- Supplementary -

1 Derivation for Similarity Transformation

Firstly, we calculate the relative scale using the method in GraphSfM:

$$s_{ij} = \frac{(C_j^{k_1} - C_i^{k_2})}{(C_j^{k_1} - C_j^{k_2})} \tag{1}$$

$$C_i^k = -R_{ki}^T t_{ki} \tag{2}$$

where C_i^k represents the center of camera k in the coordinate system of cluster i, and R_{ki} and T_{ki} respectively represent the camera pose (rotation and translation).

With the relative scales known, we take a point P observed by common camera k in clusters i and j as an example to derive R_{ji}, t_{ji}. The coordinates of P in cluster i and j are denoted as P_i and P_j, respectively. The coordinates of P in the common camera k are marked as P_k. Note that P_k here is the coordinate under the scale of cluster i. Then we have

$$R_{ki}P_i + t_{ki} = P_k \tag{3}$$

$$R_{kj}P_j + t_{kj} = s_{ji}P_k \tag{4}$$

$$s_{ji}R_{ji}P_i + t_{ji} = P_j \tag{5}$$

It can be inferred that

$$R_{ji} = R_{kj}^T R_{ki} = R_j^T R_i \tag{6}$$

$$t_{ji} = R_{kj}^T (s_{ji}t_{ki} - t_{kj}) \tag{7}$$

$$= R_j^T (s_{ji}t_i - t_j) \tag{8}$$

Here, we omit the subscript k.

2 Additional Experimental Details

2.1 Datasets

The dataset utilized in this article encompasses both small-scale and large-scale scenarios. The small-scale scenarios include Gerrard Hall, South Building, Person Hall [16], and DTU [20]. Gerrard Hall, South Building and Person Hall contains hundreds of sparse high-resolution images taken around the houses. Thees dataset has sparse perspectives and there are trees obstructing the shooting process, making reconstruction difficult. Each scene in the DTU dataset is captured with fixed camera poses in a laboratory environment, providing ground truth camera poses. We selected scan106, scan110, scan114, and scan122 to evaluate the accuracy of pose recovery, with each scene containing 64 images. The large-scale scene datasets include Lund Cathedral, Duomo, San Marco [21], Rubber [22] and Aerial-20k. The first three scenarios were captured around three different churches, each consisting of over 1000 images. Rubber and Aerial-20k are aerial datasets. Aerial-20K is an in-house large-scale scene comprising 23458 images, taken by ourselves. Other datasets are publicly available.

2.2 Experiment Settings

All experiments were conducted on a PC with an Intel i9-9900KF CPU featuring 8 cores (16 threads) and 32GB of RAM. To ensure fairness, the cluster size settings for both GraphSfM and ER-SfM remained consistent across all experiments. We set the size to 80 for Person Hall and Rounxx, and 250 for Aerial-20K, and the remaining datasets were set to 40. In addition, due to memory limitations, no BA optimization was performed in the experiments on Aerial-20k.

We set the size to 80 for Person Hall and Rounxx, and 250 for Aerial-20K

3 Algorithm

Algorithm 1 describes the complete process of Graph Cut. Firstly, an initial connected image graph is constructed using images and matches (line 1), and this graph is added to the candidate queue (line 2). Processing is performed on the connected graphs in the candidate queue until the queue is empty. For each candidate graph, if the number of images in it is less than the cluster size $\mathcal{S}_{\max}$, then the graph is considered as a cluster (line 7). Otherwise, the graph is partitioned into two subgraphs using the NCuts algorithm, and they are added to the candidate queue (lines 9-10). For the subsequent merging process, the edges cut off by the NCuts algorithm need to be collected, where the endpoints of these edges belong to different subgraphs (line 11). The result is the collection of all clusters C obtained through graph cut and all edges E_{lost} cut during the graph cut process.

Algorithm2 describes the complete process of Image Expansion. The Algorithm takes the clusters set obtained by graph cut and the set of lost edges as input. It traverses all the lost edges cut off during the graph cut stage (line 1), and for each endpoint pair of each lost edge corresponding to images I_i and I_j, it finds their respective clusters and adds the other endpoint to the cluster (lines 3-6). Finally, the expanded clusters set is obtained.

Algorithm 3 describes the complete process of Image Reduction. The image reduction process requires a sparse point cloud as guidance, which is obtained by incrementally reconstructing local images after graph cut. Firstly, for any pair of clusters, all common images are identified (line 2). Cluster edges are then constructed using these common images, with each edge containing all common images (line 3). Using these edges and clusters, a cluster graph is constructed

Algorithm 1 Graph cut Algorithm

Input: images I, matchs M, maximum number of cluster size $\mathcal{S}_{\max}$
Output: clusters C, lost edges E_{lost}

$\mathcal{G}_{\text{image}} \leftarrow$ InitImageGraph(I, M) ▷ $\mathcal{G}_{\text{image}}$: image graph
$C \leftarrow \emptyset$, $Q_{graph} \leftarrow \emptyset$ ▷ Q_{graph}: queue of candidate graphs
Append $\mathcal{G}_{\text{image}}$ to Q_{graph}
while Q_{graph} not empty **do**
  $\mathcal{G}_{\text{can}} \leftarrow$ pop(Q_{graph}) ▷ $\mathcal{G}_{\text{can}}$: candidate graph
  if size of $\mathcal{G}_{\text{can}} \leq \mathcal{S}_{\max}$ **then**
    Append $\mathcal{G}_{\text{can}}$ to C
  else
    $\mathcal{G}_{\text{sub1}}, \mathcal{G}_{\text{sub2}} \leftarrow$ Ncuts($\mathcal{G}_{\text{can}}$)
    Append $\mathcal{G}_{\text{sub1}}, \mathcal{G}_{\text{sub2}}$ to Q_{graph}
    Collect lost edges $E_{lost} : \{e_{k_1 k_2} | k_1 \in \mathcal{G}_{\text{sub1}}, k_2 \in \mathcal{G}_{\text{sub2}}\}$ ▷ $e_{k_1 k_2}$: lost edge
  end if
end while

Algorithm 2 Imges Expansion Algorithm

Input: clusters C, lost edges E_{lost}
Output: expanded clusters C_e

for all lost edge e **in** E_{lost} **do**
  I_i, $I_j \leftarrow e$ ▷ I_i, I_j: images connected by e
  $c_k \leftarrow$ FindCluster(C,I_i) ▷ c_k: cluster containing I_i
  Insert Image I_j into c_k
  $c_l \leftarrow$ FindCluster(C,I_j)
  Insert Image I_i into c_l
end for

(lines 5-6). Next, the edges in the cluster graph are traversed, and weights are assigned to the common images contained in the edges, with the top s common images being marked (lines 7-14). The scoring is based on the minimum value of sparse points observed in the common images in both clusters (lines 9-11). Finally, marginal images that have not been marked are filtered out.

Algorithm 4 outlines the overall process of image clustering and local reconstruction. Firstly, images are clustered according to Algorithm 1 (line 1). At this point, the images within each cluster are considered as local images, and parallel local incremental Structure from Motion (SfM) is used to reconstruct sparse point clouds S_{local} and camera poses P_{loacl} for these images (line 2). Subsequently, clusters are expanded and reduced, and during the reduction phase, S_{local} is used to filter the expanded images (lines 3-4). Finally, camera poses $P_{marginal}$ for the reduced marginal images are recovered using the Perspective-Three-Point (P3P) algorithm.

Algorithm 5 describes the complete process of Global Merging. It takes reduced clusters C_r, a cluster graph $\mathcal{G}_{\text{cluster}}$, local camera poses P_{local}, and local point clouds S_{local} as input, and aims to generate global point clouds and camera poses. It begins by computing the relative transformations $s_{i,j}, R_{i,j}$ and $t_{i,j}$ for

Algorithm 3 Imges Reduction Algorithm

Input: expanded clusters C_e, point cloud S, maximum number of common images s
Output: reduced clusters C_r, cluster graph $\mathcal{G}_{cluster}$

for all cluster c_i, c_j **in** C_e **and** $i \neq j$ **do**
  $N_{i,j} \leftarrow$ FindCommonNodes(c_i, c_j) ▷ $N_{i,j}$: common images between c_i and c_j
  $e_{i,j} \leftarrow$ ConstructEdge$(c_i, c_j, N_{i,j})$ ▷ $e_{i,j}$: cluster edge
  Collect cluster edges $E_{cluster} : \{e_{i,j}\}$ ▷ $E_{cluster}$: all cluster edges
end for
$\mathcal{G}_{\text{cluster}} \leftarrow$ ConstructGraph$(E_{cluster}, C_e)$ ▷ $\mathcal{G}_{\text{cluster}}$: cluster graph
for all edges $e_{i,j}$**in** G_{cluster} **do**
  for all common image n_k**in** $N_{i,j}$ **do**
    $O_i \leftarrow$ CountObrs(n_k, s_i) ▷ O_i : # of points observed in s_i from n_k
    $O_j \leftarrow$ CountObrs(n_k, s_j) ▷ O_j : # of points observed in s_j from n_k
    $W_k \leftarrow$ Min(O_i, O_j) ▷ W_k : weight of common image n_k
    MarkTopKImages$(W_k, N_{i,j}, s)$
  end for
end for
for all cluster c_i **in** C_e **do**
  if marginal image I_m of c_i is not marked **then**
    Remove(I_m,c_i)
  end if
end for

each edge in the cluster graph. Then, it extracts triples from the cluster graph and applies loop constraints on rotations and scales for each triple to filter out noisy transformations. Global averaging is then performed on the filtered rotations and scales. Next, the algorithm computes the relative transformations using the global rotations and scales for each edge in the cluster graph, updating the rotations and scales sets accordingly. It then applies loop constraints on translations for each triple, using the global rotations and scales. Scales averaging is performed on the resulting translations. Finally, for each cluster in the cluster graph where $i \neq 0$, the algorithm transforms and merges the cluster using the global rotations, scales, and translations.

Algorithm 4 Image Clustering and Parallel Local Incremental Reconstruction

Input: images I, matchs M, maximum number of cluster size $\mathcal{S}_{\max}$, maximum number of common images s
Output: reduced clusters C_r, local point clouds S and camera poses P

$C, E_{lost} \leftarrow$ GraphCut() ▷ Alg. 1
$S_{local}, P_{loacl} \leftarrow$ LocalSfM(C)
$C_e \leftarrow$ ImageExpansion(C, E_{lost}) ▷ Alg. 2
$C_r \leftarrow$ ImageReduction(C_e, S, s) ▷ Alg. 3
$P_{marginal} \leftarrow$ P3P(C_r)

Algorithm 5 Global Merging Algorithm

Input: reduced clusters C_r, cluster graph $\mathcal{G}_{\text{cluster}}$, loacl camera poses P_{local}, local point clouds S_{local}

Output: global point clouds S and camera poses P

for all edges $e_{i,j}$**in** $\mathcal{G}_{\text{cluster}}$ **do**
    $s_{i,j}, R_{i,j}, t_{i,j} \leftarrow$ ComputeRelativeTransformation($e_{i,j}$)
end for
$Trps \leftarrow$ ExtractTriples($\mathcal{G}_{\text{cluster}}$) ▷ $Trps$: triples
for all triple trp **in** $Trps$ **do**
    $R_{filtered} \leftarrow$ LoopConstrain(R,ϵ_r) ▷ R: set of $R_{i,j}$,ϵ_r rotation threshold
    $s_{filtered} \leftarrow$ LoopConstrain(s,ϵ_s) ▷ s: set of $s_{i,j}$,ϵ_s scale threshold
end for
$R_{global} \leftarrow$ GlobalAveraging($R_{filtered}$)
$s_{global} \leftarrow$ ScaleAveraging($s_{filtered}$)
for all edges $e_{i,j}$**in** $\mathcal{G}_{\text{cluster}}$ **do**
    $s_{i,j}, R_{i,j} \leftarrow$ ComputeRelativeTransformationByGlobal(R_{global}, s_{global})
    Updata set R and s
end for
for all triple trp **in** $Trps$ **do**
    $t_{filtered} \leftarrow$ LoopConstrain(t, R, s,ϵ_t) ▷ ϵ_t translation threshold
end for
$t_{global} \leftarrow$ ScaleAveraging(R_{global}, s_{global})
for all cluster c_i **in** C_r **and** $i \neq 0$ **do**
    TransformAndMerge($c_i, R_{global}, s_{global}, t_{global}$)
end for