Received January 9, 2020, accepted January 28, 2020, date of publication February 4, 2020, date of current version February 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971549

Tree Skeletonization for Raw Point Cloud Exploiting Cylindrical Shape Prior

LIXIAN FU, JI LIU, JIANLING ZHOU, MIN ZHANG, AND YAN LIN
College of Computer Science, Chongqing University, Chongqing 400044, China

Corresponding author: Ji Liu (liujiboy@cqu.edu.cn)

This work was supported in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2019jcyj-msxmX0033, in part by the National Natural Science Foundation of China under Grant 61701051, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019CDYGYB012.

ABSTRACT Tree skeleton extraction plays a fundamental role in reconstructing both biological and structural models of trees. However, traditional approaches can be ineffective and problematic in guaranteeing the topological correctness and centeredness of the tree skeleton when the tree point clouds contain noise and occlusions. To overcome this limitation, we present a tree skeletonization method to generate topologically correct and well-centered tree skeletons. We extract an initial skeleton from the tree point clouds via an octree and level set method, use cylindrical prior constraint (CPC) optimization and the estimated radii of branches to yield corrected positions of improper joints, and finally obtain updated skeletons with improved smoothness. The good centeredness of our proposed method is intrinsically achieved by (1) exploiting the cylindrical shape prior and calculating the CPC in the local neighborhood and (2) feeding the prior knowledge regarding the radii of tree branches into a topology refinement algorithm to yield near-optimal estimates of the positions of the skeleton points. To evaluate our method, we construct a novel tree point cloud data set with known ground truth and propose three quantitative assessment methods: skeleton points deviation (SPD), bifurcation points coverage (BPC) and endpoints coverage (EPC). The quantitative assessment and visual assessment show that the proposed method clearly outperforms traditional ones in terms of topology correctness and centeredness of the extracted tree skeleton.

INDEX TERMS Cylindrical prior constraint, point cloud, skeleton extraction.

I. INTRODUCTION

Due to the important role of tree models in computational forestry [1], function-structure plant modeling [2], urban planning and many other fields, substantial research effort has been devoted to tree model reconstruction [3]–[6]. The existing techniques usually include the following steps. First, a point cloud of a tree is obtained and preprocessing, such as removing noise, outliers, and leaves, is performed. Second, the skeleton of trees, which capture the essential topological invariance of the 3d model, are extracted. Third, the geometry of the tree is reconstructed based on the extracted skeleton.

The topology of the reconstructed tree is usually established in the skeleton-extraction phase [7], making skeleton extraction from the point cloud the most important task in tree reconstruction. Three common challenges are associated with extracting skeletons from point clouds [8]: (1) the point clouds may contain noise and outliers; (2) there can be large amounts of missing data in point clouds due to data occlusions; (3) the point density in the point clouds can vary substantially. Many skeleton extraction algorithms have been proposed [9]–[11] to overcome these challenges. However, many of these algorithms fail to extract a topology-preserving skeleton from point clouds of trees that have complex structures. Meanwhile, because there are almost no quantitative evaluation methods that can measure topological correctness and centeredness, the real performance of existing algorithms is difficult to assess.

In this paper, we present a novel skeletonization method that yields well-centered and topology-preserving tree skeletons from point clouds with noise and occlusions. The proposed method extracts the initial skeleton points by means of an octree [12] and level set method [13]. Then, we observe that tree branches are generally cylindrical in shape in nature; thus, we employ the cylindrical prior constraint (CPC) optimization to improve the centeredness of the skeletons.

The associate editor coordinating the review of this manuscript and approving it for publication was Chintan Amrit.

Finally, we correct the skeleton topology with a topology refinement process by estimating the radii of the branches.

Compared to existing skeleton extraction methods, the incorporation of octree subdivision makes our method resistant to noise and outliers. Most importantly, our proposed method makes considerable improvements over traditional level set methods [3]–[5], [13] and greatly enhances the centeredness and topological correctness of the resulting skeleton.

Additionally, to validate the effectiveness of our proposed approach, we present a synthetic data set of tree point clouds with known ground truths and three quantitative assessment methods: skeleton points deviation (SPD), bifurcation points coverage (BPC) and endpoints coverage (EPC). We perform extensive case studies based on synthetic point clouds and real tree point clouds. The visual comparisons and detailed quantitative assessments demonstrate that our method clearly outperforms traditional baselines.

II. RELATED WORK

Curve skeletons have been widely used in computer vision, image processing, life sciences and plant morphology [7]; thus, many methods have been proposed. The latest research in this field, presented at the SkelNetOn@CVPR19 workshop, includes Demir *et al.* [14], Yang *et al.* [15], Liu *et al.* [16] and Atienza *et al.* [17]. However, the recent methods at the SkelNetOn@CVPR19 workshop focus on using deep learning to extract skeletons from 2D images, whereas our method aims to obtain curve skeletons from unorganized 3D point clouds. Therefore, in this section, we review only skeletonization algorithms for unorganized 3D point clouds. The most closely related methods can be roughly classified as level set methods, optimization-based methods and graph-based methods.

A. LEVEL SET METHODS

Verroust and Lazarus [13] propose the level set method, which is capable of extracting skeletons of tree-like objects. They use geodesic distance on the k-nearest neighbor graph to generate multiple level sets. The skeleton point is the barycenter of the connected components in the level sets. This method can capture the basic topology of tree-like objects, is easy to implement, and is thus widely used [3]–[5]. However, the level set method is limited since it does not consider the varying density or missing data in point clouds, and the centeredness of the skeleton is not guaranteed. Such limitations can be avoided by introducing octree subdivision and CPC optimization, as in our proposed method.

B. OPTIMIZATION-BASED METHODS

The L_1-median has a long history in statistics and shows good resistance to noise and outliers in point sets. [10] employs localized L_1-medians to construct a skeleton. The L_1-medial method does not impose strong requirements on the quality of the point cloud or specific topology of the captured object. These methods use a weighting function with a supporting radius to define the size of the local neighborhood. The supporting radius is gradually increased to yield clean and well-connected skeleton points. However, this method can be ineffective in yielding a tree skeleton for two reasons. First, when two tree branches are close to each other in the point cloud, the L_1-medial skeleton may contain cycles, whereas the skeleton of a tree ought to be acyclic. Second, the bridge points are occasionally not correctly connected to nearby branches: some parameters related to bridge points must be carefully set to guarantee the topological correctness of the tree skeleton. In our method, we use the L_1-median to compute the center of the local point set rather than directly generating all the skeleton points.

Mei *et al.* [11] aim to extract skeletons from point clouds with large portions of missing data. They first use the L_1-medial method to generate a coarse tree skeleton; then, they devise an iterative data-completion scheme to recover missing data. The coarse tree skeleton is finally refined through a L_1-minimum spanning tree algorithm. Song *et al.* [18] compute a distance field from a voxelized point cloud, extract an initial skeleton from the distance field, and finally use the L_1-medial method to obtain the final skeleton.

Tagliasacchi *et al.* [9] note that a curved skeleton could be regarded as a generalized rotational symmetry axis of a shape (ROSA). Therefore, the normal information can be utilized to compensate for missing data when large portions of data are missing in a point cloud. The position of a skeleton point in the local set of points is computed by minimizing the sum of the projected distances to the normal extensions of the data points.

The Laplace operator is a useful tool for mesh smoothing. Au *et al.* [19] apply the Laplace operator on the mesh to move vertices along their approximate curvature normal direction. Laplace operator-based vertex contraction is implemented by minimizing the quadratic energy. They then convert the contracted mesh into a 1d curve skeleton by means of a connectivity surgery process. Cao *et al.* [20] extend their work by applying the Laplace operator to point cloud models and replacing the connectivity surgery process with a topological thinning method. However, the results are sensitive to parameter tuning, e.g., the contraction weights update factor.

Qin *et al.* [21] introduce an optimal mass transport method to extract the mass-driven curve skeleton (MdCS) from 3D point clouds. This method minimizes the Wasserstein distance with an entropic regularization term between two probability measures: the mass distribution of MdCS and the raw point cloud. This method is robust to noise, missing data and unoriented data. However, the authors do not discuss the results of skeletons extracted from tree-shaped point clouds.

C. GRAPH-BASED METHODS

Livny *et al.* [22] propose a method for reconstructing tree skeletal structures automatically. They construct a graph from input points and then extract minimum spanning trees that represent the initial tree skeletons. They then refine the initial

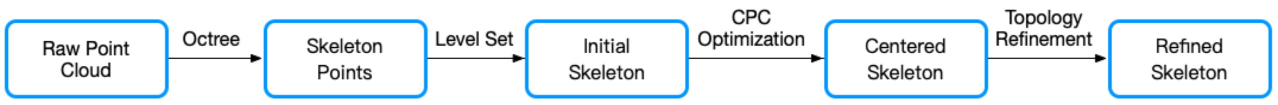

FIGURE 1. **The procedure of our skeletonization method.**

tree skeletons via a global optimization that is driven by biological priors. Their method is capable of extracting tree skeletons from point clouds of multiple overlapping trees. However, the method is based on the assumption that the graph is built from sparse TLS point clouds and is thus ineffective when applied to dense point clouds.

Bucksch *et al.* propose SkelTre [8], a graph-based method for extracting tree skeletons from point clouds with twigs and leaves. In this method, they build a graph in octree [12] subdivision and reduce the graph to a skeleton according to a set of given rules. However, the centeredness of the skeleton is sensitive to a large portion of missing data. The SkelTre method is employed in [23] to generate skeletons of tree branches.

III. OVERVIEW OF SKELETONIZATION

A point cloud is a set of data points $X = \{x_i | x_i \in R^3\}$ sampled on the surface of an object. The process of extracting a skeleton from the point cloud of a tree is called tree skeletonization. A tree skeleton is an undirected acyclic graph whose vertices are the loci of the centers of the maximal inscribed balls of the point cloud [24]. In this paper, the vertices of the tree skeleton are named skeleton points. Let $Q = \{q_j | q_j \in R^3\}$ denote the centers of the maximal inscribed balls [24]; hence, skeleton points Q represent an optimal set of points:

$$Q = argmin \sum_{i \in I} \sum_{j \in J} ||x_i - q_j|| \theta(||x_i - q_j||, r_i) \tag{1}$$

where $\theta(d, r) = e^{-d^2/(r/2)^2}$ is a rapidly decaying smooth function with support radius r, i.e., the radius of the maximal inscribed ball [24]. A major challenge in tree skeletonization is that point clouds contain noise and occlusions, which not only make the skeleton points shift from the center of the local inscribed balls but also affect the topological correctness of the connected skeleton points.

To address these issues, we propose a tree skeletonization algorithm that consists of four steps (as illustrated in Fig.1):

(1) Tree skeleton points computation. We subdivide the raw point cloud by octree and take the L_1-medians of the octree cells as the initial skeleton points. This step downsamples the point cloud, makes the skeletonization algorithm resistant to noise and varying density of the point cloud, and reduces the scale of the data.

(2) Initial tree skeleton extraction. We build a k-nearest neighbor graph from the initial skeleton points. With this k-nearest neighbor graph and a selected root point, we construct a geodesic distance function by calculating the shortest paths in the graph between all vertices and the root point. We then extract the initial tree skeleton using level sets [13] of the geodesic distance function.

(3) Cylindrical prior constraint optimization. Since some skeleton points near missing data areas can be off-centered and thus harmful to the centeredness of the resulting skeleton, we introduce CPC optimization to recenter such off-centered skeleton points.

(4) Topology refinement. Due to noise and occlusions, skeleton points in branching regions of the tree may be misconnected. These points can further lead to incorrect topology. We note that the local region of a tree branch is a generalized cylinder and design a topology refinement process. This process aims to improve the resistance to noise and occlusions by estimating the radii of branches and updating the local topology. We finally smooth the extracted tree skeleton via an interpolation algorithm and reconstruct the 3D tree model based on the smoothed skeleton.

The above steps are discussed in detail in Section IV, and the extracted skeletons and corresponding tree models are elaborated in Section V.

IV. TREE SKELETONIZATION

A. SKELETON POINTS COMPUTATION

The generated raw point cloud is a set of data points $X = \{x_i | x_i \in R^3\}$ sampled on the surface of visible tree branches, which are noisy. To remove noise, we use an octree to downsample the input point cloud (see Fig.2(b)). An octree is a tree structure for partitioning 3D space by recursively subdividing the space into eight octants. Each node in an octree is a cubic cell that contains a subset of data points. Our proposed skeletonization procedure is performed on octree cells rather than the original data points.

With an appropriate subdivision depth d_o ($d_o = 2 + log_8 |X|$ by default), the octree generation procedure is as follows:

(1) Compute the bounding box of the tree and initialize an empty cell list. To simplify our algorithm, we expand the bounding box to a cubical box of size $l \times l \times l$. Then, a cell of size $l \times l \times l$ containing all data points X is created and placed into the cell list.

(2) For each parent cell in the list, if the cell (size $m \times m \times m$) contains data points, then the cell is divided into eight child cells (size $\frac{m}{2} \times \frac{m}{2} \times \frac{m}{2}$). The data points in the parent cell are also divided and added to the corresponding child cells. Then, the parent cell is removed from the list and the eight child cells are added to the list.

(3) If the current subdivision depth for the octree is less than depth d_o, step (2) is repeated; otherwise, the subdivision is complete.

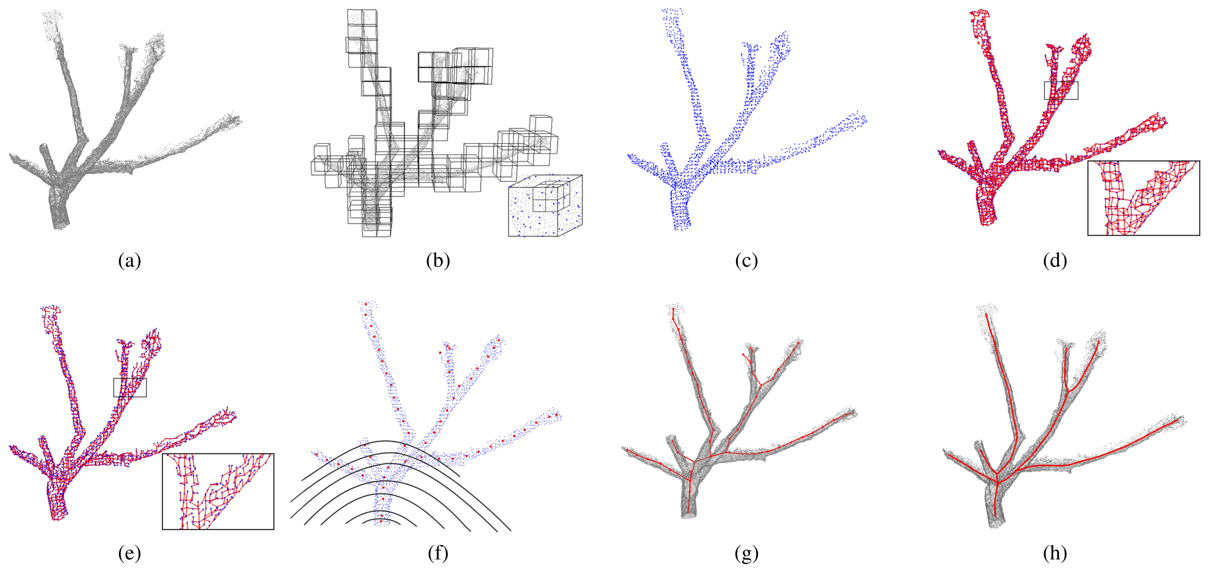

FIGURE 2. Overview of our skeletonization method. (a) Raw point cloud. (b) Octree generated from a point cloud. (c) The centroid of every octree cell. (d) Neighbor graph composed of centroids. (e) Geodesic graph extracted from the neighbor graph. (f) Original skeleton points (red) are acquired. (g) Initial skeleton extracted from the level sets. (h) Final skeleton after refinement.

In the following, we compute the skeleton points Q from octree cells. For each octree cell that contains k data points, $X_j = \{x_1^j, x_2^j, \ldots, x_k^j\}$, the spatial location of the corresponding skeleton point q_i is computed using the L_1-median (see Fig.2 (c)):

$$q_i = \arg\min_{q_i} \sum_{x_k^j \in X_j} ||q_i - x_k^j|| \tag{2}$$

B. INITIAL TREE SKELETON EXTRACTION

Given a set of skeleton points, $Q = \{q_i | q_i \in R^3\}$ computed from octree cells, we define the k-nearest neighbor ($k = 26$ by default) graph $G(Q, k)$ of a point cloud as an undirected graph (Fig.2 (d)):

(1) The vertices of $G(Q, k)$ are the points of Q;

(2) (q_i, q_j) denotes an edge of $G(Q, k)$ if q_i is one of the k nearest neighbors of q_j.

The time complexity of building the k-nearest neighbor graph assisted with octree is usually $O(nlogn)$. Because each vertex is the median point of an octree node, this procedure can be further accelerated. For each vertex q_i, we look for all the octree cells that are adjacent to the corresponding cell of q_i instead of computing and comparing Euclidean distances. We further connect the corresponding vertex of those adjacent cells and finally build the nearest neighbor graph within $O(n)$ time.

We further define $f(q_i)$ as a geodesic distance function whose value represents the length of the shortest path from point q_i to the selected root point, q_s, on $G(Q, k)$. Then, we build a geodesic graph $G_{geo}(Q)$ in which edge (q_i, q_j) represents the shortest geodesic path from q_i to q_j (Fig.2(e)).

Therefore, we extract an initial tree skeleton from the point cloud using the level sets of the geodesic distance function f, similar to [13] (see Fig.2(f)(g)). The level set $L(f, c, \rho)$ of geodesic distance function f is a set of points defined as $L(f, c, \rho) = \{q_i | c \le f(q_i) < c + \rho, q_i \in Q\}$. $q_i \in L(f, c, \rho)$ is a point with geodesic distance between c and $c + \rho$. The detailed extraction steps for the initial skeleton are as follows:

(1) We use the geodesic distance function to divide the skeleton points Q into k level sets $\{L(f, c_1, \rho), L(f, c_2, \rho), \ldots, L(f, c_k, \rho)\}$, where $c_i = c_{i-1} + \rho$.

(2) For each level set $L(f, c_i, \rho)$, we construct a fully connected graph by connecting the skeleton points in this set. Then, this fully connected graph is divided into m connected subgraphs by means of a spectral clustering algorithm [25]. If the skeleton points in $L(f, c_i, \rho)$ belong to different branches, m is equal to the number of branches; otherwise, m is equal to 1. We set the L_1-medians of the subgraphs as new skeleton points.

(3) Q' is the new skeleton point set computed from step (2), and we build a new geodesic graph $G_{geo}(Q')$. The initial skeleton is a geodesic graph $G_{geo}(Q')$ whose edges represent the topology of the tree point cloud.

C. CYLINDRICAL PRIOR CONSTRAINT

The extracted tree skeleton is an undirected acyclic graph $G_{geo}(Q')$, and Q' is the set of skeleton points that are L_1-medians of the level sets. The L_1-median is inherently robust to noise and outliers. However, the skeleton points calculated using the L_1-median are not well centered when there is a large proportion of occlusions in the point cloud (see Fig.3).

Considering the possible missing data in the occluded regions, we observe that tree branches in nature are generally cylindrical in shape, which means that the distances from each point of a cross section to the local median are similar (see Fig.4). Inspired by this characteristic, we deduce two

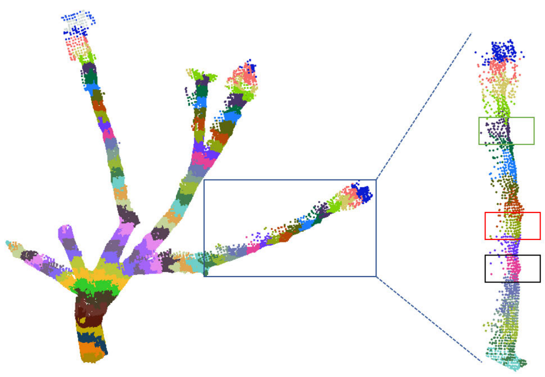

FIGURE 3. The point cloud of a tree with missing data.

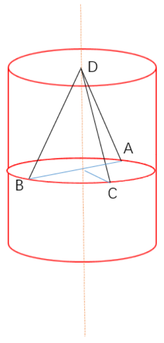

FIGURE 4. Point D is in the cylindrical axis of symmetry, and the distances from A, B, and C of a cross section to point D are similar.

constraint conditions. (1) The local median constraint makes the median point represent the location of the local points, and we choose the L_1-median as the local median constraint term because of its strong robustness. (2) The equal-distance constraint yields the median point close to the cylindrical axis of symmetry, maintaining a similar distance to each point in the neighborhood. We introduce an optimization process called CPC to recenter the skeleton points.

The CPC yields the median point that minimizes the sum of the Euclidean distances from local points to the median point and the weighted variance of these distances. Formally, for a local point set $X_s = \{x_1^s, x_2^s, \ldots, x_m^s\}$, the optimized local median υ^s is computed via the following optimization:

$$\upsilon^s = \arg\min_{\upsilon^s} \sum_{i=1}^{m} ||\upsilon^s - x_i^s|| + \lambda_s \sigma_s^2 \tag{3}$$

The local median constraint is the left term, where x_i^s is in the set X_s. The equal-distance constraint is the right term $\lambda_s \sigma_s^2$, where the weight factor λ_s controls the penalty effect. σ_s^2 is the variance of distance d_i^s from point $x_i^s \in X_s$ to the local median υ^s:

$$\sigma_s^2 = \frac{1}{m}\sum_{i=1}^{m} (d_i^s - \mu_s)^2 \tag{4}$$

The distance d_i^s is defined as:

$$d_i^s = ||\nu^s - x_i^s|| \tag{5}$$

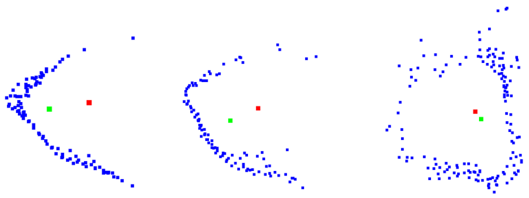

FIGURE 5. The data points shown in top view. The green point is the L_1-median. The red point is the centroid computed using CPC with $\lambda = 10^4$.

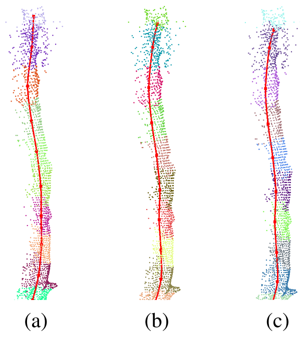

FIGURE 6. Extracted skeleton of the marked branch shown in Fig.5 using CPC with different λ. When λ is 0, the extracted red curve is the L_1-medial skeleton.

The average distance μ_s is:

$$\mu_s = \frac{1}{m}\sum_{i=1}^{m} d_i^s \tag{6}$$

Fig.5 shows three local cross sections of data points and the median computed using the L_1-median and CPC. Clearly, the equal-distance constraint prevents the L_1-median from being attracted to dense points, and the CPC median is closer to the theoretical median under the assumption that the local branch is cylindrical.

Fig.6 shows skeletons extracted using CPC with different λ_s. CPC can significantly enhance the centeredness of the skeleton when the point cloud has missing data, and the value of the weight factor λ_s plays an important role in optimizing the positions of the skeleton points. In Fig.6, as λ_s increases, the median point gradually becomes centered. In this paper, we use gradient descent to optimize Equation 3. In iteration k, λ_s^k is recomputed as follows:

$$\lambda_s^k = m^2 \mu_s^{k-1} \tag{7}$$

where $\mu_s^{(k-1)}$ is the average distance μ_s in iteration $k-1$. μ_s^0 is the average distance from the local points to the initial skeleton point. During optimization, λ_s gradually decreases until convergence.

D. TOPOLOGY REFINEMENT

The initial skeleton captures the basic topology of the tree; however, the complexity of the tree branching structure might result in unexpected vertices and edges when building the k-nearest neighbor graph (see Fig.8). Moreover, a skeleton

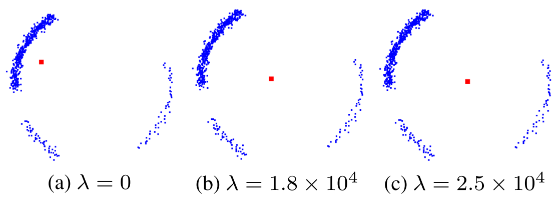

FIGURE 7. The point cloud contains 600 points. The centroid (red) is computed using CPC with different λ.

with the correct topology is difficult to extract in the bifurcation region because the distances between points in the bifurcation region are usually small. In addition, noise and occlusions can make the circumstances worse. The level set method cannot generate a good skeleton under such circumstances; therefore, we develop an algorithm to refine the topology of the tree skeleton and improperly located skeleton points, especially in the branching region.

$Q = \{q_i | q_i \in R^3\}$ is the initial skeleton points, and geodesic graph $G_{geo}(Q)$ is the initial skeleton. The pseudo code of our refinement algorithm is as follows.

Algorithm 1 Refinement

```
Input: Skeleton nodes $Q$, data points set $X$
Output: Refined skeleton
 1: for all $q_j$ in $Q$ do
 2:     if $q_j$ is a branching node then
 3:         pick up $n$ initial cluster centers
 4:         perform $k$-means clustering on $X_j$, obtain $X_{j,1}, \ldots, X_{j,n}$
 5:         compute centroids of clusters using CPC
 6:         for all $X_{j,i}$ in clusters do
 7:             compute distance $d_i$ and threshold $\tau$
 8:             if $d_i > \tau$ then
 9:                 add sibling node $q_{i,sibling}$ for $q_i$
10:                 $X_j = X_j \backslash X_{j,i}$
11:                 recompute the centroid of $X_j$
12:             end if
13:         end for
14:     else
15:         if parent of $q_j$ is a branching point then
16:             perform radius estimation for the current branch
17:         end if
18:     end if
19: end for
```

Before refinement, we sort the initial skeleton points Q according to geodesic distance value on $G_{geo}(Q)$ in descending order. Thus, the algorithm is applied to all branching regions in top-down order.

We use the neighborhood of q_i on $G_{geo}(Q)$ to determine whether q_i is a branching point: if the neighbor size of q_i is greater than 2, q_i is a branching point. For a branching point q_i that has n child branches, we perform k-means clustering on the data points in this region. The parameter k of k-means is n because the number of child branches is already known and reliable in the top-down refinement. The initial centroid for a cluster can be estimated based on known child branches. We select the n vertices closest to the n daughter branches as the initial centroid. This clustering process produces n clusters.

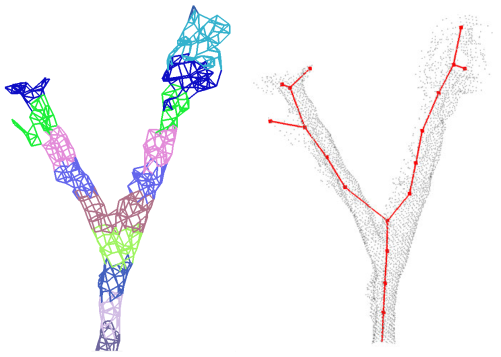

FIGURE 8. (a) Neighborhood graph colored according to level set. (b) Initial skeleton, colored red, extracted by connecting the centroids of every two adjacent components.

We propose a criterion to determine whether the k-means clustering result can be accepted. We calculate the distances $d_1, d_2, , d_n$ from the centroids of the clusters to the skeleton point q_i and compare these distances to a threshold τ. The threshold is calculated as follows:

$$\tau = \eta \sqrt{\sum_{i=1}^{n} {r_i}^2} \tag{8}$$

The value of parameter η varies depending on the current geodesic distance. The empirical value of η in our experiment is 1. r_i is the radius of skeleton point q_i, which is estimated using:

$$r_i = \frac{1}{m} \sum_{j=1}^{m} ||x_j - q_i|| \tag{9}$$

where point $x_j \in X$ is a neighbor of skeleton point q_i. This formula is also used in step 16 of the pseudo code. The method used to compute threshold τ is inspired by the observation from Leonardo da Vinci that the sum of the cross-sectional area of all child branches above a branching point is equal to the cross-sectional area of the branch below the branching point [26], [27].

If $d_i > \tau$, the centroid of the corresponding cluster is added to skeleton point set Q as a sibling node of skeleton point q_i. Later, the centroid of the cluster is recomputed using CPC. The process of steps 2-13 in the pseudo code is shown in Fig.9.

After refining the branching point, to enhance the smoothness of the skeleton, we perform cardinal spline interpolation [28]: five points are interpolated between every two adjacent skeleton points.

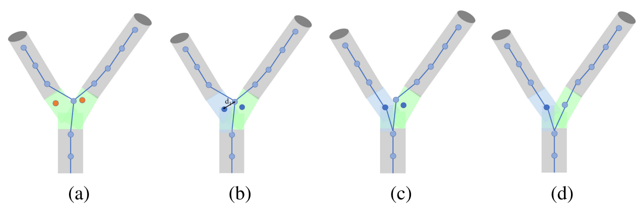

FIGURE 9. Process of refinement. (a) Pick up the initial clustering center. (b) Perform k-means clustering and compute the distance. (c) Insert sibling node. (d) Recenter original branching point.

V. RESULTS

To validate the effectiveness of our proposed skeletonization algorithm, we compare our method with L_1 [10], L_1-MST [11] and ROSA [9] in terms of the topological correctness and centeredness of the extracted skeletons. L_1 [10] shows good performance on a raw point cloud with considerable noise and outliers and large areas of missing data. Although L_1 is not designed for tree point clouds, the method can extract high-quality skeletons with complex branches. L_1-MST [11] integrates the advantages of both the L_1 and MST algorithms and can efficiently extract tree skeletons from imperfect point clouds. Same as our method, ROSA [9] uses a cylindrical shape prior and can produce robust curve skeletons over regions of a shape that are generally cylindrical. ROSA can also handle branch structures. In summary, all the methods chosen for comparison can extract skeletons from point clouds with complex branches, noise and occlusions. Moreover, L_1 and ROSA are open sourced by their authors. Although L_1-MST is not open source, it is not difficult to implement L_1-MST using the source code of L_1. Since most skeleton extraction methods are not open source, comparing our method with L_1, L_1-MST and ROSA can increase the reliability of the experimental results.

Our comparison consists of two parts: visual assessment and quantitative evaluation. For the former, we construct a new tree point cloud data set with known ground truths and propose a series of indices and approaches to conduct quantitative evaluations. For the latter, we perform a visual assessments on both real-world point clouds and synthetic point clouds.

We implement our skeletonization algorithm in the Java 8 environment. Then, we download the software provided by the authors of [10] and the code of [9] from the authors' repositories on Github. We normalize the point clouds and follow the authors' instructions before running their algorithms. To compare the performance of the algorithms, we use various point clouds that have missing data, noise, and varying point densities. We also implement the L_1-MST method according to the original paper. All the algorithms are run on a laptop with 8 GB memory and an Intel Core i7-8550U processor running at 1.80 GHz. The operating system is Windows 10. For each presented result for L_1, L_1-MST and ROSA, we have assessed at least five groups of parameters and selected the best skeleton for comparison.

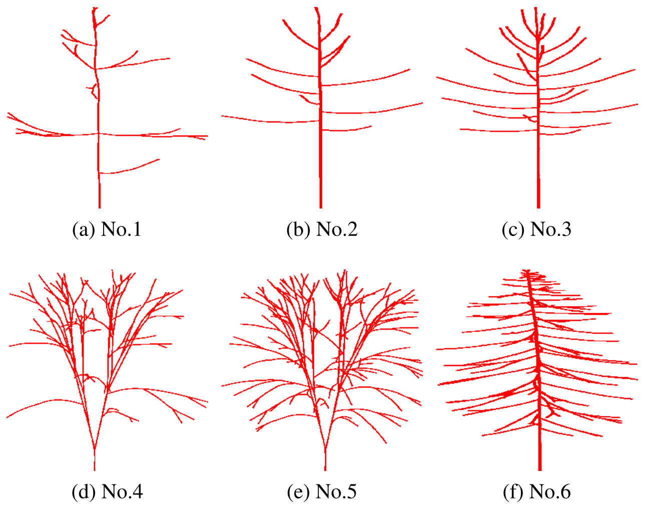

FIGURE 10. Six ground truth skeletons, indexed from 1 to 6.

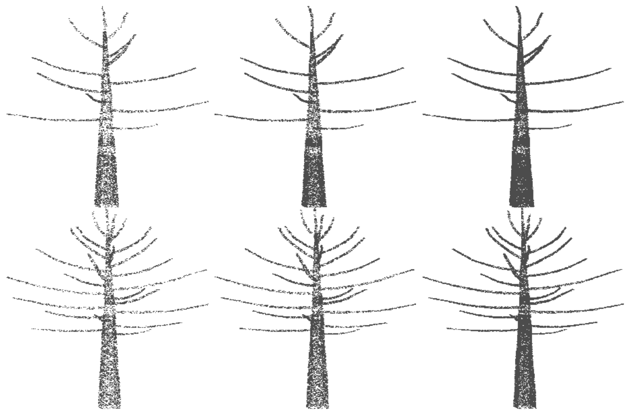

FIGURE 11. Point clouds with different numbers of data points. The point clouds in the first and second rows are generated based on ground truth skeletons No.2 and No.3, respectively. The number of points in the point cloud increases from left to right.

A. EXPERIMENTS ON THE SYNTHETIC DATA SET

1) SYNTHETIC TREE POINT CLOUD DATA SET

The lack of suitable tree point cloud data sets with known ground truth (3D skeleton models) has prevented direct quantitative evaluations. To enable such quantitative evaluations for assessing tree skeletonization algorithms, we construct a new data set that contains point cloud models of trees and the corresponding ground truth skeletons. This data set is generated via two steps. The first step is to draw a series of skeletons using particular programs. We draw six different ground truth tree skeletons, as shown in Fig.10, and for convenience of citing them in the following sections, we give each skeleton an index number. Based on these skeletons, the second step is to generate four types of point clouds: point clouds with different numbers of data points, point clouds with noise, point clouds with missing data, and point clouds with varying point density. Some examples are shown in Fig.11 - Fig.14

The details of all the tree point clouds used in our experiments are listed in Table 1.

TABLE 1. All point clouds used in the experiments.

Skeleton	Point Cloud	Point Cloud Name	Point Cloud Type
No.1	No.1-No.3	tree1_14045,tree1_28189,tree1_7088	Different Number of Points
	No.4-No.6	tree1_missing_1,tree1_missing_2,tree1_missing_3	Missing Data
	No.7-No.9	tree1_noise_1,tree1_noise_2,tree1_noise_3	Noise
	No.10-No.12	tree1_vdensity_1,tree1_vdensity_2,tree1_vdensity_3	Varying Point Density
No.2	No.13-No.15	tree2_16647,tree2_33327,tree2_8425	Different Number of Points
	No.16-No.18	tree2_missing_1,tree2_missing_2,tree2_missing_3	Missing Data
	No.19-No.21	tree2_noise_1,tree2_noise_2,tree2_noise_3	Noise
	No.22-No.24	tree2_vdensity_1,tree2_vdensity_2,tree2_vdensity_3	Varying Point Density
No.3	No.25-No.27	tree3_14830,tree3_29821,tree3_9037	Different Number of Points
	No.28-No.30	tree3_missing_1,tree3_missing_2,tree3_missing_3	Missing Data
	No.31-No.33	tree3_noise_1,tree3_noise_2,tree3_noise_3	Noise
	No.34-No.36	tree3_vdensity_1,tree3_vdensity_2,tree3_vdensity_3	Varying Point Density
No.4	No.37-No.39	tree4_13736,tree4_28117,tree4_8375	Different Number of Points
	No.40-No.42	tree4_missing_1,tree4_missing_2,tree4_missing_3	Missing Data
	No.43-No.45	tree4_noise_1,tree4_noise_2,tree4_noise_3	Noise
	No.46-No.48	tree4_vdensity_1,tree4_vdensity_2,tree4_vdensity_3	Varying Point Density
No.5	No.49-No.51	tree5_11433,tree5_23011,tree5_45680	Different Number of Points
	No.52-No.54	tree5_missing_1,tree5_missing_2,tree5_missing_3	Missing Data
	No.55-No.57	tree5_noise_1,tree5_noise_2,tree5_noise_3	Noise
	No.58-No.60	tree5_vdensity_1,tree5_vdensity_2,tree5_vdensity_3	Varying Point Density
No.6	No.61-No.63	tree6_11046,tree6_21792,tree6_44003	Different number of points
	No.64-No.66	tree6_missing_1,tree6_missing_2,tree6_missing_3	Missing Data
	No.67-No.69	tree6_noise_1,tree6_noise_2,tree6_noise_3	Noise
	No.70-No.72	tree6_vdensity_1,tree6_vdensity_2,tree6_vdensity_3	Varying Point Density

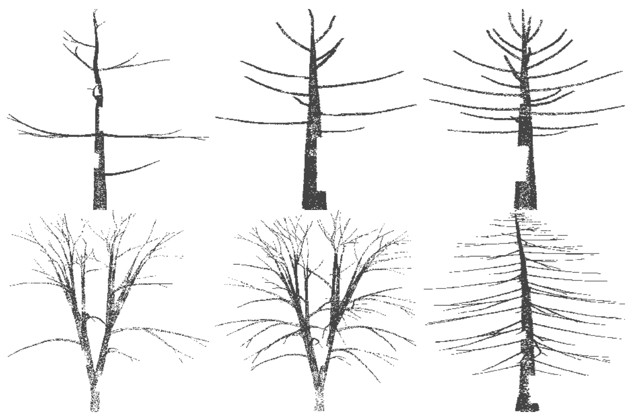
FIGURE 12. Point clouds with missing data. The point clouds in the first row are generated based on ground truth skeletons No.1-No.3. The point clouds in the second row are generated based on ground truth skeletons No.4-No.6.

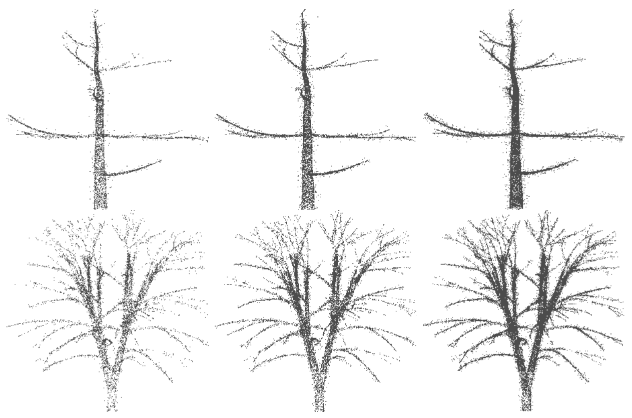
FIGURE 13. Point clouds with noise. The point clouds in the first and second rows are generated based on ground truth skeletons No.1 and No.5, respectively.

2) QUANTITATIVE EVALUATION METHODS

After constructing a synthetic tree point cloud data set, we propose an index system for quantitative evaluations. This section describes how we establish such a system by means of geometric comparison to the ground truth models and how we use the system to assess the performance of skeletonization algorithms.

We denote the ground truth model as an undirected graph $G = (V_g, E_g)$ and the corresponding skeletonization result to be evaluated as an undirected graph $S = (V_s, E_s)$. The goal of our evaluation is to assess both the centeredness and the topological correctness of S.

To measure the centeredness of S, we compute the average distance between the skeleton points in G and the skeleton points nearest to them in S. This measurement is called SPD:

$$SPD(G, S) = \frac{\sum_i dist(v_g^i, S)}{|V_g|} \tag{10}$$

where $|V_g|$ is the number of skeleton points in G and $distance(v_g^i, S)$ is the nearest distance between v_g^i and S. Cleaerly, the smaller the SPD is, the better the centeredness of the skeleton is. In practice, we calculate the nearest distance between v_g^i and edge e_s^j in S:

$$dist(v_g^i, S) = \min\{dist(v_g^i, e_s^j)|e_s^j \in E_s\} \tag{11}$$

To measure the topological correctness of S, we compute two measurements of coverage. The first measurement is the coverage of the matched bifurcation points in S and G, which

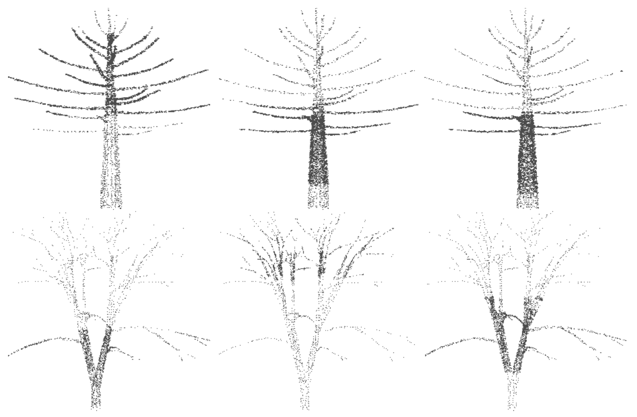

FIGURE 14. Point clouds with varying point density. The point clouds in the first and second rows are generated based on ground truth skeletons No.3 and No.4, respectively. A darker region indicates a denser distribution of data points.

we call BPC. The second measurement is the coverage of matched twig endpoints in S and G, which we call EPC.

Definition 1: Let $deg(x)$ be the degree of vertex x. Then, a bifurcation point in a skeleton is a vertex with degree greater than or equal to 3. We denote the bifurcation points in G and S as B_g and B_s, respectively.

$$B_g = \{b_g^i | b_g^i \in V_g, deg(b_g^i) \geq 3\} \tag{12}$$

$$B_s = \{b_s^i | b_s^i \in V_s, deg(b_s^i) \geq 3\} \tag{13}$$

To compute BPC, we first find matched bifurcation points in G and S. Given bifurcation points B_g and B_s, together with a distance function $dist : B_g \times B_s \rightarrow R$, which measures the Euclidean distance between b_g^i and b_s^j, we find the bijection $\phi : B_g \rightarrow B_s$ that minimizes the following cost function:

$$\sum_{b_g^i} dist(b_g^i, \phi(b_g^i)) \tag{14}$$

where $b_s^k = \phi(b_g^i)$ is the matched bifurcation point of b_g^i. This is an assignment problem, and the bijection ϕ can be found via the Hungarian algorithm [29].

Definition 2: The bifurcation point b_g^i is covered by $b_s^k = \phi(b_g^i)$ only if the distance between b_g^i and b_s^k is less than a given threshold τ.

The distance between b_g^i and $b_s^k = \phi(b_g^i)$ is represented as $d = dist(b_g^i, \phi(b_g^i))$, so the BPC can be calculated by the following equation:

$$BPC(G, S, \tau) = \frac{|\{d | d = dist(b_g^i, \phi(b_g^i)), d < \tau, b_g^i \in B_g\}|}{|B_g|} \tag{15}$$

where $|\{d | d = dist(b_g^i, \phi(b_g^i)), d < \tau, b_g^i \in B_g\}|$ is the number of covered bifurcation points, $|B_g|$ is the number of bifurcation points in G and τ is set to 0.05 by default. BPC represents the ratio of the number of matched bifurcation points in S to the total number of bifurcation points in G. The larger the BPC is, the higher the accuracy of the extracted skeleton is.

Similarly, we can calculate EPC as the following definitions.

Definition 3: A twig endpoint in a tree skeleton is a vertex with degree equal to 1. We denote the twig endpoints in G and S as T_g and T_s, respectively.

$$T_g = \{t_g^i | t_g^i \in V_g, deg(t_g^i) = 1\} \tag{16}$$

$$T_s = \{t_s^i | t_s^i \in V_s, deg(t_s^i) = 1\} \tag{17}$$

Definition 4: The twig endpoint t_g^i is covered by $t_s^k = \phi(t_g^i)$ only if the distance between t_g^i and t_s^k is less than a given threshold τ.

Then, the EPC is computed using the following equation:

$$EPC(G, S, \tau) = \frac{|\{d | d = dist(t_g^i, \phi(t_g^i)), d < \tau, t_g^i \in T_g\}|}{|T_g|} \tag{18}$$

where τ is set to 0.05 by default. As shown in the equation, EPC represents the ratio of the number of matched twig endpoints in S to the number of total matched twig endpoints in G, which is similar to the definition of BPC. Moreover, larger values of EPC indicate higher accuracy of the extracted skeleton.

3) COMPARISON OF THE RESULTS ON THE SYNTHETIC DATA SET

We first test the performance of our methods on synthetic point clouds with different numbers of points. We select point clouds No.27, No.26, No.37, and No.38 for comparison. The details of these synthetic point clouds are given in Table 1. Fig.15 shows the visual comparison of the results of four skeleton extraction algorithms: L_1, $L_1 - MST$, ROSA and our algorithm. The skeletons generated by our algorithm accurately capture the topology of the point cloud, and most of the branches in the skeleton are connected correctly. By contrast, the skeletons generated by the other three algorithms have a large number of misconnected skeleton points in the branching regions.

To further illustrate the correctness and accuracy of the skeleton generated by our algorithm, we compute the SPD, BPC, and EPC of the skeletons generated by the four skeleton extraction algorithms (as shown in Table 2-Table 10), and the number marked in bold in each row of a table represents the best value for that row. In terms of centeredness, which is measured by SPD, our algorithm performs best, followed by L_1 and L_1-MST. In terms of topological correctness, which is measured by BPC and EPC, our algorithm also has the best overall performance. L_1 and L_1-MST perform well in some cases, with BPC and EPC values greater than 0.8. ROSA performs worst in both aspects.

Next, we test our algorithm on synthetic point clouds that have large amounts of missing data. Visual comparisons are given in Fig.16. The three point clouds shown in Fig.16 are generated based on skeleton No.2. As shown in the

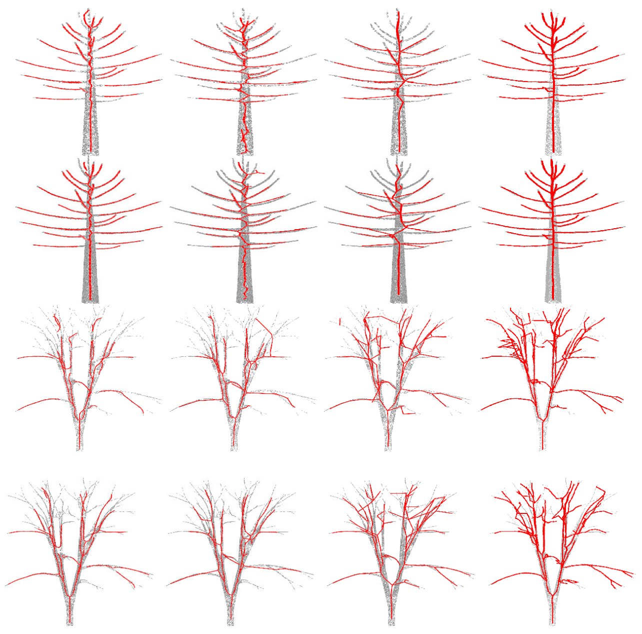

FIGURE 15. The skeletons extracted from synthetic point clouds with different numbers of points. The point clouds are indexed by No.27, No.26, No.37, and No.38 from top to bottom.

TABLE 2. SPD (different tree structure).

Name	L1	L1-MST	ROSA	Our
tree1_14045	0.047	0.054	0.043	**0.015**
tree1_28189	0.022	0.042	0.035	**0.012**
tree1_7088	0.037	0.046	0.045	**0.015**
tree2_16647	0.010	0.018	0.026	**0.005**
tree2_33327	**0.004**	0.023	0.024	0.005
tree2_8425	0.011	0.022	0.029	**0.006**
tree3_14830	0.008	0.025	0.044	**0.006**
tree3_298	0.006	0.032	0.037	**0.005**
tree3_9037	0.014	0.026	0.041	**0.006**
tree4_13736	0.121	0.126	0.059	**0.034**
tree4_28117	0.147	0.129	0.046	**0.034**
tree4_8375	0.132	0.136	0.074	**0.030**
tree5_11433	0.126	0.137	0.063	**0.019**
tree5_23011	0.138	0.128	0.056	**0.021**
tree5_45680	0.115	0.136	0.053	**0.022**
tree6_11046	0.039	0.056	0.063	**0.016**
tree6_21792	0.034	0.058	0.058	**0.015**
tree6_44003	0.036	0.065	0.052	**0.012**

blue boxes, the skeletons generated by the L_1 method and L_1-MST method deviate from the ground truth skeleton. In comparison, the skeletons generated by ROSA and our method show better centeredness. The quantitative assessment of the centeredness is shown in Table 5. Our algorithm performs best, with the highest SPD in all cases. The BPC and EPC values are also presented in Table 6 and Table 7.

We also test our algorithm on synthetic point clouds that have noise. Visual comparisons are given in Fig.17. The three other methods fail to extract skeletons from the twig area of the point cloud. Because of the existence of noise, the ROSA method fails to accurately estimate the normals of the points; hence the generated skeleton deviates considerably from the ground truth. In comparison, our method is capable of extracting a complete skeleton from the twig parts. Even in the presence of considerable noise, the skeletons generated by our method show good centeredness and topological correctness.

TABLE 3. BPC (different tree structure, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_14045	0.263	0.315	0.210	**0.736**
tree1_28189	0.315	0.421	0.157	**0.684**
tree1_7088	0.263	0.315	0.157	**0.736**
tree2_16647	0.090	**1.000**	0.454	0.909
tree2_33327	0.636	**1.000**	0.545	1.000
tree2_8425	0.000	**1.000**	0.545	1.000
tree3_14830	0.000	**1.000**	0.304	0.956
tree3_29821	0.130	**1.000**	0.347	0.956
tree3_9037	0.000	**1.000**	0.304	0.956
tree4_13736	0.000	0.153	0.046	**0.230**
tree4_28117	0.076	0.138	0.030	**0.230**
tree4_8375	0.015	0.169	0.046	**0.246**
tree5_11433	0.009	0.082	0.027	**0.339**
tree5_23011	0.018	0.073	0.082	**0.256**
tree5_45680	0.018	0.082	0.036	**0.321**
tree6_11046	0.017	**0.534**	0.034	0.310
tree6_21792	0.000	**0.551**	0.034	0.362
tree6_44003	0.051	**0.603**	0.034	0.344

TABLE 4. EPC (different tree structure, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_14045	0.409	0.045	0.000	**0.500**
tree1_28189	0.545	0.045	0.090	**0.681**
tree1_7088	0.363	0.090	0.000	**0.500**
tree2_16647	0.846	0.230	0.153	1.000
tree2_33327	**1.000**	0.076	0.230	**1.000**
tree2_8425	**0.923**	0.076	0.076	**0.923**
tree3_14830	**0.920**	0.040	0.040	0.840
tree3_29821	0.880	0.000	0.200	**0.920**
tree3_9037	**0.840**	0.120	0.080	**0.840**
tree4_13736	0.114	0.022	0.000	**0.206**
tree4_28117	0.103	0.000	0.068	**0.252**
tree4_8375	0.091	0.000	0.000	**0.321**
tree5_11433	0.078	0.045	0.026	**0.366**
tree5_23011	0.071	0.006	0.039	**0.372**
tree5_45680	0.104	0.013	0.039	**0.444**
tree6_11046	0.185	0.174	0.022	**0.421**
tree6_21792	0.224	0.123	0.033	**0.443**
tree6_44003	0.247	0.151	0.033	**0.460**

TABLE 5. SPD (missing data).

Name	L1	L1-MST	ROSA	Our
tree1_missing_1	0.030	0.041	0.035	**0.013**
tree1_missing_2	0.023	0.038	0.043	**0.013**
tree1_missing_3	0.026	0.039	0.031	**0.013**
tree2_missing_1	0.005	0.019	0.025	**0.005**
tree2_missing_2	0.005	0.018	0.027	**0.005**
tree2_missing_3	0.005	0.017	0.028	**0.005**
tree3_missing_1	0.006	0.031	0.033	**0.004**
tree3_missing_2	0.008	0.032	0.035	**0.003**
tree3_missing_3	0.006	0.029	0.038	**0.003**
tree4_missing_1	0.133	0.124	0.049	**0.018**
tree4_missing_2	0.133	0.120	0.050	**0.021**
tree4_missing_3	0.096	0.101	0.050	**0.032**
tree5_missing_1	0.139	0.138	0.053	**0.022**
tree5_missing_2	0.142	0.139	0.047	**0.021**
tree5_missing_3	0.127	0.152	0.052	**0.020**
tree6_missing_1	0.030	0.058	0.054	**0.012**
tree6_missing_2	0.036	0.057	0.064	**0.012**
tree6_missing_3	0.030	0.055	0.060	**0.011**

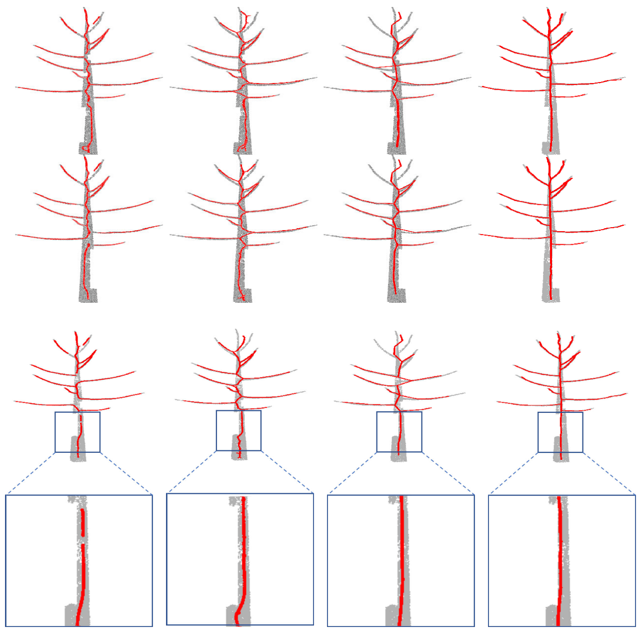
(a) L_1 (b) L_1-MST (c) ROSA (d) Ours

FIGURE 16. Skeletons extracted from point clouds with missing data. The point clouds are indexed by No.16, No.17, and No.18 from top to bottom.

TABLE 6. BPC (missing data, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_missing_1	0.210	0.315	0.210	**0.684**
tree1_missing_2	0.263	0.368	0.210	**0.684**
tree1_missing_3	0.368	0.368	0.263	**0.631**
tree2_missing_1	0.454	**1.000**	0.636	**1.000**
tree2_missing_2	0.181	**1.000**	0.454	0.909
tree2_missing_3	0.454	**1.000**	0.545	0.909
tree3_missing_1	0.304	**1.000**	0.304	0.956
tree3_missing_2	0.173	**1.000**	0.304	**1.000**
tree3_missing_3	0.391	**1.000**	0.260	**1.000**
tree4_missing_1	0.015	0.046	0.030	**0.323**
tree4_missing_2	0.076	0.153	0.015	**0.338**
tree4_missing_3	0.076	0.092	0.000	**0.369**
tree5_missing_1	0.018	0.082	0.073	**0.284**
tree5_missing_2	0.000	0.055	0.055	**0.357**
tree5_missing_3	0.045	0.064	0.027	**0.302**
tree6_missing_1	0.034	**0.586**	0.034	0.275
tree6_missing_2	0.051	**0.620**	0.051	0.344
tree6_missing_3	0.051	**0.672**	0.017	0.293

TABLE 7. EPC (missing data, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_missing_1	0.545	0.090	0.045	**0.636**
tree1_missing_2	0.500	0.045	0.045	**0.636**
tree1_missing_3	**0.590**	0.090	0.090	**0.590**
tree2_missing_1	0.923	0.153	0.230	**1.000**
tree2_missing_2	**1.000**	0.076	0.153	0.923
tree2_missing_3	**0.923**	0.153	0.153	**0.923**
tree3_missing_1	0.920	0.080	0.120	**0.960**
tree3_missing_2	0.880	0.040	0.160	**0.960**
tree3_missing_3	0.920	0.000	0.200	**0.960**
tree4_missing_1	0.080	0.011	0.022	**0.448**
tree4_missing_2	0.103	0.022	0.022	**0.471**
tree4_missing_3	0.091	0.022	0.057	**0.436**
tree5_missing_1	0.071	0.039	0.026	**0.398**
tree5_missing_2	0.058	0.032	0.032	**0.398**
tree5_missing_3	0.065	0.026	0.052	**0.418**
tree6_missing_1	0.230	0.202	0.016	**0.500**
tree6_missing_2	0.213	0.179	0.022	**0.533**
tree6_missing_3	0.241	0.230	0.011	**0.511**

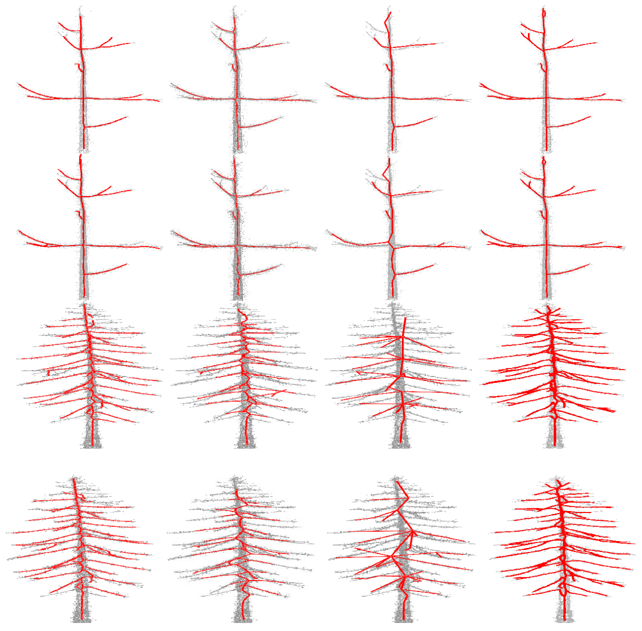
(a) L_1 (b) L_1-MST (c) ROSA (d) Ours

FIGURE 17. Skeletons extracted from point clouds with noise. The point clouds are indexed by No.8, No.9, No.68, and No.69 from top to bottom.

A related quantitative assessment is shown in Table 8 - Table 10. These three tables show the SPD, BPC, and EPC of the skeletons extracted from tree point clouds with noise, respectively. Our algorithm still outperforms the other three in terms of both centeredness and topological correctness, in most cases.

Finally, we test the performance of our algorithm on synthetic point clouds that have varying point density. Visual comparisons are given in Fig.18, and a quantitative assessment is presented in Table 11 - Table 13. Again, our algorithm produces the best results in most cases.

For a more intuitive comparison, we present three line graphs to illustrate the quantitative evaluation results of the four skeleton extraction algorithms on 72 synthetic tree point clouds in Fig.19 - Fig.21. The x-axis of these line graphs indicates the index of the synthetic point clouds, and the y-axis indicates the value of SPD, BPC or EPC. The skeletons generated by our method have a smaller SPD, which indicates better centeredness. These skeletons also have larger BPC and EPC, which indicate a topological structure that is more similar to the ground truth skeletons. In other words, our method outperforms three other skeletonization algorithms on synthetic point clouds.

4) RESULTS ANALYSIS

In the above experiments, the SPD values of our method are the best for most point clouds. Therefore, our method achieves the best centeredness because, in most situations,

TABLE 8. SPD (noise).

Name	L1	L1-MST	ROSA	Our
tree1_noise_1	0.038	0.041	0.048	**0.020**
tree1_noise_2	0.036	0.045	0.052	**0.014**
tree1_noise_3	0.025	0.035	0.059	**0.012**
tree2_noise_1	0.010	0.016	0.035	**0.005**
tree2_noise_2	0.007	0.017	0.040	**0.005**
tree2_noise_3	**0.004**	0.026	0.021	0.004
tree3_noise_1	0.009	0.024	0.043	**0.006**
tree3_noise_2	0.010	0.030	0.037	**0.006**
tree3_noise_3	0.007	0.024	0.032	**0.006**
tree4_noise_1	0.122	0.117	0.062	**0.041**
tree4_noise_2	0.163	0.124	0.048	**0.035**
tree4_noise_3	0.123	0.121	0.049	**0.023**
tree5_noise_1	0.132	0.127	0.055	**0.027**
tree5_noise_2	0.111	0.129	0.048	**0.024**
tree5_noise_3	0.141	0.128	0.057	**0.018**
tree6_noise_1	0.033	0.055	0.055	**0.016**
tree6_noise_2	0.033	0.057	0.088	**0.016**
tree6_noise_3	0.035	0.069	0.069	**0.016**

TABLE 9. BPC (noise, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_noise_1	0.263	0.263	0.157	**0.526**
tree1_noise_2	0.210	0.315	0.210	**0.631**
tree1_noise_3	0.368	0.315	0.157	**0.578**
tree2_noise_1	0.090	**1.000**	0.454	**1.000**
tree2_noise_2	0.272	**1.000**	0.545	**1.000**
tree2_noise_3	0.727	**1.000**	0.545	**1.000**
tree3_noise_1	0.000	**1.000**	0.217	0.956
tree3_noise_2	0.173	**0.956**	0.347	**0.956**
tree3_noise_3	0.478	**1.000**	0.347	0.956
tree4_noise_1	0.046	0.123	0.046	**0.169**
tree4_noise_2	0.092	0.092	0.030	**0.246**
tree4_noise_3	0.046	0.123	0.046	**0.338**
tree5_noise_1	0.000	0.055	0.091	**0.275**
tree5_noise_2	0.027	0.018	0.064	**0.229**
tree5_noise_3	0.036	0.064	0.027	**0.256**
tree6_noise_1	0.000	**0.655**	0.051	0.293
tree6_noise_2	0.017	**0.620**	0.017	0.327
tree6_noise_3	0.000	**0.586**	0.068	0.310

TABLE 10. EPC (noise, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_noise_1	0.454	0.045	0.000	**0.545**
tree1_noise_2	0.409	0.045	0.045	**0.590**
tree1_noise_3	0.590	0.090	0.045	**0.727**
tree2_noise_1	0.923	0.384	0.000	**1.000**
tree2_noise_2	0.846	0.076	0.000	**0.923**
tree2_noise_3	0.923	0.000	0.230	**1.000**
tree3_noise_1	0.880	0.040	0.040	**0.920**
tree3_noise_2	0.800	0.040	0.080	**0.960**
tree3_noise_3	0.840	0.000	0.160	**0.920**
tree4_noise_1	0.091	0.022	0.000	**0.137**
tree4_noise_2	0.091	0.000	0.057	**0.206**
tree4_noise_3	0.080	0.011	0.068	**0.425**
tree5_noise_1	0.104	0.026	0.013	**0.267**
tree5_noise_2	0.078	0.039	0.065	**0.326**
tree5_noise_3	0.084	0.026	0.019	**0.385**
tree6_noise_1	0.224	0.162	0.078	**0.410**
tree6_noise_2	0.247	0.185	0.022	**0.404**
tree6_noise_3	0.247	0.179	0.078	**0.410**

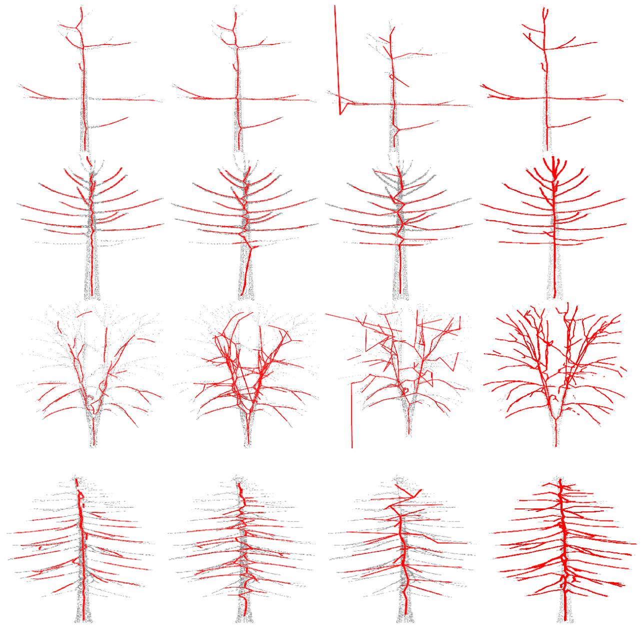

FIGURE 18. Skeletons extracted from point clouds with varying point density. The point clouds are indexed by No.12, No.34, No.60, and No.71 from top to bottom.

TABLE 11. SPD (varying point density).

Name	L1	L1-MST	ROSA	Our
tree1_vdensity_1	0.075	0.042	0.051	**0.009**
tree1_vdensity_2	0.065	0.050	0.049	**0.005**
tree1_vdensity_3	0.045	0.036	0.047	**0.011**
tree2_vdensity_1	0.033	0.041	0.039	**0.006**
tree2_vdensity_2	0.037	0.049	0.039	**0.005**
tree2_vdensity_3	0.014	0.016	0.034	**0.005**
tree3_vdensity_1	0.038	0.036	0.036	**0.006**
tree3_vdensity_2	0.075	0.067	0.040	**0.005**
tree3_vdensity_3	0.083	0.077	0.033	**0.005**
tree4_vdensity_1	0.160	0.175	0.073	**0.025**
tree4_vdensity_2	0.103	0.097	0.057	**0.017**
tree4_vdensity_3	0.148	0.159	0.072	**0.020**
tree5_vdensity_1	0.120	0.117	0.070	**0.016**
tree5_vdensity_2	0.170	0.156	0.064	**0.025**
tree5_vdensity_3	0.151	0.124	0.077	**0.023**
tree6_vdensity_1	0.056	0.068	0.082	**0.014**
tree6_vdensity_2	0.058	0.058	0.068	**0.014**
tree6_vdensity_3	0.059	0.052	0.070	**0.013**

the tree branches are generally cylindrical. Thus, we can employ CPC to optimize the positions of the skeleton points. The experiments show that even in the presence of noise or missing data (Table 5, Table 8), CPC can optimize the skeleton points to the local center. Since the other methods do not use any shape prior to recenter the skeleton points, when the point cloud is incomplete, the extracted skeleton points may deviate from the local center.

BPC and EPC are used to measure the topological correctness. As shown in the above experiments, our method also produces very good results for these values in most cases. For simple tree structures (e.g., No.1-No.3), all methods achieve good topological correctness. However, for complex tree structures (e.g., No.61-63), only L_1-MST and our method have good BPC and EPC values because L_1-MST and our method are specifically designed for tree point clouds.

TABLE 12. BPC (varying point density, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_vdensity_1	0.052	0.263	0.210	**0.684**
tree1_vdensity_2	0.105	0.315	0.263	**0.789**
tree1_vdensity_3	0.263	0.263	0.210	**0.684**
tree2_vdensity_1	0.000	0.818	0.545	**1.000**
tree2_vdensity_2	0.000	1.000	0.454	**1.000**
tree2_vdensity_3	0.000	**1.000**	0.636	0.909
tree3_vdensity_1	0.000	0.913	0.391	**0.913**
tree3_vdensity_2	0.000	0.695	0.391	**0.913**
tree3_vdensity_3	0.000	0.391	0.434	**0.913**
tree4_vdensity_1	0.000	0.061	0.061	**0.246**
tree4_vdensity_2	0.000	0.092	0.046	**0.400**
tree4_vdensity_3	0.000	0.000	0.061	**0.261**
tree5_vdensity_1	0.018	0.064	0.073	**0.385**
tree5_vdensity_2	0.009	0.073	0.100	**0.284**
tree5_vdensity_3	0.000	0.100	0.045	**0.211**
tree6_vdensity_1	0.017	**0.534**	0.034	0.258
tree6_vdensity_2	0.051	**0.500**	0.000	0.275
tree6_vdensity_3	0.017	**0.396**	0.000	0.362

TABLE 13. EPC (varying point density, $\tau = 0.05$).

Name	L1	L1-MST	ROSA	Our
tree1_vdensity_1	0.000	0.000	0.000	**0.636**
tree1_vdensity_2	0.136	0.000	0.000	**0.818**
tree1_vdensity_3	0.181	0.136	0.000	**0.681**
tree2_vdensity_1	0.615	0.000	0.000	**0.923**
tree2_vdensity_2	0.538	0.153	0.000	**0.923**
tree2_vdensity_3	0.769	0.384	0.076	**0.923**
tree3_vdensity_1	0.600	0.080	0.160	**0.960**
tree3_vdensity_2	0.400	0.120	0.080	**0.960**
tree3_vdensity_3	0.400	0.040	0.040	**0.960**
tree4_vdensity_1	0.034	0.011	0.011	**0.379**
tree4_vdensity_2	0.068	0.000	0.022	**0.459**
tree4_vdensity_3	0.045	0.011	0.034	**0.425**
tree5_vdensity_1	0.065	0.026	0.026	**0.418**
tree5_vdensity_2	0.058	0.052	0.013	**0.320**
tree5_vdensity_3	0.039	0.032	0.013	**0.313**
tree6_vdensity_1	0.123	0.078	0.011	**0.421**
tree6_vdensity_2	0.168	0.106	0.011	**0.426**
tree6_vdensity_3	0.123	0.106	0.011	**0.426**

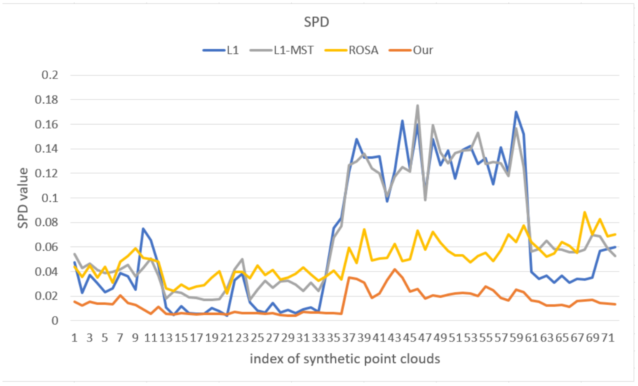

FIGURE 19. SPD of all extracted skeletons.

L_1-MST is based on the MST algorithm, and our method is an improvement of the level set method. Both the MST algorithm and the level set method can guarantee that the extracted skeleton has a tree structure. However, L_1 and ROSA do not ensure a tree structure. Therefore, even for dense point clouds

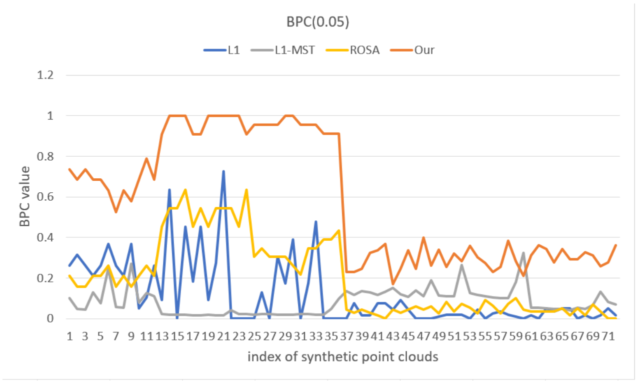

FIGURE 20. BPC of all extracted skeletons ($\tau = 0.05$).

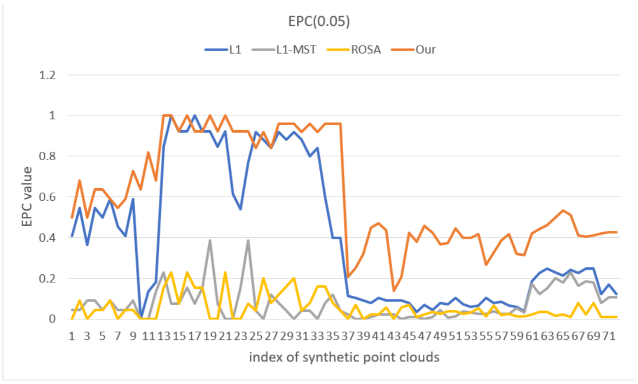

FIGURE 21. EPC of all extracted skeletons ($\tau = 0.05$).

without noise and occlusion, L_1 and ROSA may fail to extract a topologically correct skeleton.

For point clouds with noise, missing data and varying point density, our method usually has better BPC and EPC values than those of L_1-MST. One reason is that our method uses octree to downsample the point cloud, which improves the robustness to noise and varying point density. Most importantly, our method uses prior knowledge about the radii of the tree branches to refine the improper positioning of joint points. This approach enables our method to extract a topologically correct skeleton in the case of missing data and complex branch structures.

B. EXPERIMENTS ON THE REAL DATA SET

1) REAL TREE POINT CLOUD DATA SET

We also conduct experiments on a real tree point cloud data set. This section introduces how the data set is acquired.

In this paper, we employ multiview stereo (MVS) techniques to obtain point clouds of tree instances. Although MVS is widely used for image-based 3D reconstruction, its potential for point cloud-based tree reconstruction is underestimated. Different from TLS, MVS is capable of acquiring point cloud data by means of an inexpensive digital camera. To collect raw point data from a tree, we attach a digital camera to an unmanned aerial vehicle (UAV) and

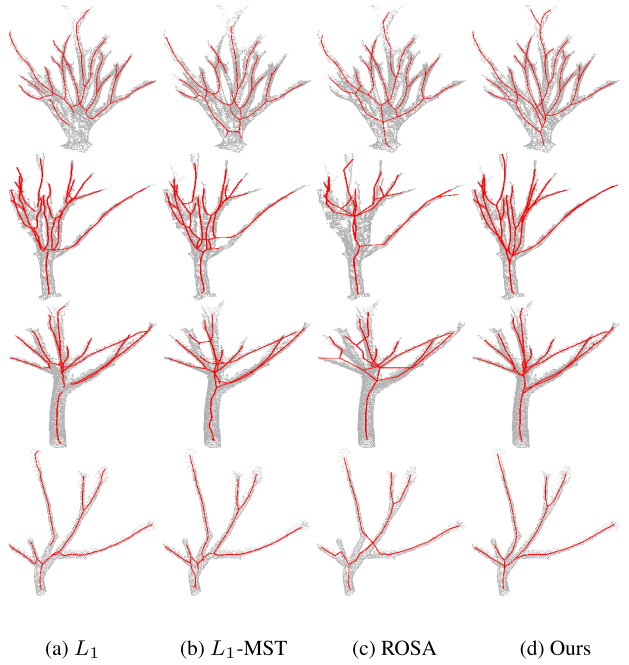

FIGURE 22. **Front view of the skeletons extracted from real tree point clouds No.4,No.5, No.7 and No.9.**

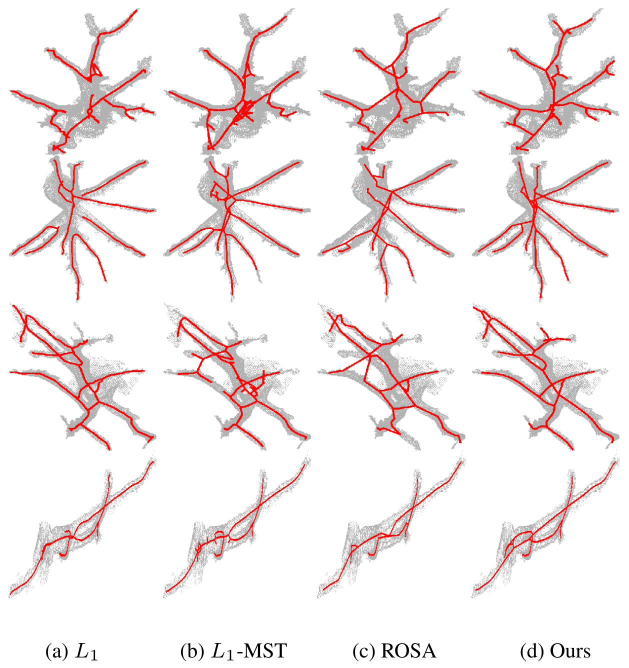

FIGURE 23. **Top view of skeletons extracted real tree point clouds No.6-No.9.**

let it take photographs around the tree. Outdoor panoramic scans of trees can easily be performed with the help of a UAV. Compared to that obtained by TLS, the point clouds obtained by MVS contain few occlusions and are dense. Such dense point clouds strongly facilitate the extraction of topology-preserving and well-centered tree skeletons.

To obtain point clouds, we took 20 to 30 photographs of each tree from different perspectives. The point cloud is generated using pen-source MVE software [30]. In our experiment, we use one Osmanthus tree, three Ficus virens tree, and five Ficus concinna trees as test cases.

2) VISUAL ASSESSMENT ON REAL TREE POINT CLOUD

Because ground truth skeletons of point clouds of real trees are difficult to obtain, we use visual comparisons to assess our proposed skeleton extraction algorithm. For each point cloud of a real tree, we provide the front view and top view of the generated skeletons to conduct visual comparisons. The point clouds of real trees and the corresponding skeletons are shown in Fig.22 - Fig.23.

From both the front view and top view, we can see that although the MVS point clouds have considerable noise and outliers, our method extracts clean and topology-preserving skeletons. The other three methods show two vulnerabilities in these cases. First, they fail to extract skeletons from the twig parts of the point clouds. Second, the skeletons generated by these methods are not well-centered in the branching regions.

VI. CONCLUSION

In this paper, we propose a novel skeletonization method for dense point clouds of trees. The proposed approach has the following characteristics: (1) accelerated skeletonization and enhanced robustness to noise using octree; (2) initial skeleton extraction using the level set method; (3) well-centered skeleton points calculated using CPC; and (4) improper position refinement of joint points based on prior knowledge about the radii of tree branches. Our method clearly outperforms traditional ones in terms of topological correctness and centeredness when the point cloud used to generate the tree skeleton is noisy and partly occluded. Extensive case studies based on real-world trees suggest that our algorithm can extract skeletons that accurately fit the tree structure, and the positions of the skeleton points are optimally centered. The extracted skeleton can not only be used to reconstruct structural models of trees but also to estimate forestry parameters. We plan to consider these topics in future work.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest in regard to this work.

REFERENCES

[1] J. Hackenberg, H. Spiecker, K. Calders, M. Disney, and P. Raumonen, "SimpleTree—An efficient open source tool to build tree models from TLS clouds," *Forests*, vol. 6, no. 11, pp. 4245–4294, 2015.

[2] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. New York, NY, USA: Springer, 2012.

[3] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, 2007, vol. 26, no. 3, p. 87.

[4] H. Xu, N. Gossett, and B. Chen, "Knowledge and heuristic-based modeling of laser-scanned trees," *ACM Trans. Graph.*, vol. 26, no. 4, p. 19, Oct. 2007.

[5] J.-F. Côté, J.-L. Widlowski, R. A. Fournier, and M. M. Verstraete, ''The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial LiDAR,'' *Remote Sens. Environ.*, vol. 113, no. 5, pp. 1067–1081, May 2009.
[6] R. Li, G. Bu, and P. Wang, ''An automatic tree skeleton extracting method based on point cloud of terrestrial laser scanner,'' *Int. J. Opt.*, vol. 2017, pp. 1–11, 2017.
[7] J. Bloomenthal, ''Modeling the mighty maple,'' *ACM SIGGRAPH Comput. Graph.*, vol. 19, pp. 305–311, Jul. 1985.
[8] A. Bucksch, R. Lindenbergh, and M. Menenti, ''SkelTre: Robust skeleton extraction from imperfect point clouds,'' *Vis. Comput.*, vol. 26, no. 10, pp. 1283–1300, 2010.
[9] A. Tagliasacchi, H. Zhang, and D. Cohenor, ''Curve skeleton extraction from incomplete point cloud,'' in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, 2009, vol. 28, no. 3, p. 71.
[10] H. Huang, S. Wu, D. Cohenor, M. Gong, H. Zhang, G. Li, and B. Chen, ''L_1-medial skeleton of point cloud,'' *ACM Trans. Graph.*, vol. 32, no. 4, p. 65, 2013.
[11] J. Mei, L. Zhang, S. Wu, Z. Wang, and L. Zhang, ''3D tree modeling from incomplete point clouds via optimization and L1-MST,'' *Int. J. Geograph. Inf. Sci.*, vol. 31, no. 5, pp. 999–1021, May 2017.
[12] H. H. Chen and T. S. Huang, ''A survey of construction and manipulation of octrees,'' *Comput. Vis., Graph., Image Process.*, vol. 43, no. 3, pp. 409–431, Sep. 1988.
[13] A. Verroust and F. Lazarus, ''Extracting skeletal curves from 3D scattered data,'' *Vis. Comput.*, vol. 16, no. 1, pp. 15–25, Feb. 2000.
[14] I. Demir, C. Hahn, K. Leonard, G. Morin, D. Rahbani, A. Panotopoulou, A. Fondevilla, E. Balashova, B. Durix, and A. Kortylewski, ''SkelNetOn 2019: Dataset and challenge on deep learning for geometric shape understanding,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2019.
[15] L. Yang, D. Oyen, and B. Wohlberg, ''A novel algorithm for skeleton extraction from images using topological graph analysis,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2019.
[16] C. Liu, D. Luo, Y. Zhang, W. Ke, F. Wan, and Q. Ye, ''Parametric skeleton generation via Gaussian mixture models,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2019.
[17] R. Atienza, ''Pyramid U-network for skeleton extraction from shape points,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR) Workshops*, Jun. 2019.
[18] C. Song, Z. Pang, X. Jing, and C. Xiao, ''Distance field guided L_1-median skeleton extraction,'' *Vis. Comput.*, vol. 34, no. 2, pp. 243–255, 2018.
[19] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee, ''Skeleton extraction by mesh contraction,'' *ACM Trans. Graph.*, vol. 27, p. 44, Aug. 2008.
[20] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su, ''Point cloud skeletons via Laplacian based contraction,'' in *Proc. Shape Modeling Int. (SMI)*, 2010, pp. 187–197.
[21] H. Qin, J. Han, N. Li, H. Huang, and B. Chen, ''Mass-driven topology-aware curve skeleton extraction from incomplete point clouds,'' *IEEE Trans. Visual. Comput. Graph.*, to be published.
[22] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. Elsana, ''Automatic reconstruction of tree skeletal structures from point clouds,'' in *Proc. Int. Conf. Comput. Graph. Interact. Techn.*, 2010, vol. 29, no. 6, p. 151.
[23] D. Zhang, N. Xie, S. Liang, and J. Jia, ''3D tree skeletonization from multiple images based on PyrLK optical flow,'' *Pattern Recognit. Lett.*, vol. 76, pp. 49–58, Jun. 2016.
[24] N. D. Cornea, D. Silver, and P. Min, ''Curve-skeleton properties, applications, and algorithms,'' *IEEE Trans. Visual. Comput. Graph.*, vol. 13, no. 3, pp. 530–548, May 2007.
[25] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*. Boca Raton, FL, USA: CRC Press, 2013.
[26] L. da Vinci, *The Notebooks of Leonardo Da Vinci*. New York, NY, USA: Addison-Wesley, 1938.
[27] R. Minamino and M. Tateno, ''Tree branching: Leonardo da Vinci's rule versus biomechanical models,'' *PLOS ONE*, vol. 9, no. 4, p. e93535, 2014.
[28] I. Schoenberg, ''Cardinal interpolation and spline functions,'' *J. Approx. Theory*, vol. 2, no. 2, pp. 167–206, 1969.
[29] H. W. Kuhn, ''The hungarian method for the assignment problem,'' *Naval Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, 1955.
[30] S. Fuhrmann, F. Langguth, and M. Goesele, ''MVE—A multi-view reconstruction environment,'' in *Proc. Eurograph. Workshop Graph. Cultural Heritage*, R. Klein and P. Santos, Eds., 2014.

LIXIAN FU received the B.S. degree in information security from Chongqing University, Chongqing, China, in 2017, where he is currently pursuing the M.S. degree in computer science. His research interests include point cloud processing, computer graphics, and computer simulation.

JI LIU received the master's and Ph.D. degrees in computer software and theory from Chongqing University, in 2005 and 2009, respectively. He started working on tree modeling during his Ph.D. thesis. He is currently on the Faculty of the College of Computer Science, Chongqing University, China. His research interests are in computer graphics, 3D reconstruction, and tree modeling.

JIANLING ZHOU received the B.S. degree in computer science from Chongqing University, Chongqing, China, in 2017, where he is currently pursuing the M.S. degree. His research interests include point cloud processing, computer graphics, and computer vision.

MIN ZHANG received the B.S. and Ph.D. degrees in computer science from Chongqing University, China. Since 2003, she has been a Lecturer with the School of Computer Science, Chongqing University. She has authored or coauthored more than 20 research publications. Her research interests are in pattern recognition, machine learning algorithms, and applied intelligence.

YAN LIN received the B.S. degree in network engineering from Chongqing University, Chongqing, China, in 2018, where she is currently pursuing the M.S. degree in computer science. Her research interests include point clouds and image processing.

• • •