A Structure-Aware Global Optimization Method for Reconstructing 3-D Tree Models From Terrestrial Laser Scanning Data

Zhen Wang, Liqiang Zhang, Tian Fang, *Member, IEEE*, P. Takis Mathiopoulos, *Senior Member, IEEE*, Huamin Qu, *Member, IEEE*, Dong Chen, and Yuebin Wang

***Abstract*—A 3-D tree structure plays an important role in many scientific fields, including forestry and agriculture. For example, terrestrial laser scanning (TLS) can efficiently capture high-precision 3-D spatial arrangements and structure of trees as a point cloud. In the past, several methods to reconstruct 3-D trees from the TLS point cloud were proposed. However, in general, they fail to process incomplete TLS data. To address such incomplete TLS data sets, a new method that is based on a structure-aware global optimization approach (SAGO) is proposed. The SAGO first obtains the approximate tree skeleton from a distance minimum spanning tree (DMst) and then defines the stretching directions of the branches on the tree skeleton. Based on these stretching directions, the SAGO recovers missing data in the incomplete TLS point cloud. The DMst is applied again to obtain the refined tree skeleton from the optimized data, and the tree skeleton is smoothed by employing a Laplacian function. To reconstruct 3-D tree models, the radius of each branch section is estimated, and leaves are added to form the crown geometry. The developed methodology has been extensively evaluated by employing a dozen TLS point clouds of various types of trees. Both qualitative and quantitative performance evaluation results have indicated that the SAGO is capable of effectively reconstructing 3-D tree models from grossly incomplete TLS point clouds with significant amounts of missing data.**

***Index Terms*—Missing data, optimization, terrestrial laser scanning (TLS), tree skeleton, 3-D tree models.**

I. Introduction

LIGHT detection and ranging techniques provide dense and accurate 3-D coordinates, as well as multiecho pulses and intensities with high horizontal and perpendicular accuracy [1]. In general terms, these techniques can be classified into airborne laser scanning (ALS) and terrestrial laser scanning (TLS) techniques. Both ALS and TLS can capture a canopy structure very effectively [2]–[5]. In practice, ALS is preferred in obtaining a large-scale canopy structure. This technology has been employed in various ecological applications, including change detection studies and forest inventories [6]. However, TLS-based techniques are suitable in reconstructing stems and branching structure (e.g., [7] and [8]) or individual trees [9]–[11]. Except for the diameter and tapering of the stem, an accurate detection of single branches is important to assess the branch dimensions [12]. Reconstruction of real trees from TLS data allows automatic estimation of the stem volume of forests, which has various uses. Detailed information regarding tree structure is a key parameter for making ecological assessments in forestry [13]. TLS-based 3-D tree modeling is also employed in agriculture to estimate various parameters, such as plant height, volume, leaf area index, and leaf area density [14].

Although TLS provides raw 3-D views of the distribution of canopy elements, there are still difficulties in accurately modeling trees because of the large variety of trees and their complex geometries [15]. The main reasons for these difficulties are as follows: 1) The tree branches or branch parts are often incompletely captured due to the limited scanning resolution or occlusions by branches or other objects between the scanner and the branch; 2) the point densities are typically not uniform in the raw data; 3) the raw data are very noisy because of the nonstationary interference created by the tree leaves; and 4) the data acquisition process for large-scale multistation scanning is very time-consuming, and the alignment of these data can accumulate additional noise. Although using a single scan avoids such problems, the disadvantage of such an approach is that a tree can be scanned from only one viewpoint. Thus, in such a case, some of the 3-D data of the stem and branches will be missing due to occlusions. Despite these difficulties and because of its importance in recent years, 3-D tree modeling from a TLS point cloud has gained considerable scientific attention (e.g., [16]–[30]). Previous 3-D tree-modeling methods can be classified roughly into crown-based and skeleton-based approaches.

On the one hand, in crown-based methods (e.g., [16] and [17]), convex hulls are used to model crown shapes, and branches are generated by the L-system or software such as

Manuscript received August 7, 2012; revised February 20, 2013, July 1, 2013, September 4, 2013, and November 1, 2013; accepted November 6, 2013. This work was supported by the National Natural Science Foundation of China under Grant 41371324.

Z. Wang, L. Zhang, and Y. Wang are with the State Key Laboratory of Remote Sensing Science, Beijing Normal University, Beijing 100875, China (e-mail: comige@gmail.com; zhanglq@bnu.edu.cn; xxgcdxwyb@163.com).

T. Fang and H. Qu are with Hong Kong University of Science and Technology, Kowloon, Hong Kong (e-mail: tianft@cse.ust.hk; huamin@cse.ust.hk).

P. T. Mathiopoulos is with the Institute for Astronomy Astrophysics Space Applications and Remote Sensing, National Observatory of Athens Metaxa and V. Pavlou, 14536 Athens, Greece, and also with the Department of Informatics and Telecommunications, National and Kapodestrian University of Athens, 15784 Athens, Greece (e-mail: mathio@hol.gr).

D. Chen is with Nanjing Forestry University, Nanjing 210037, China (e-mail: chendong@njfu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2013.2291815

PlantGL [18]. In practice, the calculated branches do not match the branches of the real trees very well. Crown-based methods are usually applied to reconstruct flourishing trees in which the detailed branch information cannot be clearly discriminated.

On the other hand, skeleton-based approaches are very effective in describing the spatial distribution of tree branches, which can be thought of as a skeletal structure. Vosselman [19] and Gorte [20] rasterized the point cloud and used Dijkstra's algorithm (DA) [21] to obtain a skeleton from their 3-D rasters. Bucksch [22] and Lindenbergh [23] segmented points into octree cells and, by connecting local extractions in adjacent cells, produced a curved skeleton. Their method can process a large number of point clouds in linear time complexity and can produce quite stable reconstructions which are not greatly influenced by small changes in the object boundaries. However, the obtained reconstructions are sensitive to varying point densities and undersampling, and lack fine details. Because their methods are based on the raster data transformed from point clouds to extract tree skeletons, the accuracy of the obtained tree skeletal structure depends on the partitioning resolution of the raster. There are some skeleton-based methods that avoid this raster resolution problem by obtaining the tree skeletal structure directly from a point cloud. For example, Pfeifer *et al.* [24] utilized a volumetric method for extracting skeletons from points and built meshes for stems by using cylinders. Their method aims at identifying the prominent structure of trees rather than at reconstructing finely detailed models. Su *et al.* [25] used a constrained Laplacian smoothing [26] to extract skeletons. This method is not computationally efficient and cannot address complex tree models. In another approach, Côté *et al.* [27] first integrated DA [21] and level sets [28] to obtain skeletons and then used the colonization algorithm [29] and a light transmission model to reconstruct the fine branch structure superimposed on the skeletons. This type of tree modeling is relatively insensitive to wind- and occlusion-induced artifacts in the TLS point cloud. However, the technique requires labor-intensive parameterization based only on the visual appearance of the tree and is thus not suitable for an automatic interpretation of TLS data. To overcome the weaknesses of [27], Côté *et al.* [10] proposed an automatic procedure to assess the accuracy of the generated tree architecture. This architectural-type model was designed to provide a practical method to synthesize and quantify the spatial distribution of tree components from the TLS point cloud, thus resulting in a clearer description of the 3-D tree architecture. Although architectural tree models reconstruct the fine branch structure to fit the biological parameters of real trees, branch skeletons cannot be accurately reconstructed by the DA and level sets alone. In [10] and [27], backscattered intensity was used to distinguish branches and leaves, but the backscattered intensity in the point clouds of our scanned trees is not an obvious difference.

The input point cloud in the previously mentioned skeleton-based methods was obtained from multistation scans. For large-scale scenes, single or mobile scans (both of which scan trees in only one direction) are more efficient than multistation scans. Because laser beams are reflected from the nearest branch surfaces along the direction of travel, the opposite side of the branches and other branches behind them are often occluded. Most trees that are scanned during winter or spring contain few leaves, making more parts of the trees visible, although occlusions are still severe in some types of trees. Using such incomplete structure information to create accurate 3-D models of trees is still a challenging research topic.

To model trees from TLS point clouds produced by a single scan, Xu *et al.* [30] proposed a modeling approach that uses general knowledge on tree structure to generate the tree meshes. In this approach, the tree skeleton is first extracted, and small twigs and leaves are then synthetically added to produce a plausible support structure for the tree crown. Because this method depends on prior knowledge about tree structure, it is less effective at creating accurate tree models when addressing a large variety of tree structure. This method therefore fails to support the tree crown due to having a sparse point cloud distribution. Livny *et al.* [31] applied a branch-structure graph, which is defined as a spatially embedded and connected directed acyclic graph, to represent tree skeletons. A series of global optimizations [32] was adopted to reconstruct the major skeletal branches; this approach avoids manually setting the parameters for different types of trees. In this way, after edge removal and fine geometry synthesis, a tree model was reconstructed. This method is robust to noise, and the reconstructed tree has finer details resolved than the approach proposed in [30]. However, it is not sufficiently flexible to describe branch stretching directions and cannot handle point clouds that have large regions of missing data.

The aforementioned literature review shows that, although previously known skeleton-based methods for tree modeling can effectively describe approximate skeleton structure, they cannot adequately extract tree skeletons from grossly incomplete point clouds. In an effort to fill this gap, a distance minimum spanning tree (DMst) and a robust structure-aware global optimization method, termed SAGO, are proposed to extract tree skeletons from imperfect TLS data. The main contributions of this paper can be summarized as follows.

1) In contrast to most tree-skeleton extraction approaches (e.g., [27], [30], and [31]), which are not well adaptive to the point cloud with varying point density, a DMst-based novel algorithm that is adaptive to the varying point density is proposed to extract tree skeletons from the TLS point cloud. The DMst integrates the advantages of the DA (being robust to noise and varying cloud point densities) and the minimum spanning tree (MST; preserving the local spatial structure of the point cloud).
2) Previous methods do not recover the regions of missing data before they reconstruct the 3-D tree models. Without the recovery of the data in occluded regions, the extraction of the tree skeleton is highly likely to fail. To resolve this problem, the branch stretching direction of each point in the TLS data is computed, which makes the proposed optimization method structure-aware. Next, SAGO is developed to place the points into appropriate regions of missing data in such a way that missing data are recovered. Various experimental evaluation results have shown that the SAGO can handle the TLS point cloud even with large regions of missing data.

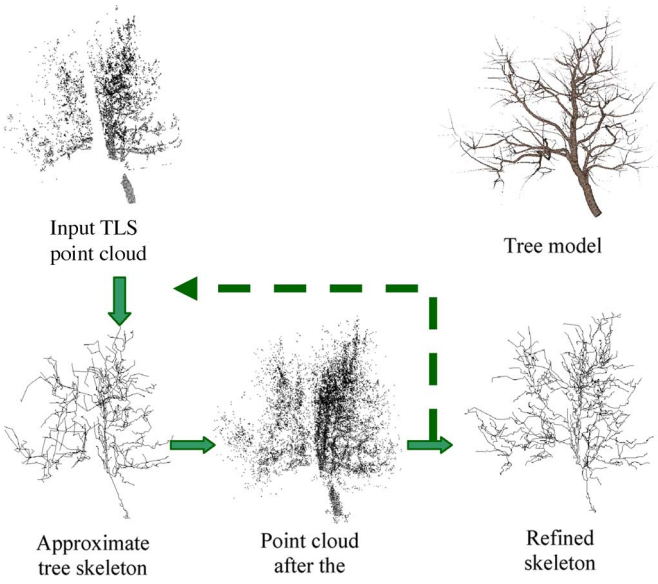


Fig. 1. Methodology for reconstructing 3-D tree models from the TLS point cloud.

II. Methodology

This section presents an overview of the methodology that is proposed for restructuring 3-D tree models from the TLS point cloud. Fig. 1 illustrates this procedure. First, preprocessing is conducted to generate a connected undirected graph from the TLS point cloud. Second, the DMst is applied to extract the approximate tree skeletons, and the branch stretching directions are computed from these skeletons. Third, the SAGO is introduced to recover regions of missing data. Through an iterative process using these three steps, the true skeletons are refined, and in this way, the geometry of the trees is finally created, and the 3-D tree models are generated. The functionality of each step will be briefly explained next.

1) Before tree models are reconstructed, the points on trees are manually recognized by users from the input point cloud. To conveniently extract tree skeletons during preprocessing, these tree point clouds are further organized into a connected undirected graph.
2) A DMst is developed to obtain the approximate tree skeleton from the previously constructed graph. The procedure of constructing the DMst is introduced. Using the approximate tree skeleton, the branch stretching directions are calculated for implementing the SAGO.
3) The SAGO that recovers missing data is presented. This process allows more accurate extractions of tree skeletons from the incomplete point cloud. To obtain the refined tree skeletons, the optimization and tree skeleton construction processes are performed alternately.
4) To generate the 3-D mesh models for trees, the reconstructed skeletons are smoothed, and then, the radii of the branches are computed in such a way that the skeletons are inflated to the 3-D tree models. Leaves are also included if the scanned trees contain them.

A. Preprocessing

Before 3-D tree models are reconstructed, the tree point cloud is identified by the user from the full TLS point cloud, which contains trees as well as other objects, including buildings, roads, and pedestrians. If the point cloud contains more than one tree, the number of trees and their base locations also need to be specified. Next, we construct an undirected graph $G(\mathbf{V}, \mathbf{E})$ to determine the connectivity of points among the point cloud, where $\mathbf{V}$ is a set of vertices and $\mathbf{E}$ is a set of edges. Each point corresponds to a vertex in G. For each point, we identify its k nearest neighbors and connect the corresponding vertices in G with edges that are associated with the Euclidean distance between the two vertices. Due to limited scanning resolution and occlusions from leaves and other branches, G is usually not a connected graph [33], [34]. To obtain a connected graph, the two nearest connected components of G are combined into one by an edge that links their two nearest points. The process ends when G becomes a connected undirected graph.

B. DMst Construction

This section presents a new algorithm for the determination of tree skeletons that is based on the graph G to adapt to the point cloud with noise and varying point densities. The DA has been typically applied in the past to construct tree skeletons (e.g., see [10], [20], and [30]). It is a graph-based search algorithm which solves the single-source shortest path problem. It does so by producing a tree that minimizes the sum of the edge weight from each vertex to the single-source vertex, which is the root node. Because the directions of the shortest paths computed by the DA visually coincide with the branch growing directions, this approach gives a good approximation of tree skeletons even if the point cloud is incomplete or noisy. Unfortunately, because the DA cannot describe the spatial distribution of points in local regions, it often combines the level set to extract the skeleton, which causes the reconstructed trees to lack important details. To describe the spatial distribution of points in local regions, the square of the Euclidean distance between two points is taken as the weight of the edge that connects the two points, and then, the DA is used to extract the tree skeleton in [31]. Unfortunately, the fixed weights of the edges prevent this algorithm from accurately handling data that have a varying point density. In contrast, the MST is a spanning tree in which the sum of the edge weights is no larger than those of any other spanning tree. To the best of our knowledge, no previous studies have used the MST to extract tree skeletons. The reason could be that the MST is sensitive to noise and often erroneously connects the points of neighboring branches. However, the main advantage of the MST is that it can preserve the local spatial structure of the point cloud.

Motivated by the individual advantages of the trees produced by the DA and the MST and by the similar algorithmic structures of the DA and Prim's algorithm [35], which is used to obtain the MST, we propose a novel spanning tree, which will be referred to as DMst, to extract the tree skeleton from the TLS

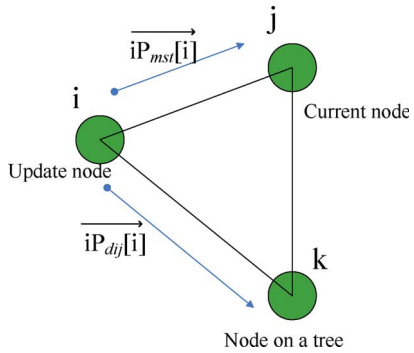


Fig. 2. Connectivity of nodes on the DMst.

point cloud. Essentially, the DMst is a compromise between the DA and the MST since it has a smaller weight sum than the tree produced by the DA, and the distance from each node to the root node is shorter than that in the MST. When constructing a DMst, we start with a root node that is located at the base of the DMst as a single-source vertex, and we continuously increase the size of the tree, one edge at a time. The process stops when the DMst spans all of the vertices of a graph. Fig. 2 illustrates the connectivity of the nodes on a DMst. Suppose that i is a node to be refreshed, j is currently selected as a tree node, and k is a node that is already on the DMst. $\mathbf{P}_{\mathrm{mst}}[i]$ is the parent node of i on the MST, and here, $\mathbf{P}_{\mathrm{mst}}[i]$ is j. $\mathbf{P}_{dij}[i]$ is the parent node of i on a tree that is produced by the DA; here, $\mathbf{P}_{dij}[i]$ is k. $\mathbf{P}_{\mathrm{DMst}}[i]$ is the parent node of i on the DMst, $\overrightarrow{iP_{\mathrm{mst}}[i]}$ is the direction from node i to $\mathbf{P}_{\mathrm{mst}}[i]$, and $\overrightarrow{iP_{dij}[i]}$ is the direction from node i to $\mathbf{P}_{dij}[i]$. Table I presents the details of the algorithm that is applied to construct a DMst. In step I, $\mathbf{D}_{\mathrm{mst}}[i] + \mathbf{D}_{dij}[i]$ is taken as the condition for selecting i. Step II shows the way to refresh a point in the MST. Step III presents the process for refreshing a point in the tree constructed by the DA. As shown in Fig. 2, $P_{\mathrm{mst}}[i]$ and $P_{dij}[i]$ are different. To maintain the DMst with a small weight sum and a short distance from each node on the DMst to the root node, in step IV, we make $\mathbf{P}_{\mathrm{DMst}}[i]$ satisfy the condition that the length sum of $(i, \mathbf{P}_{\mathrm{DMst}}[i])$ and the length of i passing through $\mathbf{P}_{\mathrm{DMst}}[i]$ to the root node is the smallest among those that i connects with other nodes.

If the point cloud contains multiple trees, first, the DMst is constructed, and then, an approach that is similar to the approach presented in [31] is used to divide the data into individual trees.

Because all of the points are on the previously constructed DMst, the DMst is a cluttered tree structure and does not clearly show the main branches. Redundant nodes must therefore be removed, as will be explained in Section II-B1. To adapt to the different point densities, two extensions are added to the DMst in Section II-B2.

1) Node Pruning: Because all of the points of the point cloud are on the DMst, the DMst is cluttered. Some points need to be pruned from the DMst to obtain an approximate tree skeleton on which we can see branch connections more clearly and compute the branch stretching directions efficiently. Assume that the weight of node i is c_i. For node i with no

TABLE I
ALGORITHM FOR CONSTRUCTING A DMst

Input: A non-empty connected weighted graph with vertices **V**, edges **E** and a root node *x*.
Initialize: $\mathbf{V}_{new}=\{x\}$ and the vertex *x* is removed from V. $\mathbf{P}_{mst}[i]$, $\mathbf{P}_{DMst}[i]$, and $\mathbf{P}_{dij}[i]$ store the connectivity information of the MST, DMst, and the tree produced by the DA, respectively. $\mathbf{D}_{mst}[i]$ is the weight of the edge $(i, \mathbf{P}_{mst}[i])$. $\mathbf{D}_{dij}[i]$ is the minimum weight from i to the root node in the graph. If i is connected with *x*, $\mathbf{P}_{mst}[i]$, $\mathbf{P}_{dij}[i]$ and $\mathbf{P}_{DMst}[i]$ are all equal to *x*. $\mathbf{D}_{mst}[i]$ and $\mathbf{D}_{dij}[i]$ of i are the edge weight between i and *x*. Otherwise, $\mathbf{P}_{mst}[i]$, $\mathbf{P}_{dij}[i]$ and $\mathbf{P}_{DMst}[i]$ are set to NULL, $\mathbf{D}_{mst}[i]$and $\mathbf{D}_{dij}[i]$ are set to infinity, and $\mathbf{E}_{new}=\mathbf{Ø}$;
Assume that the weight between vertices i and j is e_{ij}.
while V!=Ø:
Step I: Choose a vertex i with the minimal $\mathbf{D}_{mst}[i]+\mathbf{D}_{dij}[i]$ in **V**.
Add i and the edge $(i, \mathbf{P}_{DMst}[i])$ to $\mathbf{V}_{new}$ and $\mathbf{E}_{new}$.
Step II: **If** $\mathbf{D}_{mst}[i] > e_{ij}$ **then** Δ Refresh $\mathbf{P}_{mst}[i]$ and $\mathbf{D}_{mst}[i]$.
$\mathbf{P}_{mst}[i] = j$
$\mathbf{D}_{mst}[i] = e_{ij}$
End if
Step III: **If** $\mathbf{D}_{dij}[i] > \mathbf{D}_{dij}[j] + e_{ij}$ **then** Δ Refresh $\mathbf{P}_{dij}[i]$ and $\mathrm{D}_{dij}[i]$.
$\mathbf{P}_{dij}[i] = j$
$\mathbf{D}_{dij}[i] = \mathbf{D}_{dij}[j] + e_{ij}$
End if
Step IV: **If**
$$\mathbf{D}_{mst}[i]+(\mathbf{D}_{mst}[i]+\mathbf{D}_{dij}[j]) > (\mathbf{D}_{dij}[i]-\mathbf{D}_{dij}[k])+\mathbf{D}_{dij}[i]$$
then Δ Refresh $\mathbf{P}_{DMst}$.
Step IV-1: $\mathbf{P}_{DMst}[i]= \mathbf{P}_{dij}[i]$.
Else
Step IV-2: $\mathbf{P}_{DMst}[i] = \mathbf{P}_{mst}[i]$.
End if
Step VI: Remove j from **V**.
End while
Output: $\mathbf{V}_{new}$ and $\mathbf{E}_{new}$ describe the DMst.

child nodes, $c_i = 0.1$ m. For a node i with children, c_i can be obtained as follows:

$$c_i = \sum_{j \in \mathbf{\Omega}} (c_j + d_{ij}) \tag{1}$$

where $\mathbf{\Omega}$ is the set of the child nodes of i, j is a child node of i, and d_{ij} is the Euclidean distance between i and j.

It is convenient to introduce a threshold δ, which allows the removal of the children, as follows: If $c_i < \delta$, node i is removed; otherwise, it is retained. Fig. 3 shows the details of the DMst as it has been applied to a typical sample of a tree. All of the points in Fig. 3(a) are on the DMst in Fig. 3(b), which occludes the structure of the skeleton. The region in the red box in Fig. 3(b) is enlarged in Fig. 3(d). After pruning the nodes, an approximate skeleton is clearly displayed [Fig. 3(c)].

2) Adaptation to Varying Point Density: To address varying point densities and different tree species, the following two generalizations are introduced: 1) The weight of each edge in the connected undirected graph is modified, and 2) a positive parameter γ is added to the DMst.

For the first generalization, the weight of each edge in the graph is dynamically changed. In general, a tree surface is smooth, and the extensions of the branches are continuous. Because a thin twig seldom turns suddenly into a thick twig or vice versa, two basic properties are valid. The first property is that the point density around two adjacent nodes is similar because the local point density seldom varies. The second property is that a branch can exist in regions that have high point density. According to these two properties, on the one hand,

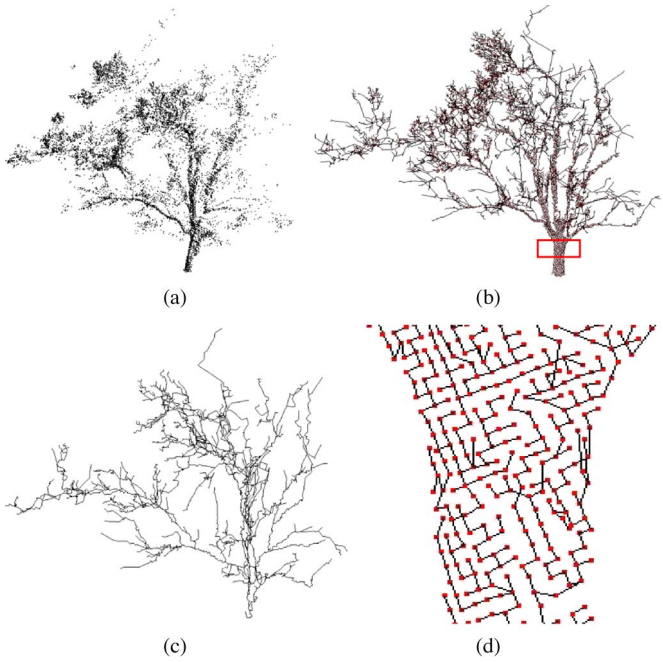


Fig. 3. DMst of a tree. (a) Raw point cloud. (b) DMst from the raw point cloud. (c) DMst after pruning nodes. (d) Details of the region in the red box in (b).

nodes with high density are connected prior to nodes with low point density. On the other hand, adjacent nodes with similar point density are connected prior to those adjacent nodes that have varying point density. In this situation, the branch connections are not affected by complex thin twigs. To satisfy the connections, the edge weight in the graph $G(\mathbf{V}, \mathbf{E})$ must be changed accordingly. The weight e_{ij} of the edge between the vertices i and j can be calculated as follows:

$$eij = dij \times vi \times vj \times (vi/vj + vj/vi) \tag{2}$$

where v_i is the average distance between vertex i and its connected vertices and v_j is the average distance between vertex j and its connected vertices. Because v_i and v_j indicate the reciprocal of the point density at i and j, the larger their values are, the lower the local densities at i and j will be. From (2), it can be observed that the weight e_{ij} is small in the graph G if the point cloud at vertices i and j has similar density. When the DMst is constructed from the graph $G(\mathbf{V}, \mathbf{E})$, edges that have small weights are added to the DMst prior to the edges that have large weights. This procedure clearly satisfies the aforementioned two properties.

To further address the point cloud with varying point density and the adaptation to different tree species, the algorithm presented in Table I is made adaptive. This goal is achieved by introducing an adaptation parameter γ, which adjusts the ratio between the MST and the tree constructed by the DA. More specifically, in Table I, the term $\mathbf{D}_{\text{mst}} + \mathbf{D}_{dij}$ should be replaced by $\mathbf{D}_{\text{mst}}[i] + \gamma \times \mathbf{D}_{dij}[i]$ to control the selected nodes in step I. When the DMst is refreshed, $\gamma \times (\mathbf{D}_{\text{mst}}[i] + (\mathbf{D}_{\text{mst}}[i] + D_{dij}[j]) - \mathbf{D}_{dij}[k]) > (\mathbf{D}_{dij}[i] - \mathbf{D}_{dij}[k] + \mathbf{D}_{dij}[i] - \mathbf{D}_{dij}[k])$ is used in step IV instead of $\mathbf{D}_{\text{mst}}[i] + (\mathbf{D}_{\text{mst}}[i] + \mathbf{D}_{dij}[j]) > \mathbf{D}_{dij}[i] - \mathbf{D}_{dij}[k] + \mathbf{D}_{dij}[i]$. If $\gamma = 0$, step IV-1 is omitted, and only step IV-2 is executed when only using the MST. When γ tends toward infinity, step IV-1 is always followed, while step IV-2 is omitted, which allows DMst to be deduced by only using the DA. For any other values of γ, steps IV-1 and IV-2 are executed, and the DMst is a tree that is somewhere between the trees obtained by the MST and the DA.

Fig. 4 shows the connectivity possibilities among nodes that have different values of γ. Let us consider the case of $\gamma = \gamma_c$ as a reference scenario [see Fig. 4(a)]. When γ_c becomes large, the angle $\angle$kij becomes small, as shown in Fig. 4(b). Otherwise, the angle $\angle$kij is large, as in Fig. 4(c). As this procedure clearly shows, the DMst constructs a tree whose branches extend in different directions depending on the way that γ changes.

The previously described method has been used to obtain approximate tree skeletons from the TLS point cloud. A typical set of performance evaluation results is illustrated in Fig. 5 for the approximate skeletons produced by the MST, the DMst with various values of γ, and the DA. Differences in the extracted branches are shown in the red rectangle. It is noted that the approximate skeleton produced by the MST preserves more local spatial features of the branches, but it introduces incorrect topologies [Fig. 5(b)]. The approximate skeleton produced by the DA gives a natural view of the tree but lacks precise local structure [Fig. 5(f)]. The approximate skeletons produced by the DMst with different γ are shown in Fig. 5(c)–(e). The smaller γ is, the more similar the approximate skeleton produced by the DMst is to that produced by the MST. For larger γ, the DMst skeleton is more similar to that produced by the DA. The approximate tree skeleton obtained by employing the DMst, as shown in Fig. 5(d), is most visually similar to the real tree because it has accurate orientations and local spatial structure. The DMst considers the distance from each point to the root node along the DMst. This arrangement can guarantee that the generated twigs extend toward the outside, which is helpful for the recovery of missing data.

C. Skeleton Refinement and Dominant Direction Extraction

As shown in Fig. 3(d), the direction that a branch stretches can differ from the directions of the individual edges contained within it. We define the dominant direction as the branch stretching direction. The dominant directions can guide the points along the branch stretching directions. Although the pruned tree is an approximate skeleton, there are still many edges that do not represent actual branches. Therefore, to determine the dominant directions, the pruned tree is refined to ensure that the edges on the skeleton represent actual branches. Fig. 6 illustrates the process of obtaining the dominant directions of the points on an interval of a resampled branch by using the pruned DMst in Fig. 5(d) as an example.

To obtain the refined tree skeleton, the pruned tree is resampled. First, the root node, the end nodes without children, and the intersection nodes, which are the nodes that have more than two children, are obtained and labeled on the pruned tree as sampling nodes. Next, starting from these sampling nodes, the parent nodes are searched for among the sampling nodes in the

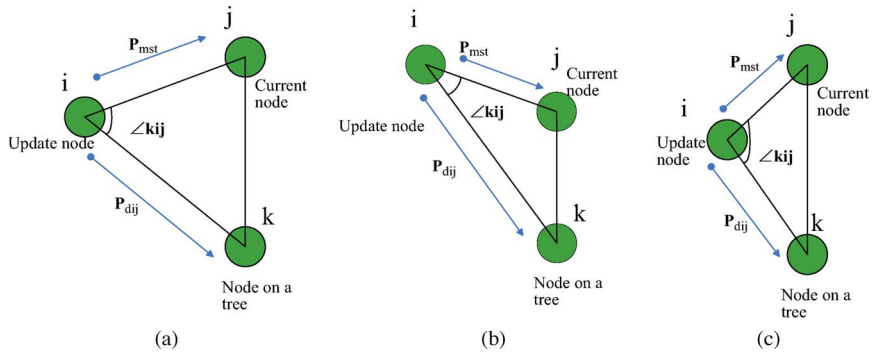


Fig. 4. Possible triangles in the DMst. (a) At $\gamma = \gamma_c$. (b) At $\gamma > \gamma_c$. (c) At $\gamma < \gamma_c$.

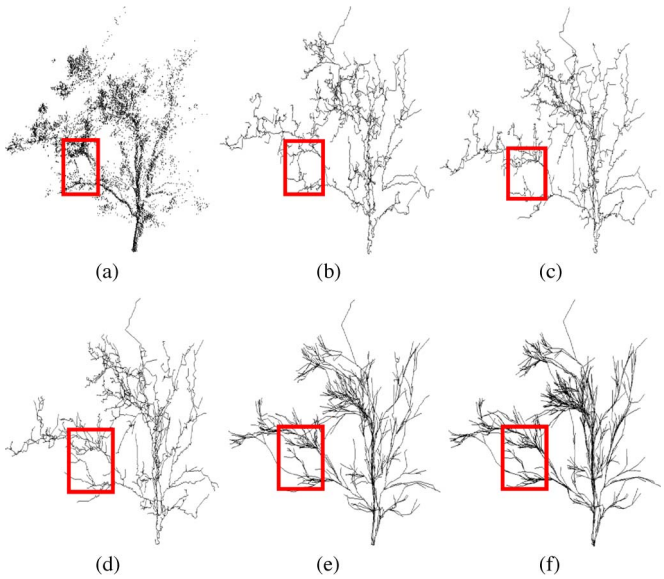


Fig. 5. Tree skeletons generated by the MST and DMst and the tree produced by the DA. (a) Raw point cloud. (b) Pruned MST. (c) Pruned DMst with $\gamma = 0.5$. (d) Pruned DMst with $\gamma = 1$. (e) Pruned DMst with $\gamma = 1.5$. (f) Pruned tree obtained by the DA.

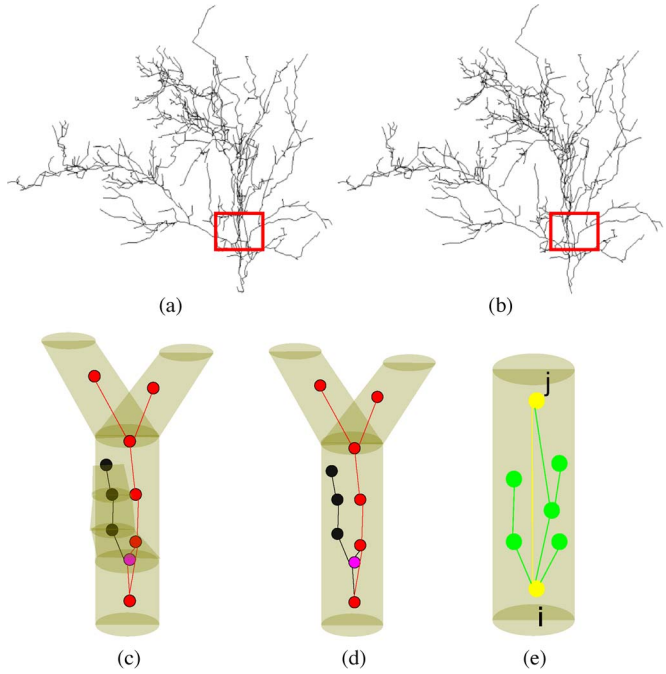


Fig. 6. Process for obtaining the dominant directions. (a) Tree skeletal structure after resampling. (b) Refined tree skeletal structure. (c) Identifying an incorrect branch (black points). (d) Deleting the intersection point of the incorrect branch. (e) Dominant directions of all points shown except node j. The black nodes and edges are deleted. The red nodes and edges are retained. The pink nodes are the intersection nodes. The green points are the children of i between i and j on the DMst. The yellow nodes are located on the refined tree. The yellow line from i to j represents the dominant direction. The green lines are the edges of the DMst.

tree over a regular distance interval s until all of the sampling nodes have been labeled. Finally, these sampling nodes are connected according to the pruned tree to become a resampled tree [Fig. 6(a)]. The tree skeletal structure after resampling can be inflated into a tree model. A small cut of branches is usually thought of as a generalized cylinder. Because the point cloud is subject to noise, it is difficult to determine the radius of the cylinder. However, the point cloud near the base of a tree is relatively dense and less noisy, which means that the radius of the base of the tree can be relatively accurately estimated by cylinder fitting. To estimate the radii of other branches, an approach that is similar to that of [30] is followed. In particular, if node i has only one child node, then the radius r_j of the child node is computed as

$$r_j = r_i \left(\frac{l_j}{l_i} \right)^{1.5} \tag{3}$$

where r_i is the radius of node i, l_j is the total branch length supported by the child, and l_i is the total branch length supported by the parent.

If node i has more than one child, the radius r_j of each child node can be obtained by

$$r_j = r_i \left(\frac{l_j}{\sum_{j \in \Omega} l_j} \right)^{\frac{1}{2.49}}. \tag{4}$$

After the radii of the branches are obtained, the generalized cylinder tree is generated. Branches whose intersection volume exceeds 50% of the volume of their branch cylinders are removed. On the resampled tree, if an original intersection node [shown as the pink node in Fig. 6(c) and (d)] is not an intersection node after these branch points are removed, it is a spurious intersection node. We search for the child and parent of the intersection node and compute the distances from these two

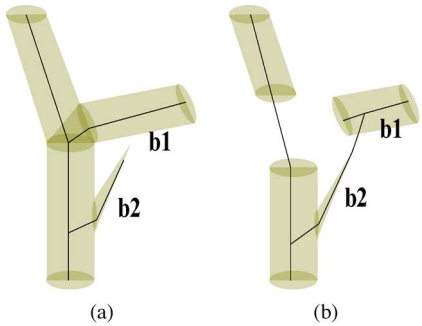


Fig. 7. Example of an intersection region with missing data. (a) Complete branches. (b) Point cloud with missing data in the region of intersection.

nodes to the intersection node. Once either of the two distances is smaller than 0.75 s (s is the distance interval introduced in Section II-C), the intersection node is removed, and the child nodes on the resampled tree are connected to the parent node, as shown in Fig. 6(d). Otherwise, the intersection node is preserved. When all of the nodes on the spurious branches and the spurious intersection nodes are removed, the refined tree skeleton is obtained [see Fig. 6(b)]. The red box highlights the major difference between Fig. 6(a) and (b). If the radii of the branches near the base of a tree are large, spurious branches there would be occluded by the thick branches, and only the main branches would be maintained. On the other hand, the branches on the crown are thin, which causes few intersections to occur; as a result, most of those branches are maintained. Then, the dominant directions can be extracted from the refined tree skeleton. Fig. 6(e) shows two neighboring nodes i and j of the refined tree skeleton, where i is closer to the tree root. Taking the direction from i to j as the dominant direction of i, this direction is also applied as the dominant direction of each child [green points in Fig. 6(e)] of i between i and j on the DMst. Because the end node does not have child nodes on the refined tree skeleton, its dominant direction and that of its children on the DMst are defined as the dominant direction of its parent.

D. SAGO

This section presents the SAGO, which improves the quality of the point cloud by recovering regions of missing data. Through optimization, the points are distributed more uniformly, and regions of missing data are filled. The skeletal structure obtained from the optimized point cloud is much better than that from the raw point cloud.

Fig. 7 illustrates a crossed region of a tree where three branches intersect with each other. If the radii of the cylinders are large, then the corresponding branches are thick; if the radii of the cylinders are small, then the corresponding branches are thin. This typically occurring example illustrates the necessity for our structure-aware optimization to improve the connectivity of the skeleton. When the tree point cloud is perfect, i.e., no branches are occluded, the skeleton can be easily extracted [see Fig. 7(a)]. However, with missing data in the regions of branch intersections, the correct connectivity of the skeleton in these regions is difficult to obtain. The black lines represent the possible skeletons. Missing data often cause incorrect skeleton connections because branches *b1* and *b2* are close to each other, and they are easily connected together [see Fig. 7(b)]. However, *b2* is thin, and *b1* is thick; thus, *b2* cannot be the parent branch of *b1*. Therefore, it is necessary to recover missing data to fill in the correct skeleton. This goal will be achieved by the SAGO.

The main motivation behind the SAGO is based on the following observations. The point cloud of a tree can be regarded as discretizations of the stem and branch surfaces. The tree surface is usually smooth, which implies that the point density varies little on a continuous branch surface. If a branch is partly occluded, data are missing. We can easily extract the whole branch skeleton by extending the branch and the branches around it up to a certain distance based on their radii and then connecting the branch to coincide with its most probable shape. If the radius is large at the end of a branch (i.e., the local point density is high), this arrangement typically indicates that part of this branch is blocked. Therefore, the end of the branch is extended by moving the points at the end of the branch along growing direction of the branch, which is the dominant direction.

It is convenient for our analysis to assume that all of the points are analogous to particles that carry the same electric charge, which repel each other, and the branch-growing movement is analogously driven by the force between these point charges. We define $\mathbf{F}_r$ as the electric force, which is a repulsive force that is exerted on a node by its connecting nodes. Points that experience $\mathbf{F}_r$ will continue to move until they meet other balancing forces. Therefore, we introduce a constraint force $\mathbf{F}_s$ to prevent points from moving significantly away from their original positions. The direction of $\mathbf{F}_s$ points toward the original position from the new positions of the points and scales, with the distance moved. Its operation can be compared with the force from a spring; the farther away that a point is from its original position, the larger $\mathbf{F}_s$ becomes. Next, Section II-D1 introduces the force-balance-based optimization algorithm for recovering regions of missing data. Section II-D2 presents case studies of regions of missing points that can be effectively recovered by this algorithm.

1) Optimization of the Incomplete Point Cloud: The repulsive force $\mathbf{F}_r$ can also be considered to be equivalent to an electric force between two particles that carry electric charges. Therefore, the electric charge q_i is defined first; it is equal to the local point density at point i as follows:

$$q_i \propto \frac{1}{v_i} \tag{5}$$

where v_i is the average distance from all of the nodes in $\mathbf{\Omega}$ to node i.

Because the distribution of the points is highly related to the structure of the branches, we further make $\mathbf{F}_r$ structure-aware by projecting it onto the dominant directions $\mathbf{O}_i$ of the points, i.e.,

$$\mathbf{F}_r(\mathbf{P}) \propto \sum_{j\in\mathbf{\Omega}} \frac{q_i q_j}{\|\mathbf{P}_i - \mathbf{P}_j\|^2}(\mathbf{P}_i - \mathbf{P}_j) \tag{6}$$

$$\mathbf{F}_r(\mathbf{P}) \propto \sum_{j\in\mathbf{\Omega}} \mathbf{O}_i^T \frac{q_i q_j}{\|\mathbf{P}_i - \mathbf{P}_j\|^2}(\mathbf{P}_i - \mathbf{P}_j)\mathbf{O}_i \tag{7}$$

where $\mathbf{P}_i$ and $\mathbf{P}_j$ are the coordinates of points i and j after the optimization, $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \dots, \}$, q_i and q_j are the analogous electric charges of i and j, and $i \neq j$. $\mathbf{\Omega}$ is the set of all nodes connecting with i, $\mathbf{O}_i$ is the dominant direction of i, and T is the transpose operator.

The constraint force $\mathbf{F}_s$ is constructed similar to Hooke's law for a spring. More specifically, the force $\mathbf{F}_s$ between a new position and its original position is proportional to the distance from its original position. Mathematically, this relationship can be expressed as

$$\mathbf{F}_s(i) \propto K_i(\mathbf{U}_i - \mathbf{P}_i) \tag{8}$$

where $\mathbf{U}_i$ is the coordinate of i before the optimization. The damping factor K_i of i is defined as

$$K_i \propto \frac{\left(\sum_{j \in \mathbf{\Omega}} q_j\right) q_i}{v_m^2} \times \mathrm{Log}_2(c_i + 1)_i \tag{9}$$

$$v_m = \frac{1}{m} \sum_{i=1}^{i=m} v_i \tag{10}$$

where m is the number of nodes on the DMst.

It is expected that the nodes near the base of the DMst do not move far, while the nodes near the end nodes in the crown should be moved long distances because they could be near to regions that have a sparse point cloud or missing data. Thus, the values of K_i should be larger near the base of the DMst and smaller near the end nodes. The trends of coefficient c_i, which is defined in Section II-B2, are also similar to the trends of K_i. The growth rate between c_i of node i and that of its children increases approximately exponentially. This growth in c_i would cause K_i to change drastically, which would make the points move too far and become too diffuse. To avoid drastic changes in K_i, we use $log_2(c_i + 1)$ in (9) instead of using only c_i.

When $\mathbf{F}_s = \mathbf{F}_r$, the points will not move. In an optimal way, the movement/positioning of the points in such a way that force balance is achieved, i.e., $\mathbf{F}_r = \mathbf{F}_s$, the optimization function is

$$\mathbf{P} = \arg\min_{\mathbf{P}} \sum \|\mathbf{F}_r(\mathbf{P}) + \lambda \mathbf{F}_s(\mathbf{P})\|^2 \tag{11}$$

where $\lambda > 0$ is a parameter that is used to balance the two forces. Equation (11) is a nonlinear optimization equation that can be solved iteratively by Newton's method.

To indicate the effectiveness of our methodology, we have tested it on the raw point cloud. After the raw point cloud [Fig. 8(a)] is optimized, the data are recovered [Fig. 8(b)]. However, the data in the blue box are not recovered in Fig. 8(c), and the end points are more diffuse than those in Fig. 8(b).

Although the data in regions of missing data can be recovered, the distribution of the points after optimization is diffuse and noisy in those regions, which prevents the skeleton from retaining the branching structure. To maintain the branching structure as much as possible, we attempt to preserve the tree skeleton while processing the raw point cloud. Because the DMst is adaptive to the point density, if the raw point cloud overlaps with the optimized point cloud to form a new point cloud as Γ_0 [Fig. 9(a)] and a new graph and DMst are regenerated on Γ_0, then the regions with complete data will have a higher point density. Because the points in these regions have the priority to be connected on the skeleton, the branching structure is retained. Furthermore, because the regions of missing data are filled by the optimized points, the tree skeleton in these regions is reconstructed more accurately. To show the result of avoiding noise on Γ_0, the optimized points that are also end points are removed iteratively until there is none left. This arrangement is shown in Fig. 9(b), where most of the diffuse points are removed and missing data are recovered. The branch structure in Fig. 9(c) in the regions of missing data is reconstructed more precisely than the branch structure in Fig. 9(d), which was reconstructed from the raw point cloud.

After the tree structure is reobtained, the dominant direction of each point is also recomputed to replace the previous one. The new dominant directions more accurately coincide with the corresponding branch stretching directions. To further improve the accuracy, the same optimization procedure is repeated on

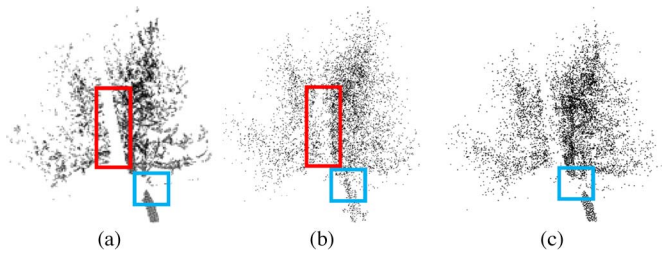


Fig. 8. Comparison of the point cloud before/after optimization. (a) Raw point cloud. (b) Point cloud after optimization. (c) Result after optimization in which c_i instead of $\log_2(c_i + 1)$ was applied in calculating K_i (9).

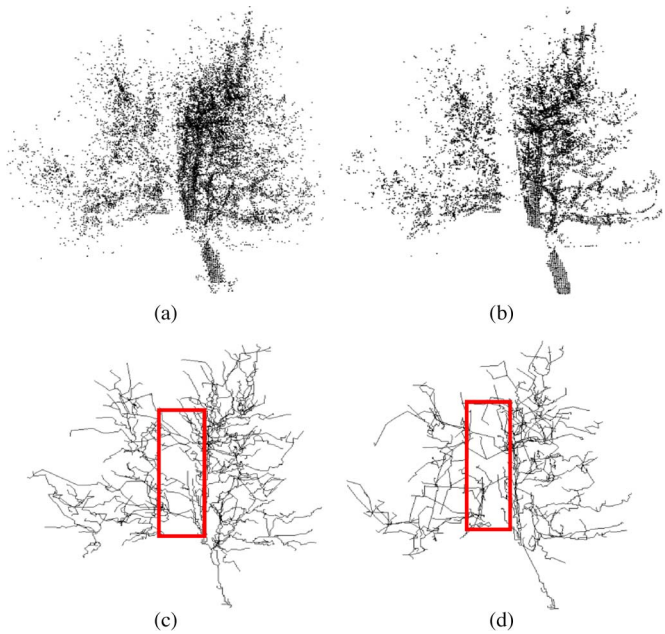


Fig. 9. Effects of different procedures for handling the point cloud. (a) Points generated by adding the optimized points to the original points. (b) Point cloud of (a) after deleting points. (c) Refined skeleton obtained from (b). (d) Refined skeleton obtained from the raw point cloud.

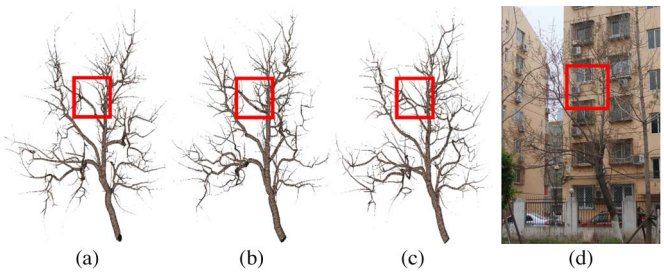


Fig. 10. Three-dimensional tree model is reconstructed in three iterations. (a) First iteration. (b) Second iteration. (c) Third iteration. (d) Photograph of the tree.

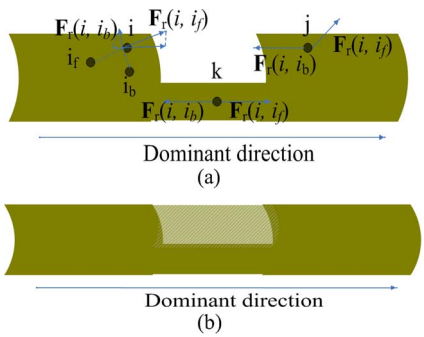


Fig. 11. Recovery of missing data in a part of a branch. (a) Part of the branch is missing, but the branch still connects. (b) Result after optimization.

the optimized point cloud Γ_0 by using the new dominant directions. The reoptimized point cloud Γ_1 is then overlain on Γ_0 to form the new point cloud Γ_2, which results in a more refined tree skeleton. This process is repeated until the difference between L_n, the sum of all of the edge lengths of the skeleton, and L_{n-1}, the sum in the previous iteration, is less than 5% of L_n or L_{n-1}. Experiments have shown that this process converges quickly, i.e., in less than four iterations. In summary, the DMst considers the local point density, and the optimization process recovers missing data to make the point cloud uniform, thus achieving precise tree skeletons from the imperfect raw original point cloud. Fig. 10 shows the visual results of this iterative optimization. Note that the skeleton in Fig. 10(c) has more details than the ones in Fig. 10(a) and (b). From the graphs illustrated in the red box, the third iteration is the most similar to the tree structure that is shown in the photograph [Fig. 10(d)].

2) Recovery of Missing Data: We will use three generic but complementary cases, which will illustrate, in a general manner, how the optimization method can be used to effectively recover the incomplete point cloud. As will become apparent and can be observed from Figs. 11–13, these three cases or their combinations can represent all of the typical spatial distributions of missing data. In particular, Fig. 11 illustrates the case in which the points of a part of a branch are missing, whereas Fig. 12 describes cases of an actual branch end and a spurious branch end. Finally, Fig. 13 shows the case of compound missing data from which it is especially difficult to reconstruct a true skeleton. For all three cases, it is assumed that points at thick branches are dense and points at thin branches are sparse. As shown in Figs. 11–13, the earth-tone-colored

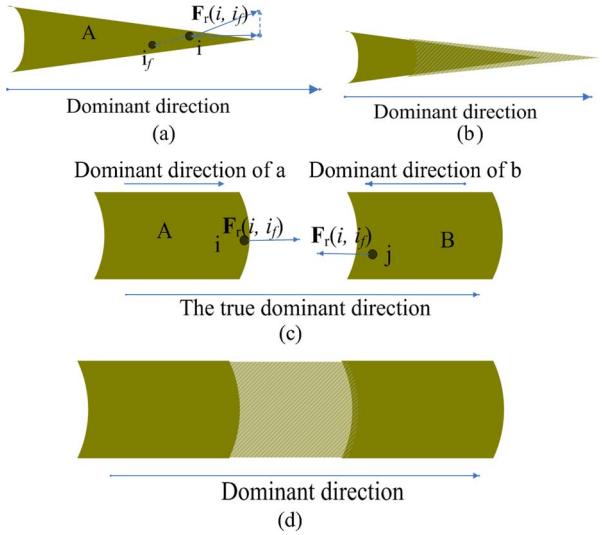


Fig. 12. Illustration of recovering missing data at a true branch end at spurious branch ends. (a) True end of a branch. (b) Result after the optimization of (a). (c) Spurious branch ends. (d) Result after the optimization of (d).

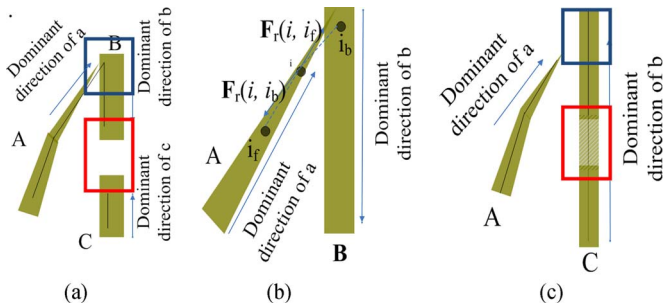


Fig. 13. Illustration of recovering missing data. (a) Compound missing data case. (b) Detail of the blue box in (a). (c) Result after optimization.

regions denote the raw point cloud, and the cross-hatched earth-tone-colored regions indicate the optimized points. Next, the detailed procedure for recovering the missing data for these three cases will be presented.

In Fig. 11(a), suppose that the point density at node i_f is higher than that at node i_b. The electric charge q_{if} of node i_f is larger than the electric charge q_{ib} of node i_b. The angle between the dominant direction and the direction between nodes i_f and i is smaller than the angle between the dominant direction and the direction between nodes i_b and i. Furthermore, because $\mathbf{F}_r(i, i_f) > \mathbf{F}_\mathbf{r}(i, i_b)$, i is forced to move to the right until the constraint force is equal to the repulsive force, while for the same reason, node j will move to the left. For node k, because $\mathbf{F}_r(i, i_f) \approx \mathbf{F}_\mathbf{r}(i, i_b)$, i will move only slightly to the right or left until these two forces balance at or near k. Through this process, the region of missing data can be recovered, as shown in Fig. 11(b).

Considering node i without any children in Fig. 12(a), then, because the repulsive force created by the parent of node i on the DMst is balanced by the constraint force, i moves along its dominant direction. Because it is a true end of a branch, the branch thins slowly along this direction, and thus, the point density here also changes slowly. Therefore, as shown in

Fig. 12(b), $\mathbf{F}_r(i, i_f)$ is small, which means that the points move only a short distance along their dominant directions to keep the end nodes from diffusing. Fig. 12(c) illustrates the point cloud of two spurious branch ends. Occlusions divide the branch into two separated branches, A and B. At the ends of the two spurious branches, which are not true branch ends, the radius of the branch changes significantly, which implies that the point density here also changes significantly. Therefore, $\mathbf{F}_r(i, i_f)$ is large, and thus, the points move far distances along their dominant directions. The dominant direction of the points on branch A points to the right, and the dominant direction of the points on branch B moves to the left. This arrangement causes node i to move to the right and j to move to the left, which fills the region of missing data, as also shown in Fig. 12(d).

In Fig. 13(a), black lines represent the edges of an incorrect DMst. Subbranches B and C are two parts of the same branch, which are separated by the region of missing data that is shown in the red box. In the blue box, A is very close to B. The simultaneous occurrence of these two situations causes the DMst to be constructed incorrectly. The process of reconstructing the missing data in the red box is the same as previously explained [see Fig. 12(c)]. Fig. 13(b) shows an enlargement in the blue box. For node i, the electric charge q_{if} of its parent on the DMst is smaller than the electric charge q_{ib} of its child; thus, $\mathbf{F}_r(i, i_f) > \mathbf{F}_{\mathbf{r}}(i, i_b)$, which causes i to move in the direction that is opposite to the dominant direction of A. For the same reason, all of the points around i also move in that direction. The final outcome of this optimization is illustrated in Fig. 13(c), where it avoids the wrong connections.

E. Construction of Tree Models

This section will present the approach used to reconstruct 3-D trees. The main procedure consists of the following steps. First, branches are extended to obtain accuracy lengths. Second, the skeleton is smoothed by a Laplace function. Third, the radii of the branches are estimated, and if needed, leaves are added.

1) Branch Extension: As mentioned in Section II-B2, some points are deleted during the point-pruning process, which affects the branch length. To retain the actual branch length, some of the original points that were pruned are now remapped onto the trees by using the following procedure. First, we find an end node on the refined skeleton and search for its child node by using the maximum weight on the DMst as a current node. If the distance along the tree skeleton from the current node to the end node is larger than the distance interval s, then this node is added to the tree skeleton as a new end node. Otherwise, its maximum weighted child node is taken as a new current node. This process continues until a node without children is found, which is then added to the tree skeleton. The refined tree skeleton is updated after all of the end nodes on the refined tree skeleton have been extended through this process.

2) Skeleton Smoothing: The branches on the refined tree skeleton have a winding shape because they are obtained from the point cloud that shows only one side of the scanned tree surface. Although the refined skeleton keeps the connections of the tree, it does not present the natural stretching shapes of the branches well. To make the refined tree skeleton smoother, the Laplace function [36] is applied to the points of the skeleton as follows:

$$\begin{bmatrix} \mathbf{W_L L} \\ \mathbf{W_H} \end{bmatrix} \mathbf{V}' = \begin{bmatrix} \mathbf{0} \\ \mathbf{W_H V} \end{bmatrix} \tag{12}$$

where $\mathbf{W}_L$ and $\mathbf{W}_H$ are the diagonal matrices that balance the divergent and attractive forces, respectively. The ith diagonal element of $\mathbf{W}_L$ is denoted $W_{L,i}$, and similarly, the ith diagonal element of $\mathbf{W}_H$ is denoted $W_{H,i}$. In (12), $\mathbf{V}'$ is the optimized point set, $\mathbf{V}$ is the original point set, and $\mathbf{L}$ is the following $n \times n$ Laplace operator:

$$L_{ij} = \begin{cases} \omega_{ij} = 1 & \text{if } (i,j) \in \mathbf{E} \\ \sum_{(i,k)\in\mathbf{E}}^{k} -\omega_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where $\mathbf{E}$ is the edge set of the skeleton and i, j, and k are the nodes of the skeleton.

The diagonal elements of $\mathbf{W}_L$ are set to unity in such a way that every node will have the same convergent ability. The values of the diagonal elements of $\mathbf{W}_H$ are selected as follows. If i is not an end node, then

$$W_{H,i} = \sum_{j\in\mathbf{\Omega}} \cos\theta_{kij} + 1 \tag{14}$$

where k is the parent of i, j is a child node of i, and θ_{kij} is $\angle kij$. If i is an end node or an intersection node, $W_{H,i}$ is assigned a relatively large value. In this way, the intersection nodes and the end nodes are maintained in such a way that the branches between the intersection nodes are smoothed.

3) Radii Estimation: The radii of the tree branches are then estimated by employing a cylinder-fitting procedure that accurately determines the radii. This method fits a cylinder function by using the nodes of the DMst that were relocated within two adjacent nodes on the refined tree skeleton. It is noted that, if the available points are not distributed on a cylindrical surface, then this approach will not provide accurate radius estimates. Additionally, if there are not enough points, the radii cannot be estimated at all. In addressing this case, the radii of the branches are also independently estimated by the allometric model introduced in Section II-C. If the radius of a branch on the refined tree can be calculated by the cylinder-fitting approach, then the size of this radius is compared with the size calculated by the allometric model. When the difference in the two estimated radii is smaller than 20% of the larger radius, the radius computed by the cylinder function is considered accurate, and the corresponding points are defined to be the true width points. Then, these values of the radii are taken as the regularization term, and the allometric model is used as the constraint. They are integrated by the following optimization:

$$\mathbf{r} = \arg\min_{\mathbf{r}} \sum_{\substack{i,j\in\text{tree}, \\ i\notin\text{intersect point}}} \left\| r_j - \left(\frac{l_j}{l_i}\right)^{1.5} r_i \right\|^2$$

$$+\sum_{\substack{i,j\in\text{tree},\\ i\in\text{intersect point}}}\left\|r_j-\left(\frac{l_j}{\sum_j l_j}\right)^{\frac{1}{2.49}}r_i\right\|^2 +\sum_{\substack{i\in\text{tree},\\ i\in\text{true width point}}}\|r_i-R_i\|^2 \quad (15)$$

where i and j are the nodes of the skeleton, j is the child of i, l_i and l_j are the total branch lengths supported by i and j, respectively, R_i is the fitting radius of i, r_i is the final radius of i, r_j is the final radius of j, and $\mathbf{r}=\{r_1,r_2,\ldots\}$.

4) Three-Dimensional Tree Reconstruction: Once the radii of trunks and branches are calculated, the skeleton can be inflated to a 3-D tree model that does not contain any leaves. For trees with leaves, it is quite common to assume that their leaves are more likely to grow at the places where the radii of the branches are relatively small and the point cloud is clustered. To identify the positions of the tree leaves, the label $g_i=q_i/r_i$ is introduced at each point. Then, the labels of all of the points are sorted in descending order, and leaves are added synthetically to nodes whose g_i exceeds a certain threshold, which controls the number of leaves to be added. After the leaves are added, the full 3-D tree models are reconstructed.

III. Performance Evaluation Results

To validate the performance of our framework, we have tested our method on a variety of tree point clouds as follows: 1) the tree point clouds with small regions of missing data and 2) the tree point clouds with large regions of missing data.

A. Experimental Data

The used point clouds have been acquired by RIEGL LMS-Z360 and RIEGL LMS-Z620 scanners. Photographs of the scenes scanned by the RIEGL LMS-Z620 scanner have been taken by a Nikon D300 camera. The horizontal angle spacing and vertical angle spacing of the RIEGL LMS-Z360 scanner are both 0.12°. The horizontal angle spacing and vertical angle spacing of the RIEGL LMS-Z620 scanner are both 0.057°. The point clouds of the scenes were acquired by a single scan. The distances from the scanned trees to the devices varied among the scans. Because all of the trees were scanned during the early spring and winter seasons, they contained relatively few leaves that consequently did not completely occlude the branches and twigs. Table II presents the data that are most important and relevant to our study, the data on scanners, the data on numbers and types of trees used, the data for the used point cloud, and the time required for tree reconstruction as well as the corresponding figures where our experimental performance evaluation results are illustrated. All of the experiments that were conducted within the framework of this study have been performed by using a computer with an Intel core2 Q6600 2.4-GHz processor equipped with a 3.5-GB RAM.

TABLE II
Information on Scanners, Trees, Point Clouds, and Time of Tree Reconstructions

Figure		14	15	17	18	19
Type of scanners		Z620	Z620	Z360	Z620	Z360
Scanning Distance (m)		41.92	18.30	14.88	48.64	10.67
Tree species		Ginkgo	Magnolia	Locust	Paulownia	Jujube
Algorithmic Parameters	λ	1	1	1	1	1
	γ	1	0.8	1	1	1
Point number		4,503	11,856	111,833	6,811	10,663
Number of trees		1	1	1	1	1
Time of tree modeling (sec.)		11	23	627	16	28

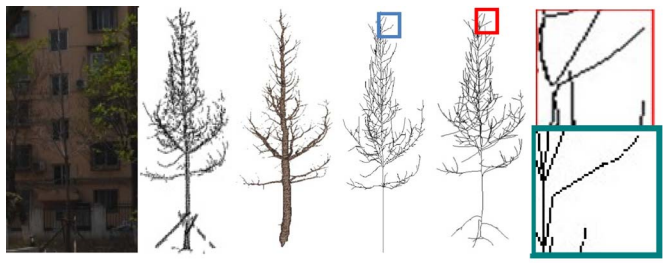


Fig. 14. Reconstruction of a 3-D leafless tree. (a) Tree photograph. (b) Raw point cloud. (c) Reconstructed 3-D tree model. (d) Tree skeleton reconstructed by the method of [31]. (e) Tree skeleton obtained by our method. (f) Details of the blue box in (d) and the red box in (e).

B. Qualitative Evaluation

To qualitatively evaluate the proposed method, we compare it with the approach presented in [31], which is the most similar approach available in the open technical literature. It is noted that the method of [31] aims to automatically construct geometric skeletons of scanned trees with few leaves; it is robust to noise and can handle small regions of missing data. However, it is not flexible enough to describe branch stretching directions, and it fails to handle the point cloud when there are large regions of missing data.

Fig. 14 illustrates the reconstruction of a 3-D tree model of a leafless tree. Some parts of the point cloud [Fig. 14(b)] are missing due to occlusions. The tree model in Fig. 14(c) that is reconstructed by our approach is similar to the tree photograph in Fig. 14(a). This finding demonstrates that our approach can model this tree reasonably well from the point cloud, e.g., clearly the intersections of the branches are described well. The tree skeleton extracted by [31] [Fig. 14(d)] is similar to ours [Fig. 14(e)]. However, as the details in Fig. 14(f), the method of [31] introduces a few unreasonable branches in Fig. 14(d).

The proposed method has been tested for modeling trees that have leaves. Fig. 15 shows the data and reconstruction results for a magnolia tree. Compared with the original photograph [Fig. 15(a)], the magnolia tree shown in Fig. 15(c) is accurately modeled from the point cloud shown in Fig. 15(b). The leaves

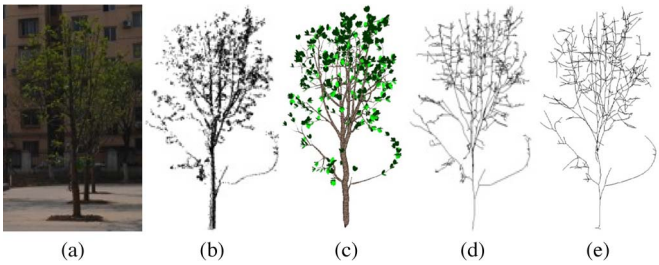


Fig. 15. Reconstruction of a 3-D tree with leaves. (a) Tree photograph. (b) Raw point cloud. (c) Reconstructed 3-D tree model. (d) Tree skeleton extracted by the method of [31]. (e) Tree skeleton extracted by our method.

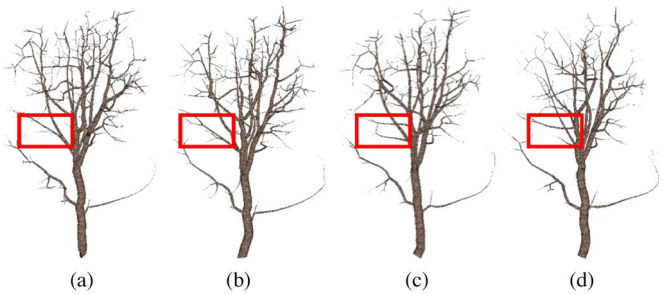


Fig. 16. Reconstructed 3-D tree models that use, from left to right, 100%, 70%, 50%, and 30% of the raw point cloud.

do not affect the extraction of the tree skeleton. It is noted that, in this case, the tree skeletons extracted by [31] and our method (see Fig. 15(d) and (e), respectively) are very similar.

To evaluate the robustness of our method to varying point densities, we have randomly removed a subset of points from the point cloud that is shown in Fig. 15(b). In Fig. 16, the 3-D tree models without leaves have been reconstructed by using 100%, 70%, 50%, and 30% of the raw point cloud. The main structure of the four resultant trees is similar, and the high-level structure is preserved despite the significant reduction in the size of the point density. Although, in general, the removal of a large number of points can reduce the connectivity, our method is quite robust because only a few tree branches (highlighted by the red boxes) are different. It is noted that most of the branches in all four trees maintain good connectivity with other branches.

Another type of tree, namely, the locust tree, which has thin but dense branches, has also been investigated. The original data, which include the original photograph [see Fig. 17(a)] and raw data [see Fig. 17(b)] as well as the various performance evaluation results, are shown in Fig. 17. Reconstructing this type of tree is difficult because of the noisiness and ambiguity of the captured point cloud in the crown [Fig. 17(b)]. In Fig. 17(d), the twig structure is retained very well. Some man-made objects (e.g., the objects that support the tree in Figs. 14 and 17) connect with the tree branches, but they do not interfere with the obtained tree modeling results. For comparison purposes, the method of [31] was also employed to generate the tree skeleton shown in Fig. 17(c), where a skeleton that is similar to the one obtained by our method can be found. We have conducted a second experiment to investigate the effects of the adjustable edge weight (see Section II-B2 for details) on the 3-D model. The tree model in Fig. 17(f) was extracted without using an adjustable edge weight. It makes a few incorrect connections compared to the tree model in Fig. 17(e), which

Fig. 17. Locust tree with thin and dense twigs. (a) Tree photograph. (b) Raw point cloud. (c) Tree skeleton extracted by the method of [31]. (d) Tree skeleton extracted by our method. (e) Reconstructed 3-D tree model. (f) Reconstructed 3-D tree model without considering the local point density.

adopts adjustable edge weights. The red box highlights the locations of the misconnections. This experiment has shown that, if the Euclidean distance is taken as the edge weight, then the branch connectivity is affected by the complex thin twigs. As can be observed from the red box of Fig. 17(f), although the thin twigs appear to be connected to the branches, in fact, they are not actually connected.

Up until now, the performance evaluation results presented in Figs. 14, 15, and 17 indicate that our method can produce accurate tree skeletons from a point cloud that has small regions of missing data. Furthermore, with the next set of results, which will be presented in Figs. 18 and 19, the ability of our method to address large regions of missing data is further validated.

The paulownia tree, the point cloud of which is shown in Fig. 18(b), has large gaps at its base and crown due to occlusion caused by trees in front of it [see Fig. 18(a)]. Nevertheless, as can be observed by comparing the results presented in Fig. 18(c) and (d), our method can accurately reconstruct the complete tree structure despite the large regions of missing data. Moreover, some branches in the tree skeleton obtained by our method in Fig. 18(f) are shown more clearly than the ones in the tree skeleton [Fig. 18(e)] obtained by the method in [31], as shown in the red box.

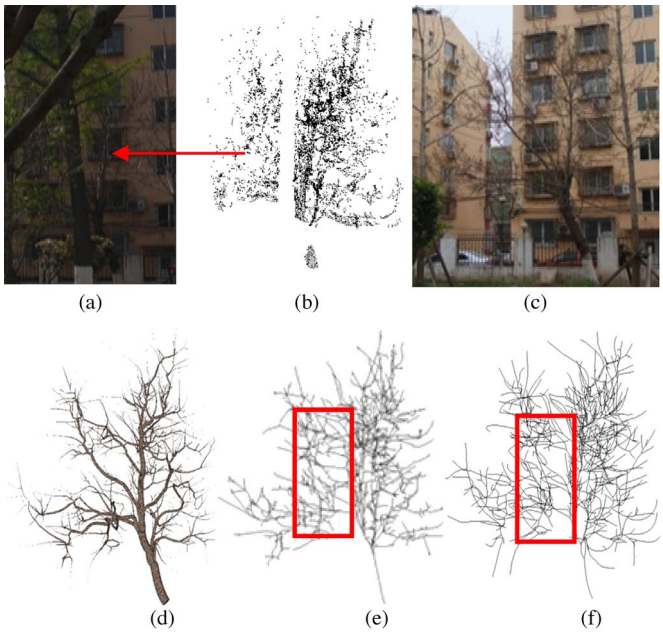


Fig. 18. Modeling a tree from an incomplete point cloud. (a) Tree photograph, with the tree to be modeled in the background, indicated by the red arrow. (b) Raw point cloud. (c) Tree photograph taken from a closer perspective. (d) Reconstructed 3-D tree model. (e) Tree skeleton obtained by the method in [31]. (f) Tree skeleton obtained by our method.

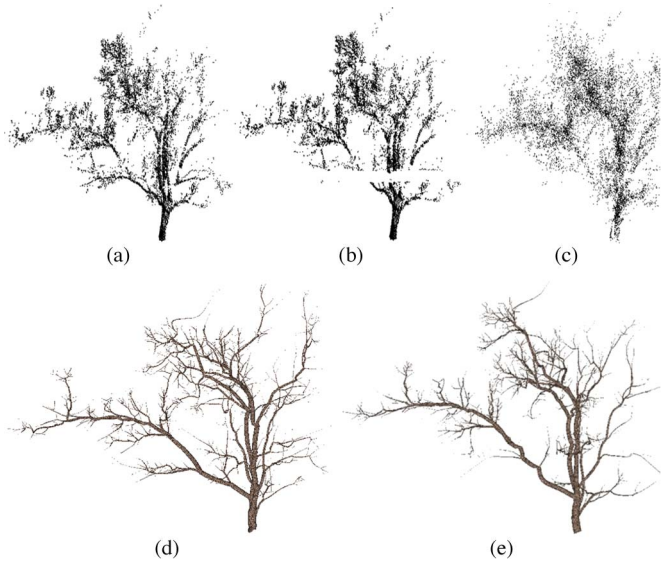


Fig. 19. Tree model that was reconstructed from a point cloud in which some of the points are intentionally removed. (a) Complete point cloud. (b) Point cloud with elevations between 1.3 and 1.8 m intentionally removed. (c) Point cloud after optimization. (d) Three-dimensional tree reconstructed from (a). (e) Three-dimensional tree reconstructed from (b).

The main reason why the method in [31] fails to compensate for severe occlusions is that its optimization and iteration process only pushes the points to shrink toward the current skeleton and forces the tree branches to stretch toward their end points. Thus, if the skeleton is extracted incorrectly in the first place, unreasonable branch connections could exist in the regions where there are missing data.

To further investigate the performance of our method in handling incomplete point clouds, we artificially removed some points from the complete input point cloud, as shown in Fig. 19(a). It is underlined that the removed points are in the region at an elevation of 1.3–1.8 m and are part of some important branch intersections [see Fig. 19(b)]. Fig. 19(c) illustrates that the region where the points are removed is reamended. Compared to the 3-D tree model that is derived from the complete point cloud and is shown in Fig. 19(d), the main branch connections are retained in Fig. 19(e), with the exception that some small twigs are missing because the points that make up those twigs have been completely removed.

C. Quantitative Evaluation

In an effort to quantitatively evaluate the reconstructed models, an approach similar to that of [27] has been employed. The 3-D tree in Fig. 19(d), which is reconstructed from the complete point cloud, is taken as the reference tree model (denoted Ref-T). The tree that is reconstructed at each iteration by using the point cloud in Fig. 19(b) is denoted Des-T, and it will be compared to Ref-T. The sampled point clouds of Des-T and Ref-T are obtained in the same way as in [27]. The sampled point clouds of the tree models are discretized in a 3-D volume whose voxels are of a 0.2-m side length. Each Des-T is compared with Ref-T by computing the normalized difference of the number of points in each voxel. The distribution of the normalized difference in the return numbers per voxel, the mean normalized differences, and the standard deviations between Ref-T and each iteration of Des-T are shown in Fig. 20(a)–(c). For all iterations, the mean values are all below zero (-0.0043, -0.0058), which shows that the wood areas of Des-T are smaller than those of Ref-T. This finding arises because the completely occluded branches cannot be reconstructed. As the dominant directions converge to the branch stretching directions after multiple iterations, the mean value approaches zero, which shows that the reconstructed model becomes more precise after each iteration. The standard deviations that are shown in Fig. 20(a)–(c) are less than 0.176. Clearly, because both the mean values and standard deviations are small, most of the voxels have very similar numbers of points in Ref-T and Des-T, and the branches of Des-T are distributed similarly to those of Ref-T. Therefore, also from a quantitative evaluation point of view, our method is quite insensitive to the incomplete point cloud and can reconstruct trees that have high quality.

It is also interesting and useful to compare the wood area, which is related to the branch radii and branch distribution of both Des-T and Ref-T. As can be observed from the related experimental results presented in Fig. 20(d)–(f), there is only a small difference between the vertical wood area profile of Ref-T and that of Des-T, except in the regions that have missing data. Therefore, our method also performs well at retaining the branch radii and branch distribution even if some of the points are removed. After each iteration, the difference in the wood area between Ref-T and Des-T becomes smaller, i.e., Des-T also becomes more precise with each iteration. The input point cloud of some of the branches is completely missing due to the point removal; thus, in the region between 1 and 2 m above the ground, it is impossible to reconstruct these branches to make the wood area of Des-T similar to that of Ref-T.

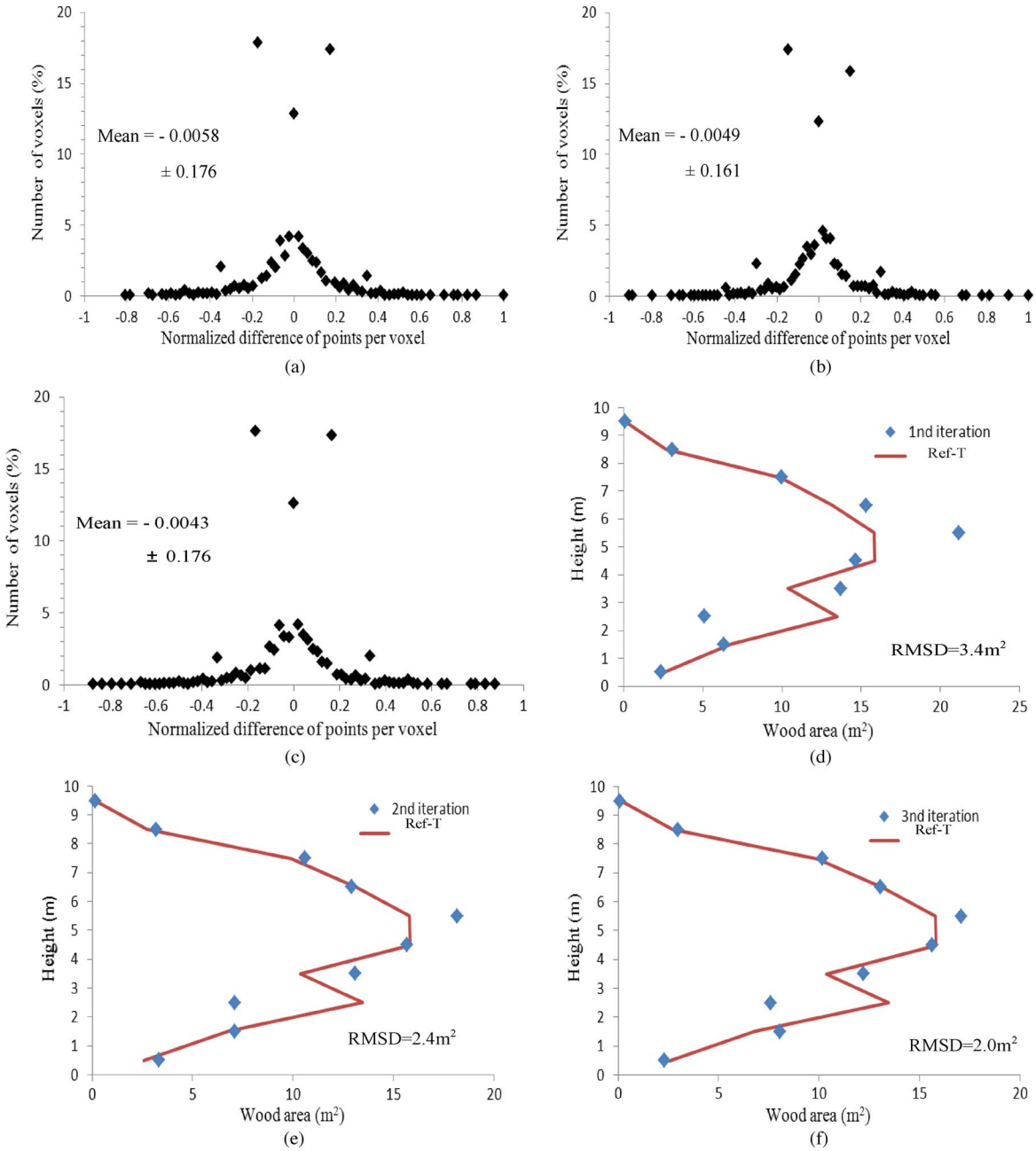


Fig. 20. Distribution of the normalized difference of the points and wood area of each iteration. (a)–(c) Normalized difference of points of the first, second, and third iterations, respectively. (d)–(f) Wood area of the first, second, and third iterations, respectively.

To further validate the accuracy of the radii of our reconstructed trees, a numerical validation is also performed. By taking field measurements, we have obtained the situ data of the diameter at breast height (DBH) of 22 trees. For each tree, the points in the region at 1.25–1.35 m height are intentionally removed because the DBH is the diameter measured at 1.3 m above the ground. The DBH of the experimental data is estimated by (15). As shown in Fig. 21, the red line is a diagonal line that indicates an ideal reconstructed DBH with respect to the measured DBH. The black line is the fitted line, which indicates the relationship between our reconstructed DBH and measured DBH. The fitting linear equation $y = 0.8256x + 5.3839$ of the points is close to the diagonal line. $R^2 = 0.9173$, which means that the measured DBHs are very close to the true DBHs. Therefore, (15) can be used to accurately estimate the DBH. However, the accuracy of the DBH estimation is affected by the shapes of the trunks. Because the shape of a tree trunk is seldom a perfect cylinder, the accuracy of the DBH estimation depends on the shape of the surface that faces the scanner. If a rough surface faces the scanner, the estimated DBH could be smaller than the measured DBH. If a flat surface faces the scanner, the estimated DBH could be larger.

IV. Discussion and Conclusion

This paper has introduced a novel method which accurately and efficiently reconstructs 3-D tree models from the incomplete TLS point cloud. This method proposes an effective DMst

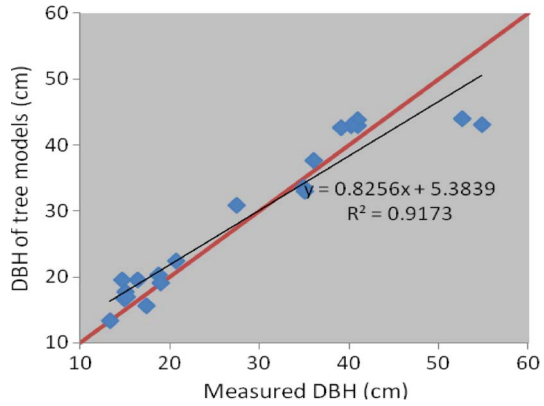


Fig. 21. Comparison of the DBH of the field measurement and the DBH of our reconstructed tree models.

algorithm to derive tree skeletons from which a tree with all of its geometrical features can be generated. These tree skeletons represent the stretching directions of branches well. Such estimated stretching directions make our structure-aware global optimization adaptive to the local structure of branches, achieving a realistic reconstruction. Owing to the flexibility of our method, we can process the TLS-captured point cloud of trees of various complexity and structure. Extensive performance evaluations indicate the robustness of our method with numerous raw scans which represent different types of trees. Compared with previous methods, our method is less sensitive to data incompleteness and noise and can more easily model trees that have large occluded regions while maintaining the characteristics of the input point cloud.

Since the proposed method is essentially data-driven, no virtual branches are added to the trees in order to enhance the visual effects. Our method algorithmically repairs the branches in regions of missing data by using only structural information on the visible parts of those branches. Thus, only the skeletons in areas where the branches and twigs are completely occluded cannot be reconstructed.

Furthermore, in cases where branches intersect with each other, they form a ring that cannot be expressed by the tree structure. Therefore, the DMst fails to address this condition well. In this case, a directed graph should be used to first describe the ring structure to construct a skeleton so that the intersecting branches can be distinguished better. To investigate new tree models which can more accurately represent the real trees, the angles of the tree branches should be included in such more generic models. The angles of the branches depend on the tree species and also on how these trees are managed, e.g., when they are in public parks or in private areas. Of course, this will depend also on the actual tree species. A possible approach to address this problem is to count the angles of the branches of a reconstructed tree and subsequently obtain the distribution of all of the angles to make the result coincide with that of the tree species. In this way, each angle can be computed by the Markov chain Monte Carlo method. An alternative approach should be that the user intervenes to semimanually adjust these angles.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments. Tree skeletons extracted by the method of [31] in this paper are implemented by Feilong Yan at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences.

REFERENCES

[1] C. Mallet and F. Bretar, "Full-waveform topographic lidar: State-of-the-art," *ISPRS J. Photogramm. Remote Sens.*, vol. 64, no. 1, pp. 1–16, Jan. 2009.

[2] J. Hyyppä, O. Kelle, M. Lehikoinen, and M. Inkinen, "A segmentation-based method to retrieve stem volume estimates from 3-D tree height models produced by laser scanners," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 5, pp. 969–975, May 2001.

[3] D. Van der Zande, W. Hoet, I. Jonckheere, J. V. Aardt, and P. Coppin, "Influence of measurement set-up of ground-based lidar for derivation of tree structure," *Agric. Forest Meteorol.*, vol. 141, no. 2–4, pp. 147–160, Dec. 2006.

[4] Y. Wang, H. Weinacker, and B. Koch, "A lidar point cloud based procedure for vertical canopy structure analysis and 3D single tree modelling in forest," *Sensors*, vol. 8, no. 6, pp. 3938–3951, Jun. 2008.

[5] F. M. Danson, D. Hetherington, F. Morsdorf, B. Koetz, and B. Allgower, "Forest canopy gap fraction from terrestrial laser scanning," *IEEE Geosci. Remote Sens. Lett.*, vol. 4, no. 1, pp. 157–160, Jan. 2007.

[6] M. Maltamo, P. Packalén, A. Suvanto, K. Korhonen, L. Mehtätalo, and P. Hyvönen, "Combining ALS and NFI training data for forest management planning: A case study in Kuortane, Western Finland," *Eur. J. Forest Res.*, vol. 128, no. 3, pp. 305–317, May 2009.

[7] N. Pfeifer and D. Winterhalder, "Modelling of tree cross sections from terrestrial laser-scanning data with free-form curves," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 36, no. (8/W2), pp. 76–81, 2004.

[8] Z.-L. Cheng, X.-P. Zhang, and B.-Q. Chen, "Simple reconstruction of tree branches from a single range image," *J. Comput. Sci. Technol.*, vol. 22, no. 6, pp. 846–858, Nov. 2007.

[9] J. Lovell, D. Jupp, G. Newnhamc, and D. Culvenor, "Measuring tree stem diameters using intensity profiles from ground-based scanning lidar from a fixed viewpoint," *ISPRS J. Photogramm. Remote Sens.*, vol. 66, no. 1, pp. 46–55, Jan. 2011.

[10] J.-F. Côté, R. A. Fournier, and R. Egli, "An architectural model of trees to estimate forest structural attributes using terrestrial lidar," *Environ. Model. Softw.*, vol. 26, no. 6, pp. 761–777, Jun. 2011.

[11] G. Zheng and L. M. Moskal, "Leaf orientation retrieval from terrestrial laser scanning (TLS) data," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 10, pp. 3970–3979, Oct. 2012.

[12] A. Bucksch and S. Fleck, "Automated detection of branch dimensions in woody skeletons of leafless fruit tree canopies," in *Proc. SILVILASER*, Austin, TX, USA, Oct. 2009, pp. 14–16.

[13] F. Morsdorf, E. Meier, B. Kötz, K. I. Itten, M. Dobbertin, and B. Allgöwer, "Lidar-based geometric reconstruction of boreal type forest stands at single tree level for forest and wildland fire management," *Remote Sens. Environ.*, vol. 92, no. 3, pp. 353–362, Aug. 2004.

[14] J. R. Rosell, J. Llorens, R. Sanz, J. Arnó, M. Ribes-Dasi, J. Masip, A. Escolá, F. Camp, F. Solanelles, F. Grócia, E. Gil, L. Val, S. Planas, and J. Palacín, "Obtaining the three-dimensional structure of tree orchards from remote 2D terrestrial lidar scanning," *Agric. Forest Meteorol.*, vol. 149, no. 9, pp. 1505–1515, Sep. 2009.

[15] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan, "Single image tree modeling," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 108:1–108:7, Dec. 2008.

[16] C. Zhu, X. Zhang, B. Hu, and M. Jaeger, "Reconstruction of tree crown shape from scanned data," in *Proc. Technol. E-Learning Digit. Entertain.*, Nanjing, China, Jun. 2008, pp. 745–756.

[17] M. Rutzinger, A. K. Pratihast, S. J. Oude Elberink, and G. Vosselman, "Tree modelling from mobile laser scanning data-sets," *The Photogramm. Rec.*, vol. 26, no. 135, pp. 361–372, Jun. 2011.

[18] C. Pradal, F. Boudon, C. Nouguier, J. Chopard, and C. Godin, "PlantGL: A Python-based geometric library for 3D plant modelling at different scales," *Graph. Models*, vol. 71, no. 1, pp. 1–21, Jan. 2009.

[19] G. Vosselman, "3D reconstruction of roads and trees for city modelling," *ISPRS—Int. Arch. Photogram. Rem. Sens. Spatial Inform. Sci.*, vol. 34, no. 3/W13, pp. 231–236, 2003.

[20] B. Gorte, "Skeletonization of laser-scanned trees in the 3D raster domain," in *Lecture Notes in Geoinformation and Cartography: Innovations in 3D Geo Information Systems*. Berlin, Germany: Springer-Verlag, 2006, pp. 371–380.

[21] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[22] A. Bucksch and R. Lindenbergh, "Campino—A skeletonization method for point cloud processing," *ISPRS J. Photogramm. Remote Sens.*, vol. 63, no. 1, pp. 115–127, Jan. 2008.
[23] A. Bucksch, R. C. Lindenbergh, and M. Menenti, "SkelTre—Fast skeletonisation for imperfect point cloud data of botanic trees," in *Proc. Eurograph. Workshop 3D Object Retrieval*, 2009, pp. 13–20.
[24] N. Pfeifer, B. Gorte, and D. Winterhalder, "Automatic reconstruction of single trees from terrestrial laser scanner data," *ISPRS—Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 35, pt. B5, pp. 114–119, Jul. 2004.
[25] Z. Su, Y. Zhao, C. Zhao, X. Guo, and Z. Li, "Skeleton extraction for tree models," *Math. Comput. Model.*, vol. 54, no. 3/4, pp. 1115–1120, Aug. 2011.
[26] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su, "Point cloud skeletons via Laplacian-based contraction," in *Proc. SMI*, Jun. 2010, pp. 187–197.
[27] 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.
[28] A. Verroust and F. Lazarus, "Extracting skeletal curves from 3D scattered data," *Visual Comput.*, vol. 16, no. 1, pp. 15–25, Feb. 2000.
[29] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Proc. Eurograph. Workshop Natural Phenomena*, 2007, pp. 63–70.
[30] H. Xu, N. Gosset, and B. Chen, "Knowledge and heuristic-based modeling of laser scanned trees," *ACM Trans. Graph.*, vol. 26, no. 4, p. 19, Oct. 2007.
[31] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana, "Automatic reconstruction of tree skeleton from point clouds," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 151:1–151:8, 2010.
[32] Z. Xiao, S. Liang, J. Wang, P. Chen, X. Yin, L. Zhang, and J. Song, "Use of general regression neural networks for generating the GLASS leaf area index product from time series MODIS surface reflectance," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 209–223, Jan. 2014.
[33] M. Zhang, L. Zhang, P. T. Mathiopoulos, W. Xie, Y. Ding, and H. Wang, "A geometry and texture coupled flexible generalization of urban building models," *ISPRS J. Photogramm. Remote Sens.*, vol. 70, pp. 1–14, Jun. 2012.
[34] M. Zhang, L. Zhang, P. T. Mathiopoulos, Y. Ding, and H. Wang, "Perception-based shape retrieval for 3D building models," *ISPRS J. Photogramm. Remote Sens.*, vol. 75, pp. 76–91, Jan. 2013.
[35] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Tech. J.*, vol. 36, no. 6, pp. 1389–1401, Nov. 1957.
[36] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa, "Laplacian mesh optimization," in *Proc. 4th Int. Conf. Comput. Graph. Interactive Tech.*, 2006, pp. 381–389.

Zhen Wang is currently working toward the Ph.D. degree in the School of Geography, Beijing Normal University, Beijing, China.

His research interests are the application of light detection and ranging in classification of ground objects and 3-D urban modeling.

Liqiang Zhang received the Ph.D. degree in geoinformatics from the Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing, China, in 2004.

He is currently a Professor with the School of Geography, Beijing Normal University, Beijing. His research interests include remote sensing image processing, 3-D urban reconstruction, and spatial object recognition. He is the corresponding author of this paper.

Tian Fang (M'13) received the Bachelor's and Master's degrees in computer science and engineering from South China University of Technology, Guangzhou, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology (HKUST), Kowloon, Hong Kong, in 2011.

He is currently a Postdoctoral Researcher with HKUST. His research interests include image-based modeling, image segmentation, recognition, and photorealistic rendering. He now works on projects related to real-time 3-D reconstruction and recognition.

P. Takis Mathiopoulos (SM'94) received the Ph.D. degree in digital communications from the University of Ottawa, Ottawa, ON, Canada, in 1989.

From 1982 to 1986, he was with Raytheon Canada Ltd., working in the areas of air navigational and satellite communications. In 1988, he joined the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, where he was a faculty member until 2003, holding the rank of Professor since 2000. He is currently the Director of Research with the ISARS/NOA and a Professor of digital communication with the Department of Informatics and Telecommunication, University of Athens, Athens, Greece. For the last 25 years, he has been conducting research mainly on the physical layer of digital communication systems for terrestrial and satellite applications. He is also interested in channel characterization and measurements, modulation and coding techniques, SIMO/MIMO, UWB, OFDM, software/cognitive radios, and green communications. He has also research activities in the fields of remote sensing and photogrammetry. He has been or currently serves on the editorial board of several archival journals, including *IET Communications* and the IEEE TRANSACTIONS ON COMMUNICATIONS (1993–2005). Since 1993, he has served on a regular basis as a Scientific Advisor and a Technical Expert for the European Commission (EC). In addition, since 2001, he has served as the Greek representative to high-level committees in the EC and the European Space Agency. He has delivered numerous invited presentations, including plenary lectures, and has taught many short courses all over the world.

Dr. Mathiopoulos was an ASI Fellow, a Killam Research Fellow, and a corecipient of two best paper awards. He has been a member of the TPC of more than 50 international conferences, as well as TPC Vice Chair for the 2006-S IEEE VTC and 2008-F IEEE VTC and Cochair of FITCE2011.

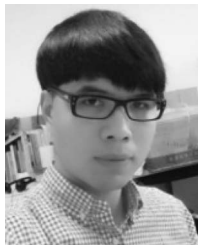
Huamin Qu (M'07) received the B.S. degree in mathematics from Xi'an Jiaotong University, Xi'an, China, and the M.S. and Ph.D. degrees in computer science from Stony Brook University, Stony Brook, NY, USA, in 2004.

He is an Associate Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong. His main research interests are visualization and computer graphics. He is the coauthor of more than 60 refereed papers, including 20 papers in the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS (TVCG).

Dr. Qu received the Honorable Mention for Best Paper Award at IEEE Visualization 2009 and was the recipient of 2009 IBM Faculty Award. He is on the steering committee of the IEEE Pacific Visualization Conferences and is an Associate Editor of IEEE TRANSACTION ON VISUALIZATION AND COMPUTER GRAPHICS.

Dong Chen received the Ph.D. degree in geographical information sciences from Beijing Normal University, Beijing, China, in 2013.

He is currently an Assistant Professor with Nanjing Forestry University, Nanjing, China. His research interests are the application of light detection and ranging in the application of remote sensing and geographic information systems in the field of forest ecosystems.



Yuebin Wang is currently working toward the Ph.D. degree in the School of Geography, Beijing Normal University, Beijing, China.

His research interests are remote sensing imagery processing and 3-D urban modeling.