remote sensing

MDPI

Article

AdTree: Accurate, Detailed, and Automatic Modelling of Laser-Scanned Trees

Shenglan Du[1], Roderik Lindenbergh[2], Hugo Ledoux[1], Jantien Stoter[1], and Liangliang Nan[1,*]

[1] 3D Geoinformation Research Group, Faculty of Architecture and the Built Environment, Delft University of Technology, 2628 BL, Delft, Netherlands

[2] Department of Geoscience and Remote Sensing, Delft University of Technology, 2628 CN, Delft, Netherlands

* Correspondence: liangliang.nan@gmail.com or liangliang.nan@tudelft.nl

Received: 30 June 2019; Accepted: 28 August 2019; Published: 4 September 2019

Abstract: Laser scanning is an effective tool for acquiring geometric attributes of trees and vegetation, which lays a solid foundation for 3-dimensional tree modelling. Existing studies on tree modelling from laser scanning data are vast. However, some works cannot guarantee sufficient modelling accuracy, while some other works are mainly rule-based and therefore highly depend on user inputs. In this paper, we propose a novel method to accurately and automatically reconstruct detailed 3D tree models from laser scans. We first extract an initial tree skeleton from the input point cloud by establishing a minimum spanning tree using the Dijkstra shortest-path algorithm. Then, the initial tree skeleton is pruned by iteratively removing redundant components. After that, an optimization-based approach is performed to fit a sequence of cylinders to approximate the geometry of the tree branches. Experiments on various types of trees from different data sources demonstrate the effectiveness and robustness of our method. The overall fitting error (i.e., the distance between the input points and the output model) is less than 10 cm. The reconstructed tree models can be further applied in the precise estimation of tree attributes, urban landscape visualization, etc. The source code of this work is freely available at https://github.com/tudelft3d/adtree.

Keywords: laser scanning; point cloud; tree modelling; precision forestry

1. Introduction

Trees are an important component throughout the world. They form and function in natural ecosystems such as forests, and also in human-made environments for instance parks and gardens [1]. Urban scenes without trees or plants are lifeless. Furthermore, satisfying environmental goals always require heavy reliance on vegetation mapping and monitoring [2]. Models of trees, therefore, have a wide range of applications, including urban landscape design, ecological simulation, forestry management, and virtual entertainment. While applications such as landscape design and visualization only require modelling virtual trees, lots of other applications relevant for ecological modelling and forestry management require accurate estimation of tree parameters (e.g., height, stem thickness). Accurate tree modelling not only enhances the realism of a scene, but also provides promising approaches to scientifically manage vegetation and forests, which will in return contribute to ecosystem protection, resource preservation, preventing degradation, and many other human activities [3]. Hence, obtaining accurate 3D tree models is necessary and of great importance to the modern society.

The traditional way of measuring trees is to manually conduct fieldwork, which is usually expensive and time-consuming [4]. Since the last several decades, remote-sensing technology has been widely exploited in extracting different information on forests and plants [5]. Both satellite sensors and airborne sensors can effectively acquire digital images with high spatial resolution, which provide viable data sources for forestry analysis at individual tree level [3]. Moreover, with the development

of digital image processing technologies, researchers considered reconstructing digital tree models from photographs [6]. The work of [7] utilized visual hulls of the original tree shape to approximate the main skeleton of the tree, based on which small twigs and leaves are synthesized to generate a plausible tree model. Reche-Martinez et al. [6] described a volumetric approach to reconstruct trees from multiple views. By combining plant images with sparse point clouds obtained from Structure From Motion (SFM), Quan et al. [8] reconstructed realistic plants with generic leaves incorporating user interaction. While the above studies can produce impressive modelling results, they do not aim to reconstruct explicit branch or leaf geometry. Reconstructing trees from photographs remains a challenging problem due to the complexity of the modelling process [9].

Recently, Light Detection and Ranging (LiDAR) technology have been widely used in forestry-related analysis and studies. As measurements from LiDAR can achieve millimeter-level of details from objects, it has become possible to directly capture 3D information and rapidly estimate tree attributes [10]. For example, LiDAR measurements are widely applied in researches such as tree height estimation [11], tree canopy analysis [12] and tree species classification [13]. Moreover, by applying LiDAR technology we are capable of acquiring highly dense point clouds, which lays the foundation for accurate tree reconstruction and modelling.

To achieve accurate tree modelling from laser scans, both the branch geometry and the tree topological structure are required. Among most literature studies, the common approach to obtain the tree branch geometry is cylinder-fitting [14]. When it comes to the reconstruction of tree topological structure, existing methods can be classified into two main categories: segmentation-based and skeleton-based. Segmentation-based approaches first segment the tree point cloud into small subsets and then connect them procedurally to reconstruct the topological structure of the tree. For example, Hackenberg et al. [15] developed a hierarchical-cylinder structure that enables parent-child neighbor relations among branches, which can efficiently extract different tree components such as the tree stem or a single branch. Raumonen et al. [16] proposed another tree modelling method based on a step-by-step collection of small connected surface patches. Bucksch et al. [17] organized the input points into an octree structure and generated skeletal curves from the octree cells. Yan et al. [18] applied a K-means clustering-based approach to extract the tree topological structure. However, these works highly rely on the quality of the input data and therefore may not be robust enough to data with quality issues such as outliers or missing data due to occlusions.

Unlike segmentation-based approaches that reconstruct the tree topological structure from small segments and subsets, skeleton-based methods directly extract skeleton curves from raw input point clouds. Some works employ a rule-based procedural modelling approach to synthesize branches [9,19], which generate the tree skeleton with high quality but require prior knowledge as well as manual parameter adjustment. Some other works proposed purely data-driven methods to automatically extract the skeleton without requiring additional user interactions. The work of [20] constructed the shortest-path map over the input point clouds to extract consecutive skeletal curves. Following this work, Delagrange et al. [21] developed a tool PypeTree to reconstruct tree branch tissues from point clouds. As an alternative for the shortest-path approach, Dey and Sun [22] utilized the medial axis to represent the skeletal structure of 3D tree-like objects. Livny et al. [23] computed a minimum spanning graph over the point cloud to obtain an initial tree skeleton and applied several global optimization techniques to refine the tree branch structure. Following this work, we further improve the fidelity of the reconstructed tree models.

In this paper, we propose a skeleton-based approach to accurately reconstruct tree branches from individual tree point clouds. Our method employs a Minimum Spanning Tree (MST) algorithm to effectively extract the initial tree skeleton over input points. By iterative skeleton simplification and cylinder fitting, we obtain a tree model with reconstructed branches. Leaves and textures are added to enhance the realism of the tree model. One novelty of our work is that we construct the initial tree skeleton based on the intrinsic spatial distribution of input points. Furthermore, we develop a specific simplification strategy to maintain the natural topological structure of tree branches while

collapsing redundant vertices and edges. Our experiments and various comparisons demonstrate both the geometrical correctness and the topological fidelity of the generated tree models.

2. Overview

This work focuses on 3D modeling of real-world trees from point clouds. Thus, the expected solution to this problem needs to fulfill the following requirements:

- ***Robust to tree species.*** The method should be able to reconstruct common tree species with clear branch structures. The reconstructed models should convey the main topological branch structure of the real world trees represented by the point clouds. Vegetation that do not demonstrate skeleton structures (e.g., bushes) are not considered the target objects.
- ***High fidelity.*** The reconstructed tree models should be topologically faithful to the input and have acceptable geometrical accuracy. This is vital for applications where important tree attributes (i.e., stem location, stem thickness, tree height) are expected to be derived from the reconstructed models.
- ***High efficiency.*** The reconstruction process should not require user intervention, i.e., it is fully automatic and can produce 3D models of individual trees promptly regardless of the size of the input point clouds. This enables large scale tree modeling when, for instance, an instance segmentation of the trees is available [24].

The input to our method is a point cloud of a single tree, which is typically contaminated by noise and outliers but is expected to convey the major branch structure of the tree. We assume that individual trees have already been segmented from the point cloud and the segmentation of multiple trees is out of the scope of this work. In practice, the segmentation can be achieved using an interactive technique or various machine learning approaches such as [24]. The output of our method is a 3D model of the same tree in a mesh representation, which can be exported in a standard file format (e.g., OBJ file format). Figure 1 shows an overview of the module design of the proposed methodology, which consists of the following major steps:

- ***Skeleton initialization.*** We triangulate the input points and apply the MST algorithm to extract the initial tree skeleton. Note that the main-branch points are identified and centralized beforehand to improve the quality of the skeleton;
- ***Skeleton simplification.*** The initial skeleton is iteratively simplified, resulting in a light-weight tree skeleton. We simplify the skeleton by retrieving and merging adjacent vertices if their distance is sufficiently small;
- ***Branch fitting.*** Based on the reconstructed tree skeleton, we fit a sequence of cylinders over the input points to approximate the geometry of the branches. We first apply non-linear least squares to obtain the accurate radius of the tree trunk. Then, we derive the radius of the subsequent branches from the main trunk geometry;
- ***Adding realism.*** We synthesize leaves at the end of tree branches and add texture to enhance realism.

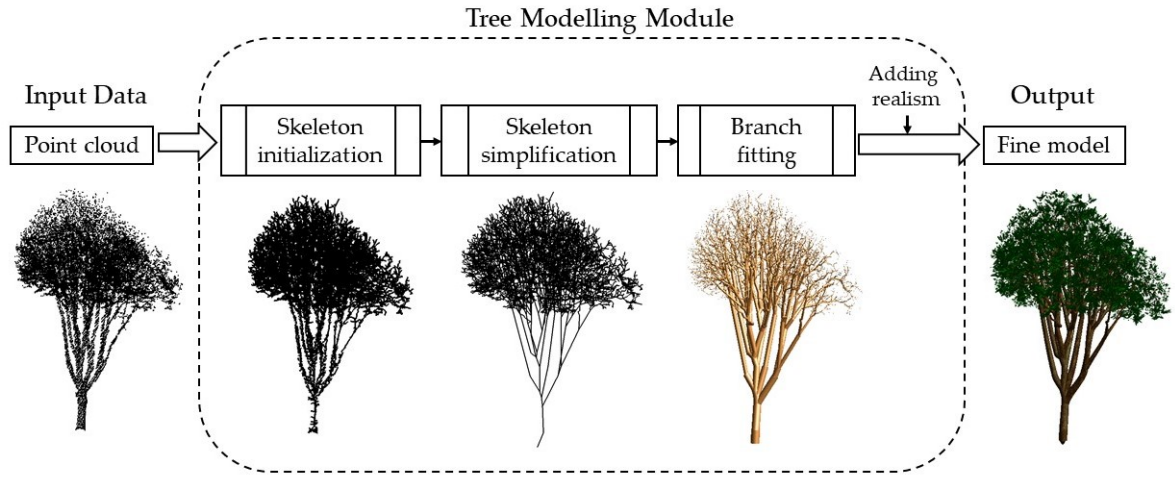


Figure 1. An overview of the proposed methodology.

In Section 3, we describe each step of the modelling module in details.

3. Materials and Methods

3.1. Skeleton Initialization

Based on the fact that points close to each other are likely to belong to the same branch, we construct an MST graph over the input point cloud to represent the initial tree skeleton. To extract an MST over input points, we first apply Delaunay triangulation to construct an initial graph. Delaunay triangulation lays the foundation for MST computation as most efficient approaches find a minimum spanning tree among edges in the Delaunay triangulation of the points [25]. Additionally, it helps to complete the missing region or incomplete branches, which ensures the robustness of our method for input point clouds of poor data quality. Having obtained the triangulation graph, we weight all the edges using their lengths defined in the Euclidean space. Then the Dijkstra shortest path algorithm is utilized to compute the MST from the triangulation, which serves as a representation of the initial skeleton of the individual tree.

Figure 2 shows the initial skeleton extracted over the input points by shortest-path computation. In most cases, the constructed MST indicates the skeletal structure of the original tree (Figure 2a). Nevertheless, special cases exist when pure MST cannot represent the tree skeleton correctly (Figure 2b). Trees with a short and wide shape typically have scattered points and branches, which leads to the computed MST growing in a horizontal manner rather than a compact vertical manner.

We address this problem by intentionally centralizing points that belong to the main branches of the tree. The aim is to generate condensed branches for better skeleton extraction. Point density near bifurcations or branch tips often changes sharply, while points within a single branch have more stable point density (Figure 3a), we can find main-branch points as they have relatively stable point density in their neighbors. Identified main-branch points are centralized through the mean-shift algorithm [26]. As illustrated in Figure 3b, the extracted skeleton after main-branch point centralization has a compact vertical growing manner.

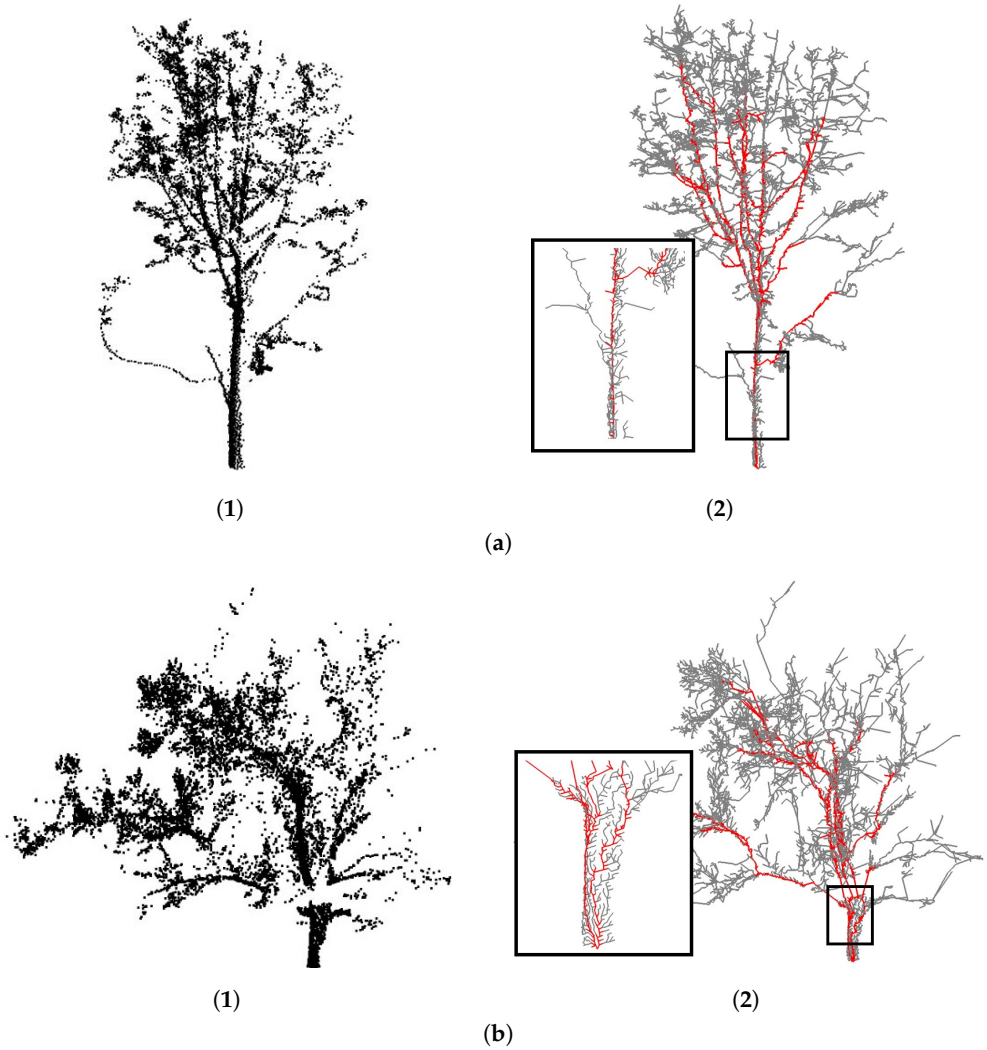


Figure 2. Skeleton initialization for two trees. (**a**) A valid tree skeleton structure (in red). (**b**) An invalid tree skeleton structure (in red). The left column shows the input point clouds.

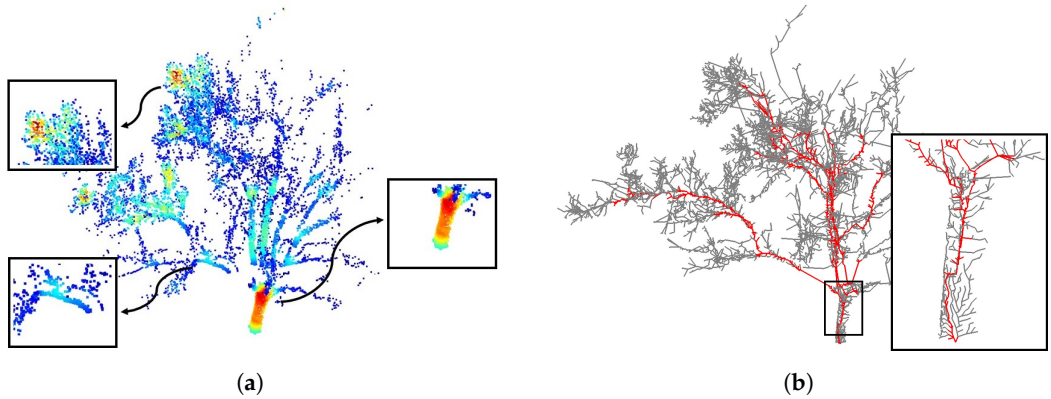


Figure 3. Skeleton extraction from the centralized points. (**a**) Density map of the raw point cloud. Red indicates high density and blue indicates low density. (**b**) Skeleton extracted after the main-branch point centralization.

3.2. Skeleton Simplification

The initial tree skeleton extracted from the input point cloud has a large number of redundant vertices and edges. Most redundant vertices and small edges do not contribute to the tree skeleton

shape and thus are of little importance and should be removed to further simplify the tree skeleton. The simplification is conducted in two major steps. We first assign importance values to vertices and edges, based on which small noisy components can be removed. Then, we iteratively check the proximity between adjacent vertices and merge nearby vertices. Figure 4 illustrates the simplification process.

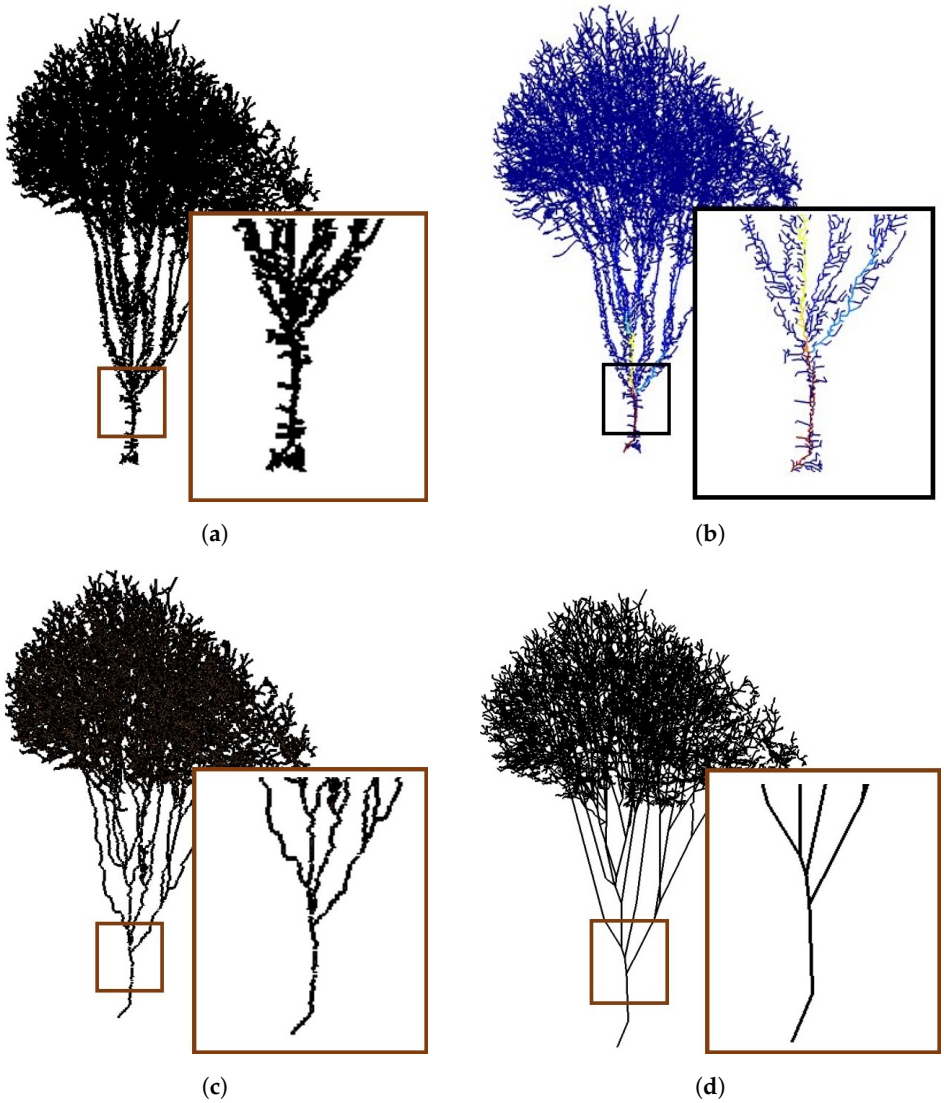


Figure 4. Skeleton simplification. (**a**) Initial skeleton. (**b**) Importance value assigned to branches. Red indicates high importance and blue indicates low importance. (**c**) Simplification by eliminating noisy small branches. (**d**) Simplification by merging similar vertices and edges.

3.2.1. Assigning Vertex and Edge Importances

We assign importance values to vertices and edges in the initial tree skeleton to further guide the simplification process. Our goal is to keep important vertices and main branch edges while ignoring short branches and noisy points. Previous work [9] suggests utilizing the point density, together with the point normal extracted from Principal Component Analysis (PCA), to indicate the importance of vertices. However, the weights evaluated in such a way significantly depend on the quality of the scanned points and thus become unreliable in case of point clouds of poor quality.

Instead of weighting points based on the local point density, we weight each vertex according to the length of the subtree. The subtree of a given vertex v is defined as v itself together with its descendant vertices and edges [27]. Accordingly, the weight of the vertex v is computed as the sum

of the length of all edges within its subtree. In such a way, high-connective vertices close to the tree base area get heavier weights while low-connective vertices near the tree crown get smaller weights. One advantage is that the weighting process is not sensitive to input point density, which makes it robust to data with different scanning qualities. Accordingly, each branch edge is weighted as the average of the subtree length of its two ending vertices. Typically, vertices and edges on the tree crown have consistent low weights [23], while near the tree base small branches have drastically small weights compared to the main trunk branches. Such a characteristic helps us to clear away noisy branches at the trunk, and at the same time, keeps small leave branches at the crown.

3.2.2. Simplifying Adjacent Vertices and Edges

Having eliminated small noisy branches with relatively low importance, we notice that many redundant vertices and edges still exist as they have similar positions and orientations as vertices and edges within their neighborhood. To simplify those similar components, we iteratively check the proximity between adjacent vertices. A similarity indicator α is defined to describe the closeness between targeted vertices.

For a vertex which has only one single child, the skeleton simplification becomes a line simplification problem. We apply Douglas-Peucker method to simplify line segments as it is regarded as the most effective line simplification algorithm [28]. Since the closer the current vertex is to the line segment formed by its parent and its child, the less important this vertex is. Hence, the indicator α is computed as follow

$$\alpha = \frac{d}{r}, \tag{1}$$

where d is the distance between the current point and the line segment formed by its parent and its child; r is the distance threshold of an edge in the tree skeleton which controls the simplification process. As illustrated in Figure 5, if the indicator value α is smaller than a given threshold σ

$$\alpha \leq \sigma. \tag{2}$$

We consider the current point unimportant and therefore it can be removed from the skeleton.

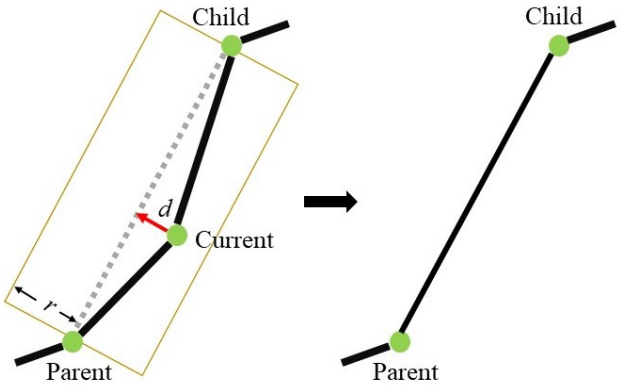


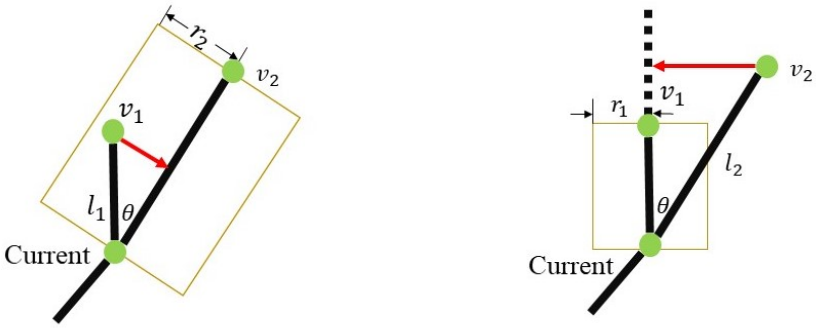
Figure 5. Single child simplification.

For a vertex having multiple children vertices, the similarity indicator α indicates how close the children vertices are, and therefore is defined as follow

$$\alpha = \min\left(\frac{l_1 \sin\theta}{r_2}, \frac{l_2 \sin\theta}{r_1}\right), \tag{3}$$

where l represents the length of the edge between one specific child vertex and its parent; θ is the angle between two edges and r is the distance threshold of a specific edge (see Figure 6). Note that

the indicator computed from different directions (i.e., from v_1 to v_2 or from v_2 to v_1) will have different values and therefore we select the minimum one to evaluate the proximity between adjacent child vertices.



(**a**) (**b**)

Figure 6. Bidirectional similarity indicators. (**a**) Indicator computed from v_1 to v_2. (**b**) Indicator computed from v_2 to v_1.

The smaller the indicator value, the more similar the two vertices. If α is smaller than a given threshold σ, we merge the pair of vertices into a new vertex. The merged new vertex position is computed as the weighted average of the original two vertices

$$p_{new} = \frac{p_1 w_1 + p_2 w_2}{w_1 + w_2}, \tag{4}$$

where p_{new} is the position of the new vertex, p_1 and p_2 are the positions of two old vertices and w represents the weight of a specific vertex, which is computed as the subtree length of the vertex, as denoted in Section 3.2.1. Figure 7 illustrates how a new vertex is created from an old pair of children vertices. After v_1 and v_2 being merged, we reassign the neighborhood relationships among the vertices which were connected to either v_1 or v_2 before. Reconstructing the structure of the tree is not necessary since the merging is a local operation within a small neighborhood and the adjacency information can be automatically updated, which maintains the tree structure.

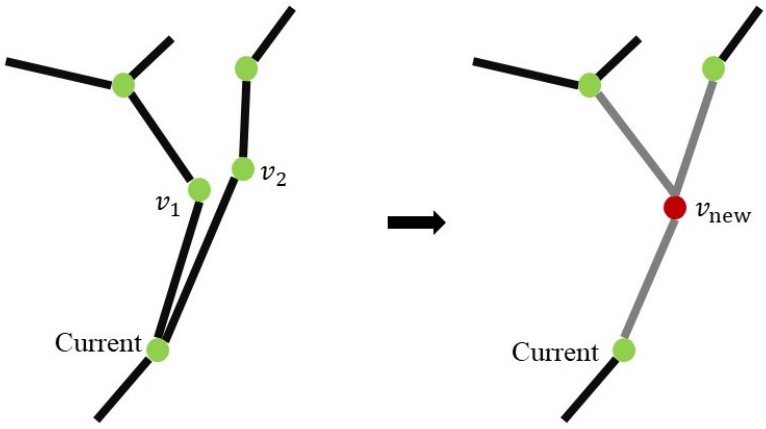


Figure 7. Multi-children simplification. Grey edges on the right indicate newly generated branches.

3.3. Branch Fitting

Based on the simplified tree skeleton, we further reconstruct the tree geometry. To precisely model the geometry of tree branches, we apply a cylinder-fitting approach. According to [29], the cylinder primitive is the most robust primitive in terms of representing the geometry of tree branches, even with holes and noises in the dataset. Moreover, compared with the complex curve fitting method, cylinder fitting is relatively easy and fast in computation [30,31]. Figure 8 shows in general how cylinder fitting is employed to obtain a tree model with branches.

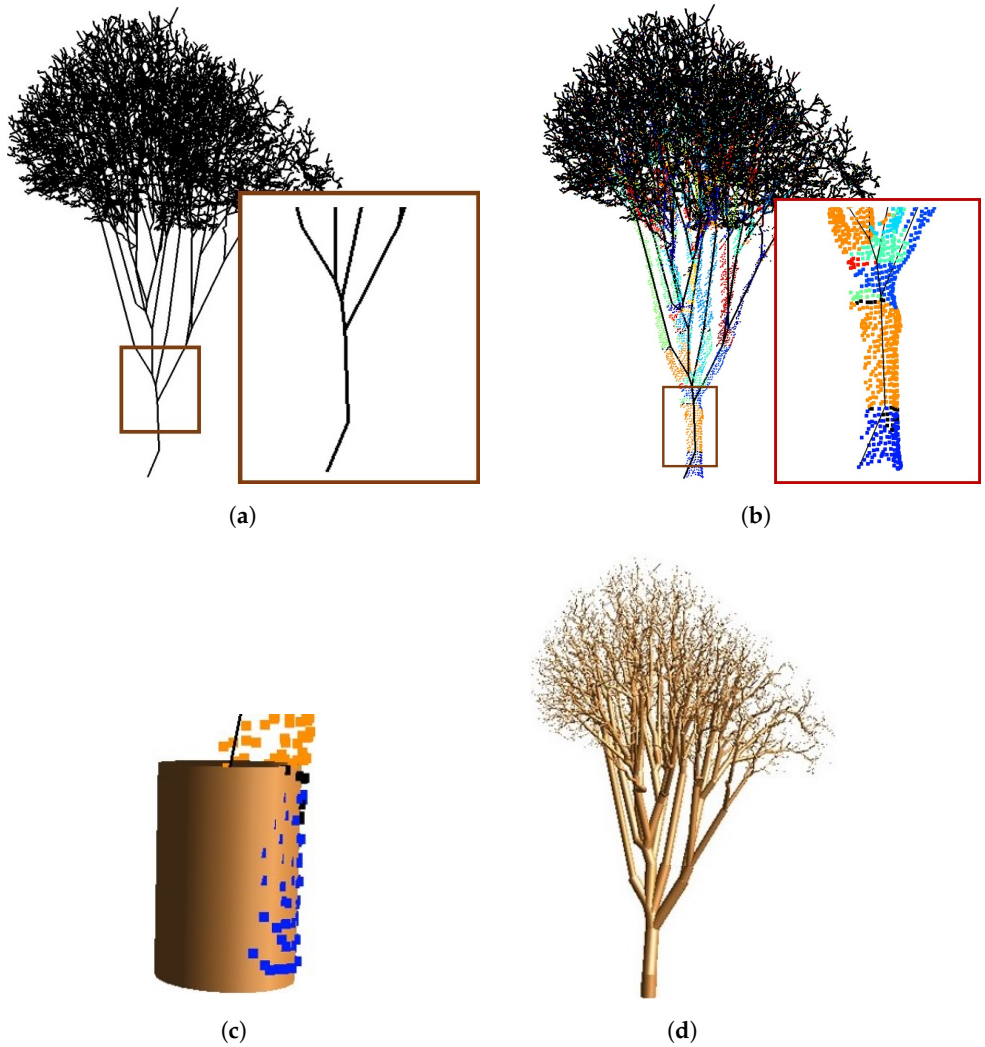


Figure 8. Branch fitting. (**a**) Tree skeleton. (**b**) Points segmented into different parts. (**c**) Cylinder fitted to the main trunk. (**d**) Geometry derived for the subsequent branches.

The main trunk close to the tree base area typically has the highest density of supportive points. We exploit an optimization-based approach to obtain accurate branch geometry. First, the neighboring points lying within the trunk part are segmented and identified (Figure 8b). We can either use a brute-force searching method or apply a kd-tree query to speed up the segmentation. Next, we fit a cylinder to approximate the branch geometry based on the corresponding trunk points. This is a typical non-linear least squares problem. We hereby define our input data, parameters to be solved, and the objective function as follows (Figure 9):

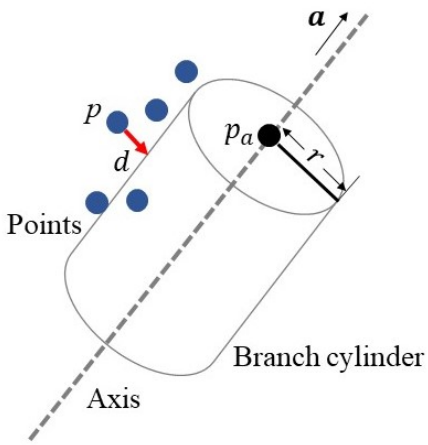


Figure 9. Parameters and objective in the cylinder fitting problem.

- ***Input data:*** position p of the input points;
- ***Parameters to be solved:*** the axis direction vector **a** of the cylinder, position p_a of the endpoint on the axis, and the radius r of the cylinder;
- ***Objective function:*** sum of squared distance d from the points to the branch cylinder, i.e.,

$$\sum_{i=1}^{n} dist(p_i), \tag{5}$$

where $dist(p_i)$ represents the distance from p_i to the branch cylinder surface. We use the Levenberg Marquardt algorithm to solve the non-linear least-squares problem [32]. Normal least-squares is sensitive to data noise and outliers. Therefore, to further improve the solution quality, we repeat the non-linear least square process and introduce weights for each point during the second iteration. We want to give heavy influence to points closer to the cylinder and relatively low influence to points that are far from the cylinder. Hence, weights are assigned according to the point's distance to the cylinder. The weight for one specific point is defined as follow

$$w_i = 1 - \frac{dist(p_i)}{dist_{max}}, \tag{6}$$

where $dist(p_i)$ represents the distance between the current i^{th} point and the cylinder obtained from the initial computation, and $dist_{max}$ is the maximum distance among all the points to the cylinder. In such a way, we normalize all point weights to the range of [0, 1]. The objective function is denoted accordingly

$$\sum_{i=1}^{n} w_i \cdot dist(p_i). \tag{7}$$

Figure 8c shows the accurate geometry of the tree trunk obtained by cylinder fitting. For the small tree branches, where points become noisier when getting close to the tree crown and branch tips, fitting an accurate cylinder is infeasible. Instead, we apply an allometric rule to obtain plausible estimates for the rest of the tree branches [9]. The radius of a branch edge is proportional to its weight, which is defined as the average of the subtree length of its two ending vertices. We compute the radius of the remaining branch edges using the following equation

$$r_{e_i} = r_t(\frac{w_i}{w_t}), \tag{8}$$

where r_{e_i} is the radius of the i^{th} branch edge; r_t is the radius of the trunk obtained by cylinder fitting and w_i is the weight of the specific i^{th} branch edge. Figure 8d shows the derived tree branch model from the constructed tree skeleton.

3.4. Adding Realism

To further add realism, we add leaves and texture to the reconstructed tree models. Since it is almost impossible for us to capture the geometry and texture characteristics of leaves from laser scans, it becomes impossible to reconstruct accurate leaves purely from the point clouds. In this work, we generate oriented leaves at the end of each branch following the method in [23]. Figure 10 shows the final reconstruction.

(**a**) Tree branches (**b**) Final tree model

Figure 10. Adding leaves and texture.

3.5. Implementation Details

We implemented our tree reconstruction algorithm in C++. We select C++ because it enables high computation efficiency, also because many necessary libraries on graph algorithms and 3D model rendering are available in C++. Boost Graph Library [33] is used for minimum spanning tree extraction and Easy3D [34] is used for visualization. Besides, tools like Mapple, which is a tool for visualizing and editing 3D point clouds [35], are used for pre-processing and segmentation of individual tree point clouds.

3.6. Test Datasets

To develop and test the proposed tree reconstruction method, 5 point cloud datasets have been collected. These test datasets contain point clouds from publicly available point cloud repositories, the Floriade Project of Almere, and the AHN dataset [36]. These point clouds include various tree shapes and types. Following the work of [37], we classify the trees into three complexity categories which include easy, medium, and difficult, based on the tree stem density and understory structure. Also, different sensors are covered, i.e., static laser scans, mobile laser scans, and airborne laser scans.

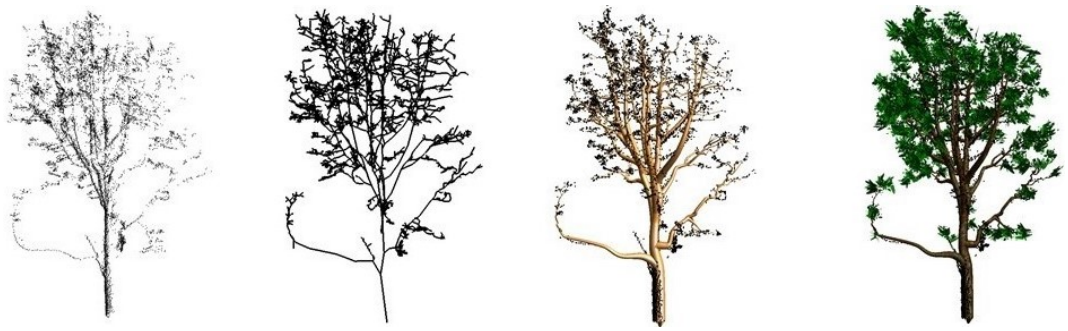
4. Results and Discussion

In this section, we provide the result analysis, aiming to test if our modelling results have fulfilled the functional and user requirements proposed in Section 3. First, a set of visual results are presented to evaluate the topological fidelity of the reconstructed tree models. Then, we compute the distance between input points and the output tree branch model to verify the geometrical accuracy of the modelling results. In Section 4.3, we illustrate the robustness and applicability of our algorithm over

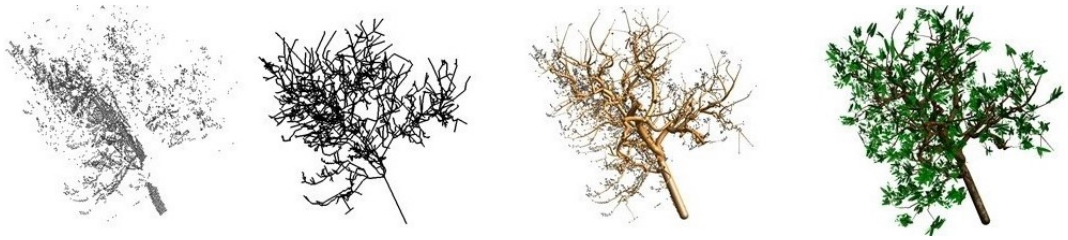
various tree types and data sources, which enables our tree modelling process to be fully automatic. Some discussions considering comparisons with the state-of-the-art methods, limitations, and future applications are made at the end of this section.

4.1. Visual Evaluation

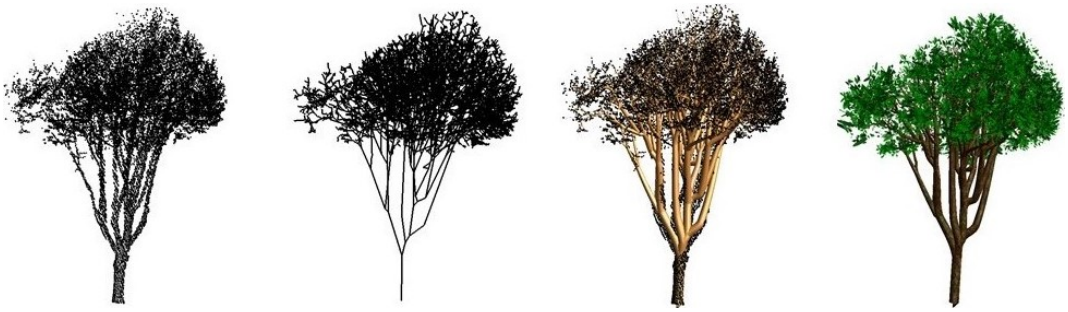
We reconstructed a variety of trees of different species, sizes, and branch structure. Figure 11a shows a vertical and slim tree with relatively simple branch structure, while Figure 11b gives an example of another tree with a tilted stem and complex branch structure. The reconstruction results of these two trees demonstrate that our method is capable of processing trees with different shapes and structures, which benefits from the skeleton-based approach that we adopt.



(**a**)



(**b**)



(**c**)

Figure 11. *Cont.*

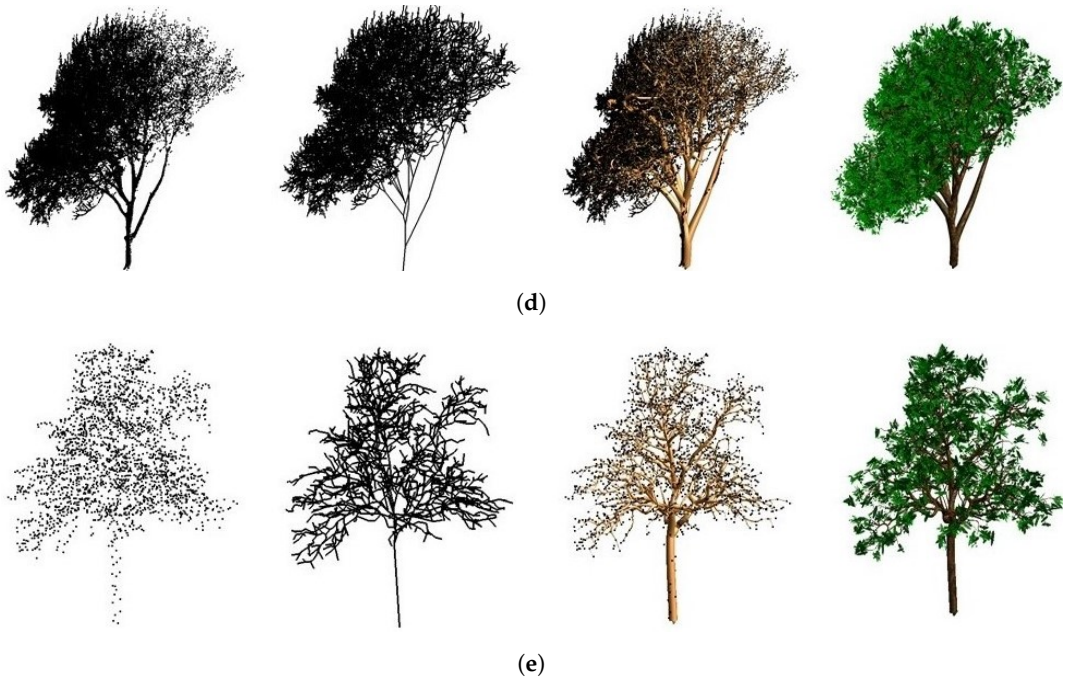


Figure 11. Five different trees (from (a) to (e)) reconstructed using our method. From left to right: point cloud, skeleton, tree branches, and final model.

Besides, we also tested our method on scanned trees from various data sources, including mobile scanning, static scanning as well as airborne scanning. It is observed that point clouds collected by mobile scanning (Figure 11c) or static scanning (Figure 11d) have a high quality and thus were all accurately reconstructed. On the other hand, Figure 11e gives an example of an input point cloud obtained by airborne scanning, which is poorly sampled and is quite sparse. Even for such a low-quality input, our approach is still able to produce a visually plausible 3D reconstruction.

4.2. Reconstruction Accuracy

We quantified the geometrical accuracy of the modelling results by computing the mean distance between the input points and the generated tree branch model [9]. The reconstruction accuracy and standard deviation in Table 1 suggested that overall our approach can generate tree models that fit closely to the input point cloud data and thus ensures high geometrical accuracy. When it comes to the individual tree level, typically a short tree with highly dense points will have a more accurate modelling result. Also, compared to trees with irregular shapes (i.e., Figure 11b), trees with a compact and standard shape usually enable higher reconstruction accuracy.

Table 1. Statistics on the tree examples shown in Figure 11. This table summarizes the height, complexity, number of points, data source of the trees and the accuracy (mean distance from the points to the surface of the reconstructed models) and the standard deviation.

Figure	Height (m)	Complexity	Point Number	Sensor Type	Accuracy (cm)	Stardard Deviation
Figure 11a	5.52	Medium	11,855	Mobile scanning	2.76	2
Figure 11b	9.87	Medium	6992	Mobile scanning	10.04	8
Figure 11c	15.99	Difficult	28,993	Mobile scanning	6.59	6
Figure 11d	21.73	Difficult	137,407	Static scanning	6.50	6
Figure 11e	13.13	Easy	2488	Airborne scanning	11.88	7

Figure 12 visualizes the per-point error distribution of the reconstructed 3D model shown in Figure 11a. In the visualization, the blue colour indicates a low error value and the red colour indicates

a high error value. From such a visualization, we can conclude that points lying within the main branches typically fit closer to the model, while points near the branch tips usually have high error values. This indicates that our method can generate highly precise main branch structure of the input tree (thanks to the non-least squares based branch fitting). However, points are getting sparser near the branch tips and thus not sufficient to reliably reconstruct these small features from the under-sampled data.

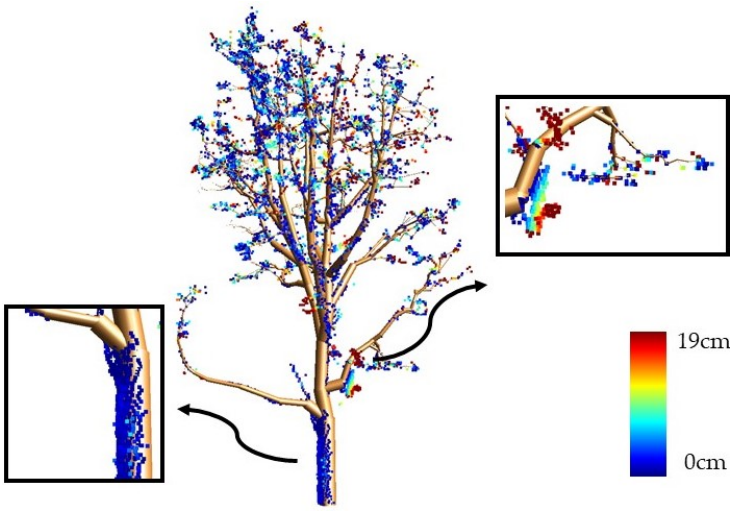
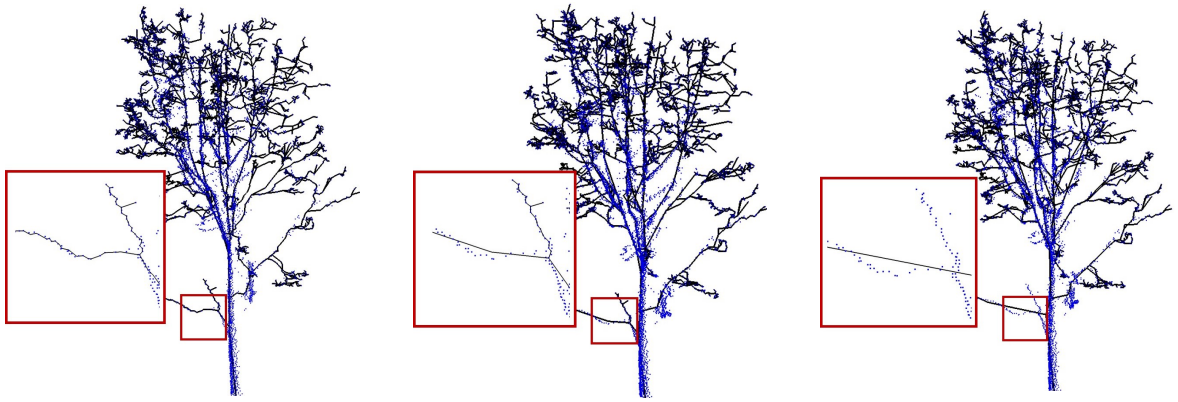


Figure 12. Visualization of the per-point error distribution.

4.3. Robustness

As described in Section 3, the simplification threshold σ is introduced during the tree skeleton simplification process, where we utilize an indicator to measure the proximity between adjacent vertices. This section discusses how different parameter values influence the modelling results, based on which, we choose the threshold values that best fit our methodology.

The simplification threshold σ controls the similarity indicator α, which determines the relative proximity between adjacent vertices. We tested the value of σ from 0.5 to 3 and the results are shown in Figure 13. According to our experiments, a very small threshold value for σ for the indicator makes it difficult to merge close-by vertices, while a very big σ causes oversimplification. Therefore, we chose 1.5 as the threshold value. It is denoted that the parameter value is pre-fixed in our algorithm, which means that we used the same parameter setting for generating all the 3D models in this paper. As σ is a relative value indicating the closeness among vertices, it is generally applicable for most trees. Users do not have to adjust the specific threshold value for specific input data, which makes our approach robust to various trees.



(**a**) $\sigma = 0.5$ (**b**) $\sigma = 1.5$ (**c**) $\sigma = 3$

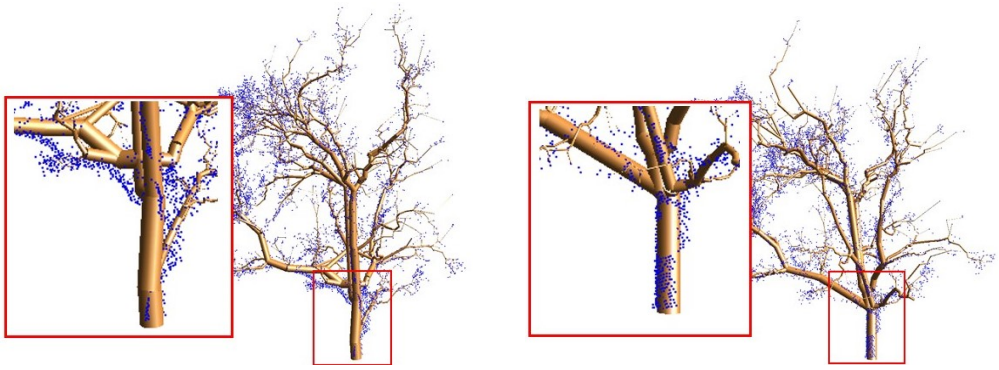
Figure 13. Simplification results using different σ values.

4.4. Comparisons

We compared our method to a few state-of-the-art approaches [15,16,21,23].

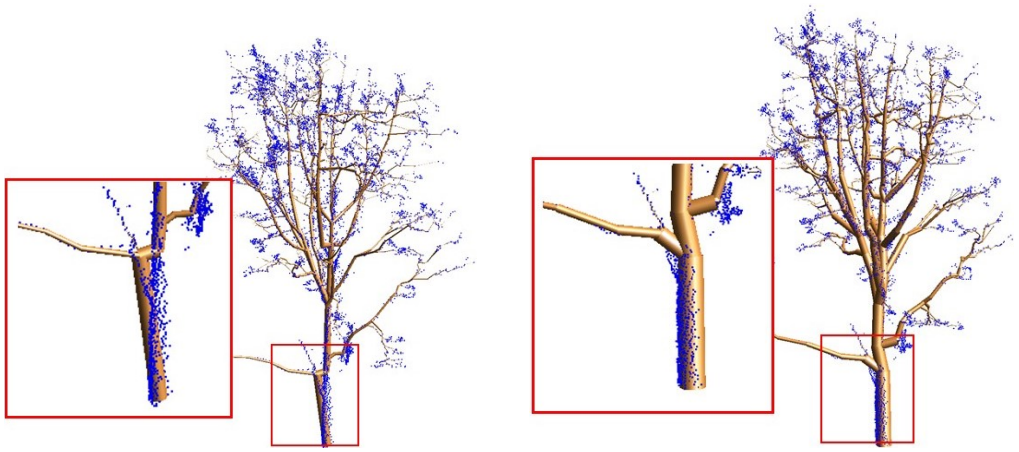
In Figure 14, we show a visual comparison of our results against the method described in [23] as it is most related to our work. It can be seen that our reconstructed models have more faithful branch structure and also fit better to the input points. Given the same point cloud, our algorithm is capable of reconstructing a tree model with higher topological and geometrical accuracy. The performance gain benefits from two improvements. Firstly, we identify and centralize main-branch points, which in return generates topologically correct tree skeletons. Secondly, our cylinder fitting exploits a distance-weighted non-linear least squares fitting, which significantly improves the geometrical accuracy.

We also compare our modelling results to other tree modelling approaches publically accessible to us, such as PypeTree [21], TreeQSM [16], and SimpleTree [15]. A visual comparison is demonstrated in Figure 15. Among these approaches, PypeTree aims to only give a rough description of the tree topological structure, which is not intended to recover the branch geometry. In contrast, TreeQSM, SimpleTree, and our method can recover the geometry of tree branches. However, TreeQSM cannot ensure consistent recovery of the tree branches. The reconstruction using SimpleTree is quite tedious as it requires user input of the key parameters such as the radii of the branches. In our comparison, we also observed that SimpleTree requires nearly perfect input, i.e., complete point clouds, which is rarely satisfied in practice. In the test data, the tree was scanned from a single viewpoint by a terrestrial laser scanner and thus the surfaces of the branches were partially sampled. SimpleTree failed to detect the branch cylinders and to recover the branch geometry. In contrast, our method successfully and faithfully recovered both the topology and the geometry of the tree branches.



(**1**) The result of [23] (**2**) Ours

(**a**) Tree 1



(**1**) The result of [23] (**2**) Ours

(**b**) Tree 2

Figure 14. Comparison between Livny's method [23] and our method demonstrated on two trees.

(**a**) PypeTree (**b**) TreeQSM (**c**) Ours

Figure 15. Comparison between PypeTree [21], TreeQSM [16], and our method.

4.5. Limitations

Our algorithm can successfully obtain accurate and detailed 3D tree models from point clouds. However, it still has some limitations. First of all, our approach depends on the quality of the input data. For poorly scanned data with sparse points, our method can reconstruct a plausible topological structure of the tree branches but is unable to achieve sufficient geometrical accuracy. Moreover, our work does not consider natural growing rules of tree branches (e.g., branch split angle, branch growing length). The incorporation of domain knowledge will further constrain the reconstructed models to be topologically correct and improve the fidelity of the models, improving both geometrical and topological accuracy.

4.6. Potential Applications

Our proposed approach enables accurate reconstruction of 3D trees from point clouds. The generated tree models can be employed in many applications. The accurate tree models with synthesized leaves and textures can be directly applied in applications such as urban landscape design and entertainment, to convey the realism of the scenes. Our method also opens up the opportunity for automatically obtaining precise tree attributes (e.g., the height of a tree, the thickness of the trunks, and the diameter at a specified height). It enables to save a significant amount of time and laboring efforts compared to the traditional manual measuring method. With accurate 3D tree models, implicit tree attributes, such as wood volume, biomass, the amount of carbon dioxide to be absorbed and oxygen to be emitted, can also be automatically derived or estimated.

5. Conclusions and Future Work

In this paper, we proposed an automatic approach to accurately reconstruct 3D tree branches from point clouds. During the reconstruction, both the geometrical accuracy and topological fidelity of the tree are taken into consideration. One novelty of our work is that we aid the skeleton construction process by the main-branch point centralization, which contributes to improving the quality of the generated tree branch structure. Moreover, an optimization-based approach is employed to accurately reconstruct the geometry of the tree branches. Experimental results revealed that our method is robust in dealing with various types and sizes of trees. As long as the input point clouds demonstrate clear branch structure, our method is capable of generating tree models of high quality.

In future work, we would like to perform automatic segmentation of trees. As our method only works for individual tree point clouds, involving existing automatic segmentation approaches will expand our algorithm to a broader range of applications. Besides, as there are many irregular shapes of tree branches in nature, we will further consider fitting free-form surfaces instead of cylinders to model the branch geometry more precisely.

Author Contributions: S.D. performed the study and implemented the algorithms. R.L., H.L., and J.S. provided constructive comments and suggestions. L.N. proposed this topic and provided daily supervision.

Funding: This research received no external funding.

Acknowledgments: We thank Yufu Zang, Kaixuan Zhang, and Agung Indrajit for valuable comments. We also thank the Floriade Project for providing test datasets.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LiDAR	Light Detection and Ranging
MST	Minimum Spanning Tree
PCA	Principal Component Analysis
QSM	Quantitative Structure Modelling
SFM	Structure From Motion

References

1. Deussen, O.; Hanrahan, P.; Lintermann, B.; Měch, R.; Pharr, M.; Prusinkiewicz, P. Realistic modeling and rendering of plant ecosystems. In Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, Orlando, Florida, USA, 19–24 July 1998; pp. 275–286.
2. Maltamo, M.; Næsset, E.; Vauhkonen, J. Forestry applications of airborne laser scanning. *Concept Case Stud. Manag. For Ecosys* **2014**, *27*, 460.
3. Ke, Y.; Quackenbush, L.J. A review of methods for automatic individual tree-crown detection and delineation from passive remote sensing. *Int. J. Remote Sens.* **2011**, *32*, 4725–4747.
4. Hyyppa, J.; Kelle, O.; Lehikoinen, M.; Inkinen, M. A segmentation-based method to retrieve stem volume estimates from 3-D tree height models produced by laser scanners. *IEEE Trans. Geosci. Remote Sens.* **2001**, *39*, 969–975.
5. Kamal, M.; Phinn, S.; Johansen, K. Object-based approach for multi-scale mangrove composition mapping using multi-resolution image datasets. *Remote Sens.* **2015**, *7*, 4753–4783.
6. Reche-Martinez, A.; Martin, I.; Drettakis, G. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Gr. (ToG)* **2004**, *23*, 720–727.
7. Shlyakhter, I.; Rozenoer, M.; Dorsey, J.; Teller, S. Reconstructing 3D tree models from instrumented photographs. *IEEE Comput. Gr. Appl.* **2001**, *21*, 53–61.
8. Quan, L.; Tan, P.; Zeng, G.; Yuan, L.; Wang, J.; Kang, S.B. Image-based plant modeling. *ACM Trans. Gr. (TOG)* **2006**, *25*, 599–604.
9. Guo, J.; Xu, S.; Yan, D.M.; Cheng, Z.; Jaeger, M.; Zhang, X. Realistic Procedural Plant Modeling from Multiple View Images. *IEEE Trans. Vis. Comput. Gr.* **2018**, doi:10.1109/TVCG.2018.2869784.
10. Liang, X.; Kankare, V.; Hyyppä, J.; Wang, Y.; Kukko, A.; Haggrén, H.; Yu, X.; Kaartinen, H.; Jaakkola, A.; Guan, F.; others. Terrestrial laser scanning in forest inventories. *ISPRS J. Photogramm. Remote Sens.* **2016**, *115*, 63–77.
11. Olofsson, K.; Holmgren, J.; Olsson, H. Tree stem and height measurements using terrestrial laser scanning and the RANSAC algorithm. *Remote Sens.* **2014**, *6*, 4323–4344.
12. Brandtberg, T.; Warner, T.A.; Landenberger, R.E.; McGraw, J.B. Detection and analysis of individual leaf-off tree crowns in small footprint, high sampling density lidar data from the eastern deciduous forest in North America. *Remote Sens. Environ.* **2003**, *85*, 290–303.
13. Holmgren, J.; Persson, Å. Identifying species of individual trees using airborne laser scanner. *Remote Sens. Environ.* **2004**, *90*, 415–423.
14. Wang, D.; Hollaus, M.; Puttonen, E.; Pfeifer, N. Automatic and self-adaptive stem reconstruction in landslide-affected forests. *Remote Sens.* **2016**, *8*, 974.
15. Hackenberg, J.; Spiecker, H.; Calders, K.; Disney, M.; Raumonen, P. SimpleTree—An efficient open source tool to build tree models from TLS clouds. *Forests* **2015**, *6*, 4245–4294.
16. Raumonen, P.; Kaasalainen, M.; Åkerblom, M.; Kaasalainen, S.; Kaartinen, H.; Vastaranta, M.; Holopainen, M.; Disney, M.; Lewis, P. Fast automatic precision tree models from terrestrial laser scanner data. *Remote Sens.* **2013**, *5*, 491–520.
17. Bucksch, A.; Lindenbergh, R.; Menenti, M. SkelTre. *Vis. Comput.* **2010**, *26*, 1283–1300.
18. Yan, D.M.; Wintz, J.; Mourrain, B.; Wang, W.; Boudon, F.; Godin, C. Efficient and robust reconstruction of botanical branching structure from laser scanned points. In Proceedings of the 2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics, Huangshan, China, 19–21 August 2009; pp. 572–575.
19. Xu, H.; Gossett, N.; Chen, B. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Gr. (TOG)* **2007**, *26*, 19.
20. Verroust, A.; Lazarus, F. Extracting skeletal curves from 3D scattered data. In Proceedings of the Shape Modeling International'99. International Conference on Shape Modeling and Applications, Aizu-Wakamatsu, Japan, 1–4 March 1999; pp. 194–201.
21. Delagrange, S.; Jauvin, C.; Rochon, P. PypeTree: A tool for reconstructing tree perennial tissues from point clouds. *Sensors* **2014**, *14*, 4271–4289.
22. Dey, T.K.; Sun, J. Defining and computing curve-skeletons with medial geodesic function. *Symp. Geom. Process.* **2006**, *6*, 143–152.

23. Livny, Y.; Yan, F.; Olson, M.; Chen, B.; Zhang, H.; El-Sana, J. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Trans. Gr. (TOG)* **2010**, *29*, 151.
24. Xu, Y.; Sun, Z.; Hoegner, L.; Stilla, U.; Yao, W. Instance Segmentation of Trees in Urban Areas from MLS Point Clouds Using Supervoxel Contexts and Graph-Based Optimization. In Proceedings of the 2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing, Beijing, China, 19–20 August 2018; pp. 1–5.
25. Zhou, H.; Shenoy, N.; Nicholls, W. Efficient minimum spanning tree construction without Delaunay triangulation. In Proceedings of the 2001 Asia and South Pacific Design Automation Conference, Yokohama, Japan, 2 February 2001; pp. 192–197.
26. Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **1995**, *17*, 790–799.
27. Chi, Y.; Muntz, R.R.; Nijssen, S.; Kok, J.N. Frequent subtree mining–an overview. *Fundam. Inf.* **2005**, *66*, 161–198.
28. Wu, S.T.; Marquez, M.R.G. A non-self-intersection Douglas-Peucker algorithm. In Proceedings of the 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003), Sao Carlos, Brazil, 12–15 October 2003; pp. 60–66.
29. Markku, Å.; Raumonen, P.; Kaasalainen, M.; Casella, E. Analysis of geometric primitives in quantitative structure models of tree stems. *Remote Sens.* **2015**, *7*, 4581–4603.
30. Panyam, M.; Kurfess, T.R.; Tucker, T.M. Least squares fitting of analytic primitives on a GPU. Journal of Manufacturing Systems. **2008**, 27(3), 130–135.
31. Nurunnabi, A.; Sadahiro, Y.; Lindenbergh, R.; Belton, D. Robust cylinder fitting in laser scanning point cloud data. *Measurement* **2019**, *138*, 632–651.
32. Marquardt, D.W. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.* **1963**, *11*, 431–441.
33. Boost. Available online: https://www.boost.org/doc/libs/1_66_0/libs/graph/doc/index.html (accessed on 1 September 2018).
34. Easy3D. Available online: https://github.com/LiangliangNan/Easy3D (accessed on 1 March 2019).
35. Mapple. Available online: https://3d.bk.tudelft.nl/liangliang/software.html (accessed on 1 September 2018).
36. AHN Dataset. Available online: https://www.pdok.nl/attenderingsservice-rss/-/asset_publisher/mvZkjafth739/content/actueel-hoogtebestand-nederland-ahn3- (accessed on 1 January 2019).
37. Zhang, W.; Wan, P.; Wang, T.; Cai, S.; Chen, Y.; Jin, X.; Yan, G. A Novel Approach for the Detection of Standing Tree Stems from Plot-Level Terrestrial Laser Scanning Data. *Remote Sens.* **2019**, *11*, 211.