An Optimal Hierarchical Clustering Approach to Mobile LiDAR Point Clouds

Sheng Xu, Ruisheng Wang, Hao Wang, and Han Zheng

***Abstract*— This paper aims to propose a new optimal hierarchical clustering approach to 3D mobile light detection and ranging (LiDAR) point clouds. The hierarchical clustering is performed on unorganized point clouds based on a proximity matrix that consists of a distance term and a direction term. In the dissimilarity calculation of two clusters, a pair of points from each of two clusters is selected, respectively, and Euclidean distances between the points are employed to define the distance term. The direction term is obtained by the differences of normal vectors at chosen points. The main contribution is that the cluster combination in the hierarchical clustering is optimized by a point-based graph model. The cluster combination is formulated as a problem of matching, optimized by finding the minimum-cost perfect matching in a bipartite graph. The results show that the proposed hierarchical clustering method succeeds in segmenting object from point clouds without any human–computer interaction and outperforms the state-of-the-art segmentation approaches in terms of completeness and correctness.**

***Index Terms*— Hierarchical clustering, MLS, segmentation, bipartite graph.**

I. Introduction

SEGMENTATION from point clouds collected by laser sensors plays an important role in the environmental analysis, 3D modeling and object tracking. Segmentation is the process of partitioning input data into multiple regions. Segmentation results can be divided into three levels, namely the supervoxel level [1], region level [2] and object level [3], [4]. Nowadays, segmentation results have been effectively used in applications related to the navigation of a self-driving car, such as the curb extraction [5], pedestrian detection [6], street light poles localization [7] and simultaneous localization and mapping (SLAM) [8]. The key challenge in segmentation of point clouds is the split of overlapping regions. Due to the fact that point clouds are noisy, uneven and unorganized i.e. lack of topology, overlapping regions are difficult to be detected.

This paper aims to propose a new optimal hierarchical clustering (OHC) approach to mobile LiDAR (Light Detection And Ranging) point clouds. The approach uses the matching in a bipartite graph to combine regions, and the optimal cluster combination is solved by the minimum-cost perfect matching in the point-based graph. In this paper, a region is regarded as a component of an object instance. Points from one region should not belong to different object instances. For example, one building instance may include one or several regions.

This paper is organized as follows. Section II reviews merits and demerits of the related segmentation methods. Section III describes the overall idea of the optimal hierarchical clustering (OHC) algorithm. Section IV focuses on the determination of the dissimilarity between clusters to form the proximity matrix. Section V uses the matching in a bipartite graph to find the combination solution and achieves the optimal combination by solving the minimum-cost perfect matching. Section VI shows experiments to evaluate the performance of the proposed method. Section VII demonstrates merging results of regions and discusses complexity of the proposed OHC. Conclusions are outlined in Section VIII.

II. Related Work

In this paper, the segmentation is related to the split of general instances from an input scene, which is different from the specific instance extraction or detection. In the object extraction, commonly used methods are rule-based. In the work of Zhang *et al.* [5], they segment road curbs based on predefined rules, including the planar surface of roads and the elevation difference between curbs and roads. In the work of Li *et al.* [6], the detection of pedestrian is based on the density requirement and geometric shape. In the work of Wu *et al.* [7], the extraction of poles is based on the feature extraction, including pole features and geometric information. Those methods are difficultly used in the segmentation of general object instances. Related work for the multi-object segmentation is shown below.

In the work of Douillard *et al.* [9], the authors propose a Cluster-All method for dense point clouds and achieve a trade-off in terms of simplicity, accuracy and computation time. The grouping process in Cluster-All is based on the k-nearest neighbors approach. The principle of KNNiPC (k-nearest neighbors in point clouds) is selecting a number of points for a given point based on the nearest Euclidean distance and assigning them the same index. To avoid under-segmentation results, the variance and mean of Euclidean distances of the points within a group are restricted to be less than a preset threshold. A similar approach is shown in

Manuscript received September 11, 2018; revised January 21, 2019; accepted March 31, 2019. The Associate Editor for this paper was J. Li. *(Corresponding author: Ruisheng Wang.)*

S. Xu is with the College of Information Science and Technology, Nanjing Forestry University, Nanjing 210037, China.

R. Wang is with the School of Geographical Sciences, Guangzhou University, Guangzhou 510006, China, and also with the Department of Geomatics Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: ruiswang@ucalgary.ca).

H. Wang is with the College of Landscape Architecture, Nanjing Forestry University, Nanjing 210037, China.

H. Zheng is with the Department of Geomatics Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada.

Digital Object Identifier 10.1109/TITS.2019.2912455

the work of Klasing *et al.* [10]. The authors present a radially nearest neighbors strategy to bound the region of neighbor points, which helps enhance the robustness of the neighbor selection. The KNNiPC works well in different terrain areas and does not require any prior knowledge of object locations. The problem is that results of KNNiPC highly depend on the selection of a "good value" for k which is the number of selected nearest neighbor points. The setting of k is very sensitive to the point cloud density. A large k fails to split overlapping regions between different object instances and a small one may cause a heavy over-segmentation.

In the point cloud segmentation, the challenging is to separate two overlapping regions. In the method of Yu *et al.* [3], the authors propose an algorithm for segmenting pole-like objects from mobile LiDAR point clouds. In order to separate a cluster with more than one object, the authors add the elevation information to extend the 2D normalized-cut method [11]. The 3DNCut (3D normalized-cut) partitions input points into two disjoint groups by minimizing the dissimilarity within each group and maximizing the dissimilarity between different groups. 3DNCut obtains an optimal solution of the binary segmentation, which demonstrates a promising method for mobile LiDAR point cloud segmentation. The shortcoming is that the number of regions has to be preset before the segmentation. In the method of Golovinskiy *et al.* [4], the authors present a min-cut based method (MinCut) for segmenting objects from point clouds. The MinCut partitions input points into two disjoint groups, i.e. background and foreground, by minimizing Euclidean distances between graph points and the source/sink node. The solution cut, which is used to separate the input scene into background and foreground, is obtained by the graph cut method [12]. The MinCut obtains competitive segmentation results in terms of the optimum and accuracy. However, users have to set the center point and radius for each object. Recently, Su *et al.* [13] propose a segmentation method for terrestrial LiDAR scans of industrial sites containing piping systems. Their split and merge procedures first separate voxel points into spatially unconnected components, and then intelligently merge these components using a series of connectivity criteria across voxels. However, the space complexity is high and the merging procedure depends on a series of connectivity criteria (proximity, orientation, and curvature), which is difficult calculated in a general traffic scene. In the work of Lin *et al.* [14], they formalize segmentation as a subset selection problem and develop an heuristic algorithm to solve the selection problem. Their extracted supervoxels preserve the object boundaries and structures with a higher boundary recall and lower under-segmentation error. The problem is that the supervoxel formulation in vegetation is difficult because the boundary of vegetation is blurred.

Clustering is a well-known technique in data mining. The commonly used is the k-means approach, which has been used for the point cloud in [15], [16]. The idea of KMiPC (k-means in point clouds) is to partition points into different sets to minimize the sum of distances of each point in the cluster to the center. In the work of Lavoué *et al.* [15], the authors use k-means for the extraction of ROI (region of interest) from mesh points. The problem is that KMiPC is easy to produce excessive segments as indicated in [15]. Moreover, the KMiPC needs to set the number of clusters and fails to segment overlapping regions [16]. In the work of Feng *et al.* [17], the authors propose a planar region extraction method based on the agglomerative clustering approach (PEAC) and succeed in segmenting regions from point clouds efficiently. The shortcoming is that input point clouds are required to be well-organized.

The accuracy of KMiPC and KNNiPC is sensitive to point cloud density and is decreased in the overlapping regions between different object instances. The accuracy of 3DNcut and MinCut depends on the scene and can be low when the number of objects is large. The accuracy of PEAC relies on the geometric information and is reduced when objects have complex shapes. In this paper, we propose a new hierarchical clustering algorithm for LiDAR point clouds. The cluster combination is optimized by solving the minimum-cost perfect matching in a bipartite graph. The proposed algorithm does not require the initial number of clusters or the location of objects, which is significant in the point cloud clustering.

III. The Overall Idea of the Optimal Hierarchical Clustering (OHC)

In this paper, the proposed OHC is based on a bottom-up strategy to group regions. It starts with a cluster set, and each set consists of only one point. Then, it measures the dissimilarity between clusters and combines similar clusters into a new cluster.

The input point set is $P = \{p_1, p_2, \ldots, p_n\}$, and the resultant cluster set is $C = \{c_1, c_2, ...c_i, \ldots, c_j, \ldots, c_t\}$, where n is the number of points in P and t is the number of clusters in C. Each cluster c_i contains one or more points from P, and $c_i \cap c_j$ is $\emptyset$ under $i \neq j$. The proximity matrix is $\mathbf{PM}$, which is used to measure the dissimilarity between clusters in C. The goal of the proposed OHC is to optimize the set C, so that each $c_i \in C$ is a cluster of a region. The objective function for the minimization is

$$\arg\min_{\Omega} \sum_{\{c_a, c_b\} \in \Omega} \mathbf{PM}(c_a, c_b), \tag{1}$$

where Ω is the set of the neighboring clusters to be combined. The matrix element $\mathbf{PM}(c_a, c_b)$ means the dissimilarity between the cluster c_a and c_b.

In the initialization, each cluster c_i contains only one point from P, and the size of the proximity matrix is $t \times t$. In the proposed OHC, similar clusters are combined into one cluster and the value of t will be reduced during the clustering process. Key steps are the determination of the proximity matrix and the optimization of the cluster combination, which will be discussed in the following sections. The corresponding flowchart is shown in Fig.1. The cluster set C^{i+1} is the update of C^i based on the optimal cluster combination. At the end of the algorithm, the converged cluster set C is regarded as the clustering result.

IV. Determination of the Proximity Matrix

For a set C with t clusters, we formulate a $t \times t$ symmetric proximity matrix. The (i, j)th element of the matrix means

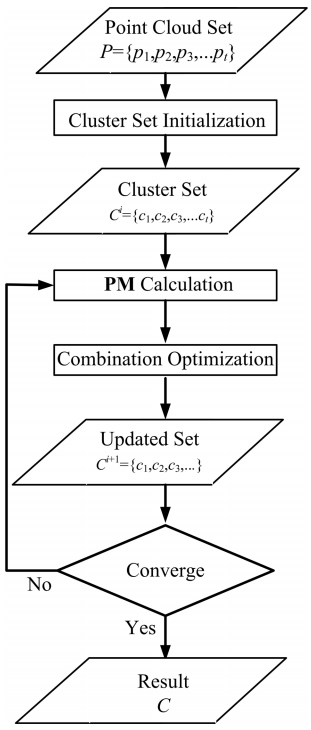


Fig. 1. The flowchart of the proposed OHC approach.

the dissimilarity between the ith and jth cluster ($i, j = 1, 2, \ldots, t$). During the hierarchical clustering, two clusters with a low dissimilarity are preferred to be combined into one cluster. In this paper, points from a region are assumed to be spatially close, and normal vectors at points from local planes are similar. The calculation of the dissimilarity contains a distance measure $\alpha\left(p_i, p_j, c_i, c_j\right)$ and a direction measure $\beta\left(p_i, p_j, c_i, c_j\right)$.

The density of LiDAR points relies on the scanning pattern and the distance to the sensor, therefore, Euclidean distances between points are various in different regions. In order to improve the robustness of the distance calculation, we normalize Euclidean distances between points. The normalization is achieved by using a scalar value. Commonly used filters for this purpose are the maximum, minimum and median filters. The maximum filter is sensitive to clusters with outliers. The minimum filter achieves 0 when there exist duplicate points in a region. In the calculation of $\alpha\left(p_i, p_j, c_i, c_j\right)$, the required scalar values $f(c_i)$ and $f(c_j)$ are based on the median filter Med, which is defined as Eq.(2).

$$f(c_i) = Med_{p \in c_i}\{\min_{p' \in c_i, p' \neq p} d(p, p')\} \tag{2}$$

For example, if one has a cluster $c_0 = \{p_1, p_2, p_3\}$ and the minimal distance from each point $p_i \in c_0$ to other points within c_0 is d_i, $f(c_0)$ is obtained by the median value of $\{d_1, d_2, d_3\}$. It is worth noting that if there is only one point in the cluster c_i, $f(c_i)$ will be 1. The distance term $\alpha\left(p_i, p_j, c_i, c_j\right)$ is formed as

$$\alpha\left(p_i, p_j, c_i, c_j\right) = \frac{d(p_i, p_j)}{\max(f(c_i), f(c_j))} \tag{3}$$

where $d(p_i, p_j)$ measures Euclidean distances between two points. Clusters c_i and c_j are two different clusters in the set C. Points p_i and p_j are obtained by

$$\arg\min_{(p_i, p_j)} d(p_i, p_j) : p_i \in c_i, p_j \in c_j \tag{4}$$

The direction measure is based on the normal vector information. The normal vector at a point is approximated as the normal to the surface estimated by its k-nearest neighborhood points. Assuming that there are k points in the estimation, based on singular value decomposition (SVD) method one has

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \cdots & \cdots & \cdots \\ x_k & y_k & z_k \end{bmatrix} = \mathbf{D}_{k\times 3} = \mathbf{U}_{k\times k}\mathbf{S}_{k\times 3}\mathbf{V}_{3\times 3}^{\top} \tag{5}$$

where $\mathbf{D}$ is the input matrix decomposed into the matrices $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$. The column vector $\mathbf{V}^{|}$ in $\mathbf{V}$, which corresponds to the smallest eigenvalue in the decomposition, is chosen as the normal vector at the given point. The direction term $\beta\left(p_i, p_j, c_i, c_j\right)$ is formed as

$$\beta\left(p_i, p_j, c_i, c_j\right) = 1 - \left|\mathbf{V}^{|}(p_i) \cdot \mathbf{V}^{|}(p_j)\right| \tag{6}$$

where $\mathbf{V}^{|}(p_i)$ and $\mathbf{V}^{|}(p_j)$ are normal vectors estimated from the k-nearest neighbors of the point p_i and p_j, respectively. Points p_i and p_j are obtained by Eq.(4).

In our work, exterior points are defined as the outer surface points of object instances, and interior points are the rest of points. If two objects are connected with each other, objects' exterior points are potential to be the overlapping points between different objects. The proximity matrix **PM** is calculated by Eq.(7), shown at the bottom of the next page. λ is a user-defined coefficient to balance $\alpha\left(p_i, p_j, c_i, c_j\right)$ and $\beta\left(p_i, p_j, c_i, c_j\right)$. If both p_i and p_j are interior points, which are assumed to be in the same cluster, the calculation of PM is based on Eq.(7a); if both of them are exterior points, which can be in different clusters, the calculation of PM is based on Eq.(7b); in all other conditions, the calculation of PM is based on Eq.(7c). The dissimilarity between two clusters is small if they are spatially close or they have consistent normal vectors at the given points obtained by Eq.(4). When both two clusters consist of interior points, the dissimilarity will be dominated by the distance measure. When both two clusters consist of exterior points, the direction measure dominates their dissimilarity.

In the work of Chaudhuri and Chaudhuri [18], the authors develop a detection approach to distinguish border points in a dot pattern. The assumption is that border points are not surrounded by other points in all directions and the authors detect border points efficiently in synthetic point clouds. However, mobile LiDAR point clouds are complex, uneven and incomplete. The threshold to determine border points is required to be adaptive for different regions. Another commonly used method for the shape calculation is called 3D α-shape [19]. It succeeds in recovering the shape of non-convex and even non-connected sets in 3D point clouds. However, since MLS point clouds are noisy and uneven and 3D α-shape is based on distances between points in deciding

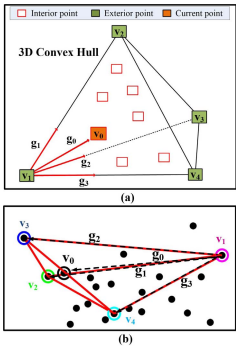


Fig. 2. Illustration of the local 3D convex hull test. (a) The testing of a vertex v_0. (b) A close-up view of a local 3D convex hull.

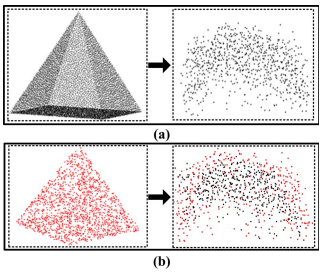


Fig. 3. Results of the exterior point extraction. (a) Input point clouds. (b) Exterior points from our method.

which points to connect by triangles or lines, it is relatively inappropriate to use this method in our non-uniform MLS point cloud sets. Therefore, we propose a local 3D convex hull testing method to mark the exterior and interior points from input data. The proposed method proceeds as follows: (1) initialize all input points as unlabeled points; (2) check if a point is not labeled as an interior point; (3) pick up its k-nearest neighbor points to construct a local 3D convex hull, and the value of k is the same as the value in the normal vector calculation; (4) label all points inside the hull as interior points; (5) repeat (2)-(4) until all input points are tested; (6) mark the rest unlabeled points as exterior points.

To implement the testing, a local 3D convex hull will be formed using four vertices, namely v_1, v_2, v_3 and v_4, as shown in Fig.2(a). The target is to test if a vertex v_0 is an interior point or not. As shown in Fig.2, the vector $\mathbf{g}_0 = (v_0 - v_1)$ can be represented by the sum of vectors $\mathbf{g}_1 = (v_2 - v_1)$, $\mathbf{g}_2 = (v_3 - v_1)$ and $\mathbf{g}_3 = (v_4 - v_1)$ as shown in Eq.(8).

$$\mathbf{g}_0 = u \times \mathbf{g}_1 + v \times \mathbf{g}_2 + w \times \mathbf{g}_3 \tag{8}$$

Based on Eq.(8), one can conclude that if the coefficient $u \not< 0, v \not< 0, w \not< 0$ and $u + v + w < 1$, v_0 will be inside the 3D convex hull. The testing relies on values of u, v and w, which can be solved efficiently by

$$[u, v, w]^\top = \left[\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\right]^{-1} \cdot \mathbf{g}_0 \tag{9}$$

If v_0 is inner the 3D convex hull, it will be labeled as an interior point. Each point from the input will be selected as a vertex and tested in a local convex hull constructed by its k-nearest neighbor points. In building the 3D convex hull, v_1 is chosen as the furthest point to v_0, v_2 is the point to obtain the largest projection of $\mathbf{g}_1$ on the $\mathbf{g}_0$ direction (i.e. $\arg\max_{v_2}(|\mathbf{g}_1| \cdot \cos < \mathbf{g}_1, \mathbf{g}_0 >)$), v_3 is the furthest point to the line containing v_2 and v_1, and v_4 is the furthest point to the plane containing v_1, v_2 and v_3. A close-up view of a local 3D convex hull is shown in Fig.2(b). To take an example, input point clouds are shown in Fig.3(a), including a synthetic pyramid point set and a vegetation point set. Fig.3(b) demonstrates the exterior points and interior points obtained from our method. All surface points from the pyramid point set are successfully labeled as exterior points. Although the vegetation point set is uneven, most surface points are marked as exterior points effectively.

In the calculation of **PM**, there is only one point in each cluster and it is easy to calculate the proximity matrix by Eq.(7) in the first step. When there is more than one point in a cluster, the user has to find the correct p_i and p_j based on Eq.(4) for the calculation of **PM**(c_i,c_j). Then, re-calculate the distance term and the direction term based on Eq.(3) and Eq.(6), respectively.

V. Cluster Combination Based on the Minimum-Cost Perfect Matching

Usually, the hierarchical clustering uses a greedy strategy to deal with the cluster combination. The idea is to find two most similar clusters based on the proximity matrix first, then combine them into one cluster. The greedy strategy is easy to incur a local optimization. Hence, this section aims to optimize the cluster combination globally.

The bipartite graph is denoted by $\mathbb{G} = \{\mathbb{V}_x, \mathbb{V}_y, \mathbb{E}\}$, where the node set $\mathbb{V}_x = \{c_1, c_2, \ldots, c_i, \ldots, c_n\}$ describes the current clusters of points in the set C and $\mathbb{V}_y$ shows clusters for the combination. In the hierarchical clustering, any two clusters can be combined into one cluster, therefore, we let $\mathbb{V}_y = \mathbb{V}_x$. There is no edges inside $\mathbb{V}_x$ or $\mathbb{V}_y$ and the edge between $c_i \in \mathbb{V}_x$ and $c_j \in \mathbb{V}_y$ is denoted as $e_{i,j} : c_i \leftrightarrow c_j$. The set $\mathbb{E} = \{e_{1,1}, \ldots, e_{1,n}, e_{2,1}, \ldots, e_{2,n}, \ldots, e_{i,j}, \ldots, e_{n,1}, \ldots, e_{n,n}\}$ shows all edges in $\mathbb{G}$. The above-modeled bipartite graph $\mathbb{G}$ is shown in Fig.4.

$$\mathbf{PM}(c_i, c_j) = \frac{\lambda - 1}{\lambda}\alpha\left(p_i, p_j, c_i, c_j\right) + \frac{1}{\lambda}\beta\left(p_i, p_j, c_i, c_j\right), \tag{7a}$$

$$\mathbf{PM}(c_i, c_j) = \frac{1}{\lambda}\alpha\left(p_i, p_j, c_i, c_j\right) + \frac{\lambda - 1}{\lambda}\beta\left(p_i, p_j, c_i, c_j\right), \tag{7b}$$

$$\mathbf{PM}(c_i, c_j) = \frac{1}{2}\alpha\left(p_i, p_j, c_i, c_j\right) + \frac{1}{2}\beta\left(p_i, p_j, c_i, c_j\right), \tag{7c}$$

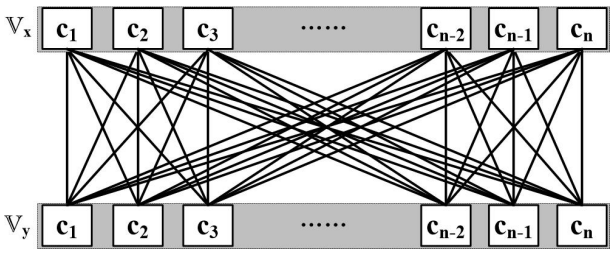


Fig. 4. The modeled bipartite graph $\mathbb{G} = \{\mathbb{V}_x, \mathbb{V}_y, \mathbb{E}\}$.

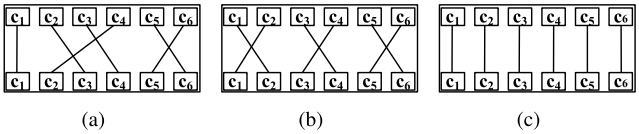


Fig. 5. Clustering results obtained by the perfect matching. (a) The combination solution: $\{c_1\}, \{c_2, c_3\}, \{c_3, c_4\}, \{c_4, c_2\}, \{c_5, c_6\}$. (b) The combination solution: $\{c_1, c_2\}, \{c_3, c_4\}, \{c_5, c_6\}$. (c) The combination solution: $\{c_1\}, \{c_2\}, \{c_3\}, \{c_4\}, \{c_5\}, \{c_6\}$.

In graph theory, the matching in a bipartite graph is a set of edges without common vertices. Assuming that a cluster can be combined with no more than one cluster at each hierarchy, the cluster combination can be solved by the perfect matching in the bipartite graph $\mathbb{G}$. The perfect matching means that every cluster $c_i \in \mathbb{V}_x$ is connected with a cluster $c_j \in \mathbb{V}_y$. In the perfect matching, the edge $e_{i,j} : c_i \leftrightarrow c_j$, which is between $c_i \in \mathbb{V}_x$ and $c_j \in \mathbb{V}_y$, means that if $i \neq j$, the cluster c_i and c_j will be combined into one cluster, otherwise c_i is not chosen for the combination at the current hierarchy. For example, the result of the hierarchical clustering can be determined by a perfect matching as shown in Fig.5.

Eq.(1) aims to find the optimal cluster combination which achieves the minimum sum of the dissimilarity in the clustering. This task can be achieved by solving the minimum-cost perfect matching in $\mathbb{G}$.

The capacity of the perfect matching in $\mathbb{G}$ is denoted by $CostPM$ which determines the sum cost of the combination. The $CostPM$ is calculated as

$$CostPM = \sum_{e_{i,j} \in \Phi} \mathbf{PM}\left(c_i, c_j\right), e_{i,j} : c_i \leftrightarrow c_j \qquad (10)$$

where Φ is the set of edges from the perfect matching. $\mathbf{PM}(c_i, c_j)$ aims to measure the cost of merging those connected clusters in the perfect matching.

The goal is to find the minimal $CostPM$ for the optimal combination. It is worth noting that $\mathbf{PM}(c_i, c_i)$ is weighted by a user-defined cut-off distance SM. Detail steps of the proposed OHC via the minimum-cost perfect matching are shown below.

The input point set P is $\{p_1, p_2, p_3, \ldots, p_n\}$. The goal is to optimize a cluster set C to achieve that each cluster in C represents an object region.

Step 1: Initialize the vertex matrix $\mathbb{V}_x$ as $\{c_1, c_2, \ldots, c_n\}$ by setting c_1 as $\{p_1\}$, c_2 as $\{p_2\}$,..., and c_n as $\{p_n\}$;

Step 2: Let $\mathbb{V}_y$ be equal to $\mathbb{V}_x$ by copying cluster elements from $\mathbb{V}_x$ to $\mathbb{V}_y$.

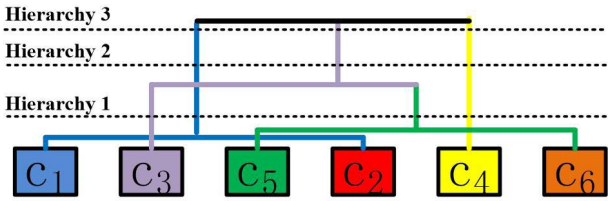


Fig. 6. The dendrogram of the hierarchical clustering.

Step 3: Connect the cluster $c_i \in \mathbb{V}_x$ with $c_j \in \mathbb{V}_y$ to form the edge $e_{i,j}$, where $1 \leq i, j \leq n$;

Step 4: Compute the proximity matrix **PM** using Eq.(7);

Step 5: Optimize Eq.(10) by solving the minimum-cost perfect matching in $\mathbb{G} = \{\mathbb{V}_x, \mathbb{V}_y, \mathbb{E}\}$ using the Kuhn-Munkres algorithm [20];

Step 6: Combine every connected clusters into one cluster;

Step 7: Update $\mathbb{V}_x$;

Step 8: Repeat steps 2–7, until $\mathbb{V}_x$ converges. Finally, the expected cluster set C is assigned by the converged $\mathbb{V}_x$.

Steps 1–3 aim to model a bipartite graph $\mathbb{G} = \{\mathbb{V}_x, \mathbb{V}_y, \mathbb{E}\}$. Edges are initialized as a full connection graph between $\mathbb{V}_x$ and $\mathbb{V}_y$, i.e. each $c_i \in \mathbb{V}_x$ is connected with all other $c_j \in \mathbb{V}_y$. The step 4 is to calculate the dissimilarity between each two clusters to form the proximity matrix. The step 5 aims to achieve the minimization of Eq.(10). The step 6 is to combine those connected clusters into one cluster. The step 7 is to update $\mathbb{V}_x$ to move up the hierarchy.

In the algorithm, spatially non-adjacent clusters are not expected to be combined, thus, edges exist between two adjacent clusters only. The following is a simulated example of clustering using the proposed OHC. The input P is $\{p_1, p_2, p_3, p_4, p_5, p_6\}$. The dendrogram of the hierarchical clustering is shown in Fig.6. In the hierarchy 1, c_1 and c_2 are combined into one cluster, c_5 and c_6 are combined into one cluster, and c_3 and c_4 are not selected in the combination. In the hierarchy 2, c_3 and the previous combination result of $\{c_5, c_6\}$ are combined into one cluster. The final hierarchical clustering result is shown in the hierarchy 3, which is $\{c_1$,$c_2\}$, $\{c_3$,c_5,$c_6\}$ and $\{c_4\}$.

Details of the hierarchical clustering process are shown below. Fig.7(a) demonstrates the input P. Step 1: initialize the node set $\mathbb{V}_x$ as $\{c_1, c_2, c_3, c_4, c_5, c_6\}$, where $c_1 = \{p_1\}$, $c_2 = \{p_2\}$, $c_3 = \{p_3\}$, $c_4 = \{p_4\}$, $c_5 = \{p_5\}$ and $c_6 = \{p_6\}$; Step 2: form the node set $\mathbb{V}_y$ the same as $\mathbb{V}_x$; Step 3: connect spatially adjacent clusters between $\mathbb{V}_x$ and $\mathbb{V}_y$ to form the edge set $\mathbb{E}$ as shown in Fig.7(b); Step 4: calculate the proximity matrix **PM**. Step 5: solve the minimum-cost perfect matching in $\mathbb{G}$ as shown in Fig.7(c). Edges in the perfect matching are shown in the set $\Phi = \{e_{1,2}, e_{2,1}, e_{3,3}, e_{4,4}, e_{5,6}, e_{6,5}\}$; Step 6: combine clusters $c_1 \in \mathbb{V}_x$ and $c_2 \in \mathbb{V}_y$ as a new cluster c'_1, and combine $c_5 \in \mathbb{V}_x$ and $c_6 \in \mathbb{V}_y$ as a new cluster c'_4. The cluster c_3 and c_4 are unchanged and renamed as c'_2 and c'_3, respectively. Step 7: update $\mathbb{V}_x$ as $\{c'_1, c'_2, c'_3, c'_4\}$; Now the first iteration is done. Repeat Step 2–7 and finally $\mathbb{V}_x$ converges to $\{c''_1, c''_2, c''_3\}$ as shown in Fig.7(i); Step 8: return $\mathbb{V}_x$ as the cluster set C.

In order to show which clusters are grouped in the iteration, different iteration results of a toy clustering are

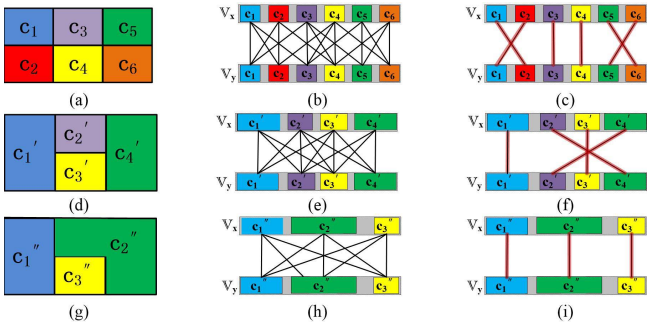


Fig. 7. A simulated example of the proposed OHC. (a) Initial clusters. (b) The bipartite graph based on (a). (c) The minimum-cost perfect matching in (b). (d) Cluster sets after the combination based on (c). (e) The bipartite graph based on (d). (f) The minimum-cost perfect matching in (e). (g) Cluster sets after the combination based on (f). (h) The bipartite graph based on (g). (i) The minimum-cost perfect matching in (h).

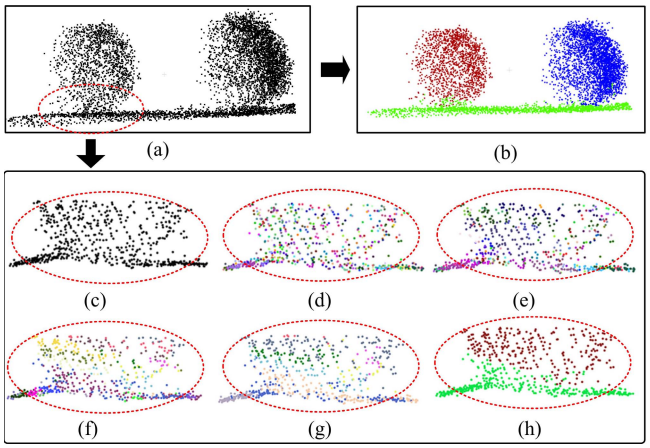


Fig. 8. The toy example of the proposed OHC. (a) Input point clouds. (b) Clustering results. (c) Initial input of 680 points. (d) Results at the first iteration of 680 clusters. (e) Results at the third iteration of 490 clusters. (f) Results at the fifth iteration of 26 clusters. (g) Results at the seventh iteration of 9 clusters. (h) Results at the ninth iteration of 2 clusters.

demonstrated in Fig.8. Fig.8(a) and (b) show the input point clouds and results, respectively. Fig.8(c)–(h) show the close-view of clustering at the iteration #1,#3,#5,#7 and #9, respectively.

The matrix **PM** after the iteration #7 according to Fig.8(g) is shown in Fig.9(a). The corresponding bipartite graph is shown in Fig.9(b). The achieved minimum-cost perfect matching is shown in Fig.8(c). Based on the connected edges in Fig.8(c), the cluster set C is updated as $\{c'_1, c'_2, c'_3, c'_4, c'_5, c'_6\}$, where $c'_1 = \{c_1\}$, $c'_2 = \{c_2, c_3\}$, $c'_3 = \{c_4, c_6\}$, $c'_4 = \{c_5\}$, $c'_5 = \{c_7\}$ and $c'_6 = \{c_8, c_9\}$. In this example, SM is chosen as 0.4.

VI. Experiments

A. Evaluation Methods

In the evaluation, the clustering result is $C = \{c_1, c_2, \ldots, c_i, \ldots, c_t\}$ and the ground truth is $C' = \{c'_1, c'_2, \ldots, c'_j, \ldots, c'_{t'}\}$. Each c_i or c'_j means a point cluster and there are t clusters in C and t' clusters in C'. For the purpose of evaluating results, we calculate the accuracy assessment n_{com}

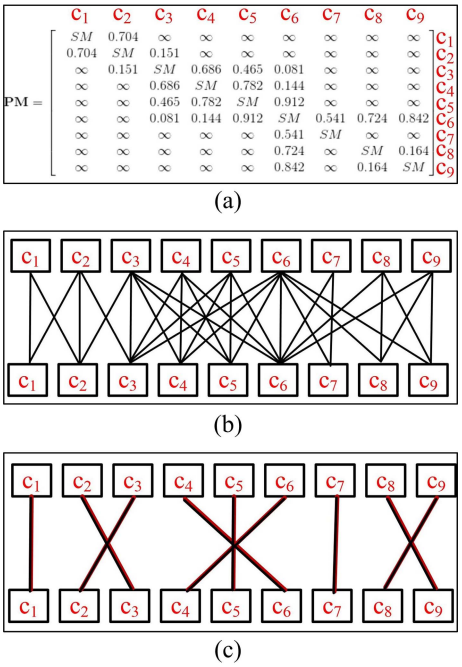


Fig. 9. The proximity matrix at the 7th iteration. (a) The numerical values of the proximity matrix. (b) The achieved bipartite graph. (c) The achieved minimum-cost perfect matching.

and n_{cor} as shown in Eq.(11) based on the Jaccard index [21], which commonly used in cluster evaluation.

$$n_{com} = \frac{1}{t'} \sum_{j=1}^{t'} \left(\frac{\max_{i=1}^{t} |c'_j \bigcap c_i|}{|c'_j|} \right)$$
$$n_{cor} = \frac{1}{t} \sum_{i=1}^{t} \left(\frac{\max_{j=1}^{t'} |c_i \bigcap c'_j|}{|c_i|} \right) \quad (11)$$

The n_{com} is to measure the completeness ratio between the correctly grouped points and the ground truth. The n_{cor} is to measure the correctness ratio between the correctly grouped points and the hierarchical clustering result. The problem of Eq.(11) is that if there is only one cluster in the ground truth C', n_{cor} will be always 1, and if there is only one cluster in the result C, n_{com} will be always 1. In order to address this problem, the minimum of n_{com} and n_{cor} is chosen as the accuracy, i.e. $n_{acc} = \min(n_{com}, n_{cor})$, to measure points' difference between C and C'.

B. Comparisons of Different Methods

This section evaluates performances of KMiPC [16], KNNiPC [9], 3DNCut [3], MinCut [4], PEAC [17] and the proposed OHC on five typical scenes. Experimental scenes are shown in Fig.10, including the HouseSet: a single object, the BushesSet: two separated sparse objects, the LamppostSet: two connected rigid objects, the TreesSet: two connected non-rigid objects and the PowerlinesSet: a complex scene with different objects. Those experimental scenes are collected by the mobile laser scanner, i.e. RIEGL VMX-450 system, which uses a narrow infrared laser beam at a very

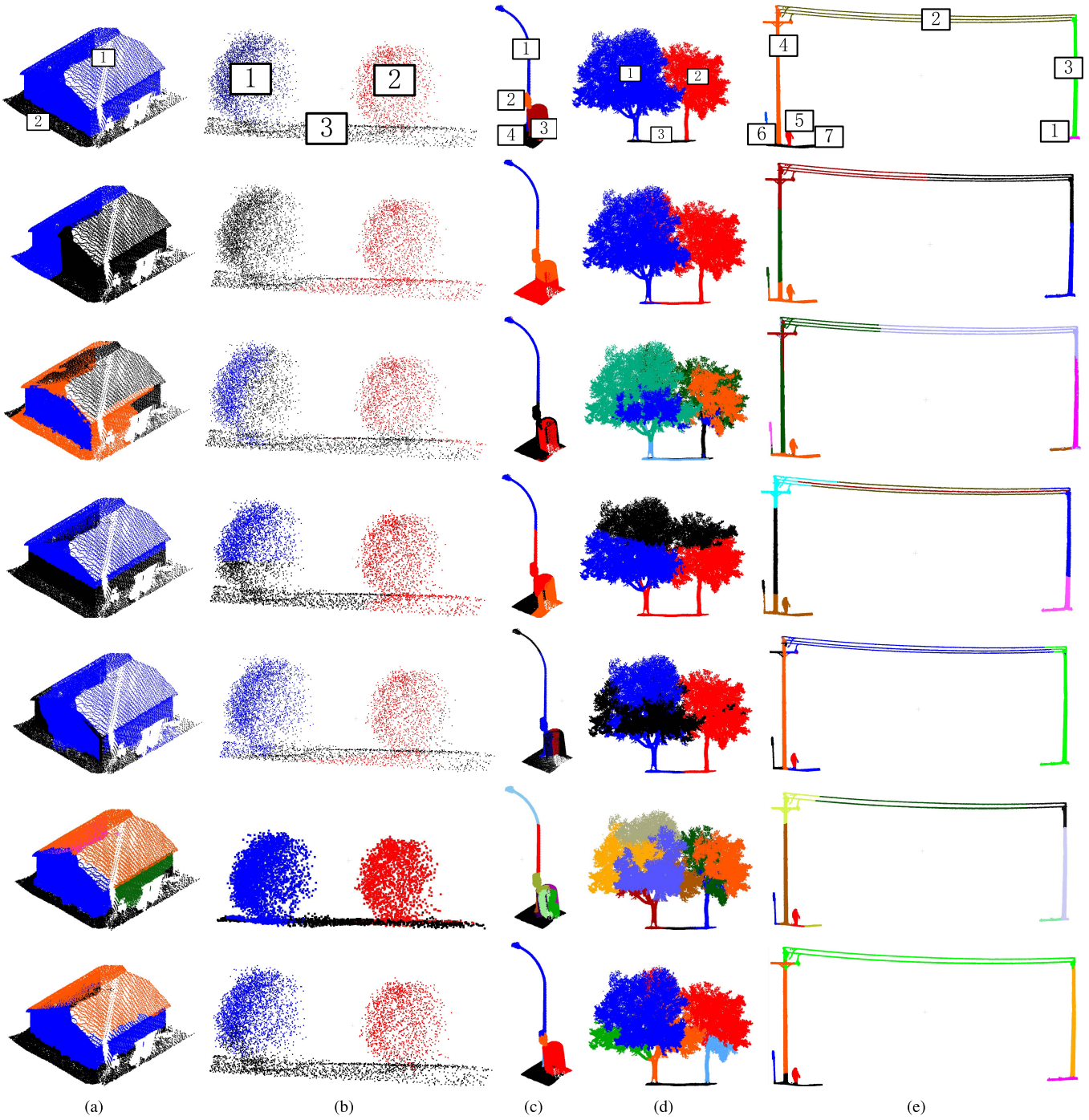


Fig. 10. Performances of different methods. (a) HouseSet. (b) BushSet. (c) LamppostSet. (d) TreeSet. (e) PowerlinesSet. Row 1: the segmentation ground truth of each scene. Row 2: performance of KMiPC [16]. Row 3: performance of KNNiPC [9]. Row 4: performance of 3DNCut [3]. Row 5: performance of MinCut [4]. Row 6: performance of PEAC [17]. Row 7: performance of the proposed OHC.

high scanning rate. The scanner can be up to 200 lines/sec and enables full 360-degree beam deflection without any gaps.

The first row of Fig.10 shows the ground truth. Since the ground-truth of object regions is difficult to reproduce, our ground-truth is achieved by segmenting visually independent object instances manually. We use the CloudCompare visualization software (http://www.danielgm.net/cc/) to segment instance one by one. From the second row to the last are the performance of KMiPC, KNNiPC, 3DNCut, MinCut, PEAC and the proposed OHC, respectively. In the visualization, different colors are used to represent different segments. KMiPC, KNNiPC and MinCut are implemented by using the Point Cloud Library (www.pointclouds.org/), 3DNCut is extended from the normalized-cut method (www.cis.upenn.edu/ jshi/software/). PEAC is achieved based on the software of Feng et al. (www.merl.com/research/?research=license-request&sw=PEAC).

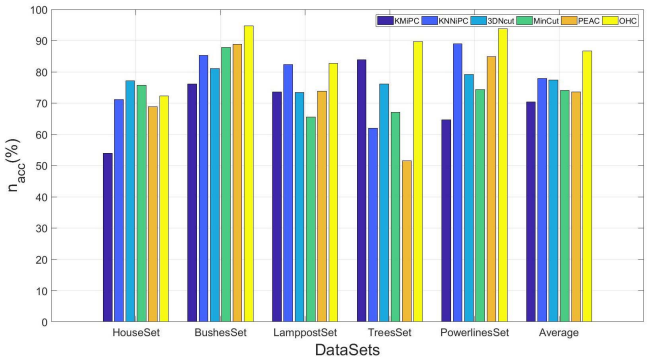


Fig. 11. Evaluation of different methods using n_{acc}.

KMiPC is suitable for segmenting symmetric objects and works well in the BushesSet and TreesSet. However, it fails to split connected objects. KNNiPC relies on the density heavily and is difficult to separate different objects as shown in the HouseSet. 3DNCut tries to normalize the difference of points' Euclidean distances in each group. It tends to group points evenly as shown in the HouseSet and TreesSet. MinCut performs well in most cases when the required center point and radius of each object are set properly. PEAC groups points based on the planar information and requires organizing data before the segmentation. The proposed OHC is not sensitive to the density of points as shown in the BushesSet. Moreover, it does not require initial number of clusters and the location of each object. The attached traffic sign in the LamppostSet is segmented successfully by using the proposed OHC. The split of the overlapping regions between non-rigid objects is rather difficult due to the rapid change of normal vectors as shown in the TreesSet. In the step of computing the normal vector at a point, when using the normal vectors for the clustering, it makes much different if points are located on a tree. Based on Eq.(7), we set a large λ to let the Euclidean distance dominate the **PM** calculation. Results are shown in the last row of Fig.10(d). Leaves points are supposed grouped into one cluster. The over-segmentation is because that there are separated branches and each tree crown is regarded as one cluster.

The quantitative evaluation is shown in Fig.11. One can observe that the average accuracy of the proposed OHC is higher than other methods through all experimental scenes. In the clustering, the proposed OHC uses the matching in a bipartite graph to combine regions from the same object instance, such as the house facade and trash bin in Fig.10(a) and Fig.10(c), respectively. The proposed method achieves the optimal cluster combination by solving the minimum-cost perfect matching in the point-based graph. This is more accurate than other methods in splitting of overlapping regions from different instances as shown in Fig.10(b) i.e. the split of bushes and ground, Fig.10(d) i.e. the split of trunks and tree leaves, and Fig.10(e) i.e. the split of power-lines and poles. The key is the proposed combination optimization in the clustering.

C. Analysis of Parameters Sensitivity

In the implementation of the proposed OHC, there are three parameters, namely the k to select the nearest neighbor points, λ to balance the weight of the $\alpha\left(p_i, p_j, c_i, c_j\right)$ and $\beta\left(p_i, p_j, c_i, c_j\right)$, and SM to formulate the proximity matrix. In this paper, λ is 4, k is 40 and SM is 0.4. For the purpose of the sensitivity analysis, we range all parameters from -30% to 30% with respect to the above-mentioned values. This experiment is conducted by fixing two parameters among λ, SM and k and alternating the rest one. Fig.12 (a) shows results by fixing $k = 40$ and $\lambda = 4$, Fig.12 (b) shows results by fixing $\lambda = 4$ and $SM = 0.4$ and Fig.12 (c) shows results by fixing $k = 40$ and $SM = 0.4$. In each case, the mean accuracy is above 85% and the standard deviation is less than 1.5%. A large k or SM may cause under-segmentation and a small k or SM will increase the over-segmentation rate. A large λ works well for segmenting connected objects and a small λ is preferred when there are less overlapping among different objects.

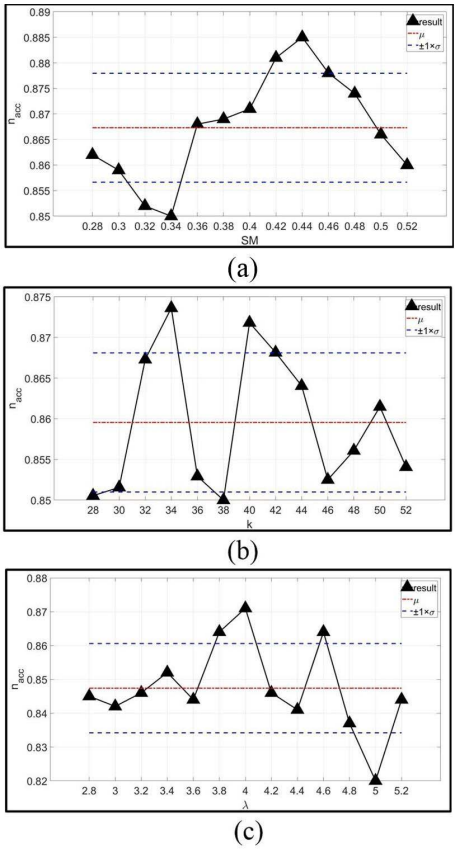


Fig. 12. Sensitivity analysis. (a) Mean μ is 0.8673, standard deviation σ is 0.0107 (fix $k = 40$ and $\lambda = 4$). (b) Mean μ is 0.8595, standard deviation σ is 0.0086 (fix $\lambda = 4$ and $SM = 0.4$).(c) Mean μ is 0.8474, standard deviation σ is 0.0132 (fix $k = 40$ and $SM = 0.4$).

Although the proposed OHC requires to be given three parameters empirically, which can be a shortcoming of our work, Fig.12 demonstrates that it is stable to different input parameters in the suggested value. In the future, we will try to figure out the adaptive parameter setting method at the risk of requiring more prior knowledge of input scenes.

D. Performance on Large-Scale Datasets

We also test the proposed OHC on two large-scale MLS datasets. Fig.13 shows the performance on a large-scale residential point set (558 MB, 12,551,837 points) scanned by the

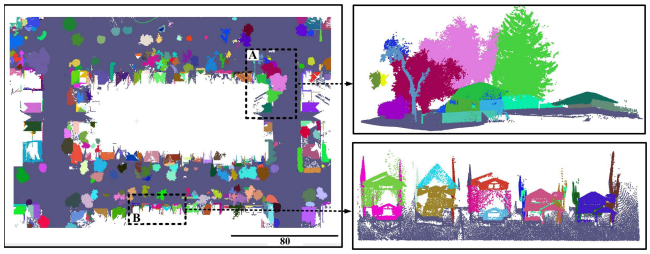


Fig. 13. Clustering results of a large-scale residential point set.

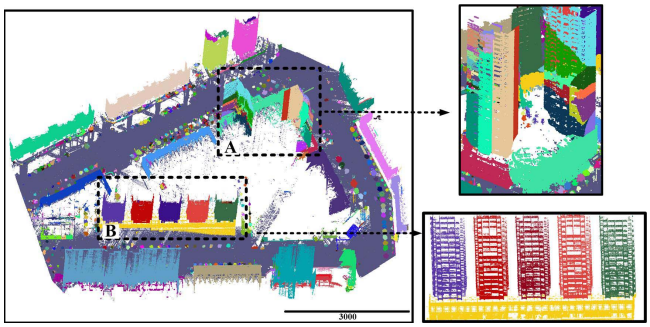


Fig. 14. Clustering results of a large-scale urban point set.

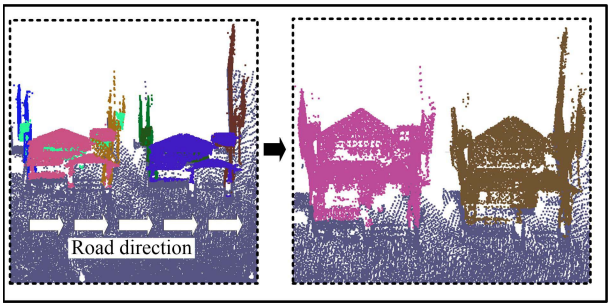


Fig. 15. Merging results based on the Euclidean distance.

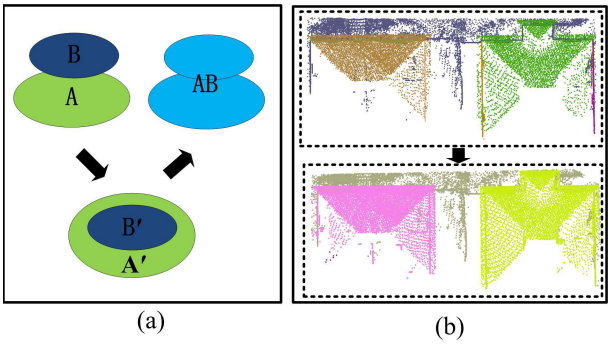


Fig. 16. Merging results based on the subset information.

RIEGL scanner. Fig.14 shows results of a large-scale urban point set (528 MB, 10,870,886 points) collected by the Optech scanner.

Segmentation challenges in the residential point set are (1) the split of overlapping trees and (2) the split of houses with different structures. As shown in Fig.13, we succeed to separate trees and houses into different clusters in region A. A house instance is split into several regions as shown in region B. Segmentation challenges in the urban point set are the split of objects with varied sizes. Various high buildings and low vegetation are segmented as shown in Fig.14 region A. A high building is separated from the base, because of the data incompleteness in connectivity regions as shown in Fig.14 region B.

Both 3DNCut and MinCut require human-computer interaction in the point cloud segmentation, for example, they require presetting the number of objects in the scene. The MinCut requires manually setting center points and radius for each object, which is extremely time-consuming in the large-scale scenes. PEAC is proposed for dealing with organized point clouds. Therefore, it is not practical to test 3DNCut, MinCut and PEAC on large-scale scenes.

VII. Discussion

A. Merging Regions

Since our clustering results can not be used in the object detection or recognition, we propose a merging strategy to combine regions into object instances. The expected outcome is that after the rule-based merging process, regions from the same object are combined into one instance. In this paper, we merge regions using the following two rules.

(1) The first rule is based on Euclidean distances between clusters. With the help of the ground information in MLS data demonstrated in the left of Fig.15, we can easily obtain road direction in a local region (i.e. 10 m by 10 m). When calculate Euclidean distances between the point p_i and p_j, if the direction from p_i to p_j is parallel to the local road direction, $d(p_i, p_j)$ will be enlarged by 5 times. This rule is designed for the distance calculation in the XOY plane. It is worth noting that spatially close clusters are merged into one component in the above-mentioned clustering process. Therefore, we increase the Euclidean distance threshold in the merging of regions. If the closest distance between two clusters is less than 5 m, they will be merged into one cluster. This rule works well in merging houses, vehicles and trees as shown in the right of Fig.15.

(2) The second rule is based on the subset information. As shown in the left of Fig.16, the cluster A and B are neighbors in the Z-axis direction. A' and B' are projection point sets of the cluster A and B in the XOY plane, receptively. If $\frac{|A' \cap B'|}{|A'|} > 0.9$ (i.e. $A' \subset B'$) or $\frac{|A' \cap B'|}{|B'|} > 0.9$ (i.e. $B' \subset A'$), the cluster A and B will be merged into one cluster AB. This rule is designed for merging clusters in the Z direction, such as roofs and facades as shown in the right of Fig.16.

We use the above-mentioned rules one-by-one in the automatic merging process. The merging results of the above residential and urban scenes are shown in Fig.17 and Fig.18, respectively. Each bounding box indicates an object instance after the rule-based merging process. Fig.17 region A demonstrates the merging of trees and Fig.17 region B shows the merging of houses. Fig.18 region A demonstrates the merging of buildings and Fig.18 region B shows the merging of buildings. Regions from one object are merged into the same instance.

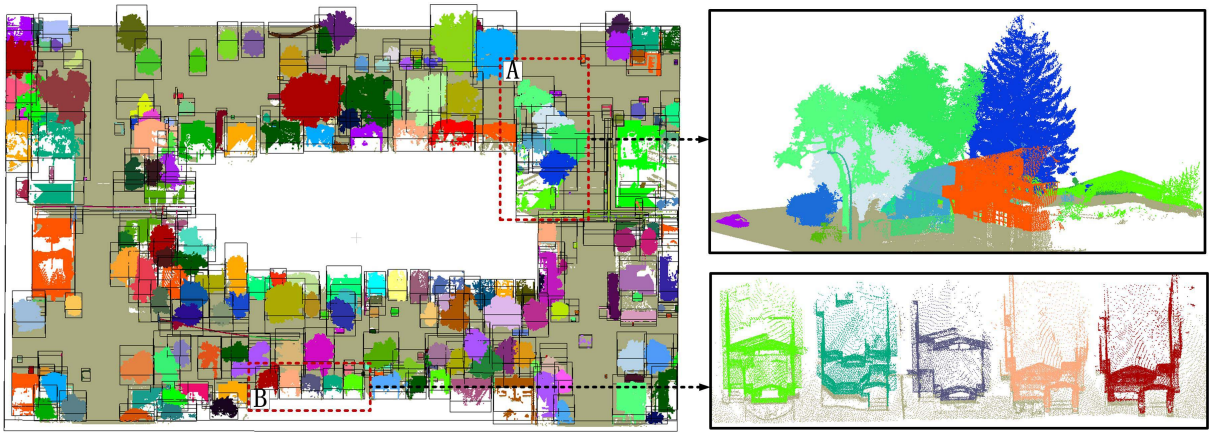


Fig. 17. Merging performance on residential scene.

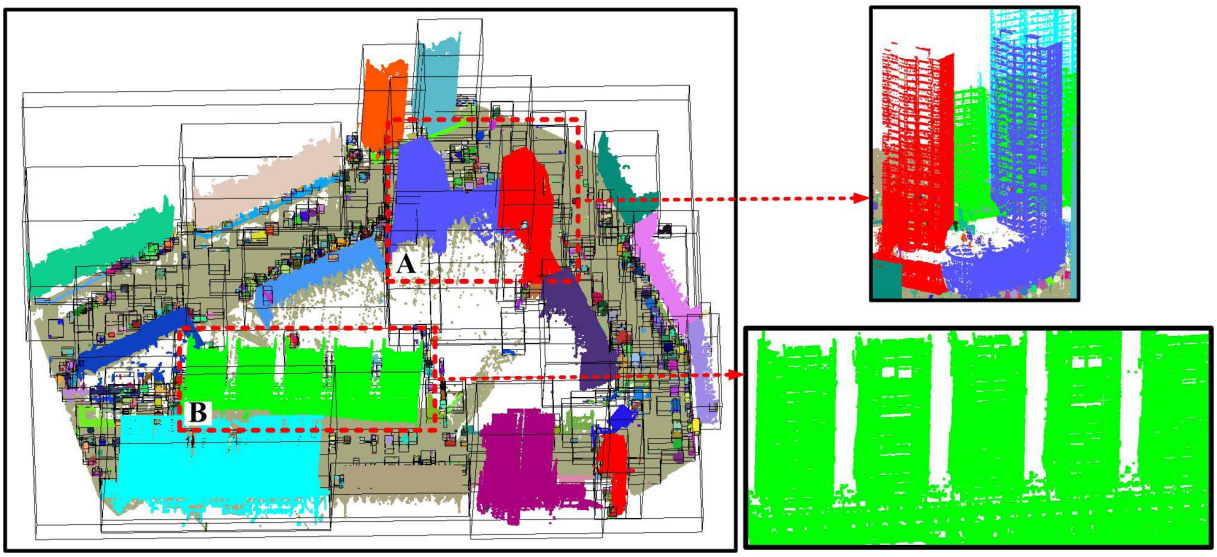


Fig. 18. Merging performance on urban scene.

TABLE I
EXECUTION TIME OF EACH ALGORITHM

Dataset	Number of points	Density (points/m^2)	Time cost (seconds)					
			KMiPC	KNNiPC	3DNCut	MinCut	PEAC	OHC
HouseSet	73899	821	0.53	0.76	61.67	15.99	4.92	10.63
BushesSet	7046	793	0.04	0.04	0.60	3.16	0.24	1.63
LamppostSet	52403	4645	0.31	0.21	29.96	31.92	3.02	9.91
TreesSet	257469	858	1.86	1.84	520.14	122.21	111.32	35.96
PowerlinesSet	154307	1836	1.07	0.58	234.35	139.53	33.65	16.23

B. Complexity Analysis

The proposed OHC works automatically for the point cloud clustering. There is no need for human-computer interaction, e.g. set initial number of clusters or locations of each object. The performance is competitive against the state-of-the-art methods.

The space complexity of the initial proximity matrix is $\mathcal{O}(N^2)$. Since the proximity matrix is symmetric and most of its elements are ∞, we use sparse matrix strategy to reduce the space complexity. The time complexity relies on the Kuhn-Munkres algorithm which is $\mathcal{O}(N^2)$. To deal with the large-scale point cloud clustering, we present a strategy to increase our efficiency: 1) remove ground points using the Cloth Simulation Filter (CSF) approach [22]; 2) down-sample the above-ground points into a sparse point by randomly removing points from input data; 3) apply the

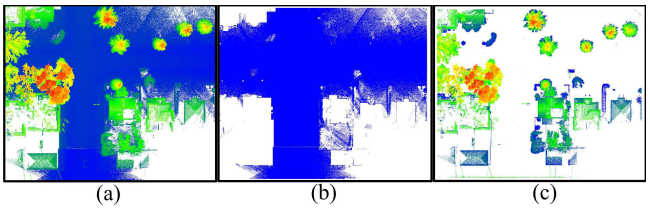


Fig. 19. Ground removal. (a) Input scene. (b) Ground points. (c) Above-ground points.

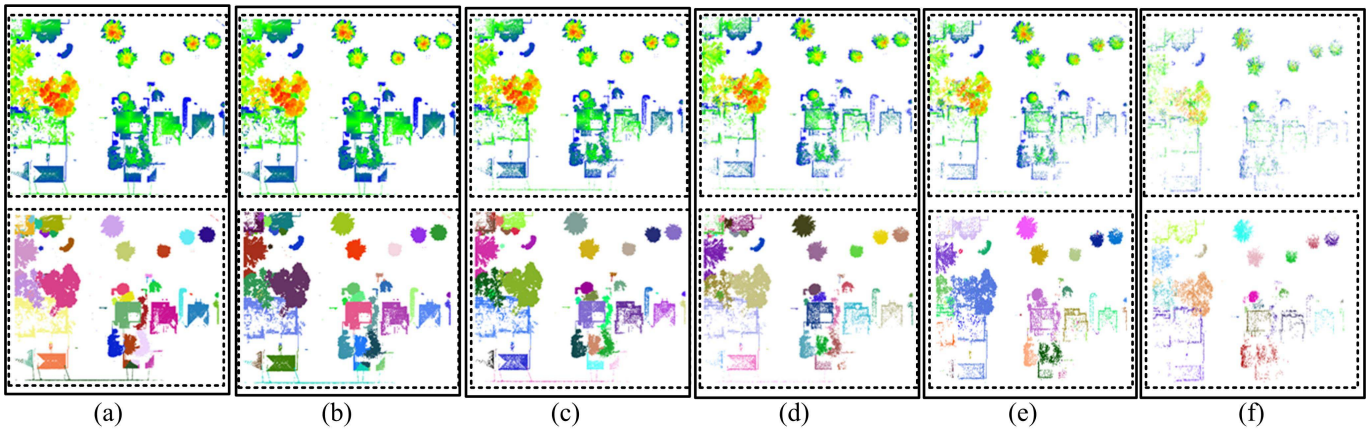


Fig. 20. Clustering results from down-sampled points. The first row shows input down-sampled data and the second row shows our results. (a) Points are down-sampled to 90%. (b) Points are down-sampled to 60%. (c) Points are down-sampled to 30%. (d) Points are down-sampled to 10%. (e) Points are down-sampled to 5%. (f) Points are down-sampled to 1%.

TABLE II
ANALYSIS OF THE SEGMENTATION FROM DOWN-SAMPLED POINTS

Test	Level(%)	Points	Clusters	$n_{acc}(\%)$	Time(mins)
a	90	1517974	50	92.7	9.566
b	60	910780	47	93.5	5.814
c	30	455390	45	91.9	3.219
d	10	151800	43	90.5	1.488
e	5	75899	30	64.2	0.983
f	1	15180	24	52.9	0.422

proposed OHC to the down-sampling data; 4) assign those discarded points from the down-sampling process to their closest clusters.

The cost time in each algorithm is shown in Table.I. In the comparison, only KMiPC and KNNiPC perform better than ours. 3DNCut is slower than the proposed method when the scale of the scene is large. For MinCut, the human-computer interaction is very time-consuming. The organization of point clouds in PEAC is not counted in Table.I, which will cost lots of time. In order to deal with the high complexity issue, Fig.19 shows the performance of OHC on down-sampling data. Fig.19 (a) shows an input scene containing vegetation, buildings and roads. Ground are removed by the CSF approach as shown in Fig.19 (b). The rest of above-ground points are shown in Fig.19 (c) which are down-sampled to different levels randomly. The first row of Fig.20 demonstrates down-sampled points by 90%, 60%, 30%, 10%, 5% and 1%. The second row of Fig.20 shows the corresponding object segmentation results.

In the case of inputting down-sampling data, although the accuracy of the overlapping region segmentation is decreased and there appear the over-segmentation and under-segmentation as shown in Fig.20 and Table II, OHC provides promising results at different sampling levels. The accuracy and cost time are shown in Table II. This experiment was done on a Windows 10 Enterprise 64-bit, Intel Core i7-6900k 3.20GHz processor with 64 GB of RAM by using Matlab R2018a. By using the down-sampling strategy, the proposed OHC succeeds in segmenting objects from a large-scale point set effectively.

VIII. CONCLUSIONS

This paper investigates a new optimal hierarchical clustering (OHC) method for achieving object instances from 3D mobile LiDAR point clouds. The cluster combination is obtained by the matching in a point-based graph, and the combination is optimized by solving the minimum-cost perfect matching in a bipartite graph. In the sensitivity analysis, the standard deviation of the accuracy is less than 1.5% through all experimental scenes, which shows the algorithm's robustness to different parameters. To merge regions into object instances, this work proposes a rule-based grouping strategy to merge regions into the same object. The drawback of the proposed OHC is the time complexity, which is addressed by using the sampling strategy. Performance on the residential and urban point sets shows that the proposed method is effective in clustering regions from different scenes and superior to the state-of-the-art methods in terms of the accuracy.

Future work will focus on adding the dissimilarity measure of the intensity and color information in the proximity matrix. Besides, we will try to improve the efficiency of the combination by using supervoxel techniques in the clustering.

REFERENCES

[1] H. Wang, C. Wang, H. Luo, P. Li, Y. Chen, and J. Li, "3-D point cloud object detection based on supervoxel neighborhood with Hough forest framework," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 4, pp. 1570–1581, Apr. 2015.

[2] L. Nan and P. Wonka, "PolyFit: Polygonal surface reconstruction from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Oct. 2017, pp. 2353–2361.

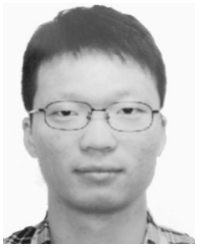
[3] Y. Yu, J. Li, H. Guan, C. Wang, and J. Yu, "Semiautomated extraction of street light poles from mobile LiDAR point-clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 3, pp. 1374–1386, Mar. 2014.

[4] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3D point clouds in urban environments," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep./Oct. 2009, pp. 2154–2161.

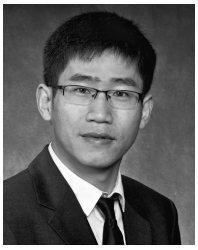
[5] Y. Zhang, J. Wang, X. Wang, and J. M. Dolan, "Road-segmentation-based curb detection method for self-driving via a 3D-LiDAR sensor," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 12, pp. 3981–3991, Dec. 2018.

[6] K. Li, X. Wang, Y. Xu, and J. Wang, "Density enhancement-based long-range pedestrian detection using 3-D range data," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1368–1380, May 2016.

[7] F. Wu *et al.*, "Rapid localization and extraction of street light poles in mobile LiDAR point clouds: A supervoxel-based approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 2, pp. 292–305, Feb. 2017.
[8] P. Corcoran, A. Winstanley, P. Mooney, and R. Middleton, "Background foreground segmentation for SLAM," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1177–1183, Dec. 2011.
[9] B. Douillard *et al.*, "On the segmentation of 3D LIDAR point clouds," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2011, pp. 2798–2805.
[10] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3D laser data," in *Proc. ICRA*, May 2008, pp. 4043–4048.
[11] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[12] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *Int. J. Comput. Vis.*, vol. 70, no. 2, pp. 109–131, 2006.
[13] Y.-T. Su, J. Bethel, and S. Hu, "Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications," *ISPRS J. Photogramm. Remote Sens.*, vol. 113, pp. 59–74, Mar. 2016.
[14] Y. Lin, C. Wang, D. Zhai, W. Li, and J. Li, "Toward better boundary preserved supervoxel segmentation for 3D point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 143, pp. 39–47, Sep. 2018.
[15] G. Lavoué, F. Dupont, and A. Baskurt, "A new CAD mesh segmentation method, based on curvature tensor analysis," *Comput.-Aided Des.*, vol. 37, no. 10, pp. 975–987, 2005.
[16] R. A. Kuçak, E. Özdemir, and S. Erol, "The segmentation of point clouds with k-means and ann (artifical neural network)," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. XLII-1/W1, pp. 595–598, Jun. 2017. [Online]. Available: http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-1-W1/595/2017/
[17] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2014, pp. 6218–6225.
[18] D. Chaudhuri and B. B. Chaudhuri, "A novel multiseed nonhierarchical data clustering technique," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 27, no. 5, pp. 871–876, Sep. 1997.
[19] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Trans. Inf. Theory*, vol. 29, no. 4, pp. 551–559, Jul. 1983.
[20] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.
[21] G. W. Milligan and M. C. Cooper, "A study of the comparability of external criteria for hierarchical cluster analysis," *Multivariate Behav. Res.*, vol. 21, no. 4, pp. 441–458, 1986.
[22] W. Zhang *et al.*, "An easy-to-use airborne LiDAR data filtering method based on cloth simulation," *Remote Sens.*, vol. 8, no. 6, p. 501, Jun. 2016.



Sheng Xu received the B.Eng. degree in computer science and technology from Nanjing Forestry University and the Ph.D. degree in digital image systems from the University of Calgary. In 2018, he joined the College of Information Science and Technology, Nanjing Forestry University, where he is currently an Assistant Professor. His current research interests include mobile mapping, vegetation mapping, and computer vision.



Ruisheng Wang received the B.Eng. degree in photogrammetry and remote sensing from Wuhan University, the M.Sc.E. degree in geomatics engineering from the University of New Brunswick, and the Ph.D. degree in electrical and computer engineering from McGill University. He has been an Industrial Researcher with Nokia, Chicago, IL, USA, since 2008. In 2012, he joined the Department of Geomatics Engineering, University of Calgary, where he is currently an Associate Professor. He has also been a Distinguished Professor of Geographical Sciences at Guangzhou University. His research interests include geomatics and computer vision.

Hao Wang received the B.Eng. degree from Nanjing Forestry University and the Ph.D. degree from Tongji University. In 1983, he joined the College of Landscape Architecture, Nanjing Forestry University, where he is currently a Professor. His research interests include landscape architecture design, urban planning, and green space system Analysis. He is a member of the Editorial Committee of the Chinese Landscape Architecture.

Han Zheng received the B.S. and M.S. degrees in geodesy and geomatics engineering from Wuhan University, Wuhan, China, in 2011 and 2013, respectively. He is currently pursuing the master's degree in digital image system with the Department of Geomatics Engineering, University of Calgary, Calgary, AB, Canada. His current research interests include laser measurement application, computer vision, digital image processing, and object recognition from three-dimensional point clouds.