Cluster-based 3D Reconstruction of Aerial Video

Scott M. Sawyer, Karl Ni, and Nadya T. Bliss
MIT Lincoln Laboratory
Emails: {scott.sawyer, karl.ni, nt}@ll.mit.edu

***Abstract*—Large-scale 3D scene reconstruction using Structure from Motion (SfM) continues to be very computationally challenging despite much active research in the area. We propose an efficient, scalable processing chain designed for cluster computing and suitable for use on aerial video. The sparse bundle adjustment step, which is iterative and difficult to parallelize, is accomplished by partitioning the input image set, generating independent point clouds in parallel, and then fusing the clouds and combining duplicate points. We compare this processing chain to a leading parallel SfM implementation, which exploits fine-grained parallelism in various matrix operations and is not designed to scale beyond a multi-core workstation with GPU. We show our cluster-based approach offers significant improvement in scalability and runtime while producing comparable point cloud density and more accurate point location estimates.**

I. Introduction

Detailed 3D reconstructions can be automatically generated from photos and video by inferring the geometry of the real-world scene based on motion between multiple 2D viewpoints. This process of creating Structure from Motion (SfM) of imagery is a very active area of research, with many techniques and applications being developed in recent years. The task is especially challenging because determining each camera pose and the matching points among the images represent an extraordinarily large solution space requiring much computation to converge on a reasonable estimate.

Solutions to the problem have gained traction in the past decade in commercial and military applications, including mapping [1], robotic navigation [2], and most prevalently, 3D geo-registration of photos and video [3]–[5]. SfM techniques are well-suited to geo-registration because they work with low-cost sensors and platforms and are robust to outliers in telemetry data, which could propagate errors in other techniques.

Automated 3D reconstruction represents a considerable computational challenge, exercising the state-of-the-art in computer vision, optimization, and parallel computing. SfM techniques rely on identifying and matching common features from multiple images, a process whose runtime grows quadratically with the input size. Moreover, a solution requires an estimate of each camera's parameters, which include both extrinsics (e.g. location and pointing direction) and intrinsics (e.g. focal length and lens distortion). Parameter estimation presents a highly non-linear and non-convex optimization problem. Compounding the computational difficulties are large data considerations and real-time processing requirements imposed by certain applications. The volume of imagery at high-resolution is quickly outpacing the current capability to process it. Furthermore, building models with high density and accuracy requires large photo sets, and computation quickly becomes intractable without efficient, parallel techniques. Hence, scalable and automated algorithms are required to process aerial imagery in appropriate computing environments.

Attempts to improve SfM efficiency have been previously proposed [6]–[8], and multi-core processors and Graphics Processing Units (GPUs) have been employed to accelerate computation [9]. However, while some work has considered SfM in a cluster environment [10], little attention has been given to processing datasets relevant to wide-area geo-registration, such as aerial video, on a large-scale computing grid.

This paper focuses on the problem of computing SfM from aerial video for the purpose of fully automated geo-registration of objects in the video. Applications require large input datasets to produce accurate results, so the implementation must be able to scale to an appropriately sized cluster in order to meet a real-time processing budget. For this study, we consider a dataset collected by MIT Lincoln Laboratory over the MIT campus in 2011. The GPS location of the sensor is available for all frames, and video frames are sampled at several rates to produce photo sets of various sizes. We propose an SfM processing chain that uses state-of-the-art SfM methodologies that can run on a large-scale cluster. We compare this chain's runtime and reconstruction quality with a freely available parallel SfM implementation [9] designed to run on a high performance desktop workstation. Results are shown for the sequential baseline implementation, the multi-core workstation processing chain, and our cluster-based processing chain. Our contributions are a novel technique for parallelizing SfM and an in-depth analysis on the impact of two parallelization methods on 3D reconstruction results.

The remaining sections will focus on describing SfM in more detail, presenting our proposed solution, and evaluating performance and scalability. Sec. II begins with a brief review of the SfM problem for geo-registration. The next section, Sec. III is a survey that outlines specific SfM implementations in the literature and on the internet. Sec. IV introduces our cluster-based SfM processing chain, while Sec. V compares it with the alternative parallel and sequential processing chains. Finally, we conclude in Sec. VI.

This work is sponsored by the Assistant Secretary of Defense for Research and Engineering under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Government.

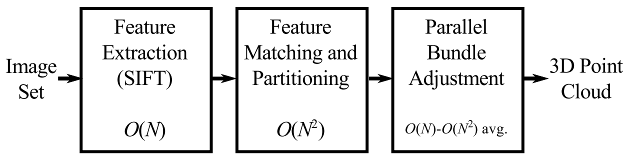
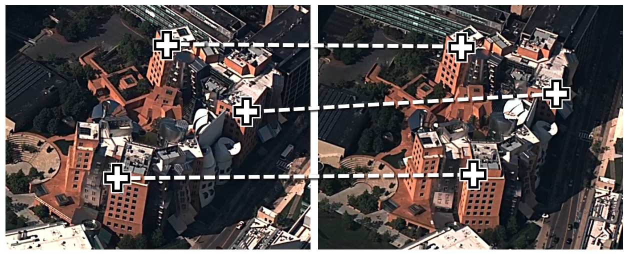
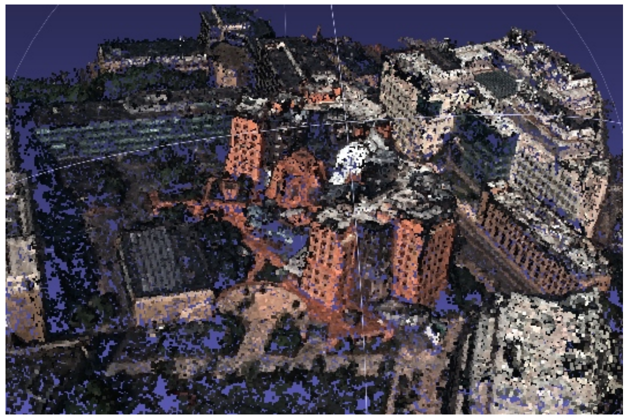


Image Set
Feature Extraction (SIFT)
$O(N)$
Feature Matching and Partitioning
$O(N^2)$
Parallel Bundle Adjustment
$O(N)$-$O(N^2)$ avg.
3D Point Cloud

(a)



(b)



(c)

Fig. 1. (a) The baseline SfM processing chain consists of several steps of varying computational complexity. (b) Key points based on SIFT are extracted and matched among images. (c) Bundle adjustment creates a 3D point cloud from the implicit scene geometry.

II. Review of Structure from Motion (SfM)

The SfM problem infers the 3D geometry of a scene based on multiple 2D views of the same scene from different aspect angles. SfM techniques and applications have been active areas of computer vision research in recent years. In general, SfM techniques follow this this series of steps: feature extraction, feature matching, and bundle adjustment (Fig. 1) [11]. The processing chain takes a series of photos as its input and it outputs a 3D model (generally a point cloud) and the estimated position and viewing frustum of each camera. In addition to the 3D model, SfM identifies a graph connecting each 3D point to a set of views corresponding to features in the input photos.

Feature extraction relates to the identification of distinct features in each photo that can be robustly matched to features in the other photos. Identifying distinctive image features is a difficult problem, with many SfM techniques relying on the Scale-Invariant Feature Transform (SIFT) developed by David Lowe, who offers an implementation that accepts an image as input and returns a list of key points described by their location in the image, scale, orientation, and a 128-byte identifying vector [12]. The computational cost of the feature extraction step grows linearly ($O(N)$) with the number of input photos. Computation also increases with photo resolution, resulting in the identification of more key points per photo. Feature extraction is embarrassingly parallel, as each photos feature set can be computed independently of the others. Additionally, the filters and dense operations performed within feature matching can be parallelized further or implemented to take advantage of hardware acceleration.

Following feature extraction, features must be matched between images. A matched feature indicates two photos have captured views of the same real-world point. In the case of SIFT, two features would ideally be considered a match when their 128-byte feature vectors have a Euclidean distance below some threshold. For efficiency, matching is generally performed using Approximate Nearest Neighbor (ANN) techniques to avoid computing the distance between all combinations of extracted features. In either case, the features from any given image must be matched against the features of every other image. This nominally leads to quadratic growth ($O(N^2)$) of computation with respect to the number of input images. Although this step is computationally intensive, it is also easily parallelized, as the feature comparisons can be performed simultaneously.

Then 3D geometry is iteratively estimated from the matched (and often sparse) features in a process termed sparse bundle adjustment. As more views of the same point are added to the bundle of points, the 3D location of each point is re-estimated in a least-squares sense. To first order, bundle adjustment has $O(N)$ complexity, but runtimes generally approach $O(N^2)$ in reality. The process is iterative and is not trivially parallelized. Nonetheless, the following section will discuss several approaches for improving runtime performance. Upon completion of bundle adjustment, a 3D point cloud has been generated in an arbitrary 3D coordinate system. Compared to a Cartesian geographic coordinate system, this coordinate system has an unknown scale, rotation and translation.

Finally, many SfM implementations conclude with a step to generate a denser point cloud by identifying likely surfaces in the 3D model. This step results in appealing dense 3D models; however, it generally does not maintain the graph relationship between 3D points and features in the original photos. In the remainder of this paper, we do not consider this step because it is not relevant for most of our applications of interest.

III. Implementation Survey

Implementing SfM has been the focus of much research aiming to improve the reconstruction quality and the computational efficiency of the processing chain. This section surveys some notable work in the field but in no way constitutes a complete list. We consider Bundler [13] as the baseline SfM implementation. Bundler and its source code are freely available on the web, and it has been cited in an abundance of SfM papers. The tool makes no assumptions about its input photo set; rather, it has been designed to handle a completely unorganized collection, such as photos downloaded from various sources on the web. Bundler runs all steps on a single processor, but the feature extraction and matching steps can be parallelized with relatively simple modifications to the source code.

In recent years, many groups, including some of the authors of Bundler, have proposed techniques for improving efficiency. In [7] a sparse point cloud is formed using a "skeletal set" of the most important points, and then subsequent points are added after significantly reducing the solution space for remaining camera parameters. Additionally, in [14], they propose matching features against a SIFT vocabulary database, which realizes computational savings compared to matching against the entire input data set. Most recently, [8] suggests finding a coarse initial estimate using a Markov random field formulation and refining the initial solution using Levenberg-Marquardt.

While Bundler has been designed for unstructured photo sets, we are considering images from a video source. In this case, one can assume the camera travels at a reasonable speed, thereby limiting the change in camera location and pose from frame to frame. It has been demonstrated that feature matching can be reduced to $O(N)$ complexity by applying these video constraints [3].

Another freely available tool, VisualSFM, is a highly optimized, parallel implementation of SfM that runs on multi-processor workstations and can use a GPU if available [9]. VisualSFM is particularly efficient at SIFT feature extraction. While Lowe's implementation operates using a single processor, VisualSFM can take advantage of a GPU to greatly accelerate extraction [15]. The tool also performs multi-threaded matching and uses Multi-Core Bundle Adjustment (MCBA) [16], which also uses a GPU if available, to generate point clouds with considerable speed-up compared to Bundler. MCBA uses a different approach for solving the non-linear least squares problem underlying bundle adjustment, and it exploits the fine-grained parallelism in the bottleneck matrix operations (e.g. computation of the Jacobian matrix and several matrix-vector multiplications).

Out-of-Core Bundle Adjustment [10] partitions the set of feature matches using graph analysis techniques. The algorithm then runs several bundle adjustment processes in parallel on different machines to create multiple point clouds. The point clouds are then combined into a single model by exploiting common points between the clouds. In addition to offering speed-up via parallelization, this implementation offers more scalability than VisualSFM because each bundle adjustment process is on a separate machine, thereby allowing the computation of large-scale point clouds that require more memory than available to a single system. Our solution is similar to [10], but we utilize the assumptions about the aerial video to partition the image set and fuse the resulting point clouds using different techniques, which achieve very high point accuracy for our particular input data constraints.

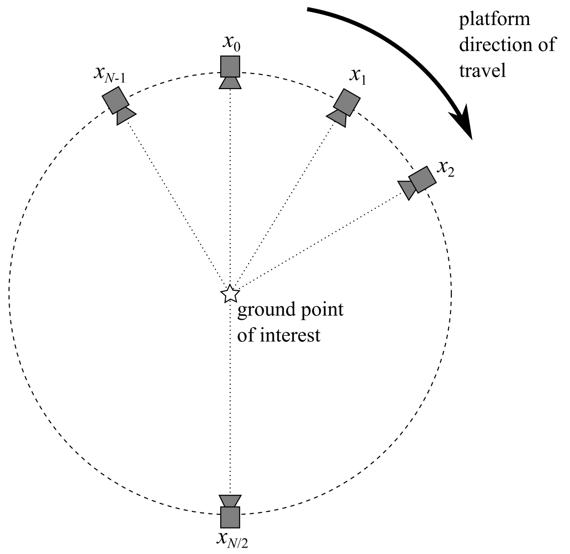


Fig. 2. Camera locations and pointing directions for N aerial video samples forming SfM input image set $x_0, x_1, ..., x_{N-1}$. The sensor continuously points at a ground point of interest while the platform completes one full revolution.

IV. Cluster-based SfM

We seek to efficiently generate 3D reconstructions from large-scale datasets of aerial video frames on a compute cluster. We assume an aerial platform captures the video while completing a full revolution around a single ground point of interest, continuously fixing the camera on the ground point, as shown in Fig. 2. Given these data constraints and the available SfM implementations (detailed in the previous section), this section describes our scalable cluster-based approach.

A cluster is defined to be a collection of commodity Linux-based machines connected via Ethernet. For the purposes of this paper, communication between processes is available only through a shared file system. A master node submits tasks to worker nodes, which are managed by a central scheduler. This computing model is scalable to thousands of nodes. In general, an algorithm may share the cluster with other processes. Therefore, the processing chain is constrained to use a maximum number of processors, N_p, specified as a runtime parameter. The algorithm is implemented in a custom software framework that dispatches each step on a set of cluster nodes, enforcing barrier synchronization between steps. Input imagery, intermediate results, and the final reconstruction are persisted in a common database, enabling results to be re-used by other processing chains.

The proposed processing chain is shown in Fig. 3. Cluster-based SfM can begin as soon as aerial video from a complete revolution becomes available for ground processing. The video is sampled at a constant rate to form a collection of N images. Each image, along with its GPS metadata, is sent to the processing chain input. Feature extraction and matching follow the same technique as [13], but use modified scripts and binaries to run in parallel. The feature matching step produces a graph $G = (V, E)$ whose vertices are the SIFT key points

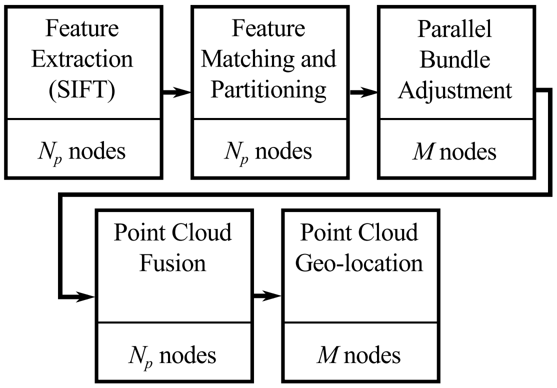


Fig. 3. Cluster-based SfM processing chain is comprised of multiple steps, each dispatched on the number of CPUs shown above. Barrier synchronization is enforced between each step.

(i.e. vertex $v_{i,j} \in V$ corresponds to the j^{th} SIFT point found in image x_i). Edge $v_m v_n$ exists if SIFT points v_m and v_n were found to match in the feature matching step.

Following feature matching, the N input images $(x_0, x_1, ..., x_{N-1})$ are partitioned into M sets $(P_0, P_1, ..., P_{M-1})$, each to undergo independent bundle adjustment in parallel. M cannot exceed N_p (the maximum number of processors available), but we apply the additional constraint that $N/M \geq 25$ such that each partition has enough images to ensure a reconstruction of reasonable quality. Images are partitioned in a block-cyclic pattern with C cycles. Photos are partitioned such that $x_i \in P_j$, where:

$$j = \left\lfloor \frac{i}{\lfloor N/(M+C) \rfloor} \right\rfloor \bmod M \tag{1}$$

Generally, $C > 12$ for large datasets ($N \gg 12M$) to limit the maximum aspect angle between consecutive photos in the same partition to no more than 30°, which promotes SIFT matching.

We then perform parallel bundle adjustment by running the Bundler binary independently on each partition. This results in M point clouds each in its own arbitrary 3D coordinate system. The point clouds can be described by the matrices $\mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_{M-1}$, each comprised of 3D points such that:

$$\mathbf{X}_i = \begin{bmatrix} x_0 & x_1 & & x_{N_i-1} \\ y_0 & y_1 & \cdots & y_{N_i-1} \\ z_0 & z_1 & & z_{N_i-1} \end{bmatrix} \tag{2}$$

where N_i is the size of the particular point cloud and each column contains the coordinates for a 3D point.

In order to combine them, we first must identify corresponding points among the partitions. Each reconstructed 3D point has a view list, $A \subset V$, which is the set of SIFT points used to determine its location. View lists from different partitions will always be disjoint because each photo—and thus each SIFT point—belongs to only one partition. However, point correspondences can be determined by using the match graph G. Two 3D points from different partitions with view lists A and B, respectively, are considered corresponding if:

$$|\Gamma(A) \cap B| > 0 \tag{3}$$

where $\Gamma(A)$ is the neighborhood of set A.

Given the point correspondences among the partitions, we can transform the point clouds to a common 3D space. Each point cloud is transformed to the coordinate system of the first partition by solving the orthogonal Procrustes problem to find the rotation matrix, $\mathbf{R}_i$, scale, s_i, and translation, $\mathbf{t}_i$, between the corresponding points in $\mathbf{X}_0$ and $\mathbf{X}_i$. The transformation is then applied to all points in the cloud $\mathbf{X}_i$ to form $\mathbf{X}'_i$ for all $i > 0$:

$$\mathbf{X}'_i = s_i \mathbf{R}_i \mathbf{X}_i + \mathbf{t}_i \mathbf{1}_{1 \times N_i} \tag{4}$$

The point correspondences also indicate redundant key point observations. Prior to combining the point clouds, we must identify and remove duplicate points. Bundle adjustment iteratively improves the estimate of each point's position, $\mathbf{x} = \{x, y, z\}$. Thus, we can combine the d duplicate points to form an improved estimate, $\mathbf{x}'$, using weight, w, equal to the number of views in the view list (i.e. $w_i = |A|$):

$$\mathbf{x}' = \frac{\sum_{i=1}^{d} w_i^2 \mathbf{x}_i}{\sum_{i=1}^{d} w_i^2} \tag{5}$$

The duplicate points are removed from all containing clouds except one, such that each duplicate $\mathbf{x}_0$ is replaced by its $\mathbf{x}'$. The point clouds are then combined.

Finally, the last step transforms the point cloud into Universal Transverse Mercator (UTM) coordinates. We again estimate rotation, scale and translation between the arbitrary SfM coordinate space and UTM, this time using the estimated camera locations from bundle adjustment and the known camera GPS locations converted to Cartesian UTM coordinates. Poor camera position estimates are removed using a RANSAC-based technique. We then apply the transformation to the cloud, resulting in a single point cloud in UTM coordinates. This technique coarsely geo-registers the point cloud with error contributions from the GPS and SfM camera placement estimates. For applications requiring precise geo-registration, a subsequent step would match the point cloud against reliable reference data, such as LIDAR or satellite imagery, using this coarse geo-registration as an initial estimate.

V. Results

We compare two parallel SfM processing chains against the sequential baseline implementation [13]. The first chain is VisualSFM [9], which is freely available online and optimized to use a multi-core processor and GPU (see Sec. III). The second chain is the novel cluster-based technique described in Sec. IV. We run each chain for various numbers of input images, evaluating the chains in terms of runtime and reconstruction quality. The cluster-based chain is also run for various numbers of compute cores.

The input image sets are comprised of extracted frames from video of one aerial revolution around the MIT Stata Center. The platform flew an approximate circle of two-mile radius at an altitude of roughly 7500 ft. Each image is 1280×720 pixels with latitude, longitude and altitude available as metadata. The camera remained fixed on the ground target with a constant field-of-view capturing the entire building and some immediate surroundings.

A. Runtime Benchmarks

Bundler and VisualSFM benchmarks were run on a quad-core Intel Xeon (2.4GHz) workstation with 6GB RAM and an nVIDIA Quadro FX 1800 GPU (64 compute cores) for math kernel acceleration. Cluster-based benchmarks were run on various numbers of compute cores, where each cluster machine contained dual-socket Intel Xeon processors (3.2GHz) with 8GB RAM and was interconnected by gigabit Ethernet.

Fig. 4 shows runtime as a function of the number of input images, for the baseline Bundler processing chain, VisualSFM, and cluster-based SfM. Feature matching and bundle adjustment dominate runtime, with feature extraction barely visible at the bottom of (a) and (b). VisualSFM offers significant speed-up of the bundle adjustment step. However, cluster-based SfM ultimately offers much faster runtime, in particular by scaling feature matching to more processing nodes. In (c), for input sizes 100, 200, 300 and 400, M values of 4, 8, 12, and 16 were used respectively.

Fig. 5 shows cluster-based SfM speed-up for various cluster sizes. For small image set sizes, the overhead of the processing chain actually increases runtime. However, for beyond 100 input images, cluster-based SfM provides significant speed-up. For the largest set sizes and cluster sizes evaluated (top right corner of plot) speed-up has converged to approximately 26x for a 64-node cluster. Speed-up is less than linear for two primary reasons: in some cases $M < N_p$, and in all cases additional processing is required to fuse the point clouds generated in parallel.

B. Reconstruction Quality

We have demonstrated the significant runtime acceleration achieved by both processing chains in the previous section. However, parallelization does not come without trade-offs in reconstruction quality. Generally, it is desirable to run SfM on large data sets because the number of reconstructed 3D points grows superlinearly with the size of the input image set. Moreover, bundle adjustment is an iterative process that refines the location of each point incrementally as more matched features are evaluated.

Fig. 6 shows the growth in the point cloud size as a function of image set size. Point cloud density for both parallel implementations grow at slower rates than the baseline. Here we see a clear trade-off of point cloud density for runtime speed-up.

Although parallelization compromises point cloud density, Table I shows that cluster-based SfM provides better convergence to the baseline point locations than VisualSFM.

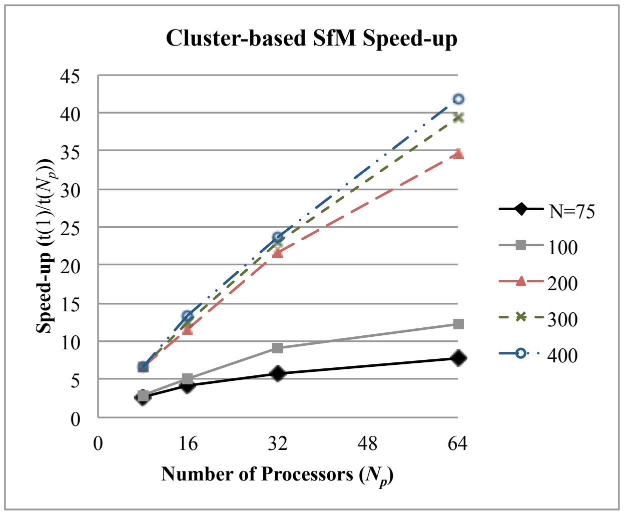


Fig. 5. Cluster-based SfM speed-up shown for various numbers of input images, N.

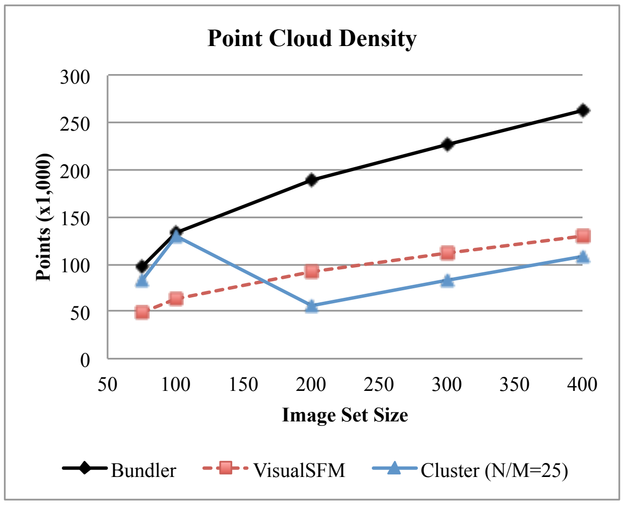


Fig. 6. Point cloud density increases with larger input image set size.

This is likely a result of the different optimization problem formulation employed by VisualSFM to improve parallelism and runtime efficiency. Error was calculated for each point cloud by comparing the 3D point location to that found by the baseline implementation. Corresponding points in the clouds produced by each processing chain were identified based on intersecting SIFT points in each points view list. Error is defined the distance (in meters) between the Bundler point

TABLE I
RMS ERROR OF RECONSTRUCTED POINTS (METERS)

Input size (N)		100	200	300	400
VisualSFM		0.20	7.15	1.05	0.44
Cluster-based SfM	$M = 4$	0.10			
	$M = 8$		0.27	0.26	0.25
	$M = 12$			0.26	
	$M = 16$				0.24

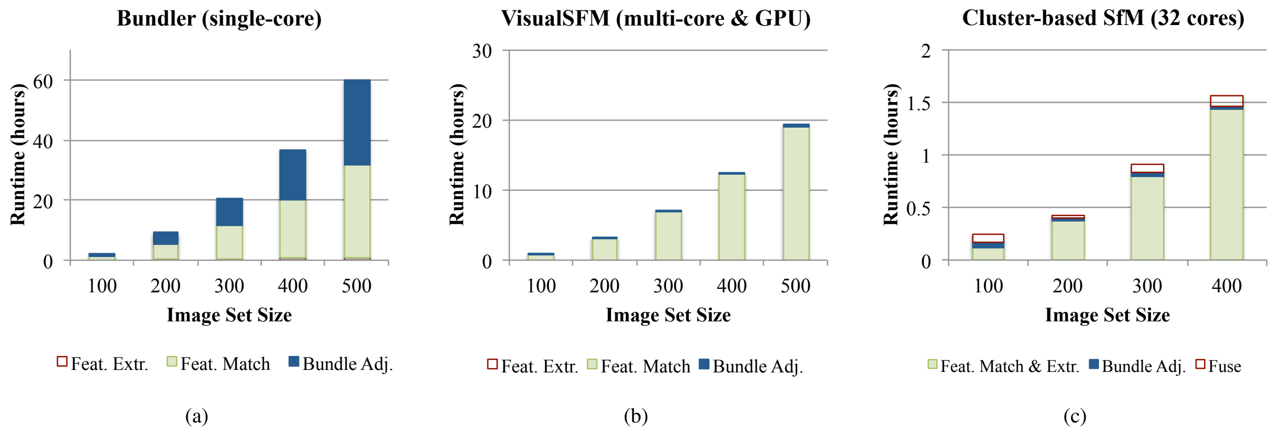


Fig. 4. Processing chain runtime (broken down by step) vs. number of input images for (a) Bundler, (b) VisualSFM and (c) cluster-based SfM (shown here for $N_p = 32$).

location and the corresponding point after scaling the clouds using the coarse geo-registration technique described in Sec. IV. Finally, RMS error is calculated for each point cloud generated.

VI. Conclusion

We have introduced a scalable, parallel SfM processing chain suitable for use with aerial video of a single ground point of interest. We have compared our technique to an alternative parallel implementation and evaluated both in terms of runtime efficiency and reconstruction quality. Practical SfM applications require scalable processing chains that can compute 3D reconstructions for very large data sets in a fixed processing time budget. We conclude that cluster-based SfM offers better scalability than VisualSFM. Our cluster-based approach can scale to run on a much larger system than VisualSFM, which is restricted to the number of CPU and GPU cores than can be integrated into a single machine and memory address space. The fine-grained parallelization exploited by multi-core bundle adjustment offers very impressive speed-up, but ultimately it is the feature matching step that poses the biggest bottleneck to both chains. Future work will focus on feature matching optimizations for different types of input data sets. Additionally, integration of fully automated precise point cloud geo-registration could be made possible by matching point clouds with reference data such as LIDAR or satellite imagery.

References

[1] "Photosynth," Microsoft Corporation, 2012, [Accessed April-2012]. [Online]. Available: http://photosynth.net

[2] H. Temeltas and D. Kayak, "SLAM for robot navigation," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 23, no. 12, pp. 16 –19, Dec. 2008.

[3] A. N. Vasile, L. J. Skelly, K. Ni, R. Heinrichs, and O. Camps, "Efficient city-sized 3D reconstruction from ultra-high resolution aerial and ground video imagery," in *Proceedings of the 7th International Conference on Advances in Visual Computing - Volume Part I*, ser. ISVC'11, 2011, pp. 347–358.

[4] K. Ni, Z. Sun, and N. Bliss, "3D image geo-registration using vision-based modeling," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, May 2011, pp. 1573 –1576.

[5] R. Kaminsky, N. Snavely, S. Seitz, and R. Szeliski, "Alignment of 3D point clouds to overhead images," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, June 2009, pp. 63 –70.

[6] R. Gherardi, M. Farenzena, and A. Fusiello, "Improving the efficiency of hierarchical structure-and-motion," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2010.

[7] N. Snavely, S. M. Seitz, and R. Szeliski, "Skeletal sets for efficient structure from motion," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2008.

[8] D. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher, "Discrete-continuous optimization for large-scale structure from motion," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2011.

[9] C. Wu, "VisualSFM: A visual structure from motion system," 2011. [Online]. Available: http://www.cs.washington.edu/homes/ccwu/vsfm/

[10] K. Ni, D. Steedly, and F. Dellaert, "Out-of-core bundle adjustment for large-scale 3D reconstruction," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, Oct. 2007, pp. 1 –8.

[11] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2003.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

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

[14] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *Proceedings of the 11th European conference on Computer vision: Part II*, ser. ECCV'10, 2010, pp. 791–804.

[15] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," http://cs.unc.edu/ ccwu/siftgpu, 2007.

[16] C. Wu, S. Agarwal, B. Curless, and S. Seitz, "Multicore bundle adjustment," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, June 2011, pp. 3057 –3064.