Joint Camera Clustering and Surface Segmentation for Large-scale Multi-view Stereo

Runze Zhang rzhangaj@cse.ust.hk
Shiwei Li slibc@cse.ust.hk
Tian Fang[1] tianft@cse.ust.hk
Siyu Zhu szhu@cse.ust.hk
Long Quan quan@cse.ust.hk
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

In this paper, we propose an decomposition approach to large-scale multi-view stereo from an initial sparse reconstruction. The success of the approach depends on the introduction of surface-segmentation-based camera clustering rather than sparse-point-based camera clustering, which suffers from the problems of non-uniform reconstruction coverage ratio and high redundancy. In details, we introduce three criteria for camera clustering and surface segmentation for reconstruction, and then we formulate these criteria into an energy minimization problem under constraints. To solve this problem, we propose a joint optimization in a hierarchical framework to obtain the final surface segments and corresponding camera clusters. On each level of the hierarchical framework, the camera clustering problem is formulated as a parameter estimation problem of a probability model solved by a General Expectation-Maximization algorithm and the surface segmentation problem is formulated as a Markov Random Field model based on the probability estimated by the previous camera clustering process. The experiments on several Internet datasets and aerial photo datasets demonstrate that the proposed approach method generates more uniform and complete dense reconstruction with less redundancy, resulting in more efficient multi-view stereo algorithm.

1. Introduction

Nowadays, the scale of multi-view stereo (MVS) grows dramatically because of the emerging of high resolution imagery and automatic imaging devices, such as unmanned vehicles mounted with video cameras. Given a large scale dataset with thousands of high resolution images, modern multi-view stereo usually takes camera poses and sparse point clouds reconstructed by structure from motion (SFM) as the input and generates dense point clouds with multiple images simultaneously. For such datasets, it is impractical and unnecessary to put all images into MVS algorithms at a time, because each image only correlates with several image locally in spaces. Taking the advantage of the localities in camera clusters, there have been several algorithms [3, 6, 7, 12, 14] to improve the scalability of multi-view stereo. In such methods, the visibility and quality of sparse SFM points are used to evaluate the locality of camera clusters. The whole reconstruction problem is then divided into several sub-problems in terms of camera clusters. Finally the MVS algorithm is performed in each camera cluster.

However, there are two major problems in these methods. Firstly, the reconstruction region is not uniformly sampled in 3D space. Since these methods conduct clustering merely based on cameras, regions seen by multiple clusters will be reconstructed many times with inconsistent qualities. Secondly, because of occlusions in the reconstruction scene and mismatching of image features, sparse SFM points can hardly provide true visibility and quality assessment for camera clustering.

Instead of using sparse SFM points, our approach takes a coarse mesh reconstructed from the sparse points as input. Such coarse meshes interpolate the regions of lower point density, and give more robust occlusion and shape information for the evaluation of the quality of camera clusters. Meanwhile, it provides a clean segmentation on surface level, which significantly reduces the redundancy of the MVS reconstruction, ensures a complete coverage of the reconstructed scene and improves the quality of the MVS reconstruction.

More concretely, we formulate the joint camera clustering and mesh segmentation as a constrained energy minimization problem to fulfill three criteria of quality control-

[1]Tian Fang is the corresponding author.

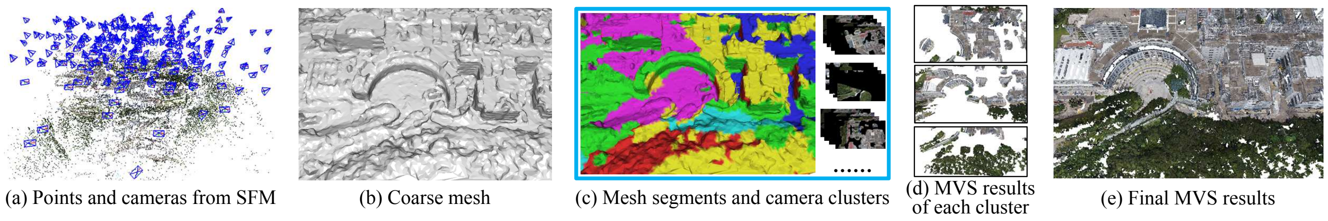
(a) Points and cameras from SFM
(b) Coarse mesh
(c) Mesh segments and camera clusters
......
(d) MVS results of each cluster
(e) Final MVS results

Figure 1. Large-scale Multi-view stereo using joint camera clustering and surface segmentation. The inputs of our method are points and cameras from SFM. Then, we use the state-of-the-art mesh triangulation algorithm to generate the coarse mesh. Next, our proposed joint camera clustering and surface segmentation method is utilised to generate mesh segments and camera clusters. Finally, the MVS algorithm is performed in each segment with its corresponding camera cluster for dense reconstruction.

s. Then we solve such a minimization problem efficiently by a hierarchical clustering framework, in which a General Expectation-Maximization (EM) algorithm and Markov Random Field (MRF) optimization are involved iteratively to find out the camera clustering and mesh segmentation respectively. Finally, each segment is reconstructed by state-of-the-art MVS algorithms with its corresponding camera cluster.

We test our method on both of the publicly available dataset and Internet dataset. The results show that our proposed algorithm outperforms the state-of-the-art method in terms of both the uniformity and completeness of the reconstruction. Moreover, since our method effectively prevents over-sampling of cameras and reconstructed regions, the redundancy and running time of MVS are reduced significantly.

In summaries, our contributions primarily are as follows:

1. We propose three criteria for the joint camera clustering and surface segmentation, which are formulated as a constrained energy minimization problem;
2. We formulate camera clustering problem as a parameter estimation problem of a probability model, which can be solved by a General Expectation-M aximization algorithm. Meanwhile, the surface segmentation is obtained using Markov Random Field optimization.
3. We present a hierarchical framework to solve the constrained energy minimization problem efficiently.

The rest of the paper is organized as follows. The problem definition will be proposed in Section 2. In Section 3 and 4, we will introduce the details of our method to solve the problem defined in Section 2. And Section 4.3 describes the details of stereo reconstruction after surface segmentation and camera clustering. Last, the experiment results in Section 5 show our method effectively solves the problem and generate better reconstruction results.

1.1. Related Work

Most of algorithms for improving the scalability of MVS can be classified into two categories. One kind of algorithms utilizes prior knowledge on the scene and the sequential attributes of video [17, 15, 9]. Pollefeys *et al.* [17] and Micusik *et al.* [15] partition the problem using nearby frames in videos and estimate the depth-maps using MVS algorithms for each sub-problem. Gallup *et al.* [9] focus on merging height map for street level images. These approaches make strong sequential assumptions on the input images.

The other kind of algorithms [12, 7, 6, 3, 10, 2, 14] divides the unordered images into several camera clusters to reduce the computation cost. An image graph is first built to find out the neighboring relationship between images. Then the quality assessment scores are computed based on either photometric similarity [12, 6], or triangulation angles/distances to the visible points [7, 14]. The whole camera graph is thereby divided into a few camera clusters according to the scores. Nevertheless, these methods only cluster cameras and cannot prevent a region from being reconstructed several times in different qualities. To address this problem, we propose to cluster images according to the region segmentation to reduce the redundancy, and guarantee both good spatial coverage and uniformity in each small region, which minimizes the number of images naturally.

Some algorithms [16, 19, 21] are targeted to decouple the SFM problem into sub-problems on the 3D space level. Unlike them, our work focuses on the division of MVS problems on the 3D space level.

2. Problem Formulation

Taking sparse point cloud $\mathcal{P}$ and camera poses $\mathcal{C}$ recovered from structure-from-motion (SFM), we target to break the to-be-reconstructed surface into a few disjoint segments each of which is associated with an optimal set of cameras for stereo reconstruction. Unfortunately, the to-be-reconstructed surface is not valid until the stereo reconstruction is done. Moreover, the SFM point cloud is so sparse, leaving many uncovered regions, and do not encode occlusion among points. Instead of starting from SFM points, we take advantage of the 3D triangulation [13] of SFM points $\mathcal{P}$ to obtain a reasonable initial rough triangle mesh $\mathcal{T} = \{t_i\}$ to approximate the to-be-reconstructed surface. Then we denote its K disjoint segmentation as $\{\mathcal{T}_i | \mathcal{T}_i \subseteq \mathcal{T}, i = 1, ..., K\}$. For each segment, a cluster of cameras $\mathcal{C}_i \subseteq \mathcal{C}$ is gathered for stereo reconstruction. Although the mesh segments are disjoint, the associated camera clusters are unnecessary to be disjoint. Now, our goal is to obtain a joint mesh segmentation and camera clustering $\mathcal{S} = \{S_i | S_i = \{\mathcal{T}_i, \mathcal{C}_i\}\}$ for large scale stereo reconstruction.

A solution to this joint mesh segmentation and camera clustering should satisfy the following criteria:

1. **Smoothness**: the mesh segment should be as smooth as possible to avoid scattered segments.

   This criteria also intends to constraint the total number of clusters implicitly, since the more clusters are there, the less smooth is the segmentation. This criteria also avoids the trivial solution that each cluster only contains a triangle.
2. **Size**: the number of cameras in each cluster should not be too many, since the scale of stereo reconstruction for each cluster is bounded by the number of images.
3. **coverage**: each mesh segment should be well covered by the selected cluster of cameras.

   We adapt the *coverage ratio* of 3D points in [7] to triangles to measure the proportion of triangles that are well observed by the cameras in the cluster for each mesh segment. This criteria ensures a good reconstruction quality.

Now our goal is mathematically cast into looking for a segmentation S^* such that

$$\begin{array}{lcl} & \mathcal{S}^* = \arg\min_{\mathcal{S}} E_s(\mathcal{S}) & \text{(smoothness)} \\ s.t. & |\mathcal{C}_i| \leq n_c, \forall \mathcal{C}_i & \text{(size)} \\ & \frac{\text{\# covered triangle in } \mathcal{T}_i}{|\mathcal{T}_i|} \geq \delta, \forall \mathcal{T}_i & \text{(coverage),} \end{array} \quad (1)$$

where n_c is the maximum number of cameras in each cluster and δ is the minimum coverage ratio for each segment. Let $L = \{1, ..., K\}$ be the label set corresponding to the mesh segments and $F = \{f_t \in L | t \in \mathcal{T}\}$ be the label set of triangles. The label f_t of triangle t is i if and only if $t \in \mathcal{T}_i$. We further define a neighborhood set $\mathcal{N} = \{(t_i, t_j) | t_i \in \mathcal{T}, t_j \in \mathcal{T}, t_i \text{ and } t_j \text{ share a common edge}\}$. Thus, the object function $E_s(\mathcal{S})$ penalizing the scattered segments is written as

$$E_s(\mathcal{S}) = \sum_{(t_i, t_j) \in N} uI(f_{t_i}, f_{t_j}), \quad (2)$$

where u is penalizing the two adjacent triangles in different clusters. We set $I(f_{t_i}, f_{t_j})$ as 1 if $f_{t_i} \neq f_{t_j}$. Otherwise, it is set as 0. The smaller is $E_s(S)$, the smoother will the segmentation be.

However, optimizing Equation 1 is NP complete. Even worse, we do not know the number of labels. Therefore, we develop the following hierarchical framework to solve this problem. Given the initial number of cluster K_0 (set as one, $E_s(S)$ is 0), the optimal camera clusters are obtained by the algorithm in section 3 and then we divide the coarse mesh into K_0 segments by the method in 4.1. After that, we check whether the clustering and segmentation satisfy the second and third criteria. If a segment does not satisfy those criteria, then the segment is divided by the above process recursively. The details of hierarchical framework will be described in 4.2.

3. Camera clustering

We first define a probability model of surface triangles given the number of camera clusters. Then we adopt the General Expectation-Maximization algorithm to compute camera clusters and the posterior probabilities of surface triangles belonging to each cluster.

3.1. Probability model of surface triangles and camera cluster

Given a camera cluster $\mathcal{C}_i$ for a triangle t to be reconstructed, the probability distribution of the surface triangle is given by

$$p(t|\mathcal{C}_i) = \frac{1}{Z} e^{-\alpha U(t, \mathcal{C}_i)},$$

where α is a prefixed scalar and

$$U(t, \mathcal{C}_i) = \begin{cases} \lambda - R(t, \mathcal{C}_i), & R(t, \mathcal{C}_i) < \lambda \\ 0, & \text{otherwise,} \end{cases}$$

where $R(t, \mathcal{C}_i) = \frac{g(t, \mathcal{C}_i)}{g(t, V(t))}$ and $V(t)$ is the set of cameras seeing triangle t. λ is a scalar to judge whether a region is sufficiently covered similar to that proposed in [7]. Apparently, the closer $R(t, \mathcal{C}_i)$ is to λ, the higher probability of triangle t belonging to cluster $\mathcal{C}_i$ is. When R exceeds the λ, the probability is the highest. When this probability is higher, it is easier to meet the coverage constraint in Eq. 1 and the clustered cameras are more suitable to reconstruct the given region according to Eq. 3, 4, 5 and 6.

The Z in above equation is the normalisation term.

$$Z = \sum_{t \in \mathcal{T}} e^{-\alpha U(t, \mathcal{C}_i)}$$

.

The $g(t, \mathcal{C}_i)$ is the MVS accuracy measurement in [7].

$$g(t, \mathcal{C}_i) = \max_{p \in \mathbf{T}(\mathcal{C}_i)} \sum_{c_k, c_l \in p} g'(t, c_k, c_l), \quad (3)$$

where

$$g'(t_j, c_k, c_l) = g_1(t_j, c_k, c_l) g_2(t_j, c_k) g_2(t_j, c_l). \quad (4)$$

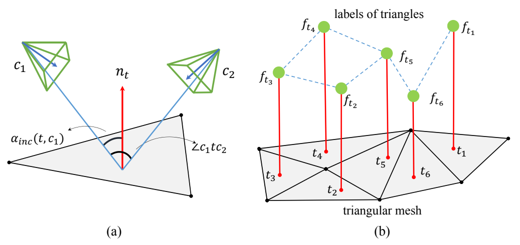

Figure 2. (a) The relationship between cameras and triangles. The angle $\angle c_1 t c_2$ is the baseline of triangulation. And $\alpha_{inc}(t, c_1)$ is the incidence angle. (b) The MRF model for surface segmentation. The circles represent label vertexes in V_f and the centers of each triangle represent triangle vertexes in V_t.

The $\mathcal{T}(C_i)$ is the set of β combination of cameras in the i-th cluster of cameras and the $g_1(t, c_k, c_l)$ is the MVS accuracy measurement in [7].

$$g_1(t, c_k, c_l) = h(\angle c_k t c_l) \min(\frac{1}{r(t, c_k)}, \frac{1}{r(t, c_l)}), \tag{5}$$

where $\angle c_k t c_l$ is the angle between two viewing rays from the center of triangle t and two camera centers, which measures the baseline. The metric of the angle is defined as [7]: $h(x) = \exp\left(-\frac{(x-\mu)^2}{2\sigma_x^2}\right)$($\mu$ is set as $20°$, when $x \leq \mu$, $\sigma_x = 5°$, otherwise, $\sigma_x = 15°$). We use $\frac{1}{r(t,c_k)}$ to present the pixel sampling rate of image c_k at the center of the triangle, where $r(t, c_k)$ is the diameter of a sphere centered at the center of triangle t whose projected diameter equals one pixel in c_k, which measures the pixel sampling rate of image c_k at the center of the triangle.

We have a coarse surface available now, we can therefore describe the relationship of regions and cameras by the surface normals, which is formulated as g_2. The g_2 is defined according to photometry [10, 2], the probability that a camera should be selected for one certain mesh region is determined by incidence angle shown in Figure 2 (a). Incidence angle is the angle α_{inc} between rays coming from the center of triangle to the camera and surface normal. If this angle is large, the triangle is very oblique when reprojected into images actually, which is unsuitable for reconstruction.

$$g_2(t, c) = e^{\frac{-\alpha_{inc}(t,c)^2}{2\sigma^2}}, \tag{6}$$

where σ is usually set as 1.

3.2. Expectation-Maximization Optimization

Camera clustering is regarded as a parameter estimation problem. And by this, the posterior probabilities $P(\mathcal{C}_i|t), t \in \mathcal{T}, i = 1, ..., K$ are also estimated. To estimate the parameter, we maximize the following likelihood with respect to a mixture model by the General Expectation-Maximization under the following constraint:

$$\Phi^* = \arg\max_{\Phi} \mathfrak{L}(\Phi|\mathcal{T})$$
$$s.t. \quad |\mathcal{C}_i| \leq n_c, \forall i,$$

and

$$\begin{aligned} \mathfrak{L}(\Phi|\mathcal{T}) &= \log \prod_{t\in\mathcal{T}} p(t|\Phi) \\ &= \sum_{t\in\mathcal{T}} \log \sum_{i=1}^{K} p(t|\mathcal{C}_i)p(\mathcal{C}_i) \end{aligned}$$

We use indicator variables $\mathbf{z}^{(t)} = (z_1^{(t)}, z_2^{(t)}, ..., z_K^{(t)})$ to express the label of each triangle. If triangle t belongs to cluster i, $z_i^{(t)} = 1$, otherwise, $z_i^{(t)} = 0$. And $\sum_{t\in\mathcal{T}} z_i^{(t)} = 1$ since each triangle can only belong to one cluster. Let the prior probabilities $P(\mathcal{C}_i) = \pi_i$, then $p(\mathbf{z}^{(t)}) = \prod_{i=1}^{K} \pi_i^{z_i^{(t)}}$. Therefore, probability of triangle's parameters given hidden variable is $p(t|\mathbf{z}^{(t)}) = \prod_{i=1}^{K} p_i(t)^{z_i^{(t)}}$, where $p_i(t)$ is the shorthand for $p(t|C_i)$. The joint density is $p(t, \mathbf{z}^{(t)}) = p(\mathbf{z}^{(t)})p(t|\mathbf{z}^{(t)})$. So, given the hidden variable Z, the likelihood function is

$$\mathfrak{L}(\Phi|Z) = -\sum_{t\in\mathcal{T}} \sum_{i=1}^{K} z_i^{(t)}(\log \pi_i + \log p_i(t)) \tag{7}$$

Since the energy minimization problem in Eq. 7 cannot be solved analytically even given the number of clusters, we use the General Expectation-Maximization algorithm to solve it.

E-step The evaluation of expectation after the τ-th iteration can be expressed as

$$Q(\Phi|\Phi^\tau) = -\sum_{t\in\mathcal{T}} \sum_{i=1}^{K} E[z_i^t|t, \Phi^\tau](\log \pi_i + \log p_i(t)), \tag{8}$$

where $E[z_i^t|t, \Phi^\tau] = p(z_i = 1|t, \Phi^\tau) = p(\mathcal{C}_i|t, \Phi^\tau)$. Define $p(\mathcal{C}_i|t, \Phi^\tau) = h_i^t$, which can be seen as soft label of triangle t and will be used in mesh segmentation in Section 4.1. And according to Bayesian rule, the posterior probability is

$$h_i^{(t)} = \frac{\pi_i p(t|\mathcal{C}_i)}{\sum_{i=1}^{K} \pi_i p(t|\mathcal{C}_i)}$$

M-step Since π_i does not depend on $\log p_i(t)$, combined with $\sum_{i=1}^{K} \pi_i = 1$, let the partial derivative of the first part $-\sum_{t\in\mathcal{T}} \sum_{i=1}^{K} h_i^t \log \pi_i$ in Eq. 8 be zero, the we get $\pi_i = \sum_{t\in\mathcal{T}} h_i^{(t)} / |\mathcal{T}|$.

The camera selection model has no analytical optimization solution in M-step, but it is the core of the optimization. Maximization problem of the second part $-\sum_{t\in\mathcal{T}}\sum_{i=1}^{K} p_i(t)\log p_i(t)$ in Eq. 8 is a combination optimization problem actually. For each cluster i, we select cameras to minimize the following function

$$\begin{aligned} \mathcal{C}_i^* &= \arg\min_{\mathcal{C}_i} \sum_{t\in\mathcal{T}} h_i^{(t)}(\alpha U(t,\mathcal{C}_i) + \log Z) \\ s.t. \quad & |\mathcal{C}_i| \le n_c \end{aligned} \tag{9}$$

Since the computation of $g(t,\mathcal{C}_i)$ needs much time, we use a suboptimal greedy algorithm to solve it in low time complexity. When computing $g(t,V(t))$, we store the combination of β cameras for triangle t, which is denoted as $C^*(t)$. For each camera c_k, we compute $s_k = \sum_{t\in\{t|c_k\in C^*(t)\}} h_i^{(t)}$. In each iteration, we select the camera with highest s_k. When one triangle's $R(t,\mathcal{C}_i)$ reaches λ, we remove its contribution from each involved s_k. The selection procedure does not stop until the n_c cameras are selected. We select the camera set with the minimum energy in Eq. 9 as the final result from all the result set of each iteration. To reduce the time complexity of the greedy algorithm, we use Fibonacci Heap to store s_k. Since each triangle can only induce the score decreasing once, the time complexity is $O(|\mathcal{T}|(|\mathcal{C}|+n_c))$.

Algorithm 1 Camera Selection in the i-th cluster

```
 1: function CameraSelection($\{h_i^{(t)}\}, \mathcal{T}, \mathcal{C}, K, n_c, \mathcal{C}^*(t)$)
 2:     Compute all $s_k$ and insert into $H$        ▷ H is the Fibonacci Heap to store $s_k$
 3:     $T_{cover} \leftarrow \emptyset, \mathcal{C}_i = \emptyset$
 4:     while $H \neq \emptyset$ and $|\mathcal{C}_i| \le n_c$ do
 5:         Extract camera $s_k$ from the top of $H$ and insert $c_k$ into $\mathcal{C}_i$
 6:         for $t \in T - T_{cover}$ do
 7:             Compute $g(t,\mathcal{C}_i)$
 8:             if $g(t,\mathcal{C}_i) \ge \lambda g(t,V(t))$ then
 9:                 Insert $t$ into $T_{cover}$
10:                 for each $s_k \in \{s_k \in H | c_k \in C^*(t)\}$ do
11:                     Decrease $s_k$ in $H$ with $h_i^{(t)}$
12:         Compute the total energy $E_n$ by Eq. 9
13:     Select the smallest $E_{n^*}$ from $E_n$
14:     Select the first $n^*$ cameras from $\mathcal{C}_i$ as the result $\mathcal{C}_i^*$
15:     Return $\mathcal{C}_i^*$
```

4. Joint optimization

In this section, the details of the joint camera clustering and surface segmentation will be described. We first introduce how the surface is segmented using the results of camera clustering in Section 3. Then the details of hierarchical framework to optimize the object function in Eq. 1 will be introduced. Last, we will describe the process to reconstruct each segmented region by MVS algorithm.

4.1. Surface Segmentation

Given the probability obtained by the algorithm in Section 3 and the fixed number of cluster K, to satisfy the coverage constraints and smoothen the segmentation as much as possible simultaneously, we use Markov Random Field model based on the probability $p(t,\mathcal{C}_{f_t}) = p(t|\mathcal{C}_{f_t})p(\mathcal{C}_{f_t})$ to divide the coarse mesh into K segments.

Define an undirected graph $G=(V,E)$ on triangles and cameras, where $V = V_T \cup V_f$ and $E = \{E_f \cup E_{tf}\}$. V_T is the set of all triangles, V_f is the set of all triangles' labels. $E_f = \{(f_{t_i}, f_{t_j}) | (f_{t_i}, f_{t_j}) \in \mathcal{N}\}$ and $E_{tf} = \{(t_i, f_{t_i}) | t_i \in \mathcal{T}\}$, where f_{t_i} is the label of the i-th triangle. The toy graph sample is shown in Figure 2 (b). The energy function of the MRF model is

$$E(\mathcal{F}) = -\sum_{t\in\mathcal{T}} \log p(t,\mathcal{C}_{f_t}) - \sum_{(t_i,t_j)\in\mathcal{N}} \log p(f_{t_i}, f_{t_j}). \tag{10}$$

In Eq. 10, $p(t,\mathcal{C}_{f_t})$ describes the coverage constraints, while $p(f_{t_i}, f_{t_j})$ essentially penalizes segmenting adjacent triangles into different segments and $-\log p(f_{t_i}, f_{t_j})$ can be calculated the same as $uI(f_{t_i}, f_{t_j})$ in Eq. 2. By graph cut algorithm [5, 4, 11], we can minimize energy function $E(\mathcal{F})$ to divide the mesh into K segments.

After segmentation, if one segment has no triangles, this segment and its corresponding camera clustering will be eliminated.

4.2. Hierarchical optimization

In this section, the objective function in Eq. 1 will be solved by a hierarchical framework to satisfy all constraints. As mentioned above, when segmenting surface, we do not consider whether the coverage constraint is satisfied. Besides, all the above algorithms require the number of clusters K should be given. These will be solved by dividing surface segments and hierarchical optimization. The initial number of clusters K_0 is set to one. The branch process in the hierarchical algorithm can be run in parallel, which will accelerate the algorithm.

Algorithm 2 Hierarchical mesh segmentation and camera selection

```
 1: function HierarchicalOptimization($\mathcal{T}, \mathcal{C}, F, K, n_c, \mathcal{S}^*$)        ▷ $F$ is the
    initial label input. K is the initial cluster number. $\mathcal{S}^*$ is the final camera clusters
    and mesh segments
 2:     Estimate model parameters to get $\mathcal{C}_1$,...,$\mathcal{C}_K$
 3:     Mesh segmentation by graph cut to get $\mathcal{T}_1$,...,$\mathcal{T}_K$
 4:     for $i \leftarrow 1$ to $K$ do
 5:         $r \leftarrow \frac{\#\text{ covered triangles in } \mathcal{T}_i}{|\mathcal{T}_i|}$
 6:         if $r \ge \delta$ then
 7:             Add $\{\mathcal{C}'_i, \mathcal{T}_i\}$ into $\mathcal{S}^*$
 8:         else
 9:             Select cameras to get camera set $\mathcal{C}'_i$ by Algorithm 1 till the coverage
    constraint is satisfied
10:             if $|\mathcal{C}_i| \le n_c$ then
11:                 Add $\{\mathcal{C}'_i, \mathcal{T}_i\}$ into $\mathcal{S}^*$
12:             else
13:                 Divide $\mathcal{T}_i$ into $\lceil \frac{|\mathcal{C}'_i|}{n_c} \rceil$ segments by K-means to get labels $F'$
14:                 HierarchicalOptimization($\mathcal{T}_i, \mathcal{C}, F', \lceil \frac{|\mathcal{C}'_i|}{n_c} \rceil, n_c, \mathcal{S}^*$)
15:     Merge segmentation with same camera clusters
```

When estimating model parameters to cluster cameras in Section 3.2, we do not know the surface segmentation.

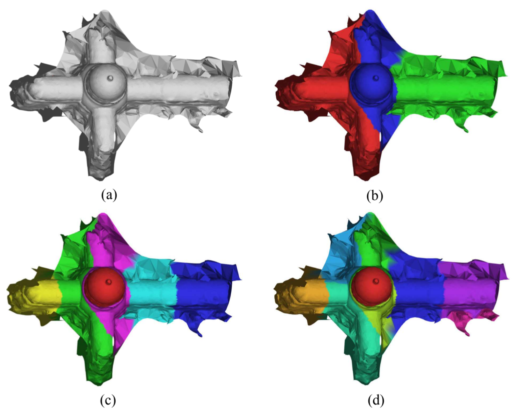

Figure 3. (a) Input coarse surface. (b) Surface segmentation on the first level. (c) Surface segmentation on the second level. (d) Surface segmentation on the third level.

Therefore, the camera clustering may not ensure the coverage constraint is satisfied for each segment by energy minimization. So, it is necessary to check the coverage constraint. If the coverage constraint is satisfied, this camera cluster and mesh segment can be used in the following dense reconstruction stage.

Otherwise, we continue to cluster cameras by the algorithm in Section 3.2 till the coverage constraint is satisfied. The new selected camera cluster is denoted as $\mathcal{C'}_i$. If $\mathcal{C'}_i \leq n_c$, the camera cluster and mesh segment can be used in the following reconstruction stage. Otherwise, we divide the segment into $\lceil \frac{|\mathcal{C'}_i|}{n_c} \rceil$ sub-segments just by K-means on the position of triangle's centers. And this segmentation is regarded as the initialization of Alg. 1 to estimate parameters and cluster cameras in the sub-segment again.

At the end of the algorithm, if the cameras in two camera clusters are identical, it is unnecessary to segment these two regions, so we need to merge these two regions.

4.3. Preparation for dense reconstruction

After segmenting the coarse surface into K regions and clustering cameras for each region, we need to process input images before dense reconstruction. Each segment is reprojected into images of its associated camera clusters to generate masks. To avoid some small regions may not be covered by the coarse mesh, the masks are dilated d (usually set as $\frac{1}{40}$ of the image size) times using 3×3 kernel. K segments will generate K sets of mask images. Then these sets of images and masks are used in the dense reconstruction. The results of all clusters are merged into the final point cloud without any filtering.

5. Experimental Results

The proposed algorithm is implemented in C++ and runs on a PC with Intel(R) Core(TM) i7-4770K 3.50GHz processors and 32GB main memory. We use Kolmogorov *et al.*'s Graph-cut software [11] for mesh segmentation and PMVS [8] for the core MVS reconstruction. The coarse mesh is reconstructed from a set of SFM points using the method in [13]. Points off the surface of mesh will be filtered out.

Throughout the experiments, we use the same parameter setting: $\alpha = 16$, $\beta = 4$, $u = 0.1$, $\lambda = 0.7$, $\delta = 0.7$. n_c is set according to the dataset scale[1].

Since the camera graph in *Rome* dataset [1] has more than one component, we select three major components to perform experiments: *Basilica*, *Trevi* and *Colosseum*. Besides, we also use the *Dubrovnik* dataset [1]. For these four datasets, we set $n_c = 150$ same as the setting in CMVS [7]. Besides, our algorithm is also tested on an aerial photo dataset called *Campus*. *Campus* contains 2060 images and the resolution is pretty high (6000×3376). Since the resolution of the aerial photo dataset is high, n_c is set as 30 to load the images in main memory. The SFM result of the aerial photo datasets is obtained by Bundler [20]. We also use *Temple* and *Dino* datasets [18] to evaluate accuracy and coverage. We set the parameter $n_c = 30$ for the above two datasets since each of them has only around 300 images. To achieve a fair comparison, we employ the same MVS algorithm to generate dense point cloud. And to compare the clustering algorithm of ours and CMVS specially, all the dense points obtained by PMVS are not filtered by the MVS filter process in CMVS.

5.1. Quantitative evaluation

We evaluate the performance of our algorithm quantitatively with regard to uniformity, redundancy and processing time. For two small benchmark datasets, the accuracy and coverage are also measured. Uniformity is evaluated by the standard deviation of point density (SD_d). The density of a point is measured by the average distance to its 10-nearest points. Redundancy is estimated by the average reconstructed times of each pixel in all clusters (P). The evaluation of coverage is referred to [14]. Dense points generated by all images are regarded as ground-truth and for each point in ground-truth and search the nearest point in our result and check whether their distance is in $4\overline{R}$. $\overline{R}$ is the average distance of points to its nearest neighbor in ground-truth. The detailed statistics are shown in Table 1. Since only the two small benchmark datasets can be reconstructed without clustering, the quantitative evaluation of coverage can only be preformed on these two datasets. Besides, only these two datasets have ground-truth for surface reconstruction, so we only evaluate accuracy qualitatively on them.

For *Basilica*, *Trevi*, *Colosseum*, *Campus* and *Dubrovnik*, these five large scale datasets, the number of used cameras selected by our algorithm is less than CMVS and the average used times of each selected camera in our algorithm

[1]The default values of n_c, β, δ and λ are derived from CMVS [7]

Dataset	*Temple*		*Dino*		*Basilica*		*Colosseum*		*Trevi*		*Campus*		*Dubrovnik*	
# cameras	312		363		1180		2043		1933		2060		6844	
	Ours	CMVS	Ours	CMVS	Ours	CMVS	Ours	CMVS	Ours	CMVS	Ours	CMVS	Ours	CMVS
# clusters	4	4	5	2	7	7	4	7	2	7	40	42	46	57
N_c	37	77	99	62	322	452	182	436	123	472	719	1485	2612	3270
Average U_c	1.08	1.06	1.11	1.02	1.57	1.34	1.15	1.10	1.18	1.54	1.94	1.49		
# points ($\times 10^5$)	1.41	1.39	1.43	1.37	106	135	72.8	76.7	49.4	54.4	1686.8	1855.5	303.1	1021.6
SD_d ($\times 10^{-4}$)	**2.26**	3.67	**1.40**	1.58	**2.31**	3.11	**0.724**	1.00	**4.57**	5.46	**2.40**	3.22	**1.02**	4.17
P	**1.080**	1.085	**1.03**	1.04	**1.09**	1.42	**1.01**	1.12	**1.00**	1.60	**1.11**	1.31	**2.20**	10.4
N_p/P ($\times 10^5$)	**1.30**	1.28	**1.39**	1.32	**97.5**	94.9	**72.1**	68.5	**49.2**	34.0	**1519.7**	1416.4	**137.78**	98.23
Accuracy(mm)	**0.52**	0.63	0.62	**0.59**	-	-	-	-	-	-	-	-	-	-
CR	**0.990**	0.987	**0.95**	0.91	-	-	-	-	-	-	-	-	-	-
$t_{cluster}$ (min)	2.2	1.4	2.3	1.2	31	5.9	49	13	25	6.7	74	18	654	282
t_{pmvs} (min)	1.5	4.3	1.9	3.2	107	179	70	114	41	70	667	981	1219	3469
t_{total} (min)	**3.7**	5.7	**4.2**	4.4	**138**	184.9	**119**	127	**66**	76.7	**741**	999	**1873**	3751

Table 1. Statistics. N_c is the number of used cameras. U_c is the used times per camera. SD_d is the standard deviation of average distance between the point and its 10-nearest points. P is the average reconstructed times of each pixel in all clusters. CR is the coverage ratio. $t_{cluster}$ is the time of clustering algorithm. t_{pmvs} is the total running time of pmvs for all clusters. t_{total} is the total time.

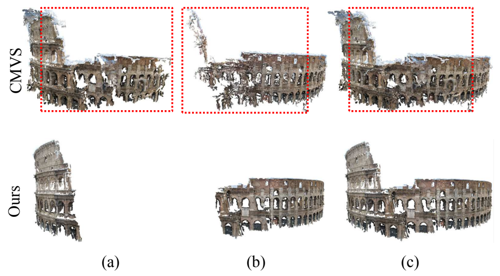

Figure 4. (a) Dense points of a cluster. (b) Dense points of another cluster. (c) Merged dense points of two clusters. The region in the red rectangle is reconstructed more than once in CMVS. In contrast, our method reduces the redundancy by surface segmentation without any post-filtering.

is higher than CMVS. However, the average reconstructed times of each point in all clusters (denoted as P) in our algorithm is much less than CMVS, which measures actually redundancy of the results. The less redundancy of our algorithm is attributed to the space division by surface segmentation, which explicitly reduces the opportunity that each region is reconstructed more than one time. The dilation of the masks also guarantees the completeness of reconstructed region.

As our algorithm usually selects less cameras than CMVS, the total number of reconstructed points of our algorithm is slightly smaller than the one of CMVS. Nevertheless, CMVS tends to produce more redundant points. Therefore, considering the redundancy, we use the ratio of the number of point to P to measure the number of the effective reconstructed points. It is clear that our algorithm generates more effective points, because our algorithm selects cameras specially for each region and assigns each triangle to the camera clusters most suitable to reconstruct it. Besides, for the same reason, our algorithm generates more uniform points by defining the uniformity in space level rather than image level.

Although our algorithm spends more time on the reconstruction and segmentation of the coarse mesh, we save significant time in multi-view stereo process. It is owed to the fact that we reduce the actual redundancy and eventually increase the actual reconstruction efficiency. As for accuracy, our method outperforms significantly CMVS in *Temple* and fall behind CMVS marginally in *Dino*. The benefit of our method is not significant on small dataset with well-condition cameras.

5.2. Qualitative evaluation

Figure 5 shows visual comparison between our method and CMVS. Overall, our method generates more complete dense point cloud than CMVS among all six datasets. The point cloud of CMVS seems to be noisier (more obvious in *Colosseum*, *Campus*). The major reason is that CMVS may reconstruct the same region multiple times with different qualities. So, these noises disturb the reconstruction of sculpture in *Campus*, which also shows that our method selects better cameras in detailed region. Moreover, the repeated dense reconstruction inevitably leads to the uneven density of point cloud, which easily causes small holes (*Basilica*) in surface reconstruction. To avert the above problem that sparse-points-based clustering methods have a bias on fine scene details, our method explicitly segments the initial mesh and ensures each segment of the surface is reconstructed once with the best visible cameras. Figure 4 shows the effectiveness of our method in reducing the redundancy. Note that in *Trevi*, since coverage criteria in the domain part is satisfied, this part is unnecessary to be partitioned. Because our method requests that each segment should be covered by enough cameras, in the margin of reconstructed regions with lower density of SFM points than the center regions, such as *Trevi*, our method generates more complete points. Please note that the image resolutions of *Dubrovnik* have a large variance. CMVS's dense points from high resolution images are degraded by the ones

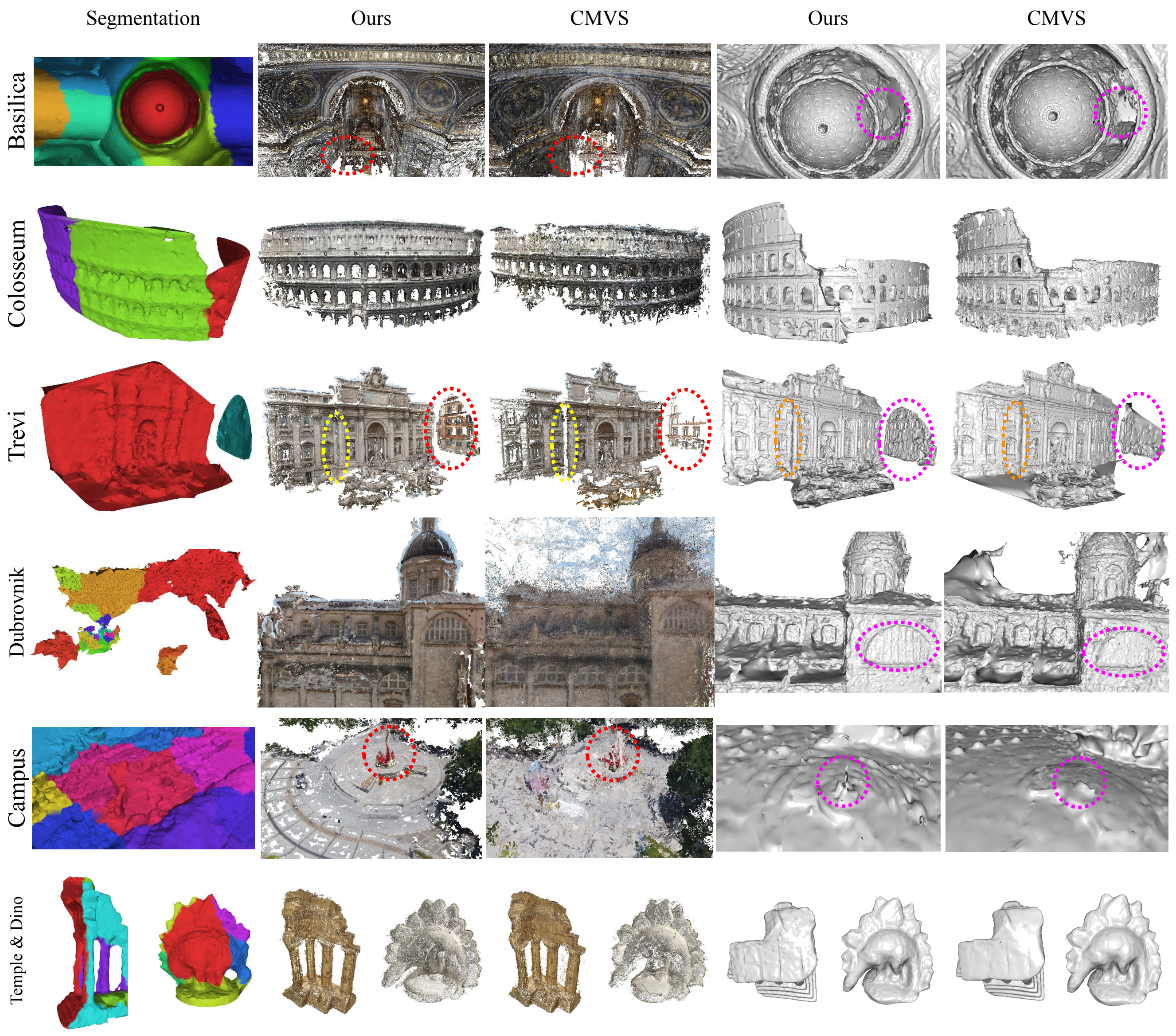

Figure 5. The visual comparison between CMVS and our method (Please refer to the color pdf for better visual quality). From top to button are respectively the segmentation results, MVS results using camera clusters from our method, MVS results using camera clusters from CMVS, our mesh results, and those of CMVS. In the region of circles, our method generates more complete or accurate dense points and surfaces.

from low resolution images, but our method only generates dense points by the best images for the region with fine details and dense points of other clusters do not degrade the good one.

6. Conclusion

In the paper, we propose a novel method to divide large scale MVS into small sub-problems not only on the image level but also on the reconstructed 3D space level. Our key contribution is to solve the problem as a coarse surface segmentation problem and regard camera clustering as the process of parameter estimation. To satisfy the scalability limitation and coverage constraints, we use a hierarchical framework to jointly optimize the camera clustering and surface segmentation. In the end, the experiment demonstrates our method can reduce the reconstruction redundancy, speed up large scale reconstruction and obtain better reconstruction results.

Acknowledgement. This work is supported by RGC-GRF 16208614 and ITC-PSKL12EG02. *Rome* and *Dubrovnik* datasets and their SFM outputs are provided by Agarwal *et al*. [1].

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] A. H. Ahmadabadian, S. Robson, J. Boehm, and M. Shortis. Image selection in photogrammetric multi-view stereo methods for metric and complete 3d reconstruction. In *SPIE*, volume 8791, pages 7–18, 2013.

[3] C. Bailer, M. Finckh, and H. P. Lensch. Scale robust multi view stereo. In *ECCV 2012*, pages 398–411. Springer, 2012.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[6] A. S. Brahmachari and S. Sarkar. View clustering of wide-baseline n-views for photo tourism. In *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*, pages 157–164. IEEE, 2011.

[7] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1434–1441. IEEE, 2010.

[8] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1362–1376, 2010.

[9] D. Gallup, J.-M. Frahm, and M. Pollefeys. A heightmap model for efficient 3d reconstruction from street-level video. In *Int. Conf. on 3D Data Processing, Visualization and Transmission*, volume 6. Citeseer, 2010.

[10] A. Hosseininaveh, M. Serpico, S. Robson, M. Hess, J. Boehm, I. Pridden, and G. Amati. Automatic image selection in photogrammetric multi-view stereo methods. In *VAST*, pages 9–16, 2012.

[11] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[12] A. Ladikos, S. Ilic, and N. Navab. Spectral camera clustering. In *Computer Vision Workshops (ICCV Workshops)*, pages 2080–2086. IEEE, 2009.

[13] M. Lhuillier and S. Yu. Manifold surface reconstruction of an environment from sparse structure-from-motion data. *Computer Vision and Image Understanding*, 117(11):1628–1644, 2013.

[14] M. Mauro, H. Riemenschneider, L. Van Gool, R. Leonardi, and I. Brescia. Overlapping camera clustering through dominant sets for scalable 3d reconstruction. In *Proc. British Machine Vision Conference*, 2013.

[15] B. Micusik and J. Kosecka. Piecewise planar city 3d modeling from street view panoramic sequences. In *Computer Vision and Pattern Recognition*, pages 2906–2912. IEEE, 2009.

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

[17] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[18] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition*, volume 1, pages 519–528. IEEE, 2006.

[19] D. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Robotics: science and systems*, 2009.

[20] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics*, volume 25, pages 835–846. ACM, 2006.

[21] S. Zhu, T. Fang, J. Xiao, and L. Quan. Local readjustment for high-resolution 3d reconstruction. In *Computer Vision and Pattern Recognition*, pages 3938–3945. IEEE, 2014.