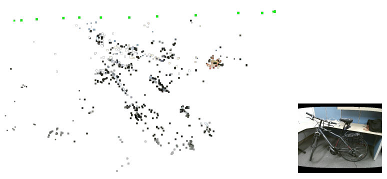


(d) L_1 feature track distance to ground truth for all metrics prior to refinement

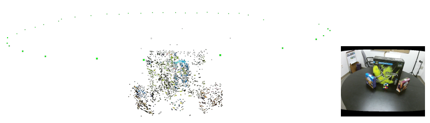
Fig. 4: This figure shows the performance of the various metrics and refinement techniques utilized in the feature track filtering algorithm. Synthetic data, consisting of 25 randomly posed cameras and 100 scene points was modified by introducing feature tracking noise. The noise was modeled using a normal distribution with $\mu = 0$ and $\sigma \in [1, 10]$. Results were recorded for all combinations of metric and refinement techniques.



(a) Kinect Table1 Dataset



(b) Kinect Bike Dataset



(c) Kinect Table2 Dataset

Fig. 5: This figure displays several reconstructions from real datasets. For the dataset in panel (a), traditional bundle adjustment required 22.97 seconds and ADBA required 35.30 seconds. Traditional bundle adjustment required 7.50631 and 34.946 seconds respectively for the datasets in panels (b) and (c), whereas ADBA executed in 9.28428 and 50.4467 seconds.

using an evaluation metric and refinement on clusters of depth points generated from the track. Finally, a depth data SfM pipeline, incorporating the proposed algorithms, produced reconstructions on real datasets.

VI. Future Work

The proliferation of multi-sensor enabled hardware, such as modern cell phones, presents interesting and unique challenges for sensor fusion. Developing algorithms to accurately and intelligently utilize the variety of data inputs is of key importance. This work addresses how depth data can be utilized to increase accuracy in SfM applications. The authors would like to pursue different error metrics and more advanced feature track filtering techniques to further improve depth reconstruction pipelines. Additional investigation into the topology of the proposed cost function is another direction of future research.

Acknowledgment

This work was supported in part by the Department of Energy, the National Nuclear Security Agency through Contract No. DE-FG52-09NA29355, the US Marine Corps SBIR program, and SURVICE Engineering's Internal R&D program. The authors thank their colleagues in the Institute for Data Analysis and Visualization (IDAV) at UC Davis and in the Applied Technology Operation at SURVICE Engineering for their support.

References

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

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

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

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

[5] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera." ACM Symposium on User Interface Software and Technology, October 2011. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=155416

[6] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal On Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[7] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, ser. Lecture Notes in Computer Science, A. Leonardis, H. Bischof, and A. Pinz, Eds. Springer Berlin / Heidelberg, 2006, vol. 3951, pp. 404–417, 10.1007/11744023.32. [Online]. Available: http://dx.doi.org/10.1007/11744023.32

[8] E. Tola, V. Lepetit, and P. Fua, "Daisy: an efficient dense descriptor applied to wide baseline stereo," in *PAMI*, vol. 32, no. 5, May 2010, pp. 815–830.

[9] D. Knoblauch, M. Hess-Flores, M. Duchaineau, and F. Kuester, "Factorization of Correspondence and Camera Error for Unconstrained Dense Correspondence Applications," *5th International Symposium on Visual Computing, Las Vegas, Nevada*, pp. 720–729, 2009.

[10] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera." ACM Symposium on User Interface Software and Technology, October 2011. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=155416

[11] J. Quiroga, T. Brox, F. Devernay, and J. Crowley, "Dense semi-rigid scene flow estimation from RGBD images," in *Computer Vision ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Springer International Publishing, 2014, vol. 8695, pp. 567–582.

[12] M. Hornacek, A. Fitzgibbon, and C. Rother, "Sphereflow: 6 DoF scene flow from RGB-D pairs," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, June 2014, pp. 3526–3533.

[13] F. Steinbrücker, C. Kerl, and D. Cremers, "Large-scale multi-resolution surface reconstruction from RGB-D sequences," in *ICCV'13*, 2013, pp. 3264–3271.

[14] D. Thomas and A. Sugimoto, "A flexible scene representation for 3D reconstruction using an RGB-D camera," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 2800–2807.

[15] C. Raposo, M. Lourenco, M. Antunes, and J. Barreto, "Plane-based odometry using an RGB-D camera," in *British Machine Vision Conference 2013*, 2013.

[16] E. Bylow, C. Olsson, and F. Kahl, "Robust camera tracking by combining color and depth measurements," in *International Conference on Pattern Recognition*, 2014.

[17] J. Stückler and S. Behnke, "Efficient dense 3D rigid-body motion segmentation in RGB-D video," 2013.

[18] B. Boom, S. Orts-Escolano, X. Ning, S. McDonagh, P. Sandilands, and R. Fisher, "Point light source estimation based on scenes recorded by a RGB-D camera," in *British Machine Vision Conference 2013*, 2013.

[19] G. Saygili, L. van der Maaten, and E. A. Hendriks, "Hybrid Kinect depth map refinement for transparent objects," in *International Conference on Pattern Recognition*, 2014 In Press.

[20] B. Zeisl, K. Köser, and M. Pollefeys, "Automatic registration of RGB-D scans via salient directions," in *ICCV'13*, 2013, pp. 2808–2815.

[21] S. Song and J. Xiao, "Tracking revisited using RGBD camera: Unified benchmark and baselines," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ser. ICCV '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 233–240. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2013.36

[22] H. Afzal, D. Aouada, D. Fofi, B. Mirbach, and B. Ottersten, "RGB-D Multi-View System Calibration for Full 3D Scene Reconstruction," in *22nd International Conference on Pattern Recognition (ICPR)*, 2014.

[23] S. Recker, M. Hess-Flores, and K. I. Joy, "Fury of the swarm: Efficient and very accurate triangulation for multi-view reconstruction," in *International Conference on Computer Vision Big Data 3D Computer Vision Workshop*, S. Recker and M. Hess-Flores, Eds., Dec. 2013.

[24] S. Agarwal, K. Mierle, and Others, "Ceres solver." [Online]. Available: https://code.google.com/p/ceres-solver/

[25] V. Lepetit, F.Moreno-Noguer, and P.Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal Computer Vision*, vol. 81, no. 2, 2009.