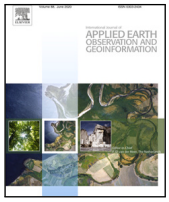
International Journal of Applied Earth Observations and Geoinformation 116 (2023) 103145

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

journal homepage: www.elsevier.com/locate/jag



Topology-based individual tree segmentation for automated processing of terrestrial laser scanning point clouds

Xin Xu [a,b,*], Federico Iuricich [c], Kim Calders [d], John Armston [a], Leila De Floriani [a,b]

[a] Department of Geographical Sciences, University of Maryland at College Park, United States
[b] University of Maryland Institutes for Advanced Computer Studies, United States
[c] Visual Computing Division, School of Computing, Clemson University, United States
[d] Computational & Applied Vegetation Ecology, Department of Environment, Ghent University, Belgium

ARTICLE INFO

MSC:
00-01
99-00

Keywords:
LiDAR
Point cloud
Tree segmentation
Topological data analysis

ABSTRACT

Terrestrial laser scanning (TLS) is a ground-based approach to rapidly acquire 3D point clouds via Light Detection and Ranging (LiDAR) technologies. Quantifying tree-scale structure from TLS point clouds requires segmentation, yet there is a lack of automated methods available to the forest ecology community. In this work, we consider the problem of segmenting a forest TLS point cloud into individual tree point clouds. Different approaches have been investigated to identify and segment individual trees in a forest point cloud. Typically these methods require intensive parameter tuning and time-consuming user interactions, which has inhibited the application of TLS to large area research. Our goal is to define a new automated segmentation method that lifts these limitations.

Our *Topology-based Tree Segmentation (TTS)* algorithm uses a new topological technique rooted in discrete Morse theory to segment input point clouds into single trees. TTS algorithm identifies distinctive tree structures (i.e., tree bottoms and tops) without user interactions. Tree tops and bottoms are then used to reconstruct single trees using the notion of relevant topological features. This mathematically well-established notion helps distinguish between noise and relevant tree features.

To demonstrate the generality of our approach, we present an evaluation using multiple datasets, including different forest types and point densities. We also compare our TTS approach with open-source tree segmentation methods. The experiments show that we achieve a higher segmentation accuracy when performing point-by-point validation. Without expensive user interactions, TTS algorithm is promising for greater usage of TLS point clouds in the forest ecology community, such as fire risk and behavior modeling, estimating tree-level biodiversity structural traits, and above-ground biomass monitoring.

1. Introduction

Forest inventory is a fundamental tool for tracking the structure, biomass and ecological condition of a forest. Since manual field surveys are expensive, time-consuming, and potentially dangerous (Zhen et al., 2016), remote sensing technologies have been adopted to assist with the generation of forest inventories. Among these, Light Detection and Ranging (LiDAR) allows to collect detailed information about the forest through sensors that emit laser signals and calculate distances based on the time delay of the returned laser pulses. *Terrestrial LiDAR*, also known as the *Terrestrial Laser Scanning*, can scan the surrounding environment and generate billions of three-dimensional (3D) points from which tree metrics, such as location, height, and diameter (Liang et al., 2018), and tree models (Raumonen et al., 2013; Hackenberg et al., 2015a,b) can be derived.

To run this type of analysis, it is first necessary to identify every single tree in the point cloud. However, inconsistent point cloud quality, diverse forest structures, and complicated plant morphology make it extremely hard to find a general, efficient, and fully automated solution (Wilkes et al., 2017; Liang et al., 2018; Calders et al., 2020; Martin-Ducup et al., 2021).

Most individual tree segmentation methods aim to extract single tree point clouds from the input forest point cloud by adding external information, such as allometric functions (Burt et al., 2019), user-defined parameters (Trochta et al., 2017) or manual inspection and correction of segmentation results (Burt et al., 2019; Raumonen et al., 2015; Calders et al., 2015). At the same time, external information makes an algorithm dataset specific, difficult to generalize, and usable only by knowledgeable experts.

* Corresponding author.
E-mail address: xinxu629@umd.edu (X. Xu).

https://doi.org/10.1016/j.jag.2022.103145
Received 31 May 2022; Received in revised form 5 December 2022; Accepted 6 December 2022
Available online 12 December 2022

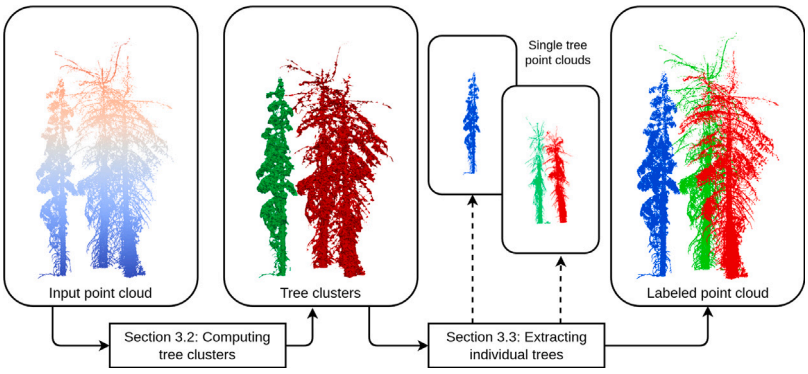


Fig. 1. Pipeline of the proposed Topology-based Tree Segmentation (TTS) approach. Input vegetation point cloud is divided into tree clusters (see Section 3.2). Single tree point clouds are then extracted from each tree cluster (see Section 3.3), which consist of the final labeled point cloud. Single tree extraction can be executed in parallel, which is marked by dashed arrows.

This work aims to define an automated approach capable of providing robust results without demanding user interactions. To achieve this goal, we define a new approach, called *Topology-based Tree Segmentation (TTS)*. TTS identifies distinctive structures in the forest, like tree bottoms and tops, and extracts individual trees using a topological structure defined via discrete Morse theory (Forman, 1998).

The performance of TTS is demonstrated on different forest types (i.e., coniferous forests as well as deciduous forests) and point characteristics (e.g., point density). We also compare our TTS method to state-of-the-art approaches, which are available in the public domain, namely *3D Forest* (Trochta et al., 2017) and *Forest Structural Complexity Tool (FSCT)* (Krisanski et al., 2021a). Multiple levels of validation are designed to examine segmentation results for tree identification and tree segmentation. Without any parameter tuning process, TTS shows superior performance demonstrating to be well-suited for a wide range of forest analysis applications, including forest inventory, tree modeling, and above-ground biomass estimation. Our main contributions are as follows:

1. A novel automated topology-based approach for individual tree segmentation on TLS point clouds.
2. Extensive experimental evaluation performed on different forest types with ground truth data.
3. Objective comparison with two state-of-the-art and open-source approaches for individual tree segmentation.
4. Proven robustness of the proposed method over different forest and points characteristics (i.e., forest type, stem density and point density).

The remainder of the paper is organized as follows. In Section 2, we introduce related work. In Section 3, we present the workflow of the proposed TTS method. Experimental results and internal parameter settings are discussed in Section 4 and Section 5, respectively. We draw some concluding remarks in Section 6.

2. Related work

Segmenting a forest TLS point cloud into individual tree point clouds is known as *individual tree segmentation* (Calders et al., 2020) or *individual tree isolation* (Martin-Ducup et al., 2021). Although manual segmentation is time-consuming, laborious, and even infeasible for areas containing many trees (Martin-Ducup et al., 2021; Burt et al., 2019), only a few algorithms have been developed to automate the individual tree segmentation process. Most automated individual tree segmentation methods are bottom-up, which means they identify and grow trees from the tree bottoms. Because TLS point clouds are scanned from the ground, stem surface points are usually complete, and tree bottoms are reliable for locating trees (Calders et al., 2020).

Trochta et al. (2017) proposed a bottom-up individual tree segmentation algorithm. Points are clustered on slices, which are parallel to the ground. Later, the tree bases are located as low-height clusters. Finally, single trees are built from detected stems by merging nearby clusters with nearby stems. In the end, many user-defined parameters are needed to guide the segmentation. Usually, manual adjustments are also needed to clean extracted single tree point clouds.

Ecological knowledge has been utilized for better segmentation results. For example, metabolic scaling theory (West et al., 1997) has been used in Tao et al. (2015) and Wang (2020). A graph of points is generated first in both works (Tao et al., 2015; Wang, 2020). Then, single tree point clouds are formed by assigning points to the detected closest stems in terms of the shortest path on the graph, where the path distance is scaled based on metabolic theory (West et al., 1997). Although the metabolic ecology theory is expected to be a universal scaling law of tree growth, many factors should be considered, like demographic traits and light, when applying the theory in practical datasets (Tao et al., 2015). For both methods from Tao et al. (2015) and Wang (2020), several parameters need to be defined by the user, such as determining the size of the tree and filtering valid results. As a consequence, it takes time to learn those methods and find appropriate settings.

Allometric relationships among tree measurements are also used in the segmentation methods. Burt et al. (2019) developed a segmentation method, namely *treeseg*, based on the *Point Cloud Library (PCL)* (Rusu and Cousins, 2011). After detecting tree stems, Burt et al. (2019) used allometric relationships between stem diameter to tree height and crown extent to extract tree crowns. As far as we have tested, the implemented software hardcoded parameters in the source code. Because of the applied allometric functions related to specific forests, treeseg may not work on datasets out-of-box.

Recently, deep learning tools have been applied to TLS point cloud analysis. Krisanski et al. extended their deep learning-based semantic segmentation method (Krisanski et al., 2021b) to extract single tree point clouds and designed *Forest Structural Complexity Tool (FSCT)* (Krisanski et al., 2021a). In FSCT, PointNet++ (Qi et al., 2017) is used to classify forest points into terrain, vegetation, coarse woody debris, and stem points. Tree skeletons are generated by grouping stem points via density-based clustering methods, such as HDBSCAN (McInnes et al., 2017) and DBSCAN (Ester et al., 1996). Cylinder fitting via RANSAC (Fischler and Bolles, 1981) is repeatedly applied on skeleton points to generate cylinders representing stems and branches. The complete tree structures are achieved by grouping close neighboring cylinders. Finally, points are assigned to the closest tree cylinder to form the single tree point clouds. Due to the applied deep learning tool, FSCT demands large computational resources, which is also an obstacle in dealing with big data. Apart from PointNet++ (Qi et al., 2017), different deep learning techniques ares used to extract tree locations from

point clouds before the segmentation. Focusing on TLS points-derived images, Fan et al. (2022) used YOLOv3 (DarkNet-53) (Redmon and Farhadi, 2018) to extract tree crown boundaries and Wang et al. (2019) applied Faster R-CNN (Ren et al., 2015) to detect trunk locations. Xi and Hopkinson (2021) used CenterNet (Duan et al., 2019) to extract tree crown boundaries on the voxel-based representations of a point cloud. While very promising, these approaches (Fan et al., 2022; Xi and Hopkinson, 2021; Wang et al., 2019) need to transfer 3D point clouds into images or voxels before applying deep learning techniques. Furthermore, deep learning-based approaches require an extensive amount of annotated data to be trained, which can be challenging to collect in real scenarios (Calders et al., 2020).

3. Topology-based tree segmentation (TTS) approach

The proposed TTS approach for individual tree segmentation is composed of two steps. The entire pipeline is presented in Fig. 1. First, in the *dividing* step, we split the input vegetation TLS point cloud into sub-regions, called *tree clusters*. Intuitively, each tree cluster is a group of trees with intersecting canopy (see Section 3.2). Then, an *extracting* step is used to find every single tree through a bottom-up segmentation method (see Section 3.3). In the remainder of this section, we first introduce briefly background notions on simplicial complexes, α-complexes and discrete Morse theory, which are helpful to understand the proposed method.

3.1. Discrete morse theory

Simplicial complexes and α-complexes. We use simplicial complexes to infer a topological (i.e., connectivity) structure on the 3D point cloud. Formally, a *k-simplex* σ is the convex hull of $k+1$ affinely independent points in the Euclidean space. For instance, a 0-simplex is a point, a 1-simplex a straight line segment, a 2-simplex a triangle, and a 3-simplex a tetrahedron. k is called the *dimension* of σ. We denote k-simplex σ spanned by the vertices $v_0, v_1, \ldots, v_k$ as $\sigma = \{v_0, v_1, \ldots, v_k\}$. Any simplex τ, which is the convex hull of a non-empty subset of the points generating a simplex σ, is a *face* of σ.

A *simplicial complex* Σ is a finite set of simplexes, such that: (i) each face of a simplex in Σ belongs to Σ; (ii) for each pair of simplexes σ and τ, either $\sigma \cap \tau = \emptyset$ or $\sigma \cap \tau$ is a face of both.

An *α-complex* is a simplicial complex constructed from the simplexes of a Delaunay tetrahedralization Σ^T (Delaunay et al., 1934). Let σ a simplex in Σ^T, and let C_σ the circumsphere (or circumcircle) of σ with radius r. The *α-complex* (Edelsbrunner, 2010) Σ^α is a subcomplex of Σ^T containing all vertices of Σ^T, plus all simplexes σ such that: (i) $r < \alpha$ and C_σ contains no points in P, or (ii) σ is a face of $\tau \in \Sigma^\alpha$.

Discrete morse theory. *Discrete Morse theory* (Forman, 1998) is a combinatorial counterpart of Morse theory (Milnor, 1963) which allows to study the topology of a simplicial complex Σ.

Given a simplicial complex Σ, a discrete vector is a pair of simplices (σ, τ), where σ is a face of τ. A *discrete vector field* V is a collection of pairs (σ, τ) such that, each simplex of Σ is in at most one pair of V. Simplices that belong to no vector are called *critical*. In a triangle mesh, critical triangles are maxima, critical edges are saddles and critical vertices are minima. Pairs are formed by a triangle and an edge; and by an edge and a vertex. A *V-path* is a sequence $\sigma_1, \tau_1, \sigma_2, \tau_2, \ldots, \sigma_r, \tau_r$, such that $(\sigma_i, \tau_i) \in V$, σ_{i+1} is a face of τ_i, and $\sigma_i \neq \sigma_{i+1}$. A V-path with $r > 1$ is *closed* if σ_1 is a facet of τ_r different from σ_{r-1}. A discrete vector field V is called *Forman gradient* if it has no closed V-paths. A *separatrix* V_i-path is a V-path connecting two critical simplexes of dimension $i+1$ and i, respectively. In a triangle mesh, we have each separatrix V_0-path connecting a critical edge to a critical vertex and separatrix V_1-path connecting a critical triangle to a critical edge.

When the discrete gradient V is computed according to an input scalar function $f : \Sigma \to \mathbb{R}$, V can serve as a combinatorial representation of the gradient of f and its critical points. For example, Fig. 2(a) shows a scalar function f defined on a triangle mesh Σ. Fig. 2(b) shows arrows indicating the *gradient pairs* while points indicate the critical simplexes. Blue points indicate minima, green points indicate saddles, and the red point indicates the maximum. Notice that all the gradient pairs, depicted with arrows, mimic the gradient of the function indicating the direction in which the function values decrease. To this end, many approaches have been defined for computing a discrete gradient on regular grids (Guylassy et al., 2008; Robins et al., 2011; Shivashankar et al., 2012; Shivashankar and Natarajan, 2012) and simplicial complexes (Weiss et al., 2013; Fellegara et al., 2014). A complete overview of these methods is presented by De Floriani et al. (2015).

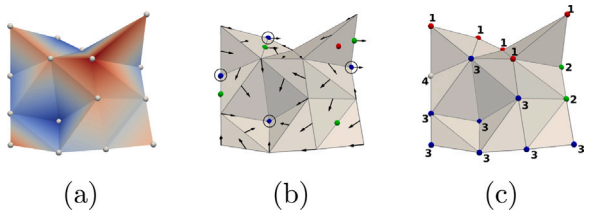


Fig. 2. (a) Scalar function f, defined on a triangle mesh Σ, color coded by means of a blue–red divergent color map. (b) Forman gradient computed on the same scalar field. Arrows indicate gradient pairs. Blue, green and red dots indicate critical minima (vertices), saddles (edges), and maxima (triangles). Minimum points are also marked by circles. (c) Four point clusters with labels corresponding to the four minima computed by navigating the gradient paths. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

A Forman gradient implicitly defines a segmentation of Σ according to the regions of influence of the critical points of f (De Floriani et al., 2015). In this work, we focus on the regions of influence of the minima of f, also called minimum-ascending regions. A *minimum-ascending region* for a minimum vertex σ is defined as the collection of vertices belonging to the V_0-paths reaching σ. Fig. 2(c) shows the minimum-ascending regions of the Forman gradient shown in Fig. 2(b). Every vertex belongs to the minimum-ascending region of one of the four minima characterizing V (see four blue points in Fig. 2(b)).

3.2. Computing tree clusters

In the remainder of this work, we refer to the *forest vegetation point cloud* as the input point set P. All parameter values introduced in this section are discussed and motivated in Section 5.

The first objective of TTS method is to divide P into *tree clusters* by dividing easily separable trees. We achieve this goal by computing an α-complex Σ on P and finding connected components of Σ.

When computing an α-complex, the value of α affects the shape of Σ and the number of connected components. Thus, the objective is to obtain a α-complex that outlines the tree shapes (i.e., branches, trunks) while disconnecting easily separable trees. Fig. 3 shows three α-complexes generated with different α values. When the α value is very small, such as $\alpha = 0.02$ m in Fig. 3(a), the α-complex is made up of isolated points and fails to outline tree shapes. However, with a larger α value ($\alpha = 0.3$ m) in Fig. 3(c), trees are connected into one component, which is hard to split. Furthermore, woody structures (i.e., branches, stems) in α-complex presented in Fig. 3(c) are not as clear as the built α-complex with $\alpha = 0.1$ m in Fig. 3(b). To this end, our experiments identified $\alpha_s = 0.1$ m as the ideal value to use.

Once α-complex Σ is computed, we process its connected components. For each component, we consider the vertex with the lowest elevation and the length of the component. The *component length* is defined as the elevation difference between the highest and lowest points in the component. Then, a component of Σ is classified as a *tree cluster* if and only if the elevation of its lowest vertex is less than $th_{low} = 1.5$ m and the length is larger than $th_{len} = 2$ m to remove noise

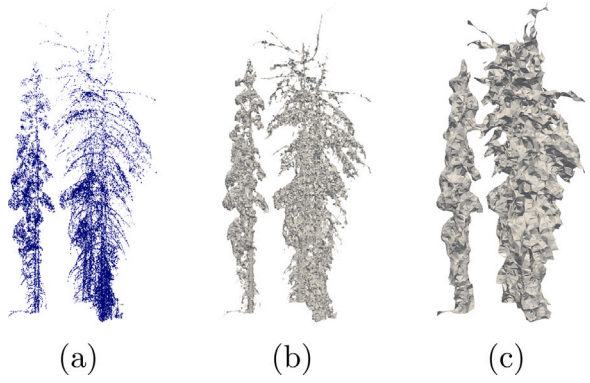


Fig. 3. Generated α-complexes with (a) α = 0.02 m, (b) α = 0.1 m, and (c) α = 0.3 m, respectively. 0-simplices (points) are colored in blue in (a), and simplices with dimension ≥ 1 are colored in gray in (b) and (c). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

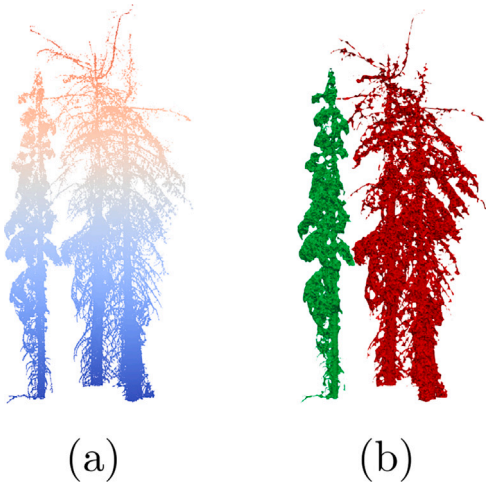


Fig. 4. Tree clusters computed from the point cloud. (a) Input forest vegetation point cloud is colored by height. (b) Two tree clusters are colored in green and red, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and small vegetation. This leads to a subdivision of the forest points into tree clusters. Fig. 4 illustrates two tree clusters computed from a point cloud with three trees. There are two components of the α-complex built on the point cloud (see Fig. 4(b)). Both components successfully outline the critical tree features, such as treetop branches and stems. Three trees are divided into two clusters, where one cluster includes a single tree (see green points in Fig. 4(b)) while the other cluster has two close trees with intersecting canopy (see red points in Fig. 4(b)).

3.3. Extracting individual trees

After computing tree clusters, the objective is to identify individual trees in each cluster. To this end, we use a bottom-up segmentation approach based on discrete Morse theory (Forman, 1998).

Let Σ_{tc} be the α-complex component corresponding to a tree cluster. We define a scalar function f on the vertices of Σ_{tc} as the height function. Formally, $f(p) = p.z$ where p is a vertex in Σ_{tc} and $p.z$ is its elevation. Then, we compute the Forman gradient V corresponding to Σ_{tc} and f (see Section 3.1), and we use critical points, namely minima, to guide the tree cluster segmentation.

Compute minimum-ascending regions. Given a Forman gradient V computed on Σ_{tc}, we extract a segmentation of the vertices of Σ_{tc} by

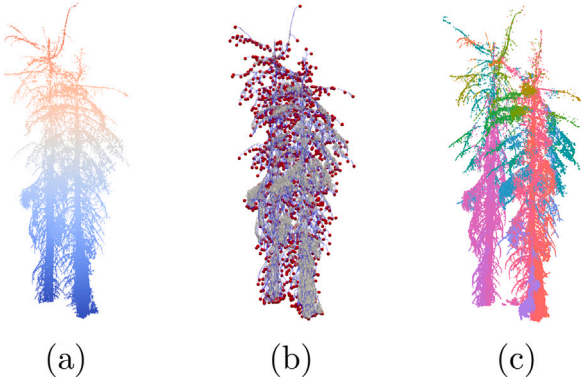


Fig. 5. Over-segmentation results on the tree cluster points. (a) Tree cluster points colored in a blue–red colormap based on point height. (b) Computed minima colored in red points. (c) Computed minimum-ascending regions in different colors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

computing influence regions corresponding to the minima of f (see Section 3.1). The region of influence of a minimum, namely the minimum-ascending region, is a collection of vertices obtained through a breadth-first traversal of the vertex-edge arrows in the Forman gradient. Starting from a minimum vertex v, we insert v into a queue Q. At each iteration we extract a vertex v_1 from Q and we compute the edges incident in v_1. For each of such edges, we retrieve if they are paired with some vertex v_2 different from v_1. If this is the case, we add v_2 to Q. The cluster corresponding to minimum vertex v is obtained by collecting all the vertices touched by such visit. Fig. 5(c) shows the minimum-ascending regions computed from the minima shown in Fig. 5(b). The numerous minimum-ascending regions imply that tree cluster points are over-segmented. Intuitively, each minimum represents one point with a minimum elevation value with respect to the neighboring points. For α-complex Σ_{tc}, minima correspond to either tree bottoms or branches facing downward (see Fig. 5(b)). Therefore, the computed minimum-ascending regions only represent small tree components instead of complete trees. Naturally, we will unite tree components to build single trees. Rather than merging points directly, we will work on the minima to group minima that are in the same single trees so that each single tree point cloud can be found from the same group of minima-ascending regions.

Locate tree seeds. We first identify minima presenting the tree bottoms. To distinguish between tree bottoms and branches from minima, we clip Σ_{tc} by removing all points and simplices (i.e., edges, triangles and tetrahedrons) above $th_{stem} = 0.5$ m. We call the remaining 3-complexes the *clipped simplicial complex* and we denote it as Σ'_{tc}.

Then, we match each component of Σ'_{tc} with the minima in V. Namely, if a component in Σ'_{tc} contains at least a minimum vertex, the component is called a *seed component* and the minimum is classified as a *seed vertex*. Please note that one seed component can include multiple minima as seeds. Fig. 6 shows seed components and seed vertices found from the tree cluster. Two stems are well captured from clipped Σ'_{tc}, as two seed components are isolated. Each seed component includes several minima colored in green and red.

Grow single trees. Once trees have been seeded, the TTS algorithm associates the remaining minima to the seeds and combines all points to form single tree point clouds.

TTS still uses the Forman gradient V as well as critical complexes (i.e., minima and 1-saddles) previously computed. We compute a graph representing the connectivity of the minima and 1-saddles. More specifically, starting each minimum, we follow the V-path and find its connected 1-saddle. If two minima are connected to the same 1-saddle, we call these minima connected. Finally, the graph is denoted as G_m =

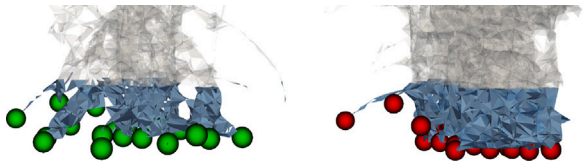


Fig. 6. Detected seed components and seed vertices from the tree cluster. Two seed components are colored in blue–gray, and corresponding seed vertices are in green and red, respectively. The α-complex is pictured with light gray as the background. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

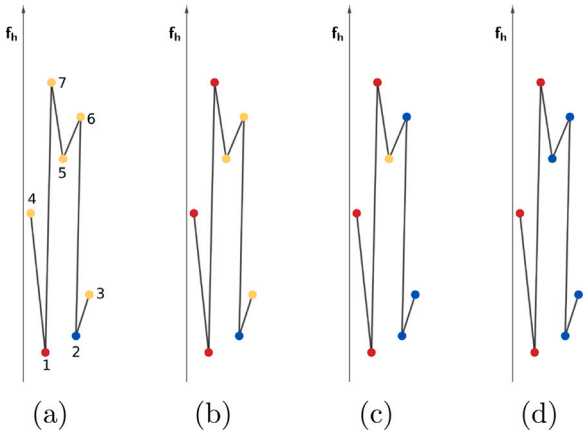


Fig. 7. Demonstration of labeling minima nodes from seeds. (a) Seeds are colored in red and blue, respectively. (b) Label neighbor nodes from the red seed. (c) Label neighbor nodes from the blue seed. (d) Final labeled nodes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(N, A, R), where nodes in N correspond to the minima of V and arcs in A connects two nodes if the corresponding minima are connected. R is a label assigned to each seed node in N. Seed nodes belonging to the same seed component have the same label. All remaining nodes are marked as unlabeled.

Then, we insert all nodes in N corresponding to tree seeds (i.e., minima labeled as seed points) in an ordered queue, keeping nodes sorted in ascending order of elevation. For each node n extracted from the queue, we retrieve the nodes in G_m connected to n. For each unlabeled node nbr connected to n, we assign to nbr the same label as n (i.e., $R(nbr) = R(n)$) and we insert nbr in the queue. Fig. 7 shows the minima labeling procedure. For illustration purposes, we place seeds (colored in red and blue, respectively) and unlabeled yellow nodes based on their f values. As red seed 1 is lower than blue seed 2 in Fig. 7(a), we have a queue initialized as $\{1, 2\}$. We first check red seed 1 and label its adjacent unlabeled nodes 4 and 7 (see Fig. 7(b)). The queue becomes $\{2, 4, 7\}$. Similarly, we grow blue seed 2 via merging its neighbor unlabeled nodes 3 and 6 in the next step in Fig. 7(c). And, we have the queue $\{3, 4, 6, 7\}$ in an ascending order based on their f values. When we check nodes 3 and 4, there are no unlabeled nodes adjacent to nodes 3 and 4, respectively. Therefore, there are no new nodes labeled. The queue becomes $\{6, 7\}$. Finally, we work on node 6 and assign its unlabeled node 5. The queue has $\{5, 7\}$ now. However, all neighbor nodes of nodes 5 and 7 are already labeled. In other words, all nodes are labeled now as presented in Fig. 7(d).

Once all nodes in G_m have been labeled, we propagate the label assigned to each minimum to its region of influence, which will start forming the tree point clouds and have input cluster points labeled. Fig. 8(a) and (b) shows the labeled minima at the initial and final states, respectively. In the beginning, only seeds at the bottom are labeled, while the remaining minima are unlabeled. After growing trees from

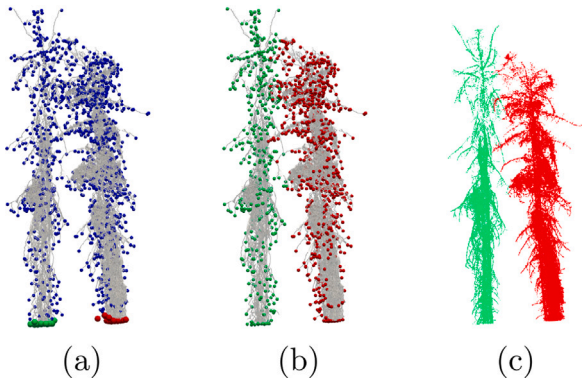


Fig. 8. Minima labeling on a α-complex built on the tree cluster points. (a) Initial minimum seeds are colored in green and red, respectively. Unlabeled minima are in blue. (b) All minima are labeled in the end. (c) Labeled single tree point clouds. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

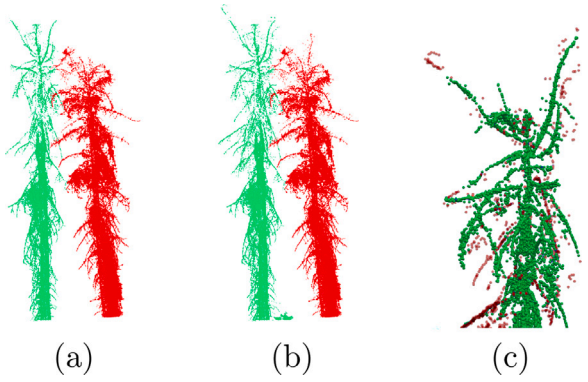


Fig. 9. Complete individual tree segmentation results. (a) Extracted single tree points from Σ. (b) Final single tree point clouds after labeling sparse points. (c) Comparison between results from (a) and (b) at the treetop. Points from (a) are colored in green, while results from (b) are colored in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the bottoms, the rest minima are successfully assigned to the stems with a tree label. Based on the minima-ascending regions, points are labeled in Fig. 8(c) where two trees are successfully extracted from the tree cluster points.

Append sparse points. The generated α-complex Σ may not include all tree points, especially for poor quality point clouds with sparse treetop points. Generally, the single tree point clouds extracted up to this step will identify hard objects of the tress (i.e., trunks and branches), but they might miss sparse treetop points. The final extraction step aims to assign sparse points to the extracted single trees. The idea is to expand the point labeling by modifying the α-complex. To ensure connectivity for all points, we generate a new α-complex Σ_l of entire forest points P with a larger value of α (i.e., $\alpha_l = 0.3$ m). Then, we reuse the same greedy search (demonstrated in Fig. 7) using the connectivity encoded in Σ_{lg} to add missed points to each extracted single tree point cloud. Fig. 9 shows an example of the trees segmented after the final step. Two trees are entirely extracted from the tree cluster points. As shown in Fig. 9(c), missed sparse treetops points are correctly added in Fig. 9(b).

4. Experimental results

This section focuses on the results obtained with our proposed *Topology-based Tree Segmentation (TTS)* method on experimental datasets. The selected experimental datasets cover different forest types (i.e., from needleleaf to deciduous broadleaf forests) and exhibit TLS

point clouds with different characteristics (i.e., point density). Experiments have been performed on a desktop computer equipped with an Intel® Core™ CPU i7-8700 @ 3.20 GHz and 32 GB memory of RAM. Due to the size of the largest dataset, *Wytham Woods* (Calders et al., 2022), we used another computer with more memory (192 GB) and an Intel® Xeon® X5660 @ 2.8 GHz to process it. We investigate the performance of our segmentation method on two levels of validations: point-level and tree-level.

4.1. Point-level experiments

In the point-level experiments, input TLS point clouds are labeled as the reference data. Each point has location coordinates and tree label information $(x, y, z, label)$. Points with the same tree label consist of one individual tree. We provide multiple point-by-point valuation methods to thoroughly evaluate the single tree point clouds extracted from labeled TLS point clouds.

4.1.1. Datasets

Four different datasets are selected to test segmentation methods comprehensively in different forest types and point densities. Following the definition from Wilkes et al. (2017) and Calders et al. (2020), we calculate the vegetation point density as the average value of four-nearest neighbor distances of points in vegetation points. Therefore, dense point clouds have smaller values of point density. *Tree set* (Hackenberg, 2022) includes four groups of single tree point clouds. There are 60 trees in total, covering evergreen sub-tropical trees, cherry, and pine trees. Each single tree point cloud is scanned from a single tree. We merge each group of single tree point clouds into one forest point cloud to simulate a labeled plot-level TLS point cloud. Therefore, the merged point clouds have the densest points in our experiments with a point density of 0.64 cm. Merged point clouds also include adjacent trees with overlapped crowns, which can be found in the following discussions (i.e., Fig. 12). *Labeled Fire data* is extracted from Ferguson Fire Mensuration TLS data and manually labeled by ourselves. The Ferguson Fire Mensuration TLS point clouds are provided by the *Global Environmental Analysis and Remote Sensing (GEARS) Laboratory* (Greenberg et al., 2022). The plots are in the Northern Sierras forest, where pines are dominated. We manually labeled 58 trees from the extracted areas. The point density of labeled point clouds is 5.07 cm. *3D Forest sample data* (Trochta et al., 2022a) is from 3D Forest software (Trochta et al., 2017, 2022b). It includes 26 labeled trees. However, it has the sparsest points in our experiments with a point density of 9.89 cm. *Wytham Woods* (Calders et al., 2022) is the largest and most challenging dataset we have in the experiment. The site is located in one broadleaf deciduous forest in Oxford, UK. This dataset includes 459 trees and very dense TLS point clouds with a point density of 2.75 cm. More details of labeled experimental datasets are provided as additional material (Xu et al., 2022).

4.1.2. Comparing accuracy

We apply each test method to extract single tree point clouds and then compare them against the reference segmentation results. We use *Rand index* (Rand, 1971), *normalized directional Hamming distance* (Huang and Dom, 1995) and *mean Intersection over Union* (Jaccard, 1908) to measure the similarity between the segmentation results R and true segmentation T.

Rand index. For each point p_i, we have two labels l_i, l_i' assigned on p_i based on T and R segmentation, respectively. We use N to count the number of paired points which have the same label relationship in T and R. Given two points p_i and p_j, if $l_i = l_j$ and $l_i' = l_j'$, we add 1 to N. When $l_i \neq l_j$ and $l_i' \neq l_j'$, we also add 1 to N. Finally, Rand index is calculated as

$$RI = \frac{N}{|T|^2},$$

where $|T|$ is the number of total points in T.

Directional hamming distance. We denote $D_H(T \Rightarrow R)$ the Hamming distance when using reference results T to check the segmentation results R. To compute $D_H(T \Rightarrow R)$, we start by constructing the correspondences between each region of T with a region of R such that $t_i \cap r_j$ is maximized. Then,

$$D_H(T \Rightarrow R) = \sum_{r_i \in R} \sum_{\substack{t_k \neq t_j, \\ t_k \cap r_i \neq 0}} |r_i \cap t_k|,$$

where $|*|$ shows the number of points in a set. Directional Hamming distance is further normalized as,

$$DHD = 1 - \frac{D_H(T \Rightarrow R) + D_H(R \Rightarrow T)}{2 \times |T|},$$

where $|T|$ is the number of total points.

Mean intersection over union (iou). Following the IoU-based assessment method used in Wang (2020), we first pair each reference tree point cloud with an extracted single tree point cloud, if it exists. Then, we use entry c_{ij} to record the number of points from reference tree i in T predicted as an extracted tree j in S. We sort extracted trees so that c_{ii} presents the number of common points between the reference tree i and its matched tree. Then the IoU of tree i is calculated as:

$$IoU_i = \frac{c_{ii}}{c_{ii} + \sum_{j \neq i} c_{ij} + \sum_{k \neq i} c_{kj}}$$

Thus, the mean IoU ($mIoU$) of all reference trees is computed by:

$$mIoU = \frac{\sum_{i=1}^{N_{ref}} IoU_i}{N_{ref}},$$

where N_{ref} is the number of reference trees in T.

All values (RI, DHD, and $mIoU$) should be one when the segmentation matches the ground truth results perfectly and are decreased when more mismatch exists between T and R.

4.1.3. Results and discussions

We compare our TTS method to two segmentation approaches, 3D Forest (Trochta et al., 2017, 2022b) and Forest Structural Complexity Tool (FSCT) (Krisanski et al., 2021a, 2022). Both are implemented as open-source packages, well-maintained and documented. 3D Forest (Trochta et al., 2017) is a classic bottom-up segmentation method, and FSCT (Krisanski et al., 2021a) focuses on plot-level tree measurements and supports single tree segmentation by applying the deep learning technique in its workflow (see Section 2).

Fig. 10 presents validation results of test methods performed on labeled TLS point clouds. Overall, our TTS method has the largest Rand index (RI), directional Hamming distance (DHD) and mean Intersection over Union (mIoU) in all test data regardless of forest types and point densities. On average, we have 0.96 of RI, 0.90 of DHD and 0.78 of mIoU. All assessment values are close to one, implying that our segmentation results are significantly similar to the reference results. In contrast, 3D Forest (Trochta et al., 2017) has 0.75, 0.79 and 0.46 for RI, DHD and mIoU, respectively. And, FSCT (Krisanski et al., 2021a) has the similar values, 0.79, 0.73 and 0.49 for RI, DHD and mIoU, respectively. Considering the forest type, we observe that test methods tend to perform better in coniferous forests from the labeled Fire data. In contrast, it is harder to extract single trees from deciduous broadleaf forests (i.e., Wytham Woods data) and mixed forests, like 3D Forest sample data, which includes multiple layers of trees and diverse forest morphology. As presented in Fig. 10(c), compared to the labeled Fire data, the IoU values of 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a) decrease by about 50% at complicated forests (i.e., 3D Forest sample data and Wythem Woods data). Unlike this drastic drop, the IoU value of our approach only declines by about 20%. A similar pattern can also be found in the Rand index and Hamming distance results. For example, 3D Forest (Trochta et al., 2017) has less than half of the Rand index in Wythm wood data compared to the Fire data (see Fig. 10(a)). It also has more than 20%

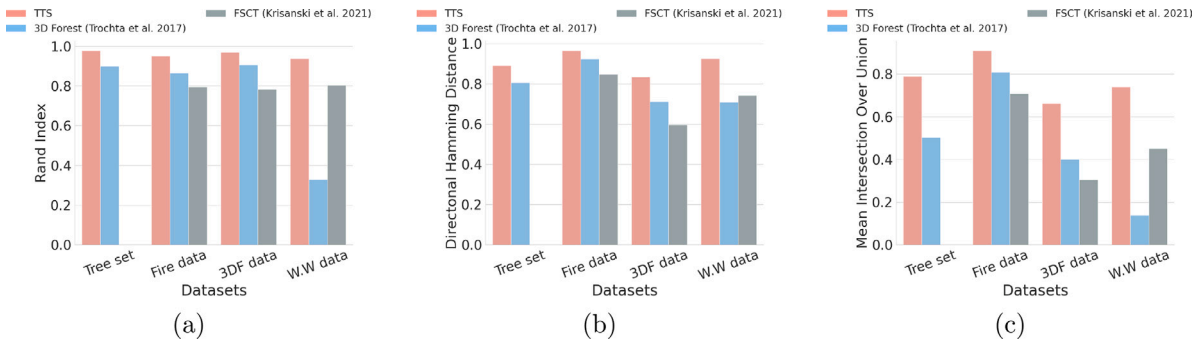


Fig. 10. (a) Rand index, (b) directional Hamming distance and (c) mean Intersection over Union values of test segmentation methods performed on labeled TLS point clouds. 3D Forest sample data (Trochta et al., 2022a) and Wytham Woods data (Calders et al., 2022) are denoted as *3DF data* and *W.W data*, respectively.

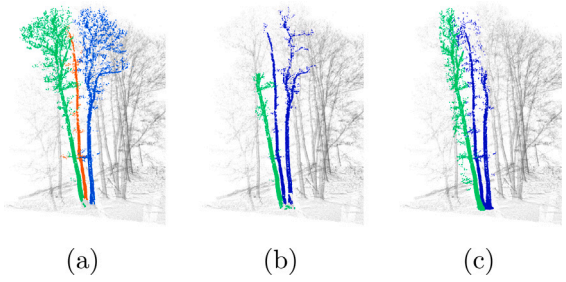


Fig. 11. Details of three extracted single trees in *3D Forest sample data* performed by (a) TTS, (b) 3D Forest (Trochta et al., 2017), and (c) FSCT (Krisanski et al., 2021a), respectively.

drop of directional Hamming distance value. Instead, our approach has a drop of less than 10% as presented in Fig. 10(b), indicating that our method is stable and resilient to different forest types. The superior performance of our TTS approach is the result of improvements to *stem detection*, *branch isolation* and *crown delineation*. Below we discuss the reasons for these improvements.

Our TTS method can identify stems correctly even in complicated forests, which is critical for bottom-up methods like TTS and 3D Forest (Trochta et al., 2017). For example, Fig. 11 shows three trees having very close stems. Only TTS can distinguish these close stems correctly as presented in Fig. 11(a). However, both 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a) fail to isolate trees and combine two trees into one tree (see Fig. 11(b) and (c)).

The applied α-complex in our TTS approach can mark tree bottoms correctly and outlines tree branches well. With discrete Morse theory-based analysis, TTS can study the tree shape and divide close branches in various conditions. Even for more challenging tree shapes, such as leaning trees and intersecting with each other (see Fig. 12), our TTS method is still able to correctly identify two trees by outlining tree branches. Unlike TTS, 3D Forest (Trochta et al., 2017) cannot reveal two trees in Fig. 12(b). Fig. 12(c)–(f) shows a small understory tree extracted by test methods. All three methods can detect this small tree without isolating its adjacent vegetation. They are acceptable results for a multiple-stem tree. However, compared to 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a) results, the tiny tree branches of our extracted tree in Fig. 12(d) is complete and more like the reference result in Fig. 12(c).

Besides better tree stem detection and branch isolation, our TTS approach can also distinguish different tree crown points better. For example, tree crown points can be relatively sparse in TLS point clouds. Because limited by laser scanning, distant objects (i.e., treetops) to the LiDAR scanners have fewer return points in TLS point clouds. Despite sparse tree crown points, our TTS method outperforms other methods for labeling tree crown points. The mixed forest from 3D Forest sample

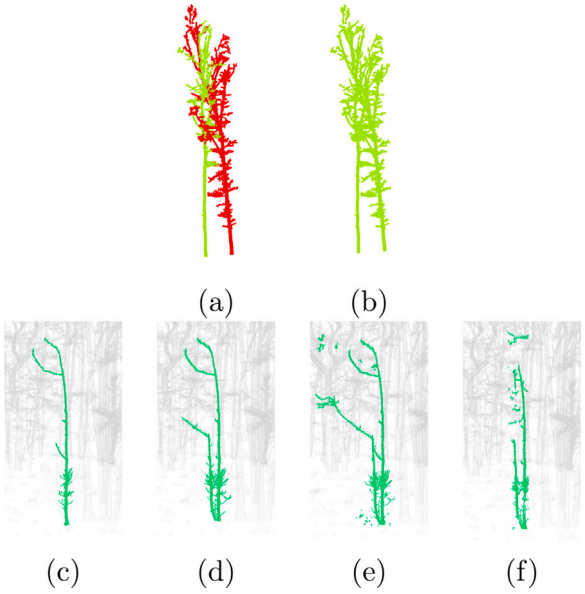


Fig. 12. Examples of two trees extracted by test methods. (a) and (b) Segmentation results of one plot (Quercus plot) in *Tree set data* achieved by TTS and 3D Forest (Trochta et al., 2017), respectively. (c) One small reference single tree point clouds in *Wytham Woods* tile 45. Entire point clouds are colored in gray as the background. (d), (e) and (f) Segmentation results from TTS, 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a), respectively.

data (Trochta et al., 2022a) is more difficult because of various trees and complicated tree structures. Our accuracy is much higher than any other methods (see Fig. 10). For example, the mIoU value of TTS is 1.7 times of that achieved by 3D Forest (Trochta et al., 2017), and 2.2 times of FSCT (Krisanski et al., 2021a). One reason is that extracted trees from other methods are incomplete and usually miss crown points. Fig. 11 already shows detailed segmentation results of three trees from 3D Forest sample data (Trochta et al., 2022a). It is clear to see that 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a) methods have larger errors labeling the sparse treetop points. We can preserve complete tree crowns and find label crown points correctly in Fig. 11(a), even though green and blue trees have very sparse treetop points. Fig. 13 shows the segmentation results of entire 3D Forest sample data (Trochta et al., 2022a) achieved by test methods. Once again, Fig. 13(c) and (d) present that most crown points are incomplete from 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a), while TTS has a better overall segmentation result with more complete crown points in Fig. 13(b).

When tree crowns have more points due to better LiDAR scanners and multiple TLS scanning, the challenge becomes distinguishing trees from neighbors. Fig. 14 presents one big tree in the Wytham

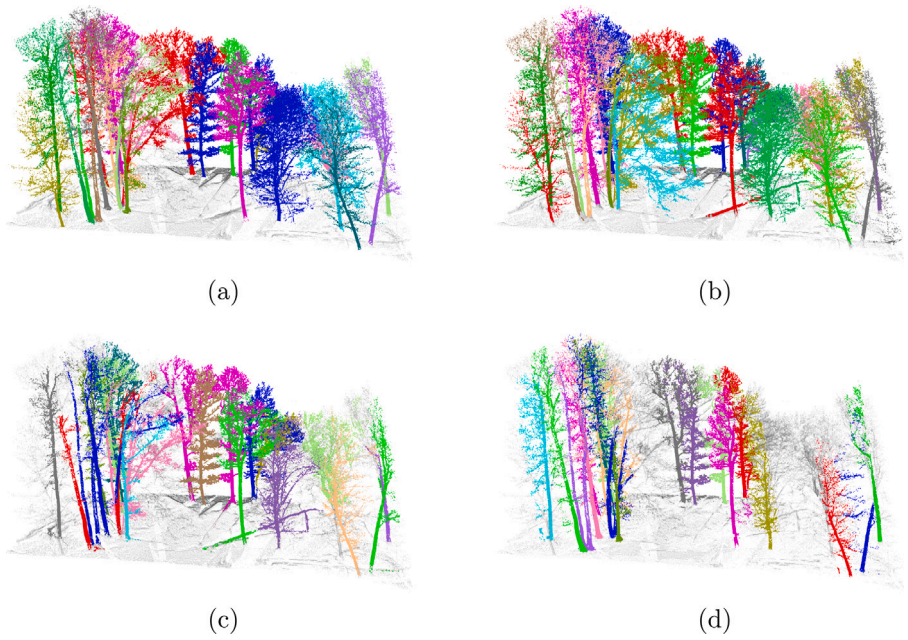


Fig. 13. Segmentation results of *3D Forest sample data*. Ground points are colored in gray. Single tree points are colored to distinguish each other. (a) Reference results. (b), (c) and (d) Segmentation results from TTS, 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a), respectively. The entire points are colored in light gray as the background.

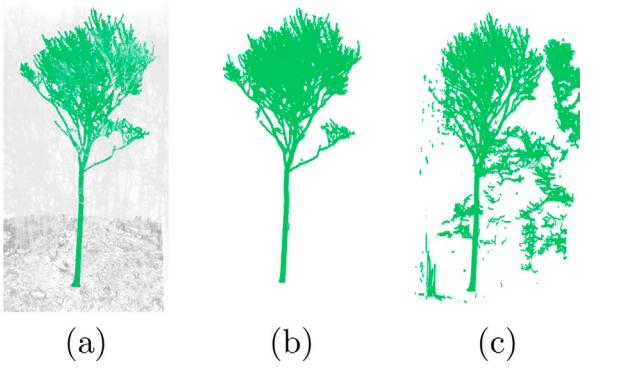


Fig. 14. Details of one big tree in *Wytham Woods* tile 45. (a) Reference single tree point clouds. Entire point clouds are colored in gray as the background. (b) Segmentation results from TTS, (c) Segmentation results from FSCT.

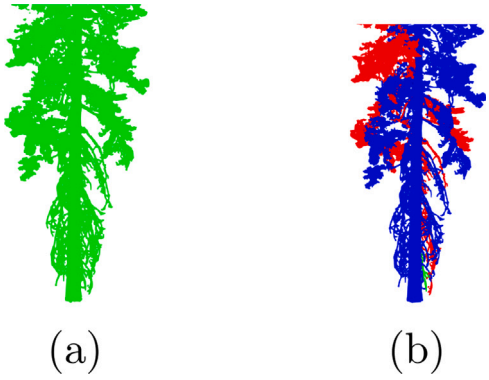


Fig. 15. Bottom points of segmentation results in single-tree study area 7 in *labeled Fire data*. (a) Input vegetation points. (b) Results performed by TTS.

Woods (Calders et al., 2022) tile 45. Our method and FSCT (Krisanski et al., 2021a) can extract this big tree, while 3D Forest (Trochta et al., 2017) cannot identify this tree from its segmentation results. Considering that other close trees surround this big tree, our automated result segmentation result is promising as our results are very similar to the reference results, even for crown branches. On the contrary, FSCT (Krisanski et al., 2021a) fails to extract single tree point correctly, as there are points from other trees in Fig. 14(c).

Compared to other test methods, TTS is aggressive at detecting stems. If low branches of trees are very close to the ground, these branches are sometimes treated as tree bottoms by TTS. Fig. 15 demonstrates the one tree with very low branches. These low branches are extremely near the ground, and the branch ends are not close to the stem trunk. Thus, TTS sets the low branch as the tree stem. The problem can be easily fixed by adjusting stem detection height th_{stem} (see Section 5) in TTS to detect tree bottoms. However, one of our motivations is to provide the tree segmentation method without an intense tuning process, which could also be used to improve the performance of 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a). Therefore, we keep using the same height thresholds in all test data to make our method easy. Despite that, our method does not over-segment single trees when tree trunks are clear. Even processing a multiple-stems tree, TTS still considers one trunk because multiple stems are clustered. For example, two trees with one connecting trunk are treated as one tree in our results presented in Fig. 12(d).

There are other exceptional cases in actual TLS data besides low branches touching the ground, such as falling trees or leaning trees. For example, one fallen tree touches two trees in the 3D Forest sample data (Trochta et al., 2022a). Extraction-focused methods, like TTS and 3D Forest (Trochta et al., 2017), still output this fallen tree in the final results and label fallen tree points to nearby trees (see Fig. 13(b) and (c)). However, measurement-focused FSCT (Krisanski et al., 2021a) uses cylinder fitting on points to filter out this fallen tree. For the sake of simplicity, we do not handle such particular scenarios at the current stage.

Considering all experiment results with labeled TLS point clouds, the experimental results show that our TTS method can achieve more accurate and more stable segmentation results in different forest types. TTS is also robust to varying point densities. More importantly, TTS works without parameter tuning in all test datasets.

Table 1
Unlabeled data overview. Column *T.Den* presents stem density (stems/ha) in the data, and Column *P.Den* (unit: cm) records the average point density of experimental TLS point clouds. Single-scanned and Multiple-scanned TLS point clouds are presented as SS and MS in EuroSDR dataset.

Dataset			T.Den	P.Den
Fire data (Greenberg et al., 2022)	Before-fire		762.17	5.31
	After-fire		589.23	4.95
EuroSDR data (Liang et al., 2018)	Easy	SS	659.18	0.92
	Easy	MS		0.87
	Medium	SS	1103.52	0.93
	Medium	MS		0.8
	Difficult	SS	1791.99	0.72
	Difficult	MS		0.66

4.2. Tree-level experiments

The tree-level experiment aims to examine the locations of extracted single trees. Many critical tree metrics are based on tree locations at the single and plot levels (i.e., stem density, spatial pattern). Therefore, tree locations are commonly used to validate a method's accuracy (Liang et al., 2018). This level experiment evaluates the tree locations derived from vegetation segmentation results with the conventional tree matching-based validation method.

4.2.1. Datasets

Two large datasets, Ferguson Fire Mensuration TLS data (namely Fire dataset in our following content) (Greenberg et al., 2022) and EuroSDR benchmark data (Liang et al., 2018), involving different forest structures and point densities, are selected in this experiment. Both datasets include TLS point clouds and reference data, such as tree locations, heights, and DBHs.

Table 1 presents an overview of experimental datasets, including the tree numbers, stem density (stems/ha), and vegetation point density. Fire dataset (Greenberg et al., 2022) includes point clouds scanned before and after the wildfire Ferguson 2018. The EuroSDR benchmarking dataset (Liang et al., 2018) includes six plots divided into three levels. Each plot has two versions of TLS point clouds in each plot: single-scanned (SS) and multiple-scanned (MS). Single-scan (SS) data is achieved from the plot center scan, while co-registering all scan data generate multiple-scan (MS) data (Liang et al., 2018). Overall, the experimental data include a broad range of stem densities, from 589 stems/ha to 1791 stems/ha. The TLS point clouds also present different point densities from 0.6 cm to 5.3 cm. More information on the two datasets can be found in Xu et al. (2022).

4.2.2. Tree matching-based validation

In this level of experiments, we follow the traditional tree matching-based validation method to check the accuracy of detected tree locations derived from segmentation results. The tree matching-based validation method searches and finds pairs of extracted trees and field trees that refer to the same trees. The details and scripts of tree matching steps can be found in Xu et al. (2022).

Based on the tree matching results, we count the field trees $fts.num$, extracted trees $sts.num$, and matched trees $mts.num$, and then compute the *Completeness*, *Correctness* and *Mean Accuracy* following the exact definition used in the EuroSDR benchmark (Liang et al., 2018). In general, the larger values, the better results.

$$Completeness = \frac{mts.num}{fts.num},$$

$$Correctness = \frac{mts.num}{sts.num},$$

$$MeanAccuracy = \frac{2 * mts.num}{fts.num + sts.num}.$$

Table 2
Pearson correlation values between characteristics of experimental datasets and accuracy achieved by test methods. Columns ρ_{acc-p} and ρ_{acc-t} show the correlation of accuracy results on point density and stem density, respectively. Both ρ_{acc-p} and ρ_{acc-t} are computed on the results of Fire data and EuroSDR Multiple-scanned data to ensure the same TLS scanning type.

Method	ρ_{acc-p}	ρ_{acc-t}
TTS	0.399	−0.857
3D Forest (Trochta et al., 2017)	0.715	−0.958
FSCT (Krisanski et al., 2021a)	−0.219	−0.653

4.2.3. Results and discussions

The tree matching-based validation results, including completeness, correctness, and mean accuracy, are presented in Fig. 16(a), (b), and (c), respectively. Overall, we have comparable accuracy of detected tree locations. The average accuracy values of our TTS method, 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a) are 0.50, 0.25 and 0.54, respectively. Our accuracy is close to the best result of FSCT (Krisanski et al., 2021a). Also, our accuracy is almost twice that of 3D Forest (Trochta et al., 2017). In the Fire dataset (Greenberg et al., 2022), we observe that all test methods have better performance from after-fire data than before-fire data. Moreover, TTS has the highest accuracy in the most challenging areas from the EuroSDR dataset (Liang et al., 2018), scoring 1.29 times better than FSCT (Krisanski et al., 2021a) and 5.5 times better than 3D Forest (Trochta et al., 2017).

We also calculated Pearson correlations to investigate the correlations between segmentation accuracy and data characteristics (i.e., point density and stem density). The Pearson correlation value ranges between −1 and 1. There is no correlation if the correlation value is 0. If the correlation value is larger than zero, the correlation type is positive. Otherwise, it is a negative correlation.

Table 2 shows that our TTS method and 3D Forest (Trochta et al., 2017) have a strong positive correlation between point density and accuracy, with the correlation values of 0.4 and 0.7, respectively. In contrast, FSCT (Krisanski et al., 2021a) has a negative correlation value of −0.22. Please note that here a smaller point density means a denser point cloud. Thus, the correlation values indicate that tree measurement-focused FSCT (Krisanski et al., 2021a) works better at dense point clouds than in sparse areas. Because, compared to TTS and 3D Forest (Trochta et al., 2017), FSCT (Krisanski et al., 2021a) is more conservative and cares more about the correctness of detected trees that are thoroughly preserved in the point clouds. At the same time, TTS and 3D Forest (Trochta et al., 2017) focus on segmenting the forest point cloud into single tree point clouds. However, our TTS method is more resilient as our correlation value is about half of the 3D Forest (Trochta et al., 2017). Table 2 also shows all methods have a strong negative correlation between stem density and accuracy. Both Fig. 16 and Table 2 show that it is more difficult to keep high performance in dense and complicated forest areas. Despite that, our TTS achieves better performance in the densest "Difficult" areas than other methods. This result shows that our method is more stable and resilient to input data making our method more valuable. Thanks to our small α-complex that can outline trees comprehensively and our Forman theory-based segmentation that can investigate trees and split trees correctly, TTS can find more trees and have the largest completeness values among test methods in "Difficult" areas (see Fig. 16(a)). At the same time, there are also more noises in dense and multiple-layer forest areas, so not all extracted trees from our results are valid in the field data. For example, detected stems of small trees are not recorded in the field data. Similar observations are also reported in Wang's work (Wang, 2020). Thus TTS has a lower correctness value than FSCT (Krisanski et al., 2021a) in Fig. 16(b). However, considering both completeness and correctness, TTS has better accuracy than FSCT (Krisanski et al., 2021a) presented in Fig. 16(c).

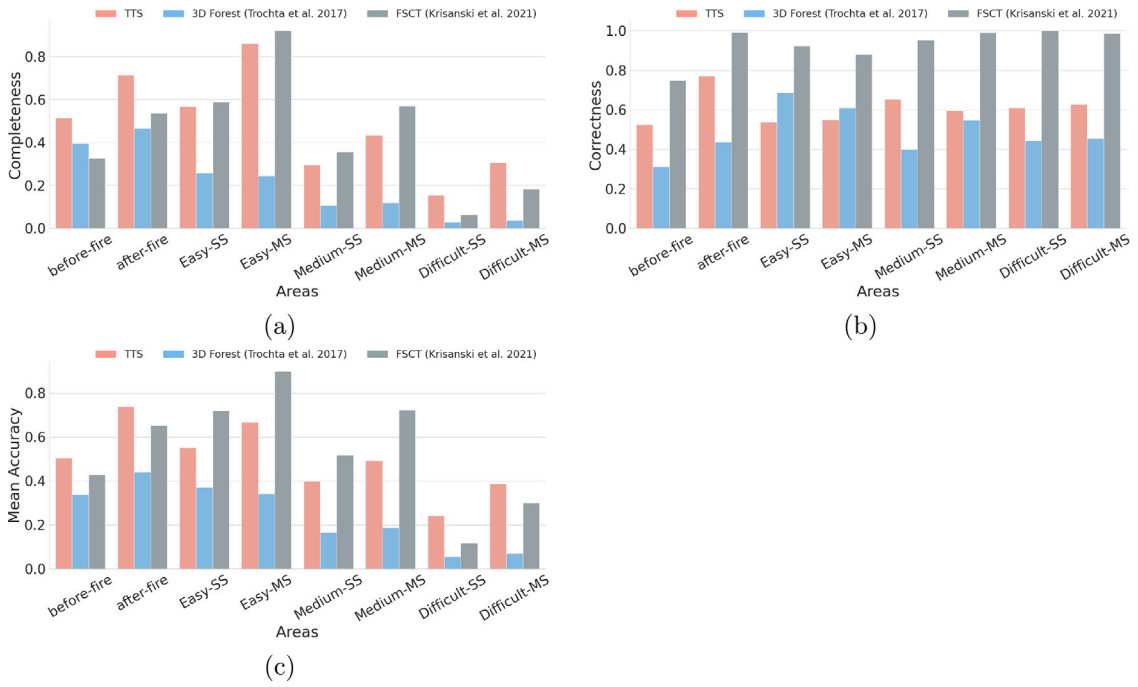


Fig. 16. Tree matching results of test methods performing on experimental datasets. (a), (b) and (c) show values of completeness, correctness, and mean accuracy, respectively.

Overall, our TTS method is robust to different forests. It is also stable in the forest after the wildfire. Moreover, automated TTS without parameter tuning achieves comparable results in the dataset and outperforms other methods in the most challenging areas, making TTS promising. Please note that 3D Forest (Trochta et al., 2017) is also tested in Liang et al. (2018) under the name of RILOG (see Appendix Section in Liang et al. (2018)). Even with the human effort (i.e., manually single tree detection), the accuracy of RILOG is still not very ideal, especially for single-scanned TLS point clouds (< 75%), which shows the difficulty of EuroSDR benchmarking datasets.

5. Setting parameters

Several internal parameters are used in TTS including values of α_s and α_l, tree validation threshold values th_{low} and th_{len}, and stem detection height th_{stem}. Those parameters are set based on theories and tested through experiments involving an eclectic mix of datasets.

Values of α_s and α_l. The value of α controls the 3D α-complex generated from points. TTS has two alpha values: a small value α_s and a large one α_l. Fixed values of α have been used to generate the α-complex of points P. For example, Bayer et al. (2013) suggest using 0.25 m while Hess et al. (2018) document better results using 0.5 m. We utilize existing ground truth data including tree locations (x,y) and DBHs to calculate boundary to boundary distances among tree stems. Given trees p_1, p_2 with DBHs d_1, d_2 respectively, the boundary to boundary distance is calculated as $dis_{bb} = distance(p_1, p_2) - (d_1 + d_2)/2$. We find that the average value of dis_{bb} can be lower than 0.1 m.

However, not all trees are well captured in TLS point clouds. Small trees are usually incompletely scanned due to occlusions (i.e., blocked by larger trees), whereas the stems of large trees are typically easier to model. At the same time, large trees have more points in the point cloud. Therefore, it is practical and critical to label large trees correctly in the segmentation results. We observe that the shape of tree with the boundary to boundary distance around 0.1 m has been persevered well in TLS point cloud. Furthermore, in the α-complex generated with a very small value of α (i.e., $\alpha < 0.1$ m), large tree stems can be over-segmented into multiple parts (see Fig. 18(b)). Considering factors

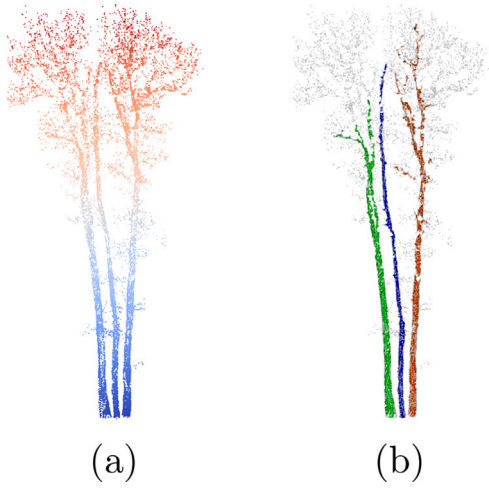


Fig. 17. Built α-complexes of point clouds with three close trees. (a) Input forest point cloud colored by height. (b) Generated 3D α-complex with $\alpha = 0.1$ m. Three valid components in the α-complex are colored.

mentioned above, it is important to acknowledge that we prioritized large trees in our validation experiments and set α_s to 0.1 m.

Fig. 17 shows one tree cluster point cloud including three trees. A 3D α-complex ($\alpha = 0.1$ m) is generated in Fig. 17(b). The clipped stem points are presented in Fig. 18(a), where the right stem points are not complete due to the common occlusion issue in TLS sampling. For the sake of simplicity and demonstration purposes, we present 2D α-complex generation results in Fig. 18, where built α-complexes with $\alpha = 0.05$ m, $\alpha = 0.10$ m, and $\alpha = 0.15$ m are presented in Fig. 18(b) Fig. 18(c) and Fig. 18(d), respectively. Fig. 18(b) shows that using a small value of α, the left tree stem points are split into multiple parts, while a large value of α, all stem points are combined into one shape, as presented in Fig. 18(d). However, an appropriate alpha value brings the precise stem shapes (see Fig. 18(c)) and outlines tree woody objects (i.e., trunks and branches) correctly (see Fig. 17(b)).

Fig. 18. Examples of built α-complexes of stem points from three close trees in Fig. 17. (a) Stem points with $p.h < 1$ m. Built 2D α-complexes of stem points with (b) $\alpha = 0.05$ m, (c) $\alpha = 0.1$ m, and (d) $\alpha = 0.15$ m.

The α-complex with value of α_l is generated to include all points, especially the sparse crown points from the input point cloud. Our method is independent of this value, as any large value should work, such as 0.25 m from Bayer et al. (2013) or 0.5 m from Hess et al. (2018). For simplicity, we fix α_l to be equal to 0.3 m in our implementation. Overall, α_s and α_l are applicable to the range of tree sizes and point cloud properties tested.

Threshold values of bottom elevation th_{low} and length th_{len}. Parameters th_{low} and th_{len} are designed to find valid trees by checking the bottom elevation and length of extracted point clouds. Recall that, tree length is defined as the elevation difference between the tree's highest point and the lowest point. Both parameters are used in two steps of TTS. The first usage is to check tree clusters after finding components in the generated α-complex (see Section 3.2). The other usage is to examine extracted single tree point clouds in Section 3.3. th_{low} is set to 1.5 m to help us get rid of "float" trees where the tree bottoms are far away from the ground level. Combined with $th_{len} = 2$ m, we can better avoid low-vegetation in our results.

Tree height threshold value th_{stem}. th_{stem} is used to find stems from the generated 3D α-complex of vegetation points (see Section 3.3). As tree bottoms are usually evident and reliable in dense TLS point clouds (Calders et al., 2020; Burt et al., 2019; Trochta et al., 2017), we set th_{stem} as 0.5 m to clip the α-complex below 0.5 m. The clipped α-complex components are considered tree stems.

6. Concluding remarks

We have proposed a general, robust, and automated *Topology-based Tree Segmentation (TTS)* approach for forest TLS point clouds with α-complex and Forman theory. Our plug-and-play method was also demonstrated to work on different out-of-box datasets without parameter tuning. However, testing across a wider range of forests, TLS instruments and survey properties is required.

We have designed experiments with multiple levels of validation. Labeled TLS point clouds have been utilized to examine the extracted single point clouds point-by-point via computing three metrics: Rand index, directional Hamming distance, and mean Intersection over Union (IoU). In the experiments, we have compared our TTS approach with open-source methods, 3D Forest (Trochta et al., 2017) and FSCT (Krisanski et al., 2021a). The results proved that TTS provides better accuracy regardless of forest types, stem densities, and point densities of input TLS point clouds. TTS achieves the highest average rand Index of 0.96, directional Hamming distance of 0.90, and mean IoU of 0.78, which indicate high-accurate segmentation results. Moreover, TTS requires no user-defined parameters and works directly in different forests and datasets. Together with the reliability and generality, TTS is promising for greater usage of TLS point clouds, such as fire risk and behavior modeling, estimating tree-level biodiversity structural traits, and above-ground biomass monitoring.

We plan to test TTS on more forests in different scanning designs in the future. Public TLS point clouds with labeled tree points are urgently needed for this aim. Considering the varying point density due to shadows and overlapped tree branches in TLS point clouds (Trochta et al., 2017), α-complex based on weighted Delaunay triangulation (Edelsbrunner, 2010) is under study with the scope of improving tree shape delineation and segmentation performance. Our region growing from the stems is also expected to benefit from metabolic scaling theory (West et al., 1997), which has also been utilized in the individual tree segmentation works in Tao et al. (2015) and Wang (2020). Lastly, the computational performance of our method is also being addressed in parallel work. Thanks to our divide-and-conquer strategy, parallel computation is naturally supported by our method and vastly improves computational performance. For scalability issue, we will use distributed data structures, like Stellar tree (Fellegara et al., 2021) to process large α-complexes built from big point clouds.

CRediT authorship contribution statement

Xin Xu: Conceptualization, Methodology, Software, Writing – original draft. **Federico Iuricich:** Supervision, Conceptualization, Writing – review & editing. **Kim Calders:** Resources, Writing – review & editing. **John Armston:** Writing – review & editing. **Leila De Floriani:** Supervision, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The experimental data are cited in the manuscript.

Acknowledgments

This work has been supported by the US National Science Foundation under grant number IIS-1910766. Inputs from John Armston were supported under NASA Biodiversity grant #20-BIODIV20-0089. We thank Dr. Ralph Dubayah, Dr. Hao Tang, and Dr. Laura Duncanson for their help and valuable comments. We also thank Dr. Jonathan Greenberg for providing Ferguson Fire Mensuration datasets.

References

Bayer, D., Seifert, S., Pretzsch, H., 2013. Structural crown properties of Norway spruce (Picea abies [L.] Karst.) and European beech (Fagus sylvatica [L.]) in mixed versus pure stands revealed by terrestrial laser scanning. Trees 27 (4), 1035–1047.

Burt, A., Disney, M., Calders, K., 2019. Extracting individual trees from lidar point clouds using treeseg. Methods Ecol. Evol. 10 (3), 438–445.

Calders, K., Adams, J., Armston, J., Bartholomeus, H., Bauwens, S., Bentley, L.P., Chave, J., Danson, F.M., Demol, M., Disney, M., et al., 2020. Terrestrial laser scanning in forest ecology: Expanding the horizon. Remote Sens. Environ. 251, 112102.

Calders, K., Newnham, G., Burt, A., Murphy, S., Raumonen, P., Herold, M., Culvenor, D., Avitabile, V., Disney, M., Armston, J., et al., 2015. Nondestructive estimates of above-ground biomass using terrestrial laser scanning. Methods Ecol. Evol. 6 (2), 198–208, https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.12301.

Calders, K., Verbeeck, H., Burt, A., Origo, N., Nightingale, J., Malhi, Y., Wilkes, P., Raumonen, P., Bunce, R.G.H., Disney, M., 2022. Laser scanning reveals potential underestimation of biomass carbon in temperate forest, Ecol. Solutions Evidence (in review).

De Floriani, L., Fugacci, U., Iuricich, F., Magillo, P., 2015. Morse complexes for shape segmentation and homological analysis: discrete models and algorithms. In: Computer Graphics Forum. Vol. 34, (2), Blackwell Publishing Ltd, pp. 761–785. http://dx.doi.org/10.1111/cgf.12596.

Delaunay, B., et al., 1934. Sur la sphere vide. Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii I Estestvennyka Nauk 7 (793–800), 1–2.

Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q., 2019. Centernet: Keypoint triplets for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6569–6578.

Edelsbrunner, H., 2010. Alpha shapes—a survey. Tessellations Sci. 27, 1–25.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. Vol. 96, (34), pp. 226–231.

Fan, H., Zhu, N., Dong, Z., et al., 2022. A Two-stage Approach for Individual Tree Segmentation from TLS Point Clouds. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens..

Fellegara, R., luricich, F., De Floriani, L., Weiss, K., 2014. Efficient Computation and Simplification of Discrete Morse Decompositions on Triangulated Terrains. In: Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. SIGSPATIAL '14, ACM, New York, NY, USA, pp. 223–232. http://dx.doi.org/10.1145/2666310.2666412.

Fellegara, R., Weiss, K., De Floriani, L., 2021. The Stellar decomposition: A compact representation for simplicial complexes and beyond. Comput. Graph. 98, 322–343.

Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: A paradigm for model fitting with. Commun. ACM 24, 381–395.

Forman, R., 1998. Morse theory for cell complexes. Adv. Math. 134, 90–145.

Greenberg, J.A., et al., 2022. Global environmental analysis and remote sensing (GEARS) laboratory. https://www.gearslab.org/.

Guylassy, A., Bremer, P.-T., Hamann, B., Pascucci, V., 2008. A Practical Approach to Morse-Smale Complex Computation: Scalability and Generality. IEEE Trans. Vis. Comput. Graph. 14 (6), 1619–1626. http://dx.doi.org/10.1109/TVCG.2008.110.

Hackenberg, J., 2022. Simpleforest - a tree modelling software. https://www.simpleforest.org/.

Hackenberg, J., Spiecker, H., Calders, K., Disney, M., Raumonen, P., 2015a. SimpleTree—an efficient open source tool to build tree models from TLS clouds. Forests 6 (11), 4245–4294.

Hackenberg, J., Wassenberg, M., Spiecker, H., Sun, D., 2015b. Non destructive method for biomass prediction combining TLS derived tree volume and wood density. Forests 6 (4), 1274–1300.

Hess, C., Härdtle, W., Kunz, M., Fichtner, A., von Oheimb, G., 2018. A high-resolution approach for the spatiotemporal analysis of forest canopy space using terrestrial laser scanning data. Ecol. Evol..

Huang, Q., Dom, B., 1995. Quantitative methods of evaluating image segmentation. In: Proceedings., International Conference on Image Processing. Vol. 3, pp. 53–56. http://dx.doi.org/10.1109/ICIP.1995.537578.

Jaccard, P., 1908. Nouvelles recherches sur la distribution florale. Bull. Soc. Vaud. Sci. Nat. 44, 223–270.

Krisanski, S., Taskhiri, M.S., Gonzalez Aracil, S., Herries, D., Muneri, A., Gurung, M.B., Montgomery, J., Turner, P., 2021a. Forest Structural Complexity Tool—An Open Source, Fully-Automated Tool for Measuring Forest Point Clouds. Remote Sens. 13 (22), 4677.

Krisanski, S., Taskhiri, M.S., Gonzalez Aracil, S., Herries, D., Turner, P., 2021b. Sensor agnostic semantic segmentation of structurally diverse and complex forest point clouds using deep learning. Remote Sens. 13 (8), 1413.

Krisanski, S., Taskhiri, M.S., Gonzalez Aracil, S., Herries, D., Turner, P., 2022. Forest Structural Complexity Tool (FSCT) software. https://github.com/SKrisanski/FSCT.

Liang, X., Hyyppä, J., Kaartinen, H., Lehtomäki, M., Pyörälä, J., Pfeifer, N., Holopainen, M., Brolly, G., Francesco, P., Hackenberg, J., et al., 2018. International benchmarking of terrestrial laser scanning approaches for forest inventories. ISPRS J. Photogramm. Remote Sens. 144, 137–179.

Martin-Ducup, O., Mofack, G., Wang, D., Raumonen, P., Ploton, P., Sonké, B., Barbier, N., Couteron, P., Pélissier, R., 2021. Evaluation of automated pipelines for tree and plot metric estimation from TLS data in tropical forest areas. Ann. Botany.

McInnes, L., Healy, J., Astels, S., 2017. Hdbscan: Hierarchical density based clustering. J. Open Source Softw. 2 (11), 205.

Milnor, J.W., 1963. Morse Theory. Princeton University Press, New Jersey, p. vi+153.

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Adv. Neural Inf. Process. Syst. 30.

Rand, W.M., 1971. Objective criteria for the evaluation of clustering methods. J. Amer. Statist. Assoc. 66 (336), 846–850.

Raumonen, P., Casella, E., Calders, K., Murphy, S., Åkerblom, M., Kaasalainen, M., 2015. Massive-scale tree modelling from TLS data. ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci. 2 (3), 189.

Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., Lewis, P., 2013. Fast automatic precision tree models from terrestrial laser scanner data. Remote Sens. 5 (2), 491–520.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. 28.

Robins, V., Wood, P.J., Sheppard, A.P., 2011. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. IEEE Trans. Pattern Anal. Mach. Intell. 33 (8), 1646–1658.

Rusu, R.B., Cousins, S., 2011. 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China.

Shivashankar, N., Natarajan, V., 2012. Parallel Computation of 3D Morse-Smale Complexes. Comput. Graph. Forum 31 (3pt1), 965–974. http://dx.doi.org/10.1111/j.1467-8659.2012.03089.x.

Shivashankar, N., Senthilnathan, M., Natarajan, V., 2012. Parallel Computation of 2D Morse-Smale Complexes. IEEE Trans. Vis. Comput. Graph. 18 (10), 1757–1770. http://dx.doi.org/10.1109/TVCG.2011.284.

Tao, S., Wu, F., Guo, Q., Wang, Y., Li, W., Xue, B., Hu, X., Li, P., Tian, D., Li, C., et al., 2015. Segmenting tree crowns from terrestrial and mobile LiDAR data by exploring ecological theories. ISPRS J. Photogramm. Remote Sens. 110, 66–76.

Trochta, J., Krucek, M., Vrska, T., Kral, K., 2017. 3D Forest: An application for descriptions of three-dimensional forest structures using terrestrial LiDAR. PLoS One 12 (5), e0176871.

Trochta, J., Krucek, M., Vrska, T., Kral, K., 2022a. 3D Forest Sample data. https://github.com/VUKOZ-OEL/3dforest-data.

Trochta, J., Krucek, M., Vrska, T., Kral, K., 2022b. 3D Forest software. https://github.com/VUKOZ-OEL/3d-forest-classic.

Wang, D., 2020. Unsupervised semantic and instance segmentation of forest point clouds. ISPRS J. Photogramm. Remote Sens. 165, 86–97.

Wang, J., Chen, X., Cao, L., An, F., Chen, B., Xue, L., Yun, T., 2019. Individual rubber tree segmentation based on ground-based LiDAR data and faster R-CNN of deep learning. Forests 10 (9), 793.

Weiss, K., Iuricich, F., Fellegara, R., De Floriani, L., 2013. A primal/dual representation for discrete morse complexes on tetrahedral meshes. Comput. Graph. Forum 32 (3 PART3), 361–370. http://dx.doi.org/10.1111/cgf.12123.

West, G.B., Brown, J.H., Enquist, B.J., 1997. A general model for the origin of allometric scaling laws in biology. Science 276 (5309), 122–126.

Wilkes, P., Lau, A., Disney, M., Calders, K., Burt, A., de Tanago, J.G., Bartholomeus, H., Brede, B., Herold, M., 2017. Data acquisition considerations for terrestrial laser scanning of forest plots. Remote Sens. Environ. 196, 140–153.

Xi, Z., Hopkinson, C., 2021. Detecting individual-tree crown regions from terrestrial laser scans with an anchor-free deep learning model. Can. J. Remote Sens. 47 (2), 228–242.

Xu, X., Iuricich, F., De Floriani, L., 2022. Individual Tree Segmentation of Forest Point Clouds. https://osf.io/hztw9/?view_only=617f20f68ac843609f814f235e2a0388.

Zhen, Z., Quackenbush, L.J., Zhang, L., 2016. Trends in automatic individual tree crown detection and delineation—Evolution of LiDAR data. Remote Sens. 8 (4), 333.