Peer-reviewed version available at *Remote Sens.* **2019**, *11*, 2074; doi:10.3390/rs11182074

*Article*

# 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, Delft, Netherland; dushenglan940128@163.com, h.ledoux@tudelft.nl, j.e.stoter@tudelft.nl, Liangliang.Nan@tudelft.nl

[2] Spatial Laser Scanning Lab, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, Netherland; r.c.lindenbergh@tudelft.nl

* Correspondence: Liangliang.Nan@tudelft.nl

**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. Nevertheless, some works don't ensure 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 tree branches from laser scans. We first extract an initial tree skeleton from the input tree point cloud, then simplify the skeleton through iteratively removing redundant components. A global-optimization 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 resulted tree models can be further applied in the precise estimation of tree attributes, urban landscape visualization, etc.

**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 entertainment. While applications such as landscape design and visualization only require modelling virtual trees, lots of other applications relevant with ecological modelling and forestry management require accurate measuring of tree parameters (e.g., height, stem thickness). Accurate tree modelling not only enhances the realism within a scene, but also provides promising approaches to scientifically manage vegetations and forests, which will in return contribute a lot 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 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, the remote-sensing technology has been widely exploited in mapping various information on forests and plants [5]. Both satellite sensors and airborne sensors can effectively acquire digital images with high spatial resolution, and that provides viable data sources for forestry analysis on the individual tree level [3]. Moreover, with the development of digital image processing technologies, researchers had tried to reconstruct 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. Combining plant images with sparse point clouds obtained from Structure From Motion (SFM), Quan et al. [8] reconstructed realistic plants with generic leaves with user interactions. While the works mentioned above can produce impressive modelling results, they don't 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, the LiDAR (Light Detection and Ranging) technology has been widely used in forestry-related analysis and studies. As measurements from LiDAR can achieve millimetre-level of details from the objects, it becomes possible to directly capture 3D information and rapidly estimate the trees attributes [10]. For example, LiDAR measurements are widely applied in further 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 skeleton are required to be taken account of. The works of [14,15] adopted a cylinder-fitting approach to obtain the geometry of tree branches. To further extract the tree skeletal structure, some works employ a rule-based procedural modelling approach to synthesize branches [9,16], which will generate the tree skeleton with high quality but requires prior knowledge as well as manual parameters adjustments. Some other works proposed purely data-driven methods to automatically extract the skeleton without requiring additional user interactions. The work of [17] constructed the shortest-path map over the input point clouds to extract consecutive skeletal curves. Served as an alternative for the shortest-path mapping, Dey and Sun [18] utilized the medial axis to represent the skeletal structure of 3D tree-like objects. Instead of extracting skeleton curves directly from point clouds, Bucksch et al. [19] applied another method that organizes points into an octree structure and generates skeletal curves from the octree cells. Similarly, Yan et al. [20] also applied a K-means clustering-based approach to generate tree skeletons. However, these works highly rely on the quality of the input data and therefore may not be robust enough to data quality issues such as holes or missing data due to occlusions. Following the work of [17], Livny et al. [21] 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. We assume that each tree is segmented out from the point clouds. Our method employs the Minimum Spanning Tree (MST) algorithm to effectively extract the initial tree skeleton over input points. By iterative skeleton simplification and cylinder fitting, we obtain the tree model with reconstructed branches. Leaves and textures are further added to enhance the realism of the tree model (Figure 1). One novelty of our work is that we construct the initial tree skeleton based on the desired characteristics 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 reconstruction approach demonstrates both the geometrical correctness and the topological fidelity of the generated tree models.

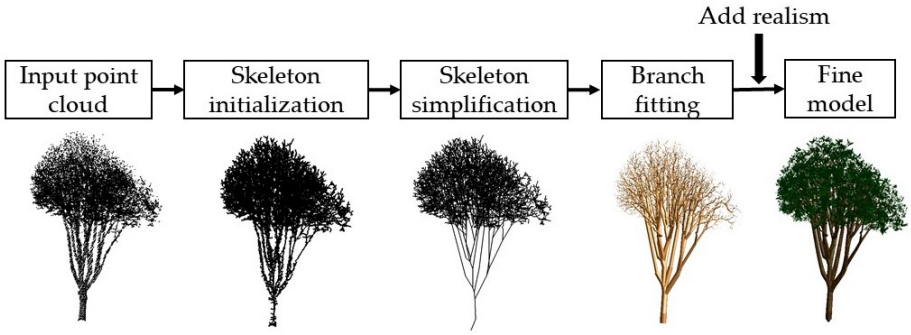



**Figure 1.** An overview of the proposed methodology.

## 2. Materials and Methods

Our input data is the point cloud of a single tree, which typically contains noises and outliers, but is expected to convey the major branch structure of a tree. Figure 1 illustrates the overall pipeline of our method, which consists of the following major steps:

- ***Skeleton initialization.*** We triangulate the tree 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 textures to enhance realism.

### *2.1. Skeleton Initialization*

Based on the fact that points close to each other are likely to belong to the same branch, we construct a MST graph over the input point cloud to represent the initial tree skeleton. To extract a 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 [22]. Additionally, it helps to complete the missing region or incomplete branches, which ensures the robustness of our method to the input point clouds with 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 MST constructed 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 fat shape typically have scattered points and branches, which leads to the computed MST growing in a horizontal manner rather than a compact vertical manner.

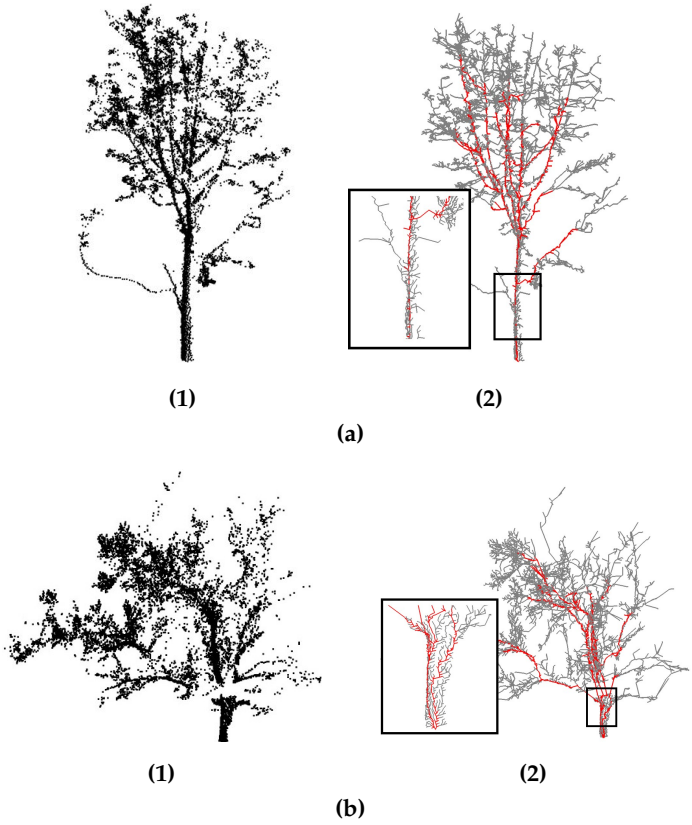



**Figure 2.** Skeleton initialization for two trees. (a) A valid tree skeleton structure (in red). (b) An invalid tree skeleton structure (in red).

We address the 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. As points near the bifurcations or the branch tips typically sharply change in terms of density, while points within a single branch share a stable density (Figure 3a), we can find main-branch points with a relatively stable density in their neighbors. Identified main-branch points are centralized through the mean-shift algorithm [23]. As illustrated in Figure 3b, the extracted skeleton after main-branch point centralization has a compact vertical growing manner.

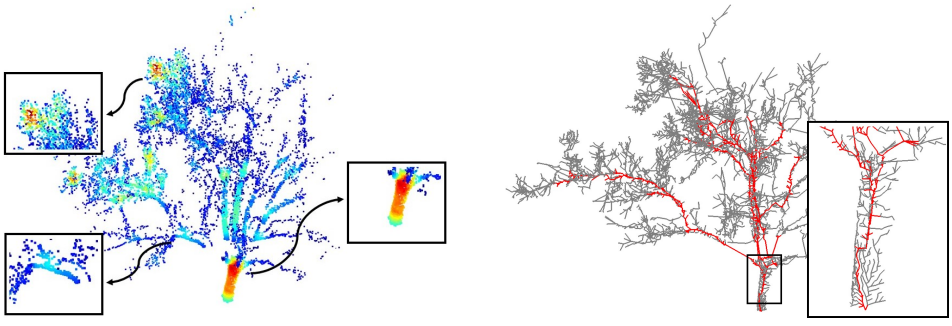
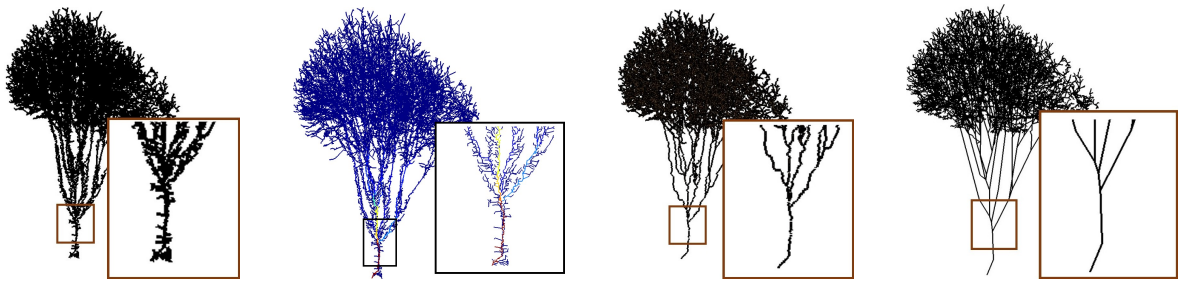



**Figure 3.** Skeleton extraction from the centralized points. (a) Density map of the raw point cloud. Red indicates high density and blue indicates low density. (b) Skeleton extracted after the main-branch point centralization.

### 2.2. Skeleton Simplification

The initial tree skeleton extracted from the input point cloud has a large amount of redundant vertices and edges. Most of the redundant vertices and small edges don't 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 accordingly. Then, we iteratively check the proximity between adjacent vertices and merge close vertices. Figure 4 illustrates the simplification process.



(a) (b) (c) (d)

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

#### 2.2.1. 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. Several previous works [9] suggest utilizing the point density, together with the point orientation vector extracted from the Principal Component Analysis (PCA), to indicate the importance of the vertex. However, the weights evaluated in such a way are significantly dependent on the quality of the scanned points and thus become unreliable when encountered with poor scanning issues.

Instead of weighting from the density, we weight each vertex according to its length of the subtree, which is computed as the sum of the length of all edges within the subtree of the vertex. Through 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 points 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 [21], while near the tree base small branches have drastically small weights compared to the main trunk branches. Such a characteristic helps us clear away noisy branches at the trunk, and at the same time, keeps the small leave branches at the crown.

#### 2.2.2. Simplifying adjacent vertices and edges

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

For vertex which has only one single child, the skeleton simplification becomes a line simplification problem. Since that the more proxy the current vertex is to the line segment formed by its parent and its child, the less important this vertex is. Hence, the indicator $\alpha$ 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 $\alpha$ is smaller than a given threshold $\sigma$:

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

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

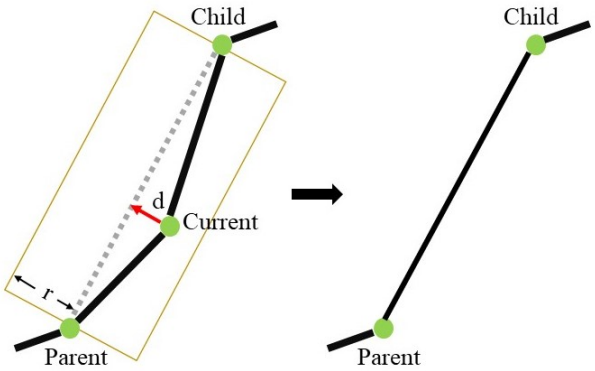



**Figure 5.** Single child simplification.

For vertex which has multiple children vertices, the similarity indicator $\alpha$ indicates the closeness of the pair of two children vertices, 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; $\theta$ 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. The smaller the indicator value, the more similarity between two vertices. If $\alpha$ is smaller than a given threshold $\sigma$, we merge the pair of vertices into a new vertex. The merged new vertex location $p_{new}$ is computed as the weighted average of the original two vertices, where the weight of each vertex is computed as the length of its subtree:

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

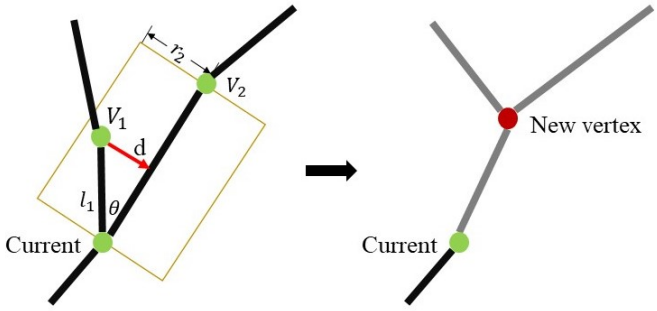
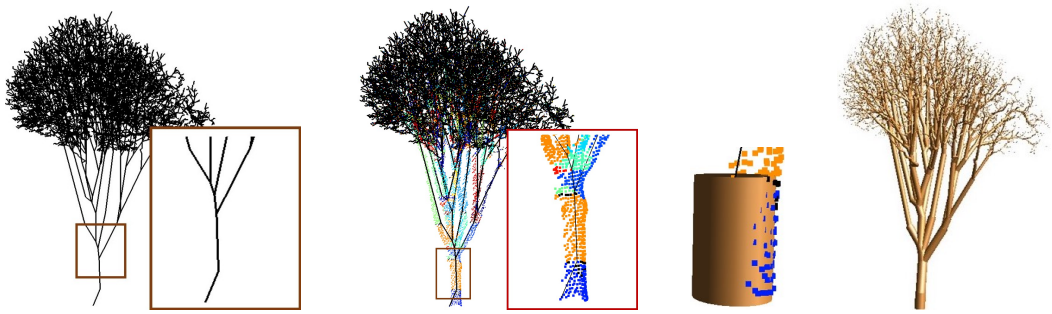



**Figure 6.** Multi-children simplification. Grey edges on the right indicates newly generated branches.

### *2.3. 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 [24], the cylinder primitive is the most robust primitive in terms representing the geometry of the 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 [25]. Figure 7 shows in general how cylinder fitting is employed to obtain the tree model with branches.



**(a)** **(b)** **(c)** **(d)**

**Figure 7.** Branch fitting. (a) Tree skeleton. (b) Points segmented to different tree 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 the accurate branch geometry. First, the neighboring points lying within the trunk part are segmented and identified (Figure 7b). 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 8):

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 July 2019 doi:10.20944/preprints201907.0058.v1

Peer-reviewed version available at Remote Sens. 2019, 11, 2074; doi:10.3390/rs11182074



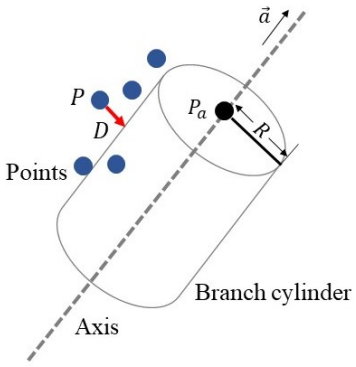



**Figure 8.** Parameters and objective in the cylinder fitting problem.

- ***Input data*** are the position $P$ of the input points;
- ***Parameters to be solved*** are the axis direction vector $\vec{a}$ of the cylinder, the position $P_a$ of the end point on the axis, and the radius $R$ of the cylinder;
- ***Objective function*** is defined as the sum of the squared distance from the points to the branch cylinder:

$$\min \sum_{i=1}^{n} D(p)_i. \tag{5}$$

We use the Levenberg Marquardt algorithm to solve the non-linear least-squares problem. Normal least-squares is sensitive to data noises and outliers. Therefore, to further improve the solution quality, we repeat the non-linear least square process and introduce the weights for each point during the second computing iteration. We want to give heavy influences to the points closer to the cylinder and relatively low influence to the 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:

$$w_i = 1 - \frac{D_i}{D_{max}}, \tag{6}$$

where $D_i$ represents the distance between the current $i_{th}$ point and the cylinder obtained from the initial computation, and $D_{max}$ is the maximum distance among all the points to the cylinder. Through such a way, we normalize all the weights of the points to the range between 0 and 1. The objective function is denoted accordingly:

$$\min \sum_{i=1}^{n} D(p)_i w_i. \tag{7}$$

Figure 7c shows the accurate geometry of the tree trunk obtained from cylinder fitting. For the sequent small tree branches, as the points become noisier when getting close to the tree crown and branch tips, fitting an accurate cylinder to small branches 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 rest 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 from cylinder fitting and $w$ is the weight of the specific branch edge. Figure 7d shows the derived tree branch model from the constructed tree skeleton.

### 2.4. Adding Realism

To further add realism, we add leaves and texture to the reconstructed tree models. Since it's 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 that of [21].

## 3. Results and Discussion

We implemented our tree reconstruction algorithm using C++. We use Boost [26] for minimum spanning tree extraction and Easy3D [27] for visualization.

### 3.1. Test Datasets

To develop and test the proposed tree reconstruction method, several 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 [28]. These point clouds include various tree species and types. Also, a wide range of data sources is covered, i.e., static laser scans, mobile laser scans, and airborne laser scans.

### 3.2. Visual Evaluation

We reconstructed a variety of trees with different species and branch structures (see Figure 9a and Figure 9b). Among them, some trees are tall and slim (Figure 9a), while others are short and fat (Figure 9b). From these results, we can see that our method is capable of processing various type of trees with different sizes. Besides, we also tested our method on scanned trees from various data sources (see Figure 9c, Figure 9d and Figure 9e), including mobile scanning, static scanning as well as airborne scanning. It is observed that point clouds collected by mobile scanning (Figure 9c) or static scanning (Figure 9d) have a high quality and thus were all accurately reconstructed. On the other hand, from Figure 9e, we can see that the input point cloud airborne scanning are poorly sampled and is extremely sparse. With such a low quality input, our approach is still able to produce a plausible 3D reconstruction.

### 3.3. Reconstruction Accuracy

We evaluated the geometrical accuracy of the modelling results by computing the mean distance between the input points and the generated tree branch model. The statistics in Table 1 suggested that our approach can generate tree models that fit closely to the input point cloud data and thus ensures high geometrical accuracy.

**Table 1.** Geometrical accuracy evaluation

| | Figure 9a | Figure 9b | Figure 9c |
|---|---|---|---|
| Height | 5.61m | 10.05m | 16.28m |
| Mean Distance | 2.76cm | 10.04cm | 6.59cm |
| Standard Deviation | 2cm | 8cm | 6cm |

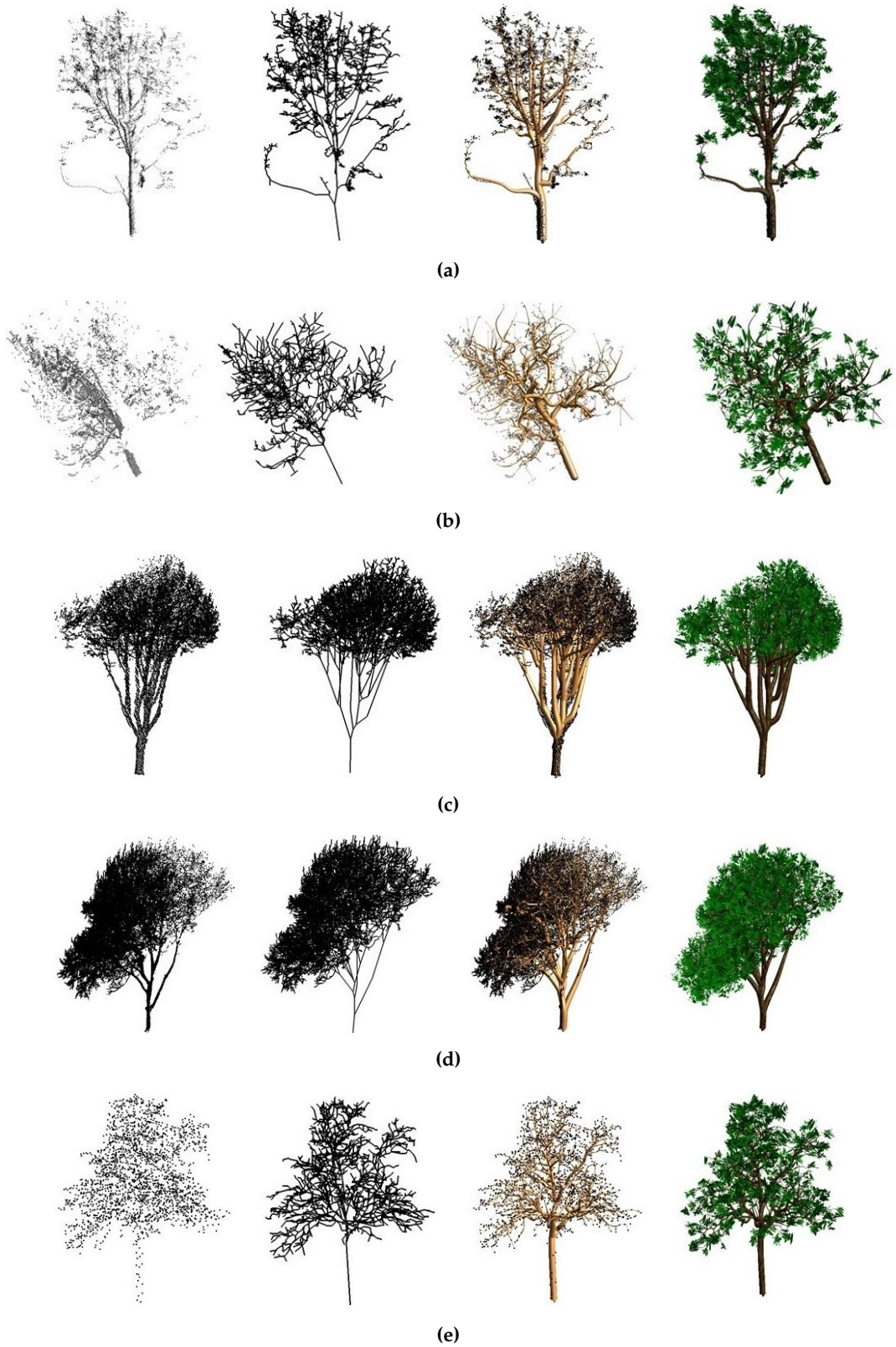
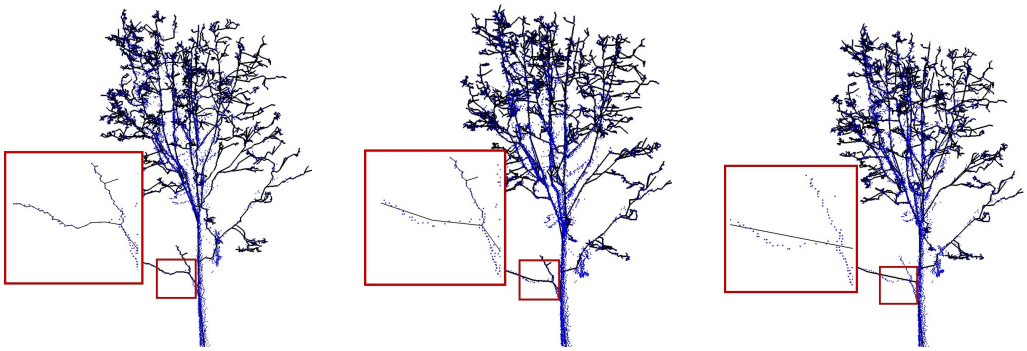


**Figure 9.** Reconstructed models for various trees. From left to right: point cloud; skeleton; tree branches; tree final model.

### 3.4. Robustness

As described in section 2, the simplification threshold $\sigma$ is introduced during the tree skeleton simplification process, where we utilize an indicator to measure the proximity between the 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 $\sigma$ controls the similarity indicator $\alpha$, which determines the relative proximity between adjacent vertices. We tested the value of $\sigma$ from 0.5 to 3 and the results are shown in Figure 10. According to our experiments, a very small threshold $\sigma$ for the indicator makes it tough for close vertices to merge, while a very big $\sigma$ 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 $\sigma$ is a relative value indicating the closeness among vertices, it is generally applicable for most trees. Users don't have to adjust the specific threshold value for specific input data, which makes our approach robust to various trees.



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

**Figure 10.** Simplification results using different $\sigma$ values. (a) $\sigma = 0.5$. (b) $\sigma = 1.5$. (c) $\sigma = 3$.

### 3.5. Comparisons

We compare our modelling results to that of [21] as it's the closest related to our work. Given the same point cloud, our algorithm is capable of reconstructing tree models with higher topological and geometrical accuracy. In Figure 11, we demonstrate the visual comparison. We can see that our reconstructed models have more reasonable branch structure and also fit better to the input points. The performance improvement benefits from two reasons. First, we identify and centralize main-branch points, which in return generates tree skeletons that are topologically correct. Besides, our cylinder fitting exploits a distance-weighted non-linear least squares fitting, which significantly improves the geometrical accuracy.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 July 2019 doi:10.20944/preprints201907.0058.v1

Peer-reviewed version available at Remote Sens. 2019, 11, 2074; doi:10.3390/rs11182074



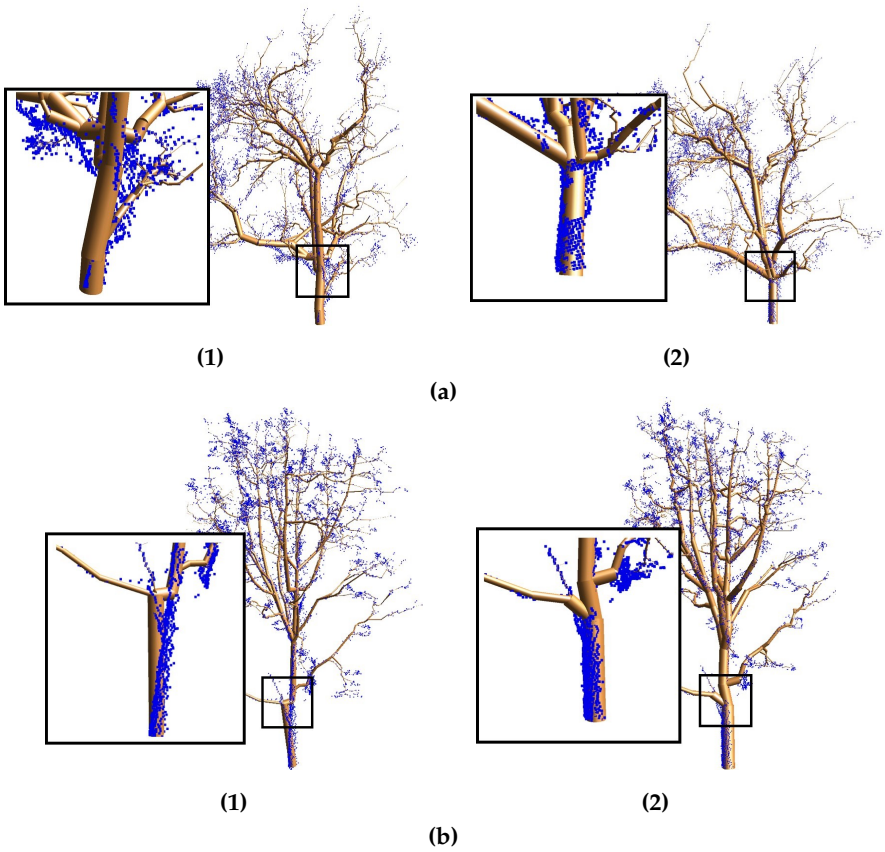



**Figure 11.** Comparison between Livny's method (left) and our method (right) on two trees.

### *3.6. Limitations*

Our algorithm can successfully reconstruct accurate and detailed 3D tree models from point clouds. However, it still has some limitations. First of all, our approach is data-driven. For poorly scanned data with sparse points, though our method can reconstruct a plausible topological structure of the tree branches, it is unable to achieve sufficient geometrical accuracy. Moreover, our work doesn't consider natural growing rules for tree branches (i.e., 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. 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 with 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 the 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 instance segmentation of trees. As our method only works for individual tree point clouds, automatic segmentation 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:** Shenglan Du performed the study and implemented the algorithms. Roderik Lindenburgh, Hugo Ledoux and Jantien Stoter provided constructive comments and suggestions. Liangliang Nan proposed this topic and provided daily supervision.

**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.

## References

1. Deussen, O.; Hanrahan, P.; Lintermann, B.; Měch, R.; Pharr, M.; Prusinkiewicz, P. Realistic modeling and rendering of plant ecosystems. Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM, 1998, pp. 275–286.
2. Maltamo, M.; Næsset, E.; Vauhkonen, J. Forestry applications of airborne laser scanning. *Concepts and case studies. 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. *International Journal of Remote Sensing* **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 Transactions on geoscience and remote sensing* **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 Sensing* **2015**, *7*, 4753–4783.
6. Reche-Martinez, A.; Martin, I.; Drettakis, G. Volumetric reconstruction and interactive rendering of trees from photographs. ACM transactions on graphics (ToG). ACM, 2004, Vol. 23, pp. 720–727.
7. Shlyakhter, I.; Rozenoer, M.; Dorsey, J.; Teller, S. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications* **2001**, *21*, 53–61.
8. Quan, L.; Tan, P.; Zeng, G.; Yuan, L.; Wang, J.; Kang, S.B. Image-based plant modeling. ACM Transactions on Graphics (TOG). ACM, 2006, Vol. 25, pp. 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 transactions on visualization and computer graphics* **2018**.
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 Journal of Photogrammetry and Remote Sensing* **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 sensing* **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 sensing of Environment* **2003**, *85*, 290–303.
13. Holmgren, J.; Persson, Å. Identifying species of individual trees using airborne laser scanner. *Remote Sensing of Environment* **2004**, *90*, 415–423.
14. Hackenberg, J.; Morhart, C.; Sheppard, J.; Spiecker, H.; Disney, M. Highly accurate tree models derived from terrestrial laser scan data: A method description. *Forests* **2014**, *5*, 1069–1105.
15. Wang, D.; Hollaus, M.; Puttonen, E.; Pfeifer, N. Automatic and self-adaptive stem reconstruction in landslide-affected forests. *Remote Sensing* **2016**, *8*, 974.
16. Xu, H.; Gossett, N.; Chen, B. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics (TOG)* **2007**, *26*, 19.
17. Verroust, A.; Lazarus, F. Extracting skeletal curves from 3D scattered data. Proceedings Shape Modeling International'99. International Conference on Shape Modeling and Applications. IEEE, 1999, pp. 194–201.
18. Dey, T.K.; Sun, J. Defining and computing curve-skeletons with medial geodesic function. Symposium on geometry processing, 2006, Vol. 6, pp. 143–152.
19. Bucksch, A.; Lindenbergh, R.C.; Menenti, M. SkelTre-fast skeletonisation for imperfect point cloud data of botanic trees. Eurographics, 2009.

20. 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. 2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics. IEEE, 2009, pp. 572–575.
21. Livny, Y.; Yan, F.; Olson, M.; Chen, B.; Zhang, H.; El-Sana, J. Automatic reconstruction of tree skeletal structures from point clouds. ACM Transactions on Graphics (TOG). ACM, 2010, Vol. 29, p. 151.
22. Zhou, H.; Shenoy, N.; Nicholls, W. Efficient minimum spanning tree construction without Delaunay triangulation. Proceedings of the 2001 Asia and South Pacific Design Automation Conference. ACM, 2001, pp. 192–197.
23. Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence* **1995**, *17*, 790–799.
24. Markku, Å.; Raumonen, P.; Kaasalainen, M.; Casella, E. Analysis of geometric primitives in quantitative structure models of tree stems. *Remote Sensing* **2015**, *7*, 4581–4603.
25. Panyam, M.; Kurfess, T.R.; Tucker, T.M. Least squares fitting of analytic primitives on a GPU. ASME 2008 9th Biennial Conference on Engineering Systems Design and Analysis. American Society of Mechanical Engineers, 2008, pp. 233–240.
26. Boost. Available online: https://www.boost.org/doc/libs/1_66_0/libs/graph/doc/index.html (accessed on 01-September-2018).
27. Easy3D. Available online: https://github.com/LiangliangNan/Easy3D (accessed on 01-March-2019).
28. AHN Dataset. Available online: https://www.pdok.nl/attenderingsservice-rss/-/asset_publisher/mvZkjafth739/content/actueel-hoogtebestand-nederland-ahn3- (accessed on 01-January-2019).