

Graph-Based Parallel Large Scale Structure from Motion

Yu Chen^a, Shuhan Shen^{b,c}, Yisong Chen^a and Guoping Wang^a

^aGraphics and Interactive Lab, Department of Computer Science and Technology, School of Electronic Engineering and Computer Science, Peking University

^bNational Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

^cUniversity of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Keywords:

clustering
Structure from Motion
Minimum Spanning Tree

ABSTRACT

While Structure from Motion achieves great success in 3D reconstruction, it still meets challenges on large scale scenes. Incremental SfM approaches are robust to outliers, but are limited by low efficiency and easy suffer from drift problem. Though Global SfM methods are more efficient than incremental approaches, they are sensitive to outliers, and would also meet memory limitation and time bottleneck. In this work, large scale SfM is deemed as a graph problem, where graph are respectively constructed in image clustering step and local reconstructions merging step. By leveraging the graph structure, we are able to handle large scale dataset in divide-and-conquer manner. Firstly, images are modelled as graph nodes, with edges are retrieved from geometric information after feature matching. Then images are divided into independent clusters by a image clustering algorithm, and followed by a subgraph expansion step, the connection and completeness of scenes are enhanced by walking along a maximum spanning tree, which is utilized to construct overlapping images between clusters. Secondly, Image clusters are distributed into servers to execute SfM in parallel mode. Thirdly, after local reconstructions complete, we construct a minimum spanning tree to find accurate similarity transformations. Then the minimum spanning tree is transformed into a Minimum Height Tree to find a proper anchor node, and is further utilized to prevent error accumulation. We evaluate our approach on various kinds of datasets and our approach shows superiority over the state-of-the-art in accuracy and efficiency. Our algorithm is open-sourced in <https://github.com/AIBluefisher/GraphSfM>.

1. Introduction

The study of Structure-from-Motion (SfM) has made rapid progress in recent years. It has achieved great success in small to medium scale scenes. However, reconstructing large scale datasets remains a big challenge in terms of both efficiency and robustness.

Since [1] has achieved great success and has become a milestone, incremental approaches have been widely used in modern SfM applications[38, 47, 30, 41, 35, 14, 27]. The geometric filtering combined with RANSAC [13] process can remove outliers effectively. Starting with a robust initial seed reconstruction, incremental SfM then adds camera one by one by PnP [24, 26]. After cameras are registered successfully, an additional bundle adjustment step is used to optimize both poses and 3D points [43], which makes incremental SfM robust and accurate. However, incremental SfM also becomes inefficient and would meet memory bottleneck on large scale datasets due to the repetitive optimization step. Besides, the manner of adding new views incrementally makes these kinds of approaches suffer from drift easily, though an additional re-triangulation step is used [47].

Global SfM approaches [17, 46, 6, 8, 3, 23, 45, 20, 18, 33, 31] have advantages over incremental ones in efficiency. When all available relative motions are obtained, global approaches first obtain global rotations by solving the rotation averaging problem efficiently and robustly [17, 19, 22, 21, 3, 4, 11, 45]. Then, global orientations and relative translations

are used to estimate camera translations(or camera centers) by translation averaging [46, 33, 16, 53]. With known camera poses, triangulation(re-triangulation might be required) can be performed to obtain 3D points and then only once bundle adjustment step is required. Though global approaches are efficiency, the shortcomings are obviously: translation averaging is hard to solve, as relative translations only decode the direction of translation and the scale is unknown; outliers are still a head-scratching problem for translation averaging, which is the main reason that prohibit the practical use of global SfM approaches.

To overcome the inefficiency problem in incremental SfM while to remain the robustness of reconstruction at the same time, a natural idea is to do reconstruction in a divide-and-conquer manner. A pioneer work that proposed this idea is [2] where images are first partitioned by graph cut and each sub-reconstruction is stitched by similarity transformation. Then followed by [51, 52] where both the advantages of incremental and global approaches are utilized in each sub-reconstruction. However, both these divide-and-conquer approaches are more focused on the local reconstructions and their pipelines are lack of global consideration, which designed the clustering step and merging step independently, thus may lead to the failure of SfM.

Inspired by these previous outstanding divide-and-conquer work [2, 40, 51, 52], we solve large scale SfM in a parallel mode while the whole pipeline is designed with a unified framework based on graph theory. We claim the novelties of the proposed framework are: (1) The image clustering algorithm, where we first cluster images in different groups, then further expand image clusters by walking along a MaxST.

✉ rainyChen@pku.edu.cn (Y. Chen); shshen@nlpr.ia.ac.cn (S. Shen); chenysisong@pku.edu.cn (Y. Chen); wgp@pku.edu.cn (G. Wang)
ORCID(s):

The image clustering allows the distribution of local SfM tasks. (2) The final local reconstructions merging algorithm, local reconstructions are accurately registered into a anchor node, by leveraging a MinST and a MHT, where the most accurate similarity transformations can be selected by the former, and the proper anchor node can be found by the latter. Specifically, first, images are divided into clusters with no overlap and each cluster is a graph node. Second, lost edges are collected and used to construct a maximum spanning tree (MaxST). Then, these lost edges are added along the MaxST to construct overlapped images and enhance the connections between clusters. Third, local SfM solvers are executed in parallel or distributed mode. At last, after all local SfM jobs finish, a novel sub-reconstructions merging algorithm is proposed for clusters registering. The most accurate $N - 1$ similarity transformations are selected within a minimum spanning tree (MinST) and a minimum height tree (MHT) is constructed to find a suitable reference frame and suppress the accumulated error.

Our contributions are mainly three folds:

- We proposed an robust image clustering algorithm, where images are clustered into groups of suitable size with overlap, the connectivity is enhanced with the help of an MaxST.
- We proposed a novel graph-based sub-model merging algorithm, where MinST is constructed to find accurate similarity transformations, and MHT is constructed to avoid error accumulation during the merging process.
- The time complexity is linearly related to the number of images, while most state-of-the-art algorithms are quadratic.

2. Related Work

Large scale SfM becomes popular since [1, 46], where they used unordered internet images as input, and utilized skeletal graph [38, 36] to avoid exhaustive feature matching [44, 48, 49].

Some exciting work in large scale reconstruction is hierarchical SfM approaches [12, 15, 42, 32, 5]. These approaches take each image as a leaf node. Point clouds and camera poses are merged from bottom to top. The principle of "smallest first" is adopted to produce a balanced dendrogram, which makes hierarchical approaches insensitive to initialization and drift error. However, due to insufficient feature matching [28, 29], the reconstructed scenes tend to lose scene details and become incomplete. Besides, the quality of reconstructions might be ruined by the selection of similar image pairs.

Some earlier work tries to solve large scale SfM via multi-cores [1], or to reduce the burden of exhaustive pairwise matching by building skeletal graphs [38]. Bhowmick [2] tried to solve large scale SfM in divide-and-conquer manner, and graph cut [9, 37] was adopted to do data partition. After all sub-reconstructions complete, additional cameras

are registered in each sub-reconstruction to construct overlapping areas and then to fuse them. It was then improved in [40] to cluster data set and merges each cluster by a distributed camera model [39, 40]. However, both [40, 2] either took no good consideration of the graph clustering strategy or neglected a careful design of clustering and merging algorithm, which makes reconstruction fragile and suffers from the drifting problem. Moreover, the loss of connections between different components makes the reconstruction fragile. Besides, the similarity score that is used as the weight in graph partition reduces the reliability of the result. One drawback that should be noticed is that the incremental merging process suffers from drifted errors, as well as traditional incremental approaches.

Follow the work of Bhowmick [2], [51, 52] augment the graph cut process in [2, 40] by a two steps process: binary graph cut and graph expansion. In their work, the graph cut step and graph expansion step alternated and then converged when both size constraint and completeness constraint are satisfied. Then, components are registered by global motion averaging [51]. However, the translation averaging at the cluster level still suffers from outliers and may lead to disconnected models. This work was further improved in [52], where it adopted the clusters registering approach in [2], and then camera poses were divided into intra-cameras and inter-cameras for motion averaging, which can improve the convergence rate of final bundle adjustment [43, 10, 34, 50].

3. Graph-Based Structure from Motion

To deal with large scale datasets, we adopt the divide-and-conquer strategy that is similar to [2, 51]. For the sake of completeness and efficiency of reconstruction, we propose to use a unified graph framework to solve the image clustering and sub-reconstructions merging problem. The pipeline of our SfM algorithm is shown in Fig.1. Firstly, we extract features and use them for matching. Epipolar geometries are estimated to filter matching outliers. After feature matching, we use our proposed image clustering algorithm to divide images into different groups. Then, clusters can be reconstructed by local SfM in parallel. After all local reconstructions are completed, we need to merge them together as each local map has their own coordinate frame. To avoid producing disconnected scenes after merging, all local reconstructions are merged with our graph-based merging algorithm. When we obtained a global map and camera poses, we can further re-triangulation more landmarks to recover more scene details. To minimize the reprojection error of the global map, a final bundle adjustment should be performed. The re-triangulation step and bundle adjustment step can be executed alternatively until convergence. In practice, it is enough to perform them once. We describe more details of our algorithm in the following subsections.

3.1. Image Clustering

We aim to group m images into k clusters, each cluster is under the memory limitation of the computer. Besides,

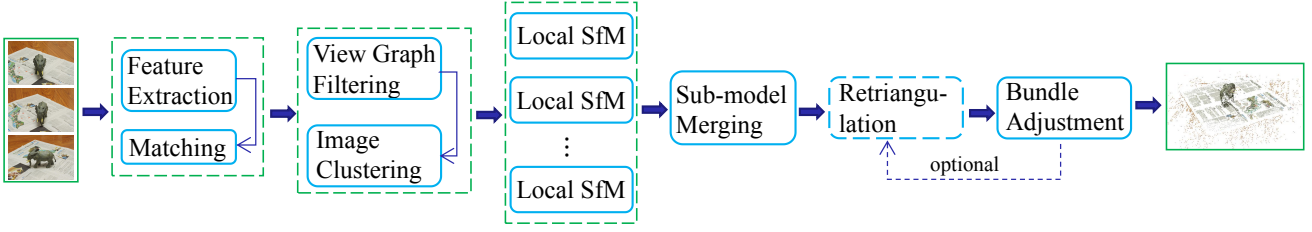


Figure 1: Our proposed SfM pipeline. We first extract and match features. Epipolar geometries are estimated to filter matching outliers. After feature matching, we use our proposed image clustering algorithm to divide images into different groups. Then, clusters can be reconstructed by local SfM in parallel. After all local reconstructions are completed, we need to merge them together-as each local map has their own coordinate frame. And to avoid producing disconnected scenes after merging, all local reconstructions are merged with our graph-based merging algorithm. When we obtained a global map and camera poses, we can further re-triangulation more landmarks to recover more scene details. To minimize the reprojection error of the global map, a final bundle adjustment should be performed. The re-triangulation step and bundle adjustment step can be executed alternatively until convergence.

each cluster should be reconstructed as accurately as possible, and not be influenced largely by the loss of geometric constraints. In this section, we share a simple but quite effective approach for a two steps image clustering: (1) Graph cutting. (2) Images expansion based on the maximum spanning tree. Both the two steps are based on the intuitive graph theory. Besides, we utilize two conditions proposed in [51] to constraint the clustering steps: size constraint and completeness constraint. Size constraint gives the upper bound of images in each cluster. Completeness constraint is defined as $\eta(i) = \frac{\sum_{j \neq i} |C_i \cap C_j|}{C_i}$, where C_i, C_j respectively represents cluster i, j . Unlike the image clustering algorithm proposed in [51], which alternates between graph cut and graph expansion, we just perform once graph cut and once subgraph expansion. And we claim the novelty of our expansion step is using a MaxST to assist the final fusing step.

3.1.1. Image Clustering

In graph cutting step, each image is deemed as a graph node, and each edge represents the connection between images. In the case of SfM, it can be represented by the results of two view geometries. The weight of edges is the number of matches after geometric filtering. To solve this problem, the connection between images can be deemed as edges inside a graph. Suppose each camera is a graph node, the connection between two cameras can be deemed as a weighted edge(Here we referred to as *image edges*). Consider the size constraint, each cluster should have a similar size. The *image clustering* problem can be solved by graph cut [9, 37]. To enhance the connection of clusters and to align them together, an additional expansion step should be executed. In our case, we need to expand these independent clusters with some common images(we referred to as overlapping area), and then to compute similarity transformations to fuse them together. As the iterative approach of Zhu [51] is time-consuming, we proposed our one step approach in expansion procedure in the next subsection.

3.1.2. Subgraph Expansion

In the subgraph expansion step, we generalize the graph into clusters level. Each cluster represents a node, the edges between clusters are the lost edges after graph cut (Here we refer as *cluster edges*). First, we collect all the lost edges $E_{\text{lost}} = \{\mathcal{E}_{k_1 k_2, i} | k_1, k_2 \in [K], i \in |\mathcal{E}_{k_1 k_2}|\}$ for cluster pairs, where K is the number of clusters. Then, we construct a graph, the weight of the cluster edge is the number of lost image edges between two clusters. Intuitively, if there are more image edges are lost inside pairwise clusters, we prefer to construct connections for them to avoid the loss of information. With that in mind, once we obtain the cluster graph, a MaxST is constructed to induce the expansion step. We gather the image edges from the MaxST and sort them by descending order. We then add the lost image edges into clusters where completeness constraint is not satisfied, and only the top- k edges are added. At last, we check all clusters and collect them together if the completeness constraint of any of them is not satisfied. For cluster edges that are not contained in the MaxST, we select them randomly and add the image edges into these clusters in a similar way.

The procedure of our image clustering algorithm is shown in Fig.2. In Fig. 2(a), The images graph is first grouped by using graph cut algorithm, where edges with weak connections tend to be removed. In Fig. 2(b), The cluster graph after graph cutting, where nodes are clusters and edges are lost edges in images graph, the number of lost edges are edge weights. In Fig. 2(c), the solid lines represent the edges of a constructed maximum spanning tree. The dotted line could be added to enhance the connectivity of clusters. Fig. 2(d) shows the final expanded image clusters. The complete image clustering algorithm is given in Alg. 1.

3.2. Graph-based Local Reconstructions Merging

After image clustering, each cluster can be reconstructed using a local SfM approach. Due to the robustness to outliers, we choose incremental SfM. As the reconstructed images are bounded below a threshold, the drift problem is alleviated. When all clusters are reconstructed, we need a final step to stitch them, as each cluster has its local coordinate

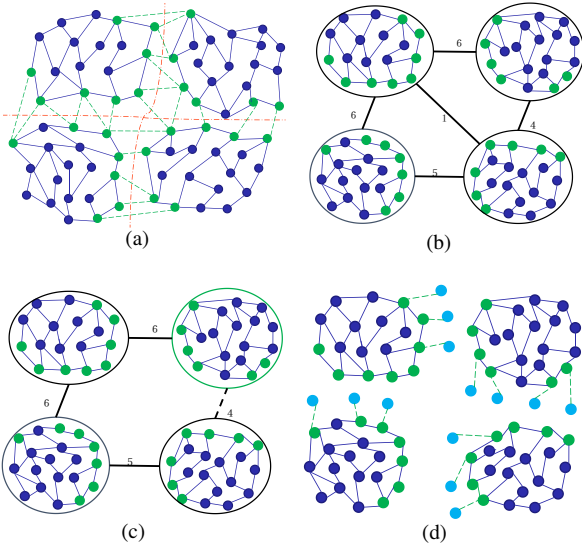


Figure 2: The image clustering procedure. (a). The images graph is grouped by NCut, where edges with weak connections are tend to be removed. (b). The cluster graph after graph cutting, where nodes are clusters and edges are lost edges in images graph, the number of lost edges are edge weights. (c). The solid lines represent the edges of constructed maximum spanning tree. The dotted line could be added to enhance connectivity of clusters. (d) The final expanded images clusters.

Algorithm 1 Image Clustering Algorithm

Input: an initial image graph $\mathcal{G} := \{(\mathcal{V}, \mathcal{E})\}$, maximum number of cluster size S_{\max} , completeness ratio C , number of overlapped images between two clusters \mathcal{O}_{\max} , number of images n .

Output: image clusters with intersection $\mathcal{G}_{\text{inter}} = \{\mathcal{G}_k\}$

- 1: $K := \lfloor n/S_{\max} \rfloor$
- 2: $\mathcal{G}_{\text{intra},i} := \text{GraphPartition}(\mathcal{G}), \mathcal{G}_{\text{intra},i} \in \mathcal{G}_{\text{intra}}, i \in [K]$
- 3: Collect lost edges $E_{\text{lost}} := \{\mathcal{E}_{k_1 k_2} | k_1, k_2 \in [K]\}$
- 4: Build cluster graph $\mathcal{G}_{\text{cluster}} := \emptyset$ from E_{lost}
- 5: $E_{\text{mst}} := \text{Kruskal}(\mathcal{G}_{\text{cluster}}), i := 0, j := 0, \mathcal{G}_{\text{inter}} := \mathcal{G}_{\text{intra}}$
- 6: **while** $i < |E_{\text{mst}}|$ **do**
- 7: edge $\mathcal{E}_{k_1 k_2, i} = E_{\text{mst}, i}$
- 8: $\mathcal{G}_{\text{inter}}^{k_1}, \mathcal{G}_{\text{inter}}^{k_2}$ add lost edges in $\mathcal{E}_{k_1 k_2, i}$
- 9: $i \leftarrow i + 1$
- 10: **while** completeness constraint not satisfied **do**
- 11: Select edge $\mathcal{E}_{k_1 k_2, r}$ from E_{lost} randomly
- 12: $\mathcal{G}_{\text{inter}}^{k_1}, \mathcal{G}_{\text{inter}}^{k_2}$ add lost edges in $\mathcal{E}_{k_1 k_2, r}$

frame.

To construct a robust merging algorithm, we consider three main problems:

- A cluster should be selected as the reference frame, which we referred to as **anchor node**.
- The merging step from other clusters to the anchor node should be as accurate as possible.

- As there may not exist overlap between anchor node and some other clusters, we have to find a path to merge them into an anchor node. Due to the accumulated errors, the path of each cluster to the anchor node shouldn't be too long.

To deal with the above problem, we construct a graph on the cluster level. The algorithm is composed of three main steps: (1) Cluster graph initialization. (2) Anchor node searching. (3) Path computation and simplification. For cluster graph initialization, we first find the common cameras between pairwise clusters, and compute the similarity transformations. Then we build a MinST to select the most accurate edges. We found the anchor node by dealing with a minimum height tree (MHT) [25] problem. We first show how the problem can be constructed into an MinST problem.

3.2.1. Pairwise Similarity Transformation

We have discussed how to construct overlapping areas in Sec. 3.1, we further utilize the overlapped information to compute the pairwise similarity transformation. Given correspondences of camera poses, i.e., $\{P_{i_{k_1}}^{k_1}\}$ and $\{P_{j_{k_2}}^{k_2}\}$, we first estimate the relative scale. With relative scale known, the similarity estimation thus degenerated to euclidean estimation.

Relative Scale Estimation To estimate relative scale, we need at least two points correspondences, $(C_{i_{k_1}}^{k_1}, C_{j_{k_1}}^{k_1})$ and $(C_{i_{k_2}}^{k_2}, C_{j_{k_2}}^{k_2})$, we can estimate the relative scale by

$$\hat{s}_{k_1 k_2} = (C_{i_{k_1}}^{k_1} - C_{j_{k_1}}^{k_1}) / (C_{i_{k_2}}^{k_2} - C_{j_{k_2}}^{k_2}). \quad (1)$$

As there may exists outliers, we choose $s_{k_1 k_2}$ as

$$s_{k_1 k_2} = \text{median}\{\hat{s}_{k_1 k_2}\}. \quad (2)$$

Euclidean Transformation Estimation When the relative scale is known, the similarity transformation degenerates to euclidean estimation. That is, we only need to estimate the relative rotation and relative translation. Suppose a 3D point X is located in global coordinate frame, and X_1, X_2 is located in local coordinate frame k_1, k_2 by two euclidean transformation $(R_1, t_1), (R_2, t_2)$ respectively. Then we have

$$X_1 = R_1 X + t_1, \quad X_2 = R_2 X + t_2. \quad (3)$$

We can further obtain

$$\begin{aligned} X_2 &= R_2 R_1^{-1} (X_1 - t_1) + t_2 \\ &= R_2 R_1^T X_1 + (t_2 - R_2 R_1^T t_1). \end{aligned} \quad (4)$$

Then, the relative transformation is

$$R_{12} = R_2 R_1^T, \quad t_{12} = t_2 - R_2 R_1^T t_1. \quad (5)$$

Because cluster k_1 and k_2 are up to a scale $s_{k_1 k_2}$, we should reformulate the relative transformation as

$$\begin{aligned} t_{k_1 k_2} &= t_{k_2} - R_{k_2} R_{k_1}^T t_{k_1} \\ &= -R_{k_2}^T C_{k_2} + s_{k_1 k_2} R_{k_2} R_{k_1}^T R_{k_1} C_{k_1} \\ &= s_{k_1 k_2} C_{k_1} - R_{k_2}^T C_{k_2}, \end{aligned} \quad (6)$$

where C_{k_1} and C_{k_2} are camera centers in cluster k_1, k_2 respectively. To handle the existence of outliers, we combined the euclidean estimation with RANSAC.

3.2.2. Cluster Graph Initialization

In our approach, each cluster is deemed as a graph node, and the edges connect the nodes sharing some common cameras. Assume that there are k clusters of cluster graph \mathcal{C} , and we denote the probability of obtaining a good transformation from the cluster pair (i, j) as p_{ij} . Consequently, the probability that all $k - 1$ edges in a spanning tree c can be reconstructed is approximated as

$$P = \prod_{i=1 \dots k-1} p_{c_{k_1} c_{k_2}}, \quad (7)$$

where c_{k_1} and c_{k_2} are two clusters associated with the k -th edge of c . P can be considered as the probability that a global 3D reconstruction can be reached provided all spanning pairs are correctly reconstructed. Then we try to maximize the probability P defined in Equ.(7). This is equivalent to minimizing the cost function

$$f = -\log \left(\prod_{i=1 \dots k-1} p_{c_{k_1} c_{k_2}} \right). \quad (8)$$

To solve for the optimal spanning tree, we define the weight of an edge connecting clusters i and j as

$$w_{ij} = -\log(p_{ij}). \quad (9)$$

Now the problem of maximizing the joint probability P is converted to the problem of finding a MinST in an undirected graph. Note that in MinST computing, the concrete value of the edge weights does not matter but the order of them does. That is, A reasonably comparable strengths of connections between clusters, instead of an accurate estimate of P , is sufficient to help generate a good spanning tree. This observation leads us to the following residual-error weight definition scheme.

Residual Error As a reliable measure of the goodness of cluster merging, we use the Mean Square Distance (MSD) to help define the edge weight. The Mean Square Error (MSE) from cluster k_1 to cluster k_2 is defined as:

$$\text{mse}_{k_1 k_2} = \frac{1}{2n} \left(\sum_{i=1}^n \|T_{k_1, k_2} X_i^{k_1} - X_i^{k_2}\|_2^2 \right)^{\frac{1}{2}}, \quad (10)$$

where T_{k_1, k_2} is the similarity transformation from cluster k_1 to cluster k_2 , X_i^k is the i -th common points in cluster k . Equ.(10) describes the transformation error from cluster k_1 to cluster k_2 . To convert MSE to a symmetric metric, we use the maximum of mse to define the MSD:

$$\text{msd}(k_1, k_2) = \max(\text{mse}_{k_1 k_2}, \text{mse}_{k_2 k_1}). \quad (11)$$

Then the edge weight between vertices k_1 and k_2 in \mathcal{C} is defined $\text{msd}(k_1, k_2)$.

3.2.3. Minimum Height Tree Construction

After computing all the weights, the graph initialization process has been completed. Then, we can construct an MinST by Kruskal algorithm to select the most accurate $N - 1$ similarity transformations. After finding an MinST, We need to find a base node as the reference of the global alignment of all clusters in the MinST. We impose restrictions on the selection of the base node: (1) The base node should be suitably large. (2) The path from the other nodes to the base node shouldn't be too long. The first constraint is considered for efficiency. The second constraint is used to avoid error accumulation. Taking a similar idea of **Minimum Height Tree (MHT)** in [25], We convert the problem of finding the base node into an MHT problem. We first introduce the concept of MHT.

Definition 1. For an undirected graph with tree characteristics, we can choose any node as the root. The resulting graph is then a rooted tree. Among all possible rooted trees, those with minimum height are called minimum height trees (MHTs).

We solve the MHT problem by merging the leaf nodes layer by layer. At each layer, we collect all the leaf nodes and merged them into their neighbors. At last, there may be two or one nodes left. If there are two nodes left, then we choose the node that has a larger size as the base node. If there is only one node left, then the node is the base node.

The merging process is depicted by Fig.3. The advantage of using algorithm to find the base node is depicted in Fig.4. Owing to the robustness of our algorithm, which can find accurate similarity transformations and the edges which have large msd are filtered, we are able to merge all sub-reconstructions accurately. The full sub-model merging algorithm is illustrated in Alg.2.

Algorithm 2 Graph-Based Local Reconstructions Merging Algorithm

Input: Clusters $\mathcal{C} := \{C_k\}$, Corresponding 3D points $\mathcal{X} := \{X_{ik} \mid i \in [n]\}$, $\mathcal{X}' := \{X'_{ik} \mid i \in [n]\}$, where $k \in [m]$

Output: The final merged cluster C_{final}

- 1: Initialize Cluster Graph $\mathcal{G} := \{\mathcal{N}, \mathcal{E}\}$
- 2: Construct an MinST \mathcal{M} by Kruskal algorithm from Cluster Graph \mathcal{G}
- 3: $i := 0$
- 4: **while** $|\mathcal{M}| > 1$ **do**
- 5: Find all leaf nodes $\{\mathcal{N}_{\text{leaf}}^i\}$ and their connected nodes $\{\mathcal{N}_{\text{con}}^i\}$
- 6: Replace $\{\mathcal{N}_{\text{con}}^i\}$ with $\{\mathcal{N}_{\text{leaf}}^i + \mathcal{N}_{\text{con}}^i\}$
- 7: Remove nodes $\mathcal{N}_{\text{leaf}}^i$
- 8: **if** $|\mathcal{M}| = 2$ **then**
- 9: Select the cluster that has a larger size as the base node
- 10: Remove the other node
- 11: **break**
- 12: $i := i + 1$

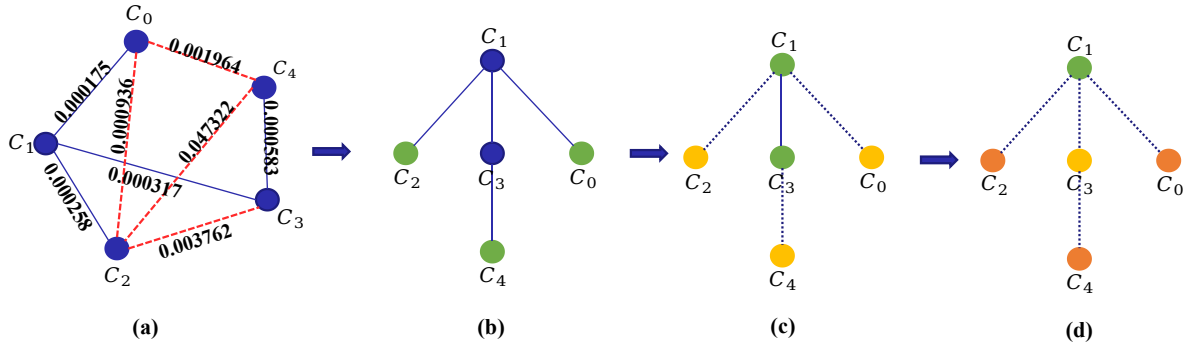


Figure 3: The sub-reconstructions merging process. (a) shows the constructed MinST, where dotted lines represent the edges that have large msd. In (b), leaf nodes (c_0, c_2, c_4) are denoted by green, c_0 and c_2 are merged into c_1 , c_4 is merged into c_3 . In (c), the nodes that have been merged are marked by yellow, c_1 and c_3 are leaf nodes now. In (d), the leaf nodes which have been merged in the first layer are marked by dark yellow, and leaf nodes merged in second layer is marked by yellow. As there are two leaf nodes left, we choose the node with larger size (suppose $|c_1| > |c_3|$, then c_3 should be merged into c_1 , and c_1 is the base node.)

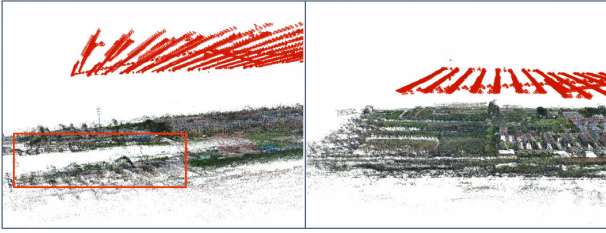


Figure 4: The alignment results with and without our graph-based local reconstructions merging algorithm.

4. Experiments

In this section, we evaluate our GraphSfM on different kinds of datasets, including ambiguous datasets and large scale aerial datasets.

4.1. Experimental Environments

Our *GraphSfM* algorithm is implemented based on COLMAP [35], and we use the incremental pipeline of COLMAP to reconstruct local clusters. All the experiments are performed on a PC with 4 cores Intel 7700 CPU and 32GB RAM. Besides, we use SIFT [28] that was implemented by vlfeat¹ to extract feature points for all the evaluated SfM approaches.

4.2. Datasets Overview

To evaluate the robustness and efficiency of our algorithm, we first construct and collect some different kinds of datasets. The first kind of datasets are collected from 9 outdoor scenes, which include small scale and medium scale datasets, and the number of images is from 60 to 2248. The second kind of datasets are collected from public datasets, which include outdoor scenes (Gerrard Hall, Person Hall, South Building) [35] and ambiguous scenes (Stadium and Heaven Temple) [36]. The last kind of datasets are 3 large

¹<https://www.vlfeat.org/api/sift.html>

scale aerial datasets, where the memory requirement and efficiency are challenges for traditional approaches.

4.3. Efficiency and Robustness Evaluation

We evaluated the efficiency of our algorithm over 2 state-of-the-art incremental SfM approaches (TheiaSfM [41] and COLMAP [35]), and 2 state-of-the-art global SfM approaches (1DSfM [46] and LUD [33]). For sake of fairness, our GraphSfM runs on one computer, though it can run in a distributed mode. The evaluation results are shown in Fig. 5 and table 1. It's not surprising that the incremental approaches take more time for reconstruction than global approaches. As the dataset scale increases, the time that is taken by COLMAP [35] grows rapidly, due to the repetitive and time-consuming bundle adjustment [43] step. Though our approach is a kind of incremental one, the scale of the images can be controlled to a constant size in each cluster. Thus the time of bundle adjustment can be highly reduced, and the time grows linearly as the number of images grows. Though TheiaSfM [41] is also an incremental SfM, it selects some good tracks [7] to perform bundle adjustment, which saves a lot of time but might become unstable in some cases. Besides, the time taken by TheiaSfM surpasses our GraphSfM when the scale of the image is over 2000. Table 1 gives more details of reconstruction results. It is obvious that our GraphSfM is as robust as COLMAP in terms of reconstructed cameras and is more accurate than other approaches in terms of reprojection errors. These facts illustrate the superior performance of our GraphSfM to handle large scale datasets. We emphasize that our algorithm just run on one computer and the reconstruction time could be reduced largely if we run it on more computers in distributed manner.

4.4. Evaluation on Public Datasets

We evaluated our algorithm on several public datasets [35, 36]. For these small scale datasets, we only run our GraphSfM on one computer. Some visual results are shown in Fig. 6 and statistics are given in table 2. COLMAP again is the most inefficient approach, and our approach is 1.2 - 3

Table 1

Efficiency and accuracy evaluation with datasets which have different scales. N_p , N_c , T represent the number of 3D points, the number of recovered cameras and the reconstruction time, respectively. Err represents the reprojection error and the best results are highlighted by bold font.

dataset	Images	COLMAP [35]				TheiaSfM [41]				1DSfM [46]				LUD [33]				Ours			
		N_c	N_p	Err	T	N_c	N_p	Err	T	N_c	N_p	Err	T	N_c	N_p	Err	T	N_c	N_p	Err	T
DS-1	60	60	16387	0.48	26.22	60	8956	1.92	10.93	60	8979	1.92	1.32	60	8979	1.92	1.36	60	13923	0.46	24.48
DS-2	158	158	68989	0.42	170.34	158	39506	1.91	87.71	157	39527	1.92	7.76	158	39517	1.92	7.95	158	62020	0.44	168.48
DS-3	214	214	71518	0.512	122.64	138	6459	1.70	45.89	187	7080	1.54	4.25	162	5099	1.45	1.69	214	68882	0.49	121.56
DS-4	319	319	154702	0.50	529.14	204	11550	1.80	186.08	290	142967	1.82	17.53	270	13484	1.76	18.77	319	151437	0.47	482.40
DS-5	401	370	166503	0.58	568.68	305	23742	1.97	241.61	348	23081	1.88	18.89	316	22160	1.85	17.93	370	164495	0.55	562.74
DS-6	628	628	268616	0.39	562.74	628	133300	1.92	421.23	610	133146	1.91	34.80	628	133747	1.91	35.03	628	259333	0.39	605.58
DS-7	704	703	345677	0.58	1918.86	449	35659	1.86	603.85	641	42716	1.93	108.15	547	34296	1.91	97.82	703	346394	0.55	1839.90
DS-8	999	980	419471	0.52	1918.86	733	40246	1.86	731.84	745	172864	1.77	77.80	611	31254	1.74	70.78	980	416512	0.50	2570.34
DS-9	2248	2248	1609026	0.63	71736	2248	187392	2.47	7255.70	2247	188102	2.48	667.59	2248	188134	2.47	694.74	2242	1445227	0.65	6108.06

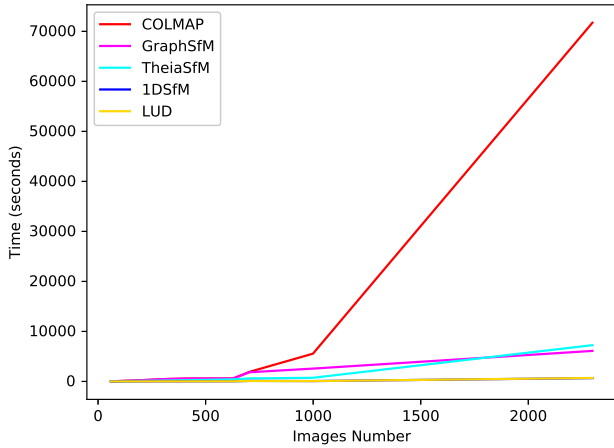


Figure 5: Efficiency evaluation on datasets with different scales.

times faster than COLMAP, though we only run it by one computer. TheiaSfM selects good tracks [7] for optimization and the two global approaches are most efficient. However, as shown in Fig. 6, we can see that both the global approaches failed in Person Hall and Guangzhou Stadium datasets, which shows global approaches are easily disturbed by outliers. As an incremental approach, TheiaSfM also failed in Person Hall and Guangzhou Stadium datasets. Our approach is as robust as COLMAP, however, more efficient than it.

Ambiguous Datasets It's a challenging work to reconstruct on ambiguous datasets for SfM approaches. Though feature matches are filtered by geometric constraints, there are still lots of wrong matches pass the verification step. As is shown in Fig.7, our GraphSfM shows advantages to traditional SfM approaches in this kind of datasets. Due to the image clustering step, some wrong edges are discarded in clusters, thus the individual reconstructions are not affected by the wrong matches. However, it is hard to detect wrong matches in traditional SfM approaches, especially in self-similar datasets or datasets with repeated structures, which is the major reason for the failure in ambiguous datasets.

4.5. Evaluation on Large Scale Aerial Datasets

Our approach is also evaluated on large scale aerial datasets. We evaluated our algorithm both on one computer sequentially (each cluster is reconstructed one by one) and on three computers in parallel mode. The reconstruction results are given in table 3. Our algorithm can recover the same number of cameras as well as COLMAP. Besides, when running on a single computer, our approach is about 6 times faster than COLMAP; when running on three computers in distributed manner, our algorithm is about 17 times faster than COLMAP. Our algorithm can be further accelerated if we own more computer resources. TheiaSfM is slightly slower than our approach, however, the the number reconstructed 3D points is one magnitude less than our approach. Though 1DSfM and LUD are still the most efficient, their robustness would meet challenge in large scale aerial datasets.

5. Conclusion

In this article, we proposed a new SfM pipeline called GraphSfM, which is based on graph theory, and we designed a unified framework to solve large scale SfM tasks. Our two steps graph clustering algorithm enhances the connections of clusters, with the help of a MaxST. In the final fusing step, the construction of MinST and MHT allows us to pick the most accurate similarity transformations and to alleviate the error accumulation. Thus, our GraphSfM is highly efficient and robust to large scale datasets, and also show superiority in ambiguous datasets when compared with traditional state-of-the-art SfM approaches. Moreover, GraphSfM can be implemented on a distributed system easily, thus the reconstruction is not limited by the scale of datasets.

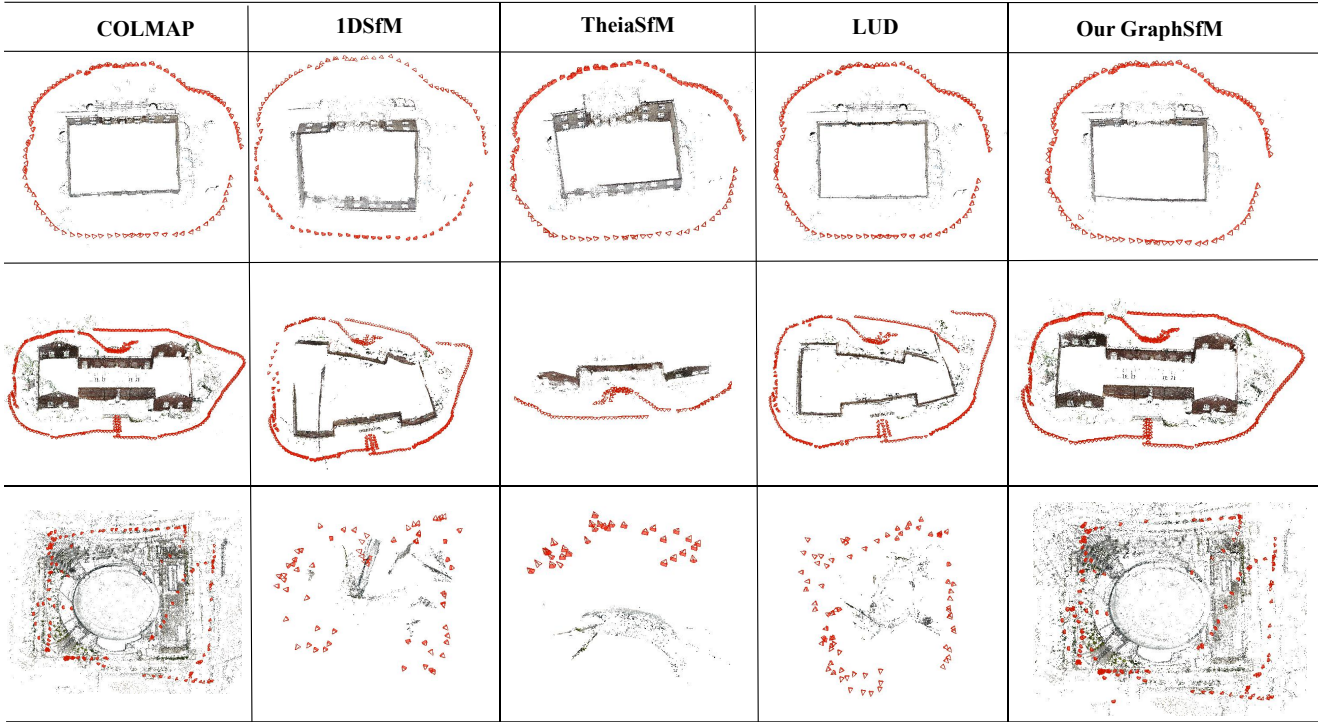


Figure 6: Reconstruction results on temple public datasets. From top to bottom are respectively Gerrard Hall, Person Hall and Guangzhou Stadium datasets.

Table 2

Comparison of reconstruction results. N_p, N_c represents the number of 3D points and the number of recovered cameras, respectively. $T_{GC}, T_{SfM}, T_{PCA}, T_{\Sigma}$ respectively denotes the time cost (seconds) of graph cluster step, local SfM step, point clouds alignment step, and the total time.

dataset	Images	COLMAP [35]			TheiaSfM [41]			1DSfM [46]			LUD [33]			Ours					
		N_c	N_p	T_{Σ}	N_c	N_p	T_{Σ}	N_c	N_p	T_{Σ}	N_c	N_p	T_{Σ}	N_c	N_p	T_{GC}	T_{SfM}	T_{BA}	T_{Σ}
Gerrard Hall	100	100	42795	303.07	100	50232	93.35	99	49083	15.82	100	44844	13.82	100	42274	0.01	114	3.85	118.68
Person Hall	330	330	141629	1725.80	113	39101	157.42	42	6239	768.75	325	93386	107.56	330	140859	0.04	713.94	25.13	742.92
South Building	128	128	61151	303.06	128	68812	155.84	128	436640	27.71	128	69110	34.70	128	58483	0.03	125.28	4.75	131.28
Stadium	157	157	85723	418.74	30	6345	18.73	65	6319	5.56	77	4549	4.74	157	71605	0.03	403.86	16.167	421.62
Heaven Temple	341	341	185750	8678.76	336	1201	46039	339	13356	40.96	340	14019	44.09	341	181583	0.04	2784.38	46.737	2856.26

References

- [1] Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R., 2009. Building rome in a day, in: IEEE International Conference on Computer Vision, pp. 72–79.
- [2] Bhowmick, B., Patra, S., Chatterjee, A., Govindu, V.M., Banerjee, S., 2014. Divide and conquer: Efficient large-scale structure from motion using graph partitioning, in: Asian Conference on Computer Vision, pp. 273–287.
- [3] Chatterjee, A., Govindu, V.M., 2013. Efficient and robust large-scale rotation averaging, in: IEEE International Conference on Computer Vision, pp. 521–528.
- [4] Chatterjee, A., Govindu, V.M., 2018. Robust relative rotation averaging. IEEE Trans. Pattern Anal. Mach. Intell. 40, 958–972.
- [5] Chen, Y., Chan, A.B., Lin, Z., Suzuki, K., Wang, G., 2017. Efficient tree-structured sfm by RANSAC generalized procrustes analysis. Computer Vision and Image Understanding 157, 179–189.
- [6] Crandall, D.J., Owens, A., Snavely, N., Huttenlocher, D.P., 2013. Sfm with mrfs: Discrete-continuous optimization for large-scale structure from motion. IEEE Trans. Pattern Anal. Mach. Intell. 35, 2841–2853.
- [7] Cui, H., Shen, S., Hu, Z., 2017. Tracks selection for robust, efficient and scalable large-scale structure from motion. Pattern Recognition 72, 341–354.
- [8] Cui, Z., Tan, P., 2015. Global structure-from-motion by similarity

Table 3

Comparison of reconstruction results. N_p, N_c represents the number of 3D points and the number of recovered cameras, respectively. T denotes the total time and T_d denotes the total time that is evaluated in a distributed system.

dataset	Images	COLMAP [35]			TheiaSfM [41]			1DSfM [46]			LUD [33]			Ours			
		N_c	N_p	T	N_c	N_p	T	N_c	N_p	T	N_c	N_p	T	N_c	N_p	T	T_d
Aerial-5155	5155	5155	1798434	41823.27	4383	203942	3527.43	4591	243490	482.45	4723	278924	390.59	5155	1834875	2491.78	936.74
Aerial-7500	7500	7455	5184368	95007.59	5327	432347	6237.91	6264	478234	931.84	5934	467230	832.40	7455	4968142	5834.37	2166.59
Aerial-12306	12306	11259	3934391	146172	8347	478237	25783.12	8923	509543	4941.27	8534	489238	4589.73	11259	3916724	22663.86	8970.01

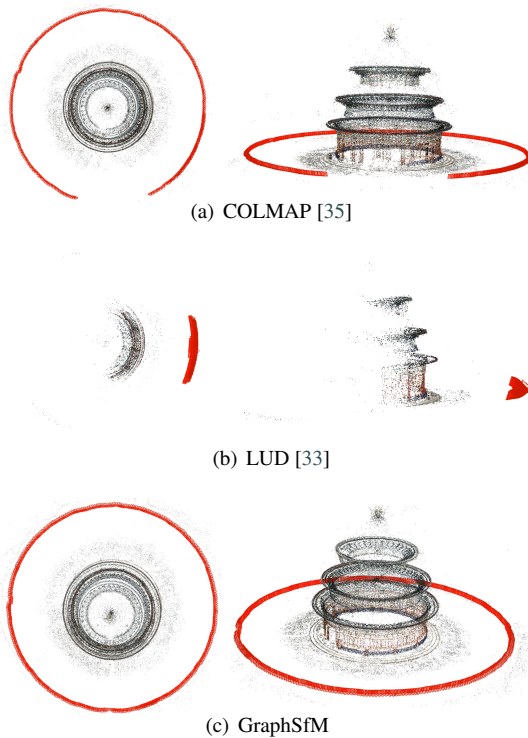


Figure 7: Reconstructions on temple heaven dataset. (a): The result reconstructed by COLMAP [35]. (b): The result reconstructed by LUD [33]. (c): The result reconstructed by our GraphSfM.

averaging, in: IEEE International Conference on Computer Vision, pp. 864–872.

[9] Dhillon, I.S., Guan, Y., Kulis, B., 2007. Weighted graph cuts without eigenvectors A multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 1944–1957.

[10] Eriksson, A.P., Bastian, J., Chin, T., Isaksson, M., 2016. A consensus-based framework for distributed bundle adjustment, in: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1754–1762.

[11] Eriksson, A.P., Olsson, C., Kahl, F., Chin, T., 2018. Rotation averaging and strong duality, in: *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 127–135.

[12] Farenzena, M., Fusiello, A., Gherardi, R., 2009. Structure-and-motion pipeline on a hierarchical cluster tree, in: *IEEE International Conference on Computer Vision Workshops*.

[13] Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 381–395.

[14] Gao, X., Hu, L., Cui, H., Shen, S., Hu, Z., 2018. Accurate and efficient ground-to-aerial model alignment. *Pattern Recognit.* 76, 288–302.

[15] Gherardi, R., Farenzena, M., Fusiello, A., 2010. Improving the efficiency of hierarchical structure-and-motion, in: *Computer Vision & Pattern Recognition*.

[16] Goldstein, T., Hand, P., Lee, C., Voroninski, V., Soatto, S., 2016. Shapefit and shapekick for robust, scalable structure from motion, in: *Computer Vision - ECCV 2016 - 14th European Conference*, pp. 289–304.

[17] Govindu, V.M., 2001. Combining two-view constraints for motion estimation, in: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 218–225.

[18] Govindu, V.M., 2004a. Lie-algebraic averaging for globally consistent motion estimation, in: *IEEE Computer Society Conference on Computer Vision & Pattern Recognition*.

[19] Govindu, V.M., 2004b. Lie-algebraic averaging for globally consistent motion estimation, in: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 684–691.

[20] Govindu, V.M., 2006. Robustness in motion averaging, in: *Asian Conference on Computer Vision*.

[21] Hartley, R.I., Aftab, K., Trunpf, J., 2011. L1 rotation averaging using the weiszfeld algorithm, in: *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, pp. 3041–3048.

[22] Hartley, R.I., Trunpf, J., Dai, Y., Li, H., 2013. Rotation averaging. *International Journal of Computer Vision* 103, 267–305.

[23] Havlena, M., Torii, A., Pajdla, T., 2010. Efficient structure from motion by graph optimization, in: *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision*, Heraklion, Crete, Greece, September 5–11, 2010, Proceedings, Part II, pp. 100–113.

[24] Kneip, L., Scaramuzza, D., Siegwart, R., 2011. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation, in: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2969–2976.

[25] Laber, E.S., Nogueira, L.T., 2004. On the hardness of the minimum height decision tree problem. *Discrete Applied Mathematics* 144, 209–212.

[26] Lepetit, V., Moreno-Noguer, F., Fua, P., 2009. *Eppn*: An accurate $O(n)$ solution to the pn p problem. *International Journal of Computer Vision* 81, 155–166.

[27] Liu, H., Tang, X., Shen, S., 2020. Depth-map completion for large indoor scene reconstruction. *Pattern Recognit.* 99.

[28] Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 91–110.

[29] Ma, J., Jiang, X., Jiang, J., Gao, Y., 2019. Feature-guided gaussian mixture model for image matching. *Pattern Recognition* 92, 231–245.

[30] Moulon, P., Monasse, P., Marlet, R., 2012. Adaptive structure from motion with a *Contrario* model estimation, in: *Asian Conference on Computer Vision*, pp. 257–270.

[31] Moulon, P., Monasse, P., Marlet, R., 2013. Global fusion of relative motions for robust, accurate and scalable structure from motion, in: *IEEE International Conference on Computer Vision*.

[32] Ni, K., Dellaert, F., 2012. Hypersfm, in: *International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pp. 144–151.

[33] Özyesil, O., Singer, A., 2015. Robust camera location estimation by convex programming, in: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2674–2683.

[34] Ramamurthy, K.N., Lin, C., Aravkin, A.Y., Pankanti, S., Viguier, R., 2017. Distributed bundle adjustment, in: *2017 IEEE International Conference on Computer Vision Workshops*, pp. 2146–2154.

[35] Schönberger, J.L., Frahm, J., 2016. Structure-from-motion revisited, in: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104–4113.

[36] Shen, T., Zhu, S., Fang, T., Zhang, R., Quan, L., 2016. Graph-based consistent matching for structure-from-motion, in: *Computer Vision European Conference*, pp. 139–155.

[37] Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 888–905.

[38] Snavely, N., Seitz, S.M., Szeliski, R., 2008. Skeletal graphs for efficient structure from motion, in: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[39] Sweeney, C., Fragoso, V., Höllerer, T., Turk, M., 2014. gdl: A scalable solution to the generalized pose and scale problem, in: *Computer Vision European Conference*, pp. 16–31.

[40] Sweeney, C., Fragoso, V., Höllerer, T., Turk, M., 2016. Large scale sfm with the distributed camera model, in: *International Conference on 3D Vision*, pp. 230–238.

[41] Sweeney, C., Höllerer, T., Turk, M., 2015. Theia: A fast and scalable structure-from-motion library, in: *Annual ACM Conference on Multimedia Conference*, pp. 693–696.

[42] Toldo, R., Gherardi, R., Farenzena, M., Fusiello, A., 2015. Hierarchical structure-and-motion recovery from uncalibrated images, pp. 127–143.

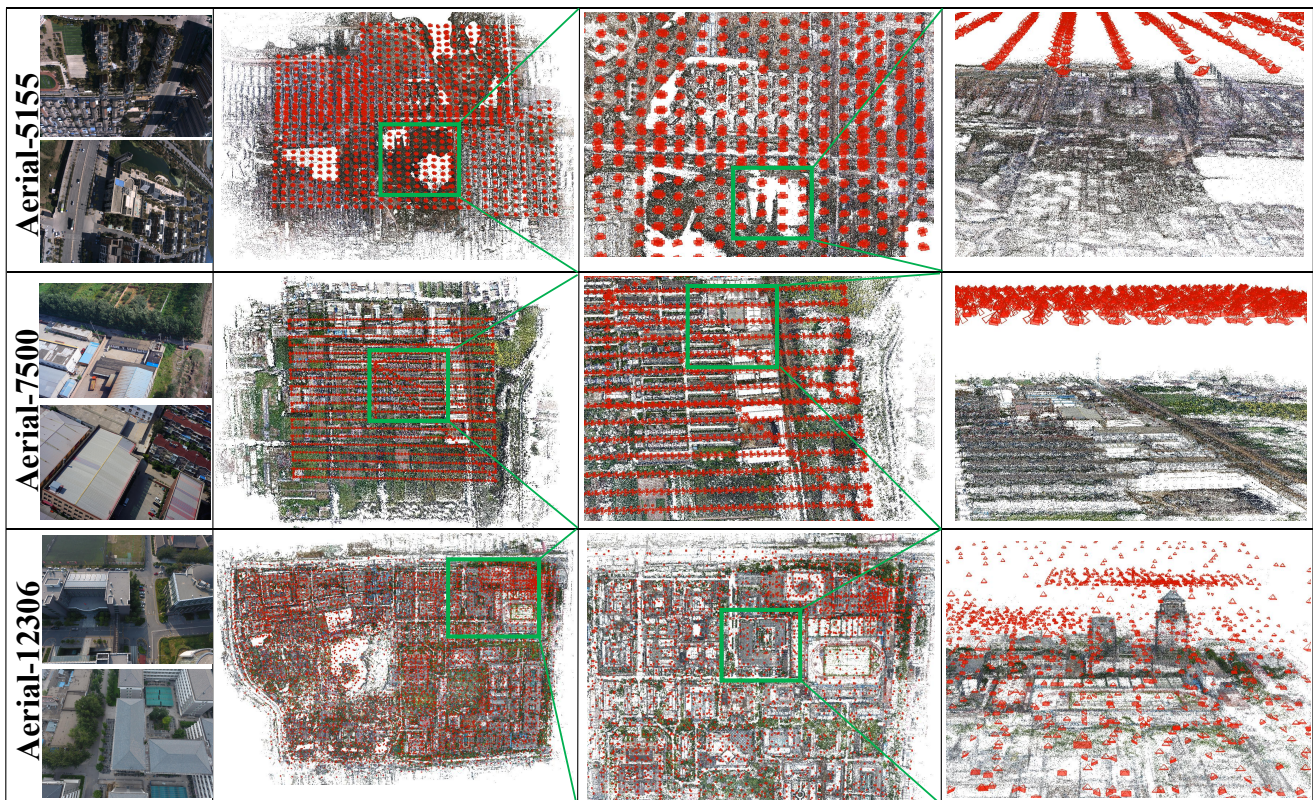


Figure 8: Reconstructions on large scale aerial datasets.

- [43] Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W., 1999. Bundle adjustment - A modern synthesis, in: *Vision Algorithms: Theory and Practice*, International Workshop on Vision Algorithms, pp. 298–372.
- [44] Wang, R., Yan, J., Yang, X., 2019. Learning combinatorial embedding networks for deep graph matching, in: *2019 IEEE/CVF International Conference on Computer Vision*, IEEE. pp. 3056–3065.
- [45] Wilson, K., Bindel, D., Snavely, N., 2016. When is rotations averaging hard?, in: *Computer Vision European Conference*, pp. 255–270.
- [46] Wilson, K., Snavely, N., 2014. Robust global translations with ldsfm, in: *Computer Vision European Conference*, pp. 61–75.
- [47] Wu, C., 2013. Towards linear-time incremental structure from motion, in: *International Conference on 3D Vision*, pp. 127–134.
- [48] Yan, J., Cho, M., Zha, H., Yang, X., Chu, S.M., 2016. Multi-graph matching via affinity optimization with graduated consistency regularization. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 1228–1242.
- [49] Yan, J., Wang, J., Zha, H., Yang, X., Chu, S.M., 2015. Consistency-driven alternating optimization for multigraph matching: A unified approach. *IEEE Trans. Image Processing* 24, 994–1009.
- [50] Zhang, R., Zhu, S., Fang, T., Quan, L., 2017. Distributed very large scale bundle adjustment by global camera consensus, in: *IEEE International Conference on Computer Vision*, pp. 29–38.
- [51] Zhu, S., Shen, T., Zhou, L., Zhang, R., Wang, J., Fang, T., Quan, L., 2017. Parallel structure from motion from local increment to global averaging.
- [52] Zhu, S., Zhang, R., Zhou, L., Shen, T., Fang, T., Tan, P., Quan, L., 2018. Very large-scale global sfm by distributed motion averaging, in: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4568–4577.
- [53] Zhuang, B., Cheong, L., Lee, G.H., 2019. Baseline desensitizing in translation averaging. *CoRR* abs/1901.00643.