forests

MDPI

Article

Improved Tree Segmentation Algorithm Based on Backpack-LiDAR Point Cloud

Dongwei Zhu [1,2,3,†], Xianglong Liu [4,†], Yili Zheng [1,2,3,5,*], Liheng Xu [4,*] and Qingqing Huang [1,2]

[1] School of Technology, Beijing Forestry University, Beijing 100083, China; zhudongwei0318@bjfu.edu.cn (D.Z.); huangqingqing@bjfu.edu.cn (Q.H.)
[2] Key Laboratory of National Forestry and Grassland Administration on Forestry Equipment and Automation, Beijing 100083, China
[3] Institute of Intelligent Sensing for Ecological Carbon Neutrality in Forestry and Grassland, Beijing Forestry University, Beijing 100083, China
[4] Qingyang Forestry Science Research Institute, Qingyang 745000, China; lxl20221122@163.com
[5] State Key Laboratory of Efficient Production of Forest, Beijing 100083, China
* Correspondence: zhengyili@bjfu.edu.cn (Y.Z.); xlh8017@163.com (L.X.)
† These authors contributed equally to this work.

Abstract: For extracting tree structural data from LiDAR point clouds, individual tree segmentation is of great significance. Most individual tree segmentation algorithms miss segmentation and misrecognition, requiring manual post-processing. This study utilized a hierarchical approach known as segmentation based on hierarchical strategy (SHS) to improve individual tree segmentation. The tree point cloud was divided into the trunk layer and the canopy layer to carry out trunk detection and canopy segmentation, respectively. The effectiveness of SHS was evaluated on three mixed broadleaf forest plots. The segmentation efficacy of SHS was evaluated on three mixed broadleaf forest plots and compared with the point cloud segmentation algorithm (PCS) and the comparative shortest-path algorithm (CSP). In the three plots, SHS correctly identified all the trunk portion, had a recall (r) of 1, 0.98, and 1, a precision (p) of 1, and an overall segmentation rate (F) of 1, 0.99, and 1. CSP and PCS are less accurate than SHS. In terms of overall plots, SHS had 10%–15% higher F-scores than PCS and CSP. SHS extracted crown diameters with R^2s of 0.91, 0.93, and 0.89 and $RMSE$s of 0.24 m, 0.23 m, and 0.30 m, outperforming CSP and PCS. Afterwards, we evaluate the three algorithms' findings, examine the SHS algorithm's parameters and constraints, and discuss the future directions of this research. This work offers an enhanced SHS that improves upon earlier research, addressing missed segmentation and misrecognition issues. It improves segmentation accuracy, individual tree segmentation, and provides both theoretical and data support for the LiDAR application in forest detection.

Keywords: LiDAR; point cloud; trunk detection; crown clustering; individual tree segmentation; Backpack-LiDAR; forestry inventory

Citation: Zhu, D.; Liu, X.; Zheng, Y.; Xu, L.; Huang, Q. Improved Tree Segmentation Algorithm Based on Backpack-LiDAR Point Cloud. *Forests* **2024**, *15*, 136. https://doi.org/10.3390/f15010136

Academic Editor: Nikolay Strigul

Received: 23 November 2023
Revised: 13 December 2023
Accepted: 22 December 2023
Published: 9 January 2024

1. Introduction

Forest resources are one of the most important natural resources on Earth, serving functions such as preventing wind, fixing sand, conserving water, and maintaining ecological balance. The traditional forest inventory algorithm requires a lot of labor and material resources, while at the same time necessitating the damage or even cutting down of the forest [1]. In the last two decades, the application of LiDAR (Light Detection and Ranging) as an active remote sensing technology in forestry resource investigation has become more and more extensive. Compared with traditional optical remote sensing algorithms, LiDAR-acquired point cloud data can accurately acquire the vertical structure of forests and obtain accurate tree structural parameters, such as tree position, tree height, diameter at breast height (DBH), crown diameter, etc. [2,3], which are useful for extracting

quantitative structural models (QSM) [4], as well as calculating forest biomass and forest carbon stocks [5].

Individual segmentation is an important prerequisite for extracting structural parameters of forest trees, and individual tree segmentation algorithms have also been the main direction of forest LiDAR point clouds research in recent years [6]. Early individual tree segmentation algorithms have been developed based on airborne LiDAR point cloud data. Wallace, L. et al. developed an airborne LiDAR system [7], and, at the same time, by identifying individual trees in the study area and estimating forest survey indicators at the plot scale and individual tree scale, data such as tree height, individual tree position [8], and canopy diameter were extracted, and the results of the survey were obtained with higher accuracy. The individual tree identification algorithm of airborne LiDAR is mainly based on the rasterized canopy height model (CHM), using the local maximum search algorithm to determine the individual tree position, and applying the segmentation algorithm to split the individual trees [9]. Popescu, S.C. et al. introduced a novel approach, referred to as variable-window individual tree identification, which is based on the linear regression connection equation between the measured height of trees and their crown diameter [10]. In their study, Koch, B. et al. put forward a method for identifying potential locations of treetops by employing a local maximum filter on the canopy height model (CHM) [11]. They subsequently utilized a watershed segmentation algorithm to delineate the canopy, enabling the automated extraction of individual trees. The findings of their research demonstrated that the application of an image segmentation algorithm on the CHM yielded superior performance in coniferous forest stands. However, it should be noted that in broadleaf forest stands with significant depression, this approach may lead to the merging of canopies. Li, W. et al. introduced a novel technique called the individual tree segmentation technique (PCS) [12]. This approach employs the local maximum algorithm to identify seed points as canopy vertices and utilizes the top-down region growth algorithm to separate individual trees. The results of their evaluation indicate that the PCS algorithm achieves an impressive overall accuracy of 90%.

However, airborne LiDAR acquires relatively low point densities (usually a few to tens of points per square meter) and a small number of individual tree point clouds, and the top-down data acquisition algorithm makes it difficult to acquire points from trunks and branches, limiting its further research and application at the single-plant scale [13,14]. Compared to airborne LiDAR (UAV-LiDAR), which can acquire large-scale forest point clouds above the canopy, terrestrial LiDAR (T-LiDAR) and mobile LiDAR (M-LiDAR) are able to penetrate deeper into the understory. T-LiDAR and M-LiDAR use side-view scanning modes with high point densities (usually several hundred to thousands of points per square meter) [15,16] and can obtain rich tree-side information, such as stems, branches, and even internal canopy details [17], which is very important for individual tree scales. Due to the aforementioned advantages, more and more studies focus on the individual tree segmentation of T-LiDAR or M-LiDAR point clouds. Tao, S. et al., addressing the problem of how difficult it is to segment the canopy of TLS and MLS point cloud data, proposed an individual tree segmentation algorithm for canopy segmentation, the comparative shortest path (CSP) algorithm [18]. The segmentation accuracy of the algorithm in broadleaf and coniferous forest plots far exceeds that of the PCS algorithm. However, the algorithm is only applicable to forest point clouds with simpler understory environments, and under more complex forest point cloud plot conditions, the algorithm suffers from misidentification. Thus, it identifies non-tree point cloud targets as segmentation targets and has even lower accuracy for the less accurate M-LiDAR point clouds. M-LiDAR can be categorized into vehicle-based scanning, hand-held, and other personal laser scanning technologies according to the scanning algorithm, including backpack-based laser scanning (BLS) [19]. M-LiDAR is more suitable for point cloud data acquisition on small and medium-sized complex plots than T-LiDAR. The difference in principle and density leads to the poor performance of individual tree segmentation algorithms based on UAV-LiDAR and T-LiDAR point clouds when applied to M-LiDAR point cloud data, leading to problems

such as missed segmentation and misrecognition. Comesaña-Cebral, L. et al. proposed an algorithm to segment the Backpack-LiDAR point cloud using the DBSCAN algorithm with a cylinder voxelization algorithm. This approach achieved a segmentation rate close to 90%, but encountered a case of individual tree over-segmentation in the traditional algorithm [20]. Liu, L. et al. successfully segmented 140 individual trees in a natural forest plot using a relative density segmentation algorithm, but there were eight missed segmentations and five misidentified trees [21].

In order to improve the accuracy of individual tree segmentation, as well as the precision of extracting tree parameters, and to reduce the problems of missed segmentation and misrecognition in individual tree segmentation, an improved individual tree segmentation algorithm is proposed in this study. This algorithm has been developed on the basis of the hierarchical idea and based on the characteristics of the height change of the vertical structure of the forest. The individual tree segmentation is divided into two major parts, including the trunk detection in the lower layer and the crown clustering in the upper layer. This process, achieved through point cloud layering, enables accurate identification of the trunk part of trees and the clustering of crowns from different trees. This method exhibits high segmentation accuracy in the region where the crowns are more closely overlapped and, at the same time, it has strong robustness. Additionally, it is able to exclude point cloud targets other than trees, avoiding the problem of misidentification. Evaluating the accuracy of segmentation results using tree-level evaluation metrics as well as point-level evaluation metrics.

2. Materials and Methods

2.1. Study Area and Data Acquisition

The data collection site is located in Ximazhuang Park (39°55′ N, 116°37′ E), Tongzhou District, Beijing, China, and the researchers in this paper scanned the entire park during September–October 2022 using Backpack-LiDAR. Data acquisition was conducted by the researcher with the Backpack-LiDAR in the plot, following an approximate S-shaped route. This approach ensured the collection of tree information from multiple angles, thereby reducing occlusion, as well as noise interference. At the same time, the route was designed in order to ensure the establishment of an accurate point cloud map. The route concluded by returning to the starting point to close the loop. The study encompassed a total area of approximately 3200 m^2, consisting of two plots of 30 × 30 m^2 each, which were representative of mixed broadleaf forests, and one irregularly shaped plot with an area of around 1400 m^2. The total number of trees in the plots, as well as parameters such as diameter at breast height (DBH), tree height, coordinates, and crown diameter of each tree were measured manually. The tree species in the plots mainly contain catalpa, ginkgo, phacelia, and golden-leaf elm. The plots contain a tree layer, a shrub layer, and an herb layer, with a relatively regular distribution of individual trees and high canopy overlap (Figure 1).

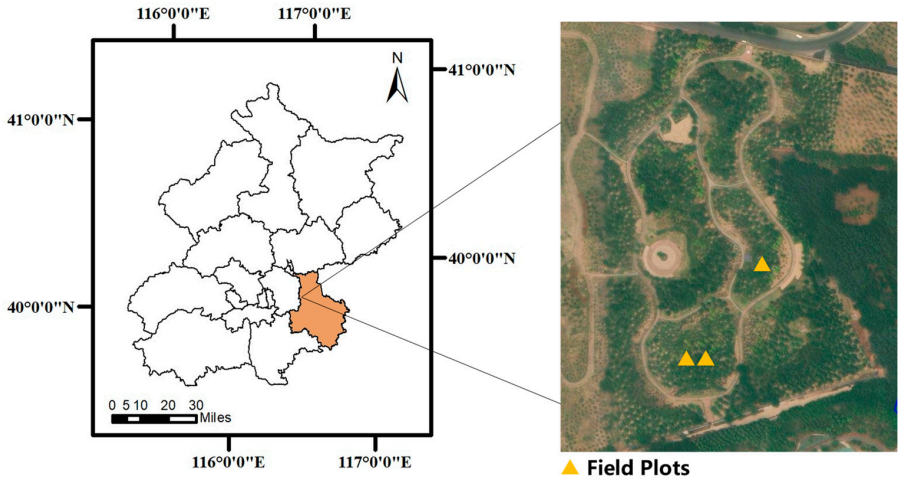

Figure 1. Location map and satellite images of Ximazhuang Park.

The Backpack-LiDAR scanning system used to collect data is mainly composed of three parts: the LiDAR is RoboSense's RS-LiDAR-16, the IMU is ALUBI's 9-axis high-precision attitude sensor LPMS-IG1, and the main control machine is selected from the Intel NUC11PAHi50Z. The LiDAR wavelength is 905 nm, the horizontal field of view is 360°, the ranging capability is 150° (80m@10% NIST), the horizontal resolution is 0.1°, the vertical field of view is 30°, and the vertical angle resolution is 2.0°. The maximum output frequency of the IMU is 500 Hz, and the open-source SLAM algorithm is used to build a 3D point cloud map. The specifications of the LiDAR and the IMU are shown in detail in Table 1.

Table 1. The specification of LiDAR and IMU.

Name	Content	Parameter
RS-LiDAR-16	Field of View	Horizontal: 360° Vertical: 30°
	Detection Range	Up to 150 m
	Laser Wavelength	905 nm
	Scanning Accuracy	±2 cm
	Resolution	Horizontal: 0.1°/0.2°/0.4° Vertical: 2.0°
LPMS-IG1	Attitude Angle Range	Roll: ±180°; Pitch: ±90°; Yaw: ±180°;
	Dual Gyroscope Parameters	#1: 3-axis, ±400, 24 bits #2: ±1000/±2000 dps, 16 bits
	Magnetic Field Sensor Parameters	3-axis, ±2/±8 gauss, 16 bits
	Output Frequency	5~500 Hz
	Resolution	0.01°

2.2. Individual Tree Segmentation Based on a Hierarchical Strategy

We propose an individual tree segmentation algorithm based on hierarchical strategy (SHS), which mainly includes three parts: data preprocessing, trunk detection, and canopy clustering. Figure 2 shows the specific process of individual tree segmentation. Firstly, data preprocessing is performed on the initial plot point cloud data, including downsampling, statistical outlier removal (SOR), cloth simulation filtering (CSF), and point cloud normalization. Secondly, the pre-processed point clouds are stratified via pass-through filtering to obtain trunk slices and crown slices, which are used as inputs for trunk detection and crown clustering, respectively. Next, the DBSCAN algorithm is used to cluster the trunk slices, perform trunk detection, and calculate the trunk center of mass. A KDTree structure is built for the crown slices. The nearest-neighbor search is used to build the undirected graphs of the crown slices and the trunk centroid. A search table is built, and the crown point cloud is clustered through the heap-optimized Dijkstra's algorithm. Finally, the clustered tree trunks and crowns are combined one by one to complete the individual tree segmentation.

2.3. Data Preprocessing

In order to ensure the accuracy of individual tree segmentation and to increase the computational speed of the algorithm, this study performed a series of data preprocessing operations to remove noise from the plots' clouds, reduce the number of point clouds in the plots, and highly normalize the plots' clouds. In order to reduce the impact of the huge number of point clouds on the data complexity of the segmentation algorithm, this study used the random sampling algorithm [22] to accomplish data downsampling. This algorithm accomplishes sampling by randomly selecting the number of point clouds within a threshold range, where each point in the point cloud has the same probability of being sampled, and random sampling does not change the original positional distribution of the point cloud.

A statistical outlier removal (SOR) algorithm was used to reduce the impact of noise on the segmentation accuracy of the sample point cloud (especially in the canopy). The SOR algorithm statistically analyzes the neighborhood of each point in the point cloud and

calculates the average distance from the point to the neighboring points, which is defined by the following equation:

$$\overline{d(x_i)} = \frac{1}{n}\sum_{j=1}^{n} d(x_i, x_{i,j}) \tag{1}$$

$$S_i = \sqrt{\frac{\sum_{j=1}^{n}\left(d(x_i, x_{i,j}) - \overline{d(x_i)}\right)^2}{n-1}} \tag{2}$$

where $x_{i,j}$ is a point in the neighborhood and $d(x_i, x_{i,j})$ is the distance from point x_i to $x_{i,j}$. Assuming that the distance from the point x_i to all points in its neighborhood follows a Gaussian distribution $N(\mu, \sigma^2)$, the average distance $\overline{d(x_i)}$ is the mean of the Gaussian distribution μ, the sample standard deviation S_i is the standard deviation of the Gaussian distribution σ, and the point x_i, whose distance is outside the standard range $(\mu - \sigma, \mu + \sigma)$, is defined as a noisy point and removed from the point cloud. Downsampling and SOR filtering change the point cloud numbers of Plot 1 and Plot 2 from 583,072 and 467,612 to 257,431 and 212,342, respectively, and their densities from 42.54 pts/m^3 to 19.02 pts/m^3 and from 32.11 pts/m^3 to 15.66 pts/m^3. Plot3's point cloud number is changed from 1,532,492 to 656,870 and its density is changed from 64.46 pts/m^3 to 29.88 pts/m^3.

The cloth simulation filtering (CSF) algorithm [23] is employed for the purpose of segmenting ground and non-ground point clouds. The CSF method leverages the inherent characteristics of the fabric to adjust the point cloud filtering process by adjusting the simulated physical processes associated with the fabric. Equation (2) in Newton's second law provides a direct relationship between the force applied to the fabric particle and its position. The researchers analyze the force exerted on the particle by dividing it into two separate stages. Then, they calculate Equation (2) without considering any internal force in order to obtain Equation (3). The positions of the fabric particles can be determined by solving Equation (3). The following equations are displayed: Equations (2) and (3).

$$m\frac{\partial P(t)}{\partial t^2} = F_{ext}(P,t) + F_{int}(P,t) \tag{3}$$

$$P(t+\Delta t) = 2P(t) - P(t-\Delta t) + \frac{G}{m}\Delta t^2 \tag{4}$$

where m is the particle mass (usually m is set to 1), P represents the position of the particle at time t, $F_{ext}(P,t)$ is the external force on the particle, $F_{int}(P,t)$ is the internal force on the particle, Δt is the time interval, and G is the gravitational constant.

This approach yields very accurate filtering results with minimal parameter settings. The proposed methodology involves the initial inversion of the input point cloud, followed by the subsequent placement of a textile material over the inverted surface. The determination of the final shape of the cloth and subsequent classification of the original points into ground and non-ground components can be achieved by the analysis of the interactions between the nodes of the fabric and the related LiDAR points. In certain instances, the simulated cloth may serve as the ultimate Digital Terrain Model (DTM) that is generated, thereby bypassing the need for interpolating ground points and effectively restoring areas where data are absent. Figure 3 illustrates the outcomes of the segmentation process, specifically the differentiation between ground points and non-ground points by the utilization of CSF filtering.

The height normalization of the point cloud uses the simulated fabric generated by the CSF algorithm as the DTM, and the normalized point cloud height is obtained by subtracting the DTM height from the original point cloud height. After normalization, the height value of the point cloud indicates the relative height from zero to the point. The step of height normalization does not change the spatial distribution of trees and branches [18]. Figure 4 shows the comparison of the point cloud before and after height normalization.

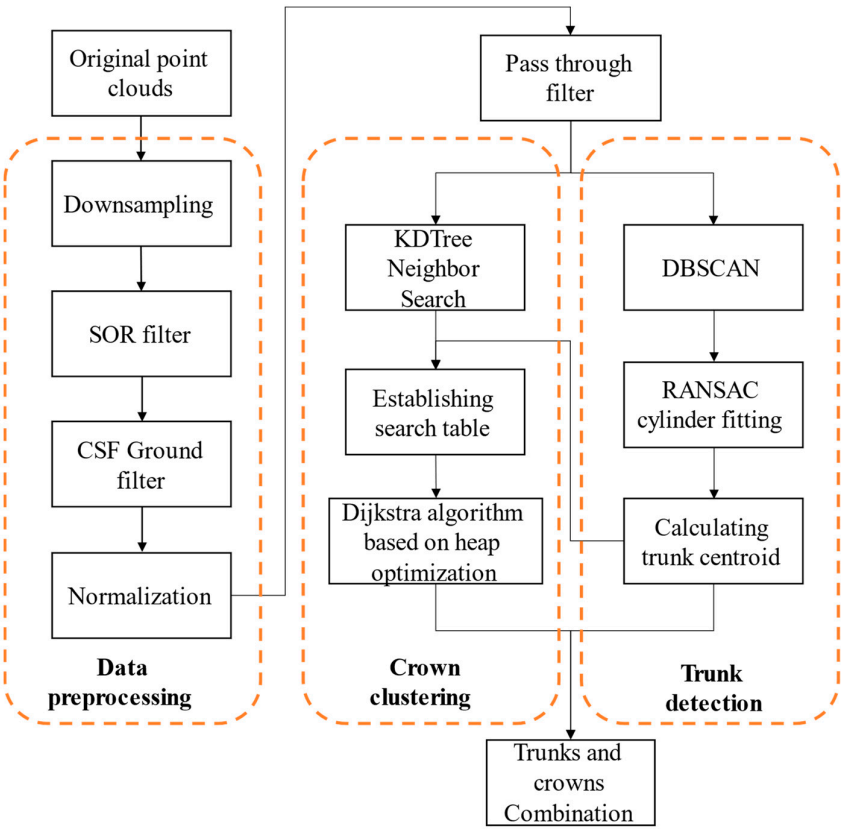

Figure 2. Flowchart of individual segmentation algorithm based on hierarchical strategy (SHS).

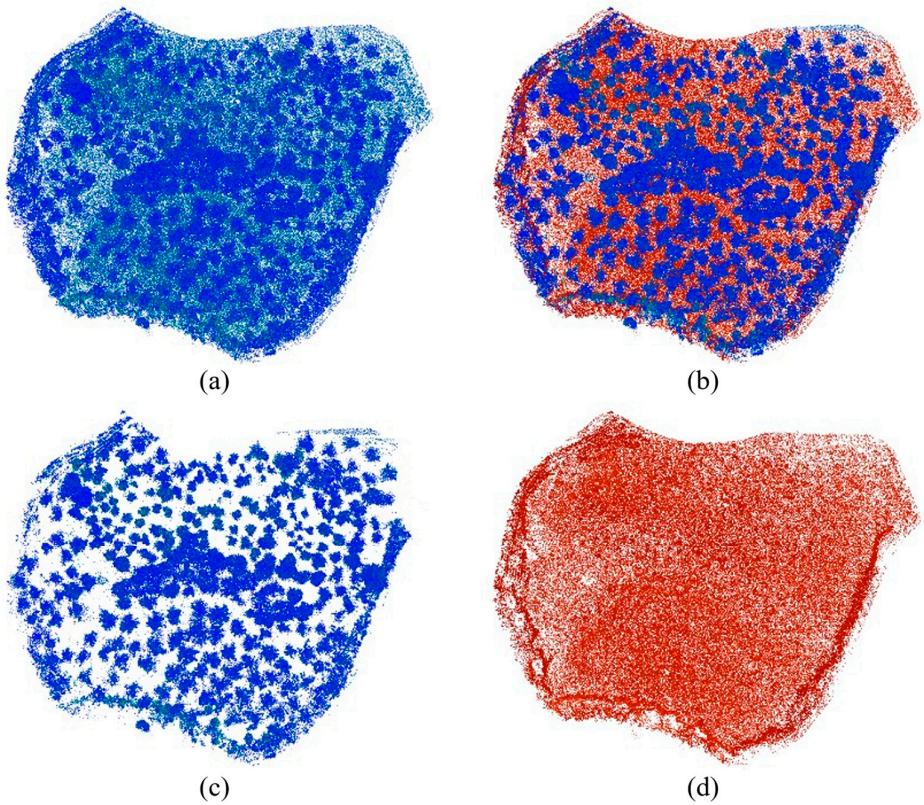

Figure 3. Using CSF filtering to split the ground and non-ground points in the plot in which Plot 1 and Plot 2 are located. (**a**) is the initial point cloud, (**b**) is the point cloud after CSF filtering, (**c**) is the filtered ground point cloud, and (**d**) is the filtered ground point cloud.

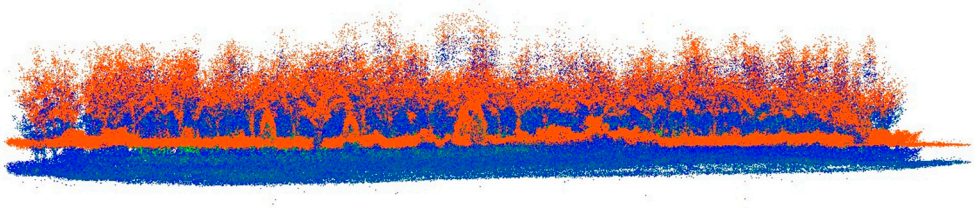

Figure 4. Point cloud comparison before and after point cloud normalization. Red: point cloud after normalization. Blue: point cloud before normalization.

2.4. Individual Tree Segmentation

2.4.1. Point Cloud Stratification

In order to segment the tree trunk and tree crown separately, this paper uses the straight pass filtering algorithm to layer the pre-processed point cloud, which is divided into a trunk point cloud layer and a crown point cloud layer by setting the height field and a fixed height value. The trunk point cloud layer contains most of the tree trunk points, and the crown point cloud layer contains all the tree crown points and some of the tree trunk points.

2.4.2. Trunk Detection

The density-based spatial clustering of applications with noise (DBSCAN) algorithm is able to discover sample points with similar density and cluster them. Compared to the K-means algorithm [24], DBSCAN does not need to pre-determine the number of clusters, is less susceptible to noise, and is able to cluster clusters of arbitrarily shaped point clouds [25]. The definition of the DBSCAN algorithm is as follows: neighborhood radius ε and the minimum number of points MinPts are the criteria used by the DBSCAN algorithm to describe density. Here, ε denotes the range of a sample point's neighborhood, and the minimum number of samples MinPts denotes the smallest number of samples within that neighborhood. DBSCAN determines the type of a sample point based on the density of the sample points. When a sample point has a number of sample points within the neighborhood radius ε that is greater than or equal to the number of MinPts, the points are classified as core points. Points that are not core points but are within the neighborhood of a core point are classified as boundary points, and points that are neither core nor boundary points are classified as noise points. When a sample point is contained in a neighborhood ε of a core point, the sample point is said to be density-direct to that core point; if there is a sequence $x_{i,2}, \cdots, x_{i,n-1}$ between the sample point $x_{i,1}$ and the sample point $x_{i,n}$ and $x_{i,j+1}$ is density-direct from $x_{i,j}$, then $x_{i,n}$ is said to be density accessible from $x_{i,1}$.

The process of the DBSCAN algorithm is as follows: According to the neighborhood radius ε and the minimum number of points MinPts, calculate all the core points in the sample. Starting from these core points, extend the clustering by considering the direct density of reachable sample points to be added to the current clusters. Traverse all sample points, excluding the unvisited points and the points that are not clustered, which are considered as outlier points.

Due to the presence of low shrubs and other non-trunk point clouds in the understory, in order to exclude the interference of non-trunk point clouds, this paper uses the cylindrical fitting algorithm of Random Sample Consensus (RANSAC) [26], combining the height of the point cloud with the number of point clouds as a constraint to further test the clustering results of DBSCAN. The RANSAC algorithm was first proposed in 1981 by Fischler and Bolles as an iterative algorithm for fitting data models for linear fitting, planar fitting, and cylindrical fitting problems. The general cylindrical equation can be expressed as:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = \frac{[l(x - x_0) + m(y - y_0) + n(z - z_0)]^2}{l^2 + m^2 + n^2} \quad (5)$$

where (x_0, y_0, z_0) is a point on the cylindrical axis L, (l, m, n) is the direction vector of the cylindrical axis L, and r is the radius of the cylinder. In this algorithm, the clustering results of DBSCAN are used as inputs. The RANSAC cylindrical fitting algorithm is used to randomly extract three points from the input point cloud to solve x_0, y_0, z_0, l, m, n. A cylindrical model is determined, and the deviation of all the input point clouds from the cylindrical model is calculated. The process is repeated until the desired error value is reached or the specified number of iterations is reached. The final cylinder fitting is accomplished with the output model parameters. Continue to set the point cloud with model parameter output as a pending trunk point cloud, and the point cloud without cylindrical parameter output as non-trunk point clouds. Take 90% of the height of the trunk layer and 90% of the average value of the number of points of the input point cloud as constraints to obtain the target trunk point cloud, and calculate the center of mass of each trunk point cloud as the starting point of the tree crown clustering.

Calculate the centroid of each trunk as the starting point for canopy clustering. Use the KDTree to accelerate the creation of a point cloud undirected graph of tree crowns and build a search table by adding the trunk centroids to the undirected graph through nearest neighbor search. Use the centroid of each trunk as the starting point to establish the core of the canopy clustering and use the Dijkstra algorithm with heap optimization to calculate the shortest distance from each point in the canopy layer to the starting point, and then use this information to classify the clusters.

The canopy point cloud is stored using an adjacency table structure. The adjacency table has a lower space complexity than the adjacency matrix, which is suitable for scenarios with a large amount of point cloud data. By traversing each point in the tree canopy layer, searching for the nearest neighbor points within the neighborhood of the point, and calculating the distance between the points using the Euclidean distance as the distance calculation algorithm, the formula is as follows:

$$d(p_i, p_j) = \sqrt{\left(x_{p_i} - x_{p_j}\right)^2 + \left(y_{p_i} - y_{p_j}\right)^2 + \left(z_{p_i} - z_{p_j}\right)^2},\ i \neq j \tag{6}$$

where $p_i = (x_i, y_i, z_i)$ and $p_j = (x_j, y_j, z_j)$. The neighbor edges of p_i and p_j are established with distance $d(p_i, p_j)$ as the weight of the adjacency table. In order to add the trunk center of mass to the undirected graph, first add the trunk center of mass points to the canopy point cloud. Next, search the trunk center of mass for the nearest neighbors and use the distance between the trunk center of mass and the nearest neighbor points as weights to establish the adjacency edge relationship. Finally, complete the establishment of the search table.

The Dijkstra algorithm, based on heap optimization, uses a priority queue to optimize the traversal phase. The main idea of the algorithm is to expand the shortest path tree step-by-step by continuously selecting the node closest to the starting point. First, a minimum heap is created as a priority queue, which is used to store the nodes to be processed. Create a distance array, dist, to record the shortest distance from the starting node source to other nodes. Set the shortest distance of the start node to 0 and the shortest distance of the other nodes to infinity. In each loop, the node with the smallest distance is selected from the priority queue heap, noted as the current node, and marked as visited. Iterate through all the near-neighbor points of the current node; if the path distance of the current node to reach the near-neighbor point is less than the current recorded shortest distance of the near-neighbor point, then update the shortest distance of the near-neighbor point; if the near-neighbor point is not in the priority queue, then add it to the priority queue. Repeat the above steps until the priority queue is empty. Finally, get the shortest path from each starting point to each point in the canopy layer. The pseudocode is shown in Algorithm 1.

Algorithm 1. Pseudo-code for Dijkstra's algorithm based on heap optimization.

```
Require: graph: the input graph; source: the source node;
Ensure: dist: an array with the shortest distances from the source to all other nodes;
dist[source] ← 0
heap ← MinHeap()
heap.insert(source, dist[source])
while heap is not empty do
    current ← heap.extractMin()
    for each neighbor in graph.neighbors(current) do
        distance ← dist[current] +graph.edgeWeight(current, neighbor)
        if distance < dist[neighbor] then
            dist[neighbor] ← distance
            if neighbor is not in heap then
                heap.insert(neighbor, dist[neighbor])
            else
                heap.decreaseKey(neighbor, dist[neighbor])
            end if
        end if
        end for
end while
return dist
```

SHS clusters the canopy point cloud based on the shortest path tree, compares the shortest path distance from each point in the canopy point cloud to each starting point. Then, it clusters the point to the cluster of starting points corresponding to the minimum value of the shortest path distance until each point is clustered.

2.5. Point Cloud Segmentation and Comparative Shortest Path

In this study, the PCS algorithm [12] and the CSP algorithm [18] are selected as comparison algorithms of the SHS algorithm so as to verify the superiority of the SHS algorithm in segmenting the forest Backpack-LiDAR point cloud. The PCS algorithm utilizes the intrinsic three-dimensional structure of the airborne LiDAR point cloud to determine the shape of an individual tree by the difference in the spacing between the top of the tree and the bottom of the tree. The CSP algorithm pioneers the use of a new segmentation algorithm to segment a canopy point cloud by computationally identifying paths in the tree trunk based on the vascular structure of the tree and using this to segment individual trees in a forest point cloud.

2.6. Implementation and Accuracy Assessment

For the individual tree segmentation algorithm based on hierarchical strategy (SHS) proposed in this paper, this study uses the open-source software CloudCompare (https://www.cloudcompare.org/, accessed on 5 October 2022) to complete the implementation of the data preprocessing part. In this process the random sampling part is set to one-third or one-half of the original number of points after sampling, the SOR filtering mean distance estimation points and the standard deviation multiplier threshold are set to the default value, the fabric resolution of CSF filtering is set to 0.5, and the maximum number of iterations is 1000. Point cloud stratification, trunk detection, and canopy clustering are realized using the point cloud library (PCL) based on C++. The PCS and CSP algorithms were implemented using LiDAR360 software (https://www.lidar360.com/, accessed on 13 February 2022).

In order to accurately evaluate the accuracy of this segmentation algorithm, this study uses the recall rate (r), precision rate (p), and overall segmentation rate (F) for evaluation with the following formulas:

$$\begin{aligned} r &= \frac{TP}{TP+FN} \\ p &= \frac{TP}{TP+FP} \\ F &= 2 \times \frac{r \times p}{r+p} \end{aligned} \tag{7}$$

where *TP* is a true positive indicating the number of trees that were correctly segmented, *FN* is a false negative indicating the number of neighboring trees that were assigned incorrect segmentation, and *FP* is a false positive indicating the trees that were segmented but did not actually exist. High *TP* values and low *FN* and *FP* values correspond to high segmentation accuracy.

In order to assess the accuracy of segmentation in terms of point level, we extracted the crown widths of individual trees segmented using each of the three methods, compared them using field-measured crown width data. We evaluated the accuracy of the extracted crown widths using the coefficient of determination, R^2, with the root mean square error (*RMSE*), two common metrics used to verify the precision of the segmentation algorithms in terms of side-by-side validation:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(d_i - \hat{d}_i\right)^2}{\sum_{i=1}^{n} \left(d_i - \overline{d}\right)^2} \tag{8}$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(d_i - \hat{d}_i\right)^2} \tag{9}$$

where d_i is the measured crown diameter of the plot numbered i, $\hat{d}_i$ is the extracted crown diameter of the plot numbered i, and $\overline{d}$ is the mean of the measured crown diameter of plot.

3. Results

For trunk detection, SHS detected 43 trunks in Plot 1, 51 trunks in Plot 2, and 62 trunks in Plot 3. SHS counted the same number of detections as the number of manual inspections, with 0 missed detections and 0 false detections. That is, each individual tree trunk was correctly detected in Plots 1–3, with a 100% correct detection rate. Figure 5. shows the detection results of SHS on tree trunks in Plots 1–3.

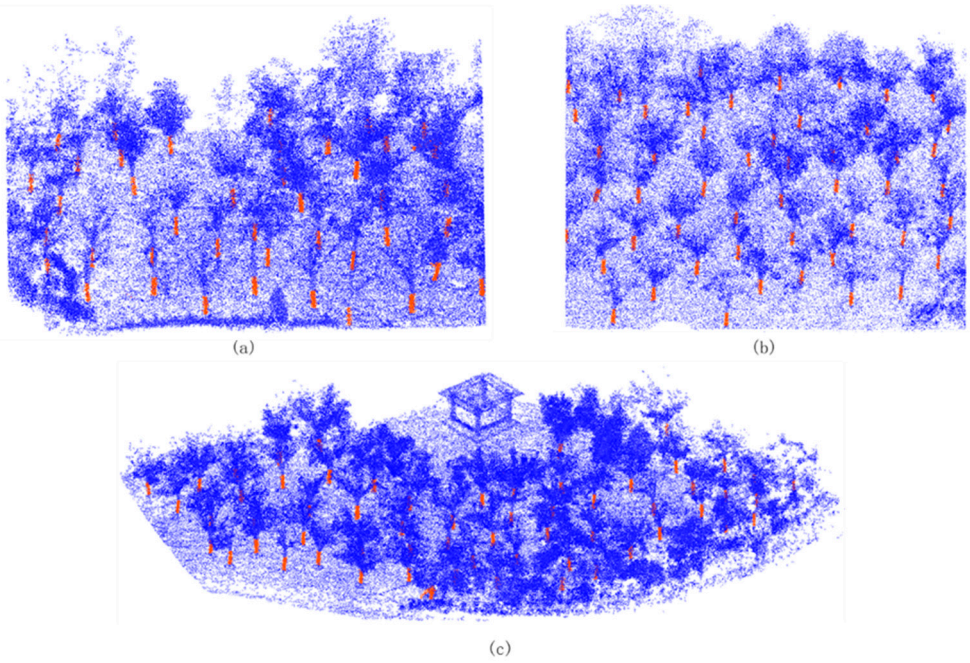

Figure 5. The findings of trunk detection for Plots 1–3 have been graphically represented. The segment highlighted in orange within the depicted picture is the trunk component identified by the algorithm provided in this scholarly article. The location of Plot 1 is denoted by (**a**), Plot 2 is denoted by (**b**), and Plot 3 is denoted by (**c**).

The specific segmentation results (*TP*, *FN*, and *FP*) and evaluation metrics (*r*, *p*, and *F*) using the three algorithms for the three plots are shown in detail in Table 2. SHS has recall values (*r*) of 1, 0.98, and 1; precision values (*p*) of 1, 1, and 1; and overall segmentation rate (*F*) of 1, 0.99, and 1. The total recall value (*r*) is 0.99, total precision value (*p*) is 1, and total overall segmentation rate (*F*) is 0.99. In Plots 1–3, the PCS algorithm has *r* values of 0.88, 0.96, and 0.96; *p* values of 0.84, 0.98, and 0.64; and *F* values of 0.86, 0.97, and 0.77. The total recall value (*r*) is 0.94, total precision value (*p*) is 0.79, and total overall segmentation rate (*F*) is 0.86. The CSP has *r* values of 1, 1, and 1 in the plots; *p* values of 0.93, 0.98, and 0.53; and *F* values of 0.96, 0.99, and 0.69. Total recall (*r*) is 1, total precision (*p*) is 0.73, and total overall segmentation rate (*F*) is 0.84. The segmentation results of the proposed algorithm in this paper and the comparative shortest-path algorithm (CSP), as well as the point cloud segmentation (PCS) algorithm for the individual trees of Plots 1–3, respectively, are shown in Figure 6.

Table 2. Segmentation results and accuracy assessments in Plot 1, Plot 2, and Plot 3.

Plot	Algorithm	Actual Number	Segmentation Number	*TP*	*FN*	*FP*	*r*	*p*	*F*
1	SHS	43	43	43	0	0	1	1	1
	CSP	43	46	43	0	3	1	0.93	0.96
	PCS	43	45	38	5	7	0.88	0.84	0.86
2	SHS	51	51	50	1	0	0.98	1	0.99
	CSP	51	51	51	0	1	1	0.98	0.99
	PCS	51	49	48	2	1	0.96	0.98	0.97
3	SHS	62	62	62	0	0	1	1	1
	CSP	62	116	62	0	54	1	0.53	0.69
	PCS	62	76	49	2	27	0.96	0.64	0.77

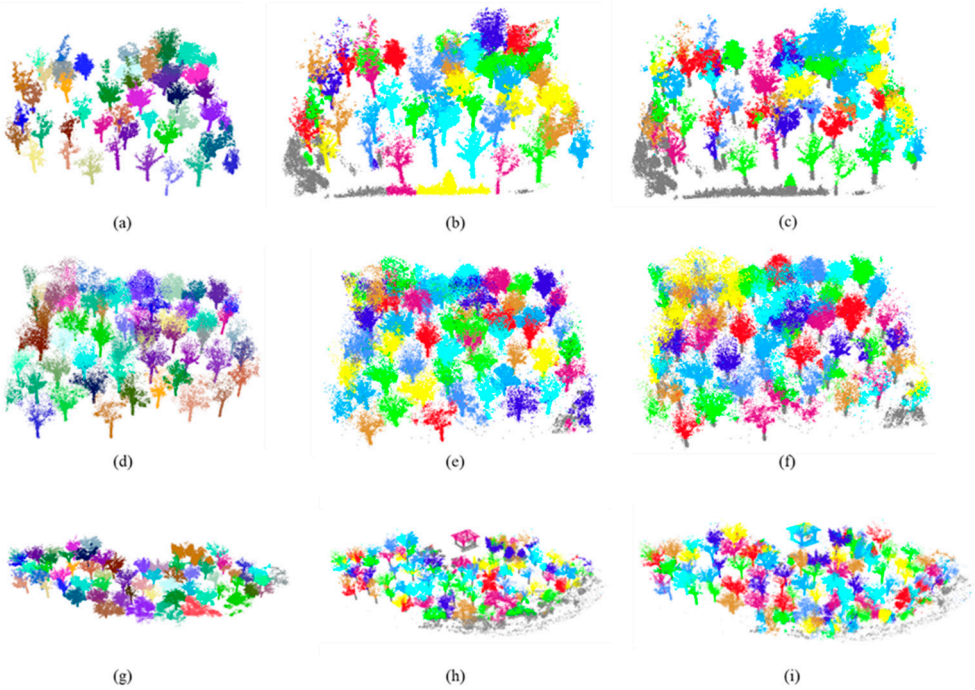

Figure 6. The plots depict the outcomes of the individual tree segmentation process in Plots 1–3, wherein distinct colors were assigned to the segmented individual trees. The plots labeled (**a**–**c**) represent the results obtained from processing Plot 1 using the methods SHS, CSP, and PCS, respectively. On the other hand, the plots labeled (**d**–**f**) and (**g**–**i**) correspond to Plot 2 and Plot 3, respectively, processed using the same three algorithms.

In terms of missed segmentation and misidentification, SHS has only one instance of missed segmentation for a tree in Plot 2, accounting for 0.6% of the total number of trees. PCS has 5, 2, 2, accounting for 5.8% of missed segmentation in the three plots, and 7, 1, 27, accounting for 22.4% of misidentifications. CSP has 0, 0, 2, accounting for 1.2% of missed segmentation, and 3, 1, 54, accounting for 37.2% of misidentifications. Figure 7 shows some typical cases of missed segmentation and misidentification.

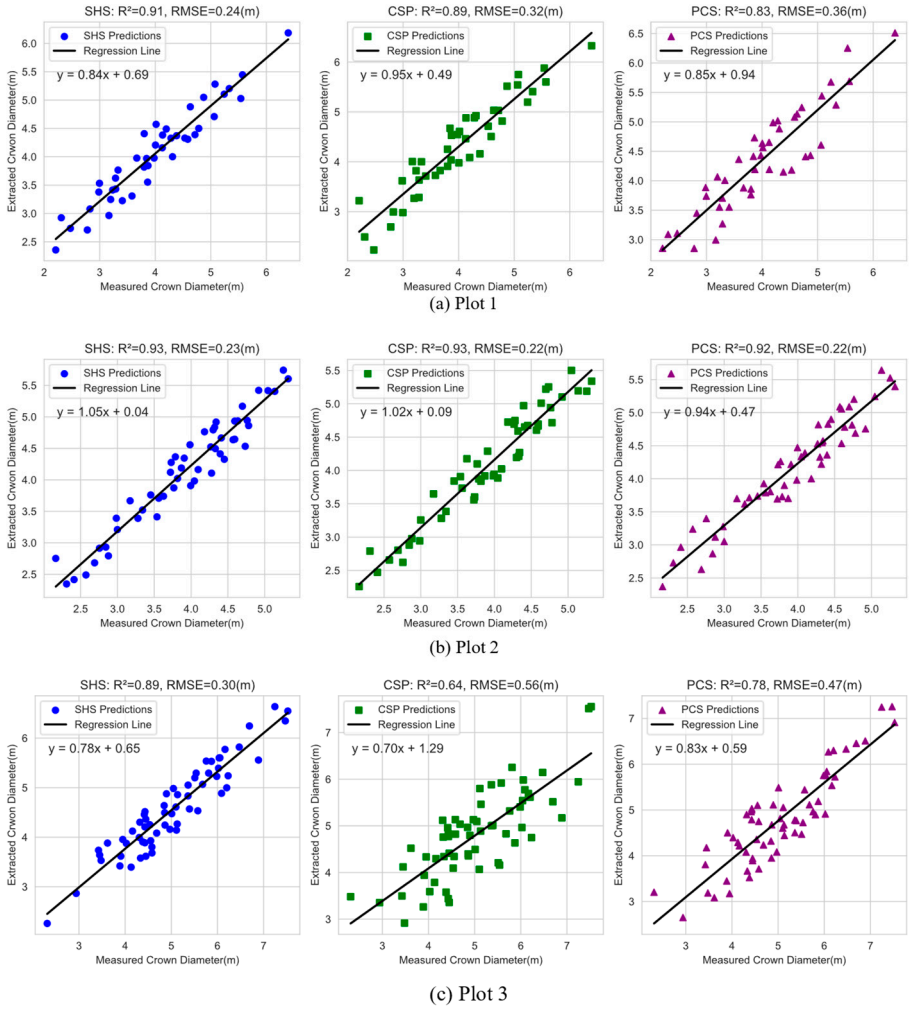

Figure 7. The scatter plots depict the comparison between the extracted crowns and the measured crowns in individual trees. These scatter plots represent the findings produced from three segmentation algorithms, labeled as (**a**), (**b**) and (**c**), respectively. The sample plots used for this analysis are numbered as 1, 2, and 3.

Figure 7 illustrates the linear correlation between the tree-extracted crowns derived from the three segmentation methods and the crowns measured in the field. The results indicate that the R^2 values for sample plots 1 and 3 using the SHS technique are 0.91 and 0.89, respectively, surpassing those of the other two algorithms. Additionally, the RMSE values for sample plots 1 and 3 using the SHS method are 0.24 m and 0.30 m, respectively, demonstrating lower values compared to the other two algorithms. In Plot 2, the coefficient of determination (R^2) for the SHS algorithm was found to be 0.93, which was comparable to the R^2 value obtained using the CSP technique. However, the R^2 value for the SHS algorithm was somewhat higher than that of the PCS 1% approach. In terms of root mean square error (RMSE), the SHS algorithm yielded a value of 0.23 m, which was slightly lower than the RMSE value of 0.22 m obtained using the other two methods. The percentage in question is 4%. These plots depict the outcomes of the segmentation process for individual

trees in Plots 1–3. Each segmented tree is represented by a distinct color. The first three plots, labeled as (a) to (c), were processed using the algorithms SHS, CSP, and PCS. The remaining plots, labeled as (d) to (f) and (g) to (i), represent Plot 2 and Plot 3, respectively, and were processed using the same three algorithms.

4. Discussion

4.1. Comparison and Analysis of Results from Individual Tree Segmentation Algorithm Based on Stratification Strategy with PCS and CSP

The PCS and CSP algorithms successfully segmented individual trees with a high degree of accuracy in Plots 1 and 2, but their performance degraded significantly in Plot 3. The primary determinant of segmentation accuracy for the PCS algorithm is the threshold [12]. Plot 3 exhibits a densely vegetated understory and a heavily overlapping tree canopy. The scanning capabilities of the Backpack-LiDAR render a portion of the treetop point cloud inaccessible [27]. Consequently, the PCS adaptive thresholding algorithm is rendered ineffective, and the segmentation accuracy declines, leading to the occurrence of misrecognition [28]. According to Wang, Y. et al., the precision of segmenting individual trees is influenced by factors such as the heights, species, and canopy morphologies of adjacent trees [29]. The segmentation results of PCS provide additional support for the aforementioned literature. The results of PCS's partial misidentification are illustrated in Figure 8b,d. Due to the sensitivity of the CSP algorithm to non-tree point clouds at greater altitudes, a considerable quantity of vegetation in Plot 3 is identified as individual tree targets, resulting in a decline in the accuracy of segmentation. Consistent with our experimental findings, Burt, A. et al. contend that CSP is unique to non-ground LiDAR point clouds and is only pertinent to structurally simple and sparse forest types [30].

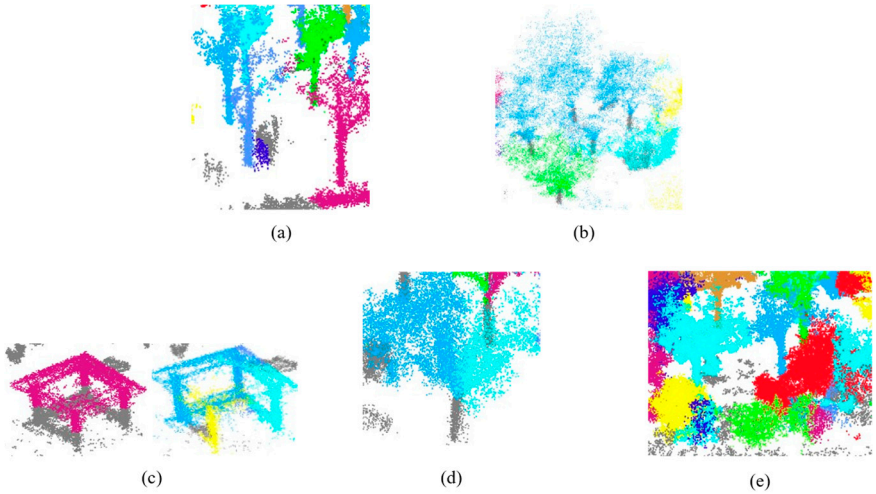

Figure 8. Typical problems in segmentation. (**a**) There is a scanner figure in the plot, which is misidentified as a tree trunk by the CSP algorithm. (**b**) The PCS algorithm recognizes four trees as the same tree, resulting in a missed segmentation. (**c**) The PCS algorithm and the CSP algorithm misidentified the building in the plot as an individual tree. (**d**) The PCS algorithm segmented the crown of one tree into the crowns of two trees. (**e**) The CSP algorithm recognized taller shrubs as individual trees.

Strictly enhancing the precision of individual tree segmentation remains a formidable task. For segmenting Backpack-LiDAR point clouds, Comesaña-Cebral, L. et al. proposed a fusion of the DBSCAN and cylinder voxelization algorithms [20]. The resulting segmentation accuracy was approximately 90%; however, a considerable number of misidentified and omitted segments persisted. Using a relative density segmentation algorithm, Liu, L. et al. effectively segmented 140 individual trees in a natural forest plot [21]. However, eight segmentations were missed and five were misidentified. The application of the SHS algorithm, as outlined in the referenced publication, results in a notable enhancement in

the segmentation precision of the point cloud as compared to the PCS and CSP algorithms. Furthermore, SHS demonstrates effective performance in mitigating the occurrence of missed segmentation and misclassification, while also exhibiting resilience in handling non-tree point clouds (refer to Figure 8c). To summarize, the accuracies of the individual tree segmentation and crown diameter measurements for the three algorithms discussed in Section 3 clearly show that SHS has the capability to enhance the accuracy of individual tree segmentation and reduce errors in identifying and recognizing trees.

4.2. Important Parameters in the Individual Tree Segmentation Algorithm Based on Stratification Strategy

The crown-trunk layered height interval and the cylindrical fitting nearest neighbor threshold are critical factors in the individual tree segmentation technique that relies on the layering strategy. These parameters have a direct impact on the quality of the segmentation results obtained using the segmentation based on hierarchical strategy (SHS) method. The optimization of the height interval for canopy-trunk layering has been shown to enhance the computational efficiency of the algorithm [24]. Additionally, the DBSCAN algorithm has demonstrated its effectiveness in detecting trunk point clouds when the trunk layer consists of a lower proportion of non-trunk point clouds [20]. This, in turn, reduces the number of pending trunk point clouds for input cylindrical fitting. On the other hand, the SHS algorithm is capable of efficiently detecting canopy point clouds when the canopy layer contains a larger quantity of such point clouds. When the canopy layer contains a higher density of canopy point cloud data, the Spatial Hierarchical Structure (SHS) forms a tightly connected undirected graph. By utilizing the Dijkstra algorithm with heap optimization, it becomes possible to precisely compute the shortest path from the canopy points to the center of mass of the trunk. This enables the successful completion of the process of clustering the canopy.

The morphology of tree trunks inside real-world plots is approximately cylindrical, albeit with deviations from the ideal cylindrical shape [31,32]. The precise identification of tree trunks within plots that have intricate understory habitats, characterized by the presence of numerous non-tree disturbances resembling tree trunks, poses a significant obstacle. In a study conducted by [19], it was shown that the precision of segmenting individual trees was diminished in plots characterized by intricate understory settings, as opposed to plots with uncomplicated understory environments. Additionally, the extraction of structural data of the trees was also influenced by this disparity. This study achieved improved recognition of distinct tree trunk shapes and non-tree trunk shapes in the SHS algorithm by implementing a nearest-neighbor threshold for cylinder fitting. Additionally, we carefully selected a suitable range of tree trunks to accurately fit the residuals to the cylinder model.

4.3. Limitations of Individual Tree Segmentation Algorithms Based on Hierarchical Strategy and Possible Future Research Directions

As indicated in Section 4.2, the outcomes of the SHS algorithm exhibit sensitivity to the parameter configurations pertaining to stratification height, cylindrical fitting, and the estimation of tree height and diameter at breast height. The park forest sample data employed in the research demonstrates the ease of adjusting these parameters. Nevertheless, the accuracy of the SHS algorithm's segmentation is significantly compromised when the research sample plot encompasses trees exhibiting substantial morphological disparities. The PCS algorithm and the CSP algorithm encounter the same issues [33]. Traditional algorithms for segmenting individual trees rely on a combination of a priori knowledge and machine learning techniques. However, when dealing with sample plots that exhibit significant variations in tree attributes, it becomes necessary to change different parameters in order to get segmentation results that are more accurate when compared to each other. In recent times, a considerable number of scholars have employed deep learning networks in their investigations pertaining to the field of agroforestry. The study conducted by [34] presents a novel approach that leverages convolutional neural networks (CNN)

and LiDAR data to facilitate the process of 3D maize phenotyping. The proposed method effectively partitions maize stems and leaves in field settings, exhibiting notable efficacy with an F-score exceeding 0.82 for individual organs and 0.99 for entire plants. The study conducted by [35] introduces a novel methodology that utilizes PointNet++ to effectively separate several components of tree structures, including the canopy, trunk, and branches, based on LiDAR point cloud data. The efficiency of the model was assessed by utilizing LiDAR data obtained from a total of 435 tree samples, encompassing various species such as Korean red pine, Korean pine, and Japanese larch. In contrast to alternative methodologies such as PointCNN [36] and PointNet [37], the PointNet++ [38] model employed in this investigation showed superior performance, notably in the realm of trunk segmentation. Nevertheless, the accuracy of canopy segmentation was comparatively lower, measuring at 90.3%. The deep learning network model possesses a notable capacity for generalization, which is advantageous in minimizing the need for extensive post-processing efforts in individual tree segmentation. Additionally, this model facilitates the extraction of tree parameters by integrating a compact network. In future research, our aim is to integrate the deep learning network with the SHS algorithm. We intend to utilize the SHS algorithm to generate a comprehensive dataset for tree segmentation. Additionally, we plan to enhance the current network architecture by incorporating principles from the traditional individual tree segmentation algorithm. Through rigorous training, our objective is to develop a segmentation model that can effectively handle variations in tree sizes.

5. Conclusions

This work presents an improved individual tree segmentation approach, referred to as the individual tree segmentation approach based on hierarchical strategy (SHS), with the aim of addressing the issues of missing segmentation and misrecognition, commonly encountered in existing individual tree segmentation methods. The process involved in this study included the stratification of point clouds into trunk and crown components. The trunk point cloud was subjected to RANSAC cylindrical fitting for constraint purposes, while the crown component was segmented using the Dijkstra algorithm with heap optimization. The resulting segmentation outcomes were then compared using both the PCS algorithm and the CSP algorithm. The SHS algorithm demonstrates a significant improvement in overall segmentation accuracy (F) compared to the PCS and CSP algorithms in the context of missed segmentation and misrecognition. Specifically, in the three broadleaf mixed forest samples, the SHS algorithm achieves segmentation accuracies of 1, 0.99, and 1, respectively. In contrast, the overall segmentation rates (F) of the PCS and CSP algorithms are consistently 10%–15% lower than those of the SHS algorithm. In this study, the crown width data were obtained by extracting information from the point clouds of individual trees using three different algorithms. These extracted crown widths were then compared to the crown widths measured in the sample plots. The results indicated that the SHS algorithm had higher coefficients of determination (R^2) compared to the PCS and CSP algorithms. Additionally, the root mean square error (RMSE) of the SHS algorithm was smaller than that of the PCS and CSP algorithms in Plot 1 and Plot 3. However, it is worth noting that the results of the SHS algorithm were similar to the results of the other two algorithms in terms of both R^2 and RMSE. The coefficient of determination (R^2) and root mean square error (RMSE) values obtained from the SHS algorithm in Plot 2 exhibit similarity to the corresponding values obtained from the other two techniques. This paper examines the disparities in segmentation outcomes among three algorithms, along with their specific variations. It further explores the significant parameters involved in the segmentation process of the SHS algorithm. Lastly, it addresses the existing constraints of the SHS algorithm and outlines potential avenues for future research in this field. In conclusion, the SHS algorithm, as presented in this study, demonstrates notable enhancements in addressing the challenges of missed segmentation of individual tree segmentation and misrecognition. This algorithm contributes to the theoretical foundation of LiDAR research in forest detection and facilitates the progress towards achieving precise and accurate forest

detection methodologies. In next investigations, our focus will be on integrating deep learning networks as a means to enhance the challenging issue of parameter adjustment in the segmentation of individual trees.

Author Contributions: Conceptualization, D.Z. and X.L.; methodology, D.Z. and X.L.; software, D.Z. and X.L.; validation, D.Z. and Y.Z.; formal analysis, L.X.; investigation, D.Z. and L.X.; resources, L.X. and Q.H.; data curation, D.Z. and Q.H.; writing—original draft preparation, D.Z. and X.L.; writing—review and editing, Y.Z. and L.X.; visualization, D.Z. and Q.H.; supervision, Y.Z. and L.X.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Science and Technology Program Project (QY-STK-2022A-034) of Qingyang City, Gansu Province, and the Research Project of the JiangXi Province Department of Forestry ([2022] 38).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the consideration of data security.

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

References

1. Haoran, L.; Weiwei, F.; Yongsheng, X.; Wenshu, L. Estimation of Individual Tree Biomass Based on Unmanned Aerial Vehicle LiDAR Point Cloud. *J. Cent. South Univ. For. Sci. Technol.* **2021**, *41*, 92–99.
2. Hu, T.; Sun, X.; Su, Y.; Guan, H.; Sun, Q.; Kelly, M.; Guo, Q. Development and Performance Evaluation of a Very Low-Cost UAV-Lidar System for Forestry Applications. *Remote Sens.* **2021**, *13*, 77. [CrossRef]
3. Kai, Z.; Lin, C. Current Status and Prospects of Remote Sensing Applications in Precision Forest Cultivation. *J. Remote Sens.* **2021**, *25*, 423–438.
4. Ping, L.; Zhong, F. Quantitative structural modeling for ground-based LiDAR individual tree segmentation applications. *Surv. Mapp. Sci.* **2022**, *47*, 151–156+199.
5. Torre-Tojal, L.; Bastarrika, A.; Boyano, A.; Lopez-Guede, J.M.; Graña, M. Above-Ground Biomass Estimation from LiDAR Data Using Random Forest Algorithms. *J. Comput. Sci.* **2022**, *58*, 101517. [CrossRef]
6. Huo, L.; Zhang, X. Individual tree information extraction based on multi-layer clustering of airborne LiDAR point cloud and its accuracy evaluation. *For. Sci.* **2021**, *57*, 85–94.
7. Wallace, L.; Lucieer, A.; Watson, C.; Turner, D. Development of a UAV-LiDAR System with Application to Forest Inventory. *Remote Sens.* **2012**, *4*, 1519–1543. [CrossRef]
8. Yu, H.; Feng, S.; Shen, Y.; Liu, P. Research on individual tree segmentation algorithms for UAV-borne LiDAR in plantation forests. *Laser Infrared* **2022**, *52*, 757–762.
9. Liu, H.; Zhang, X.; Zhang, Y.; Zhu, Y.; Liu, H.; Wang, L. Advances in airborne lidar individual tree recognition research. *Adv. Lasers Optoelectron.* **2018**, *55*, 40–48.
10. Popescu, S.C.; Wynne, R.H. Seeing the Trees in the Forest. *Photogramm. Eng. Remote Sens.* **2004**, *70*, 589–604. [CrossRef]
11. Koch, B.; Heyder, U.; Weinacker, H. Detection of Individual Tree Crowns in Airborne Lidar Data. *Photogramm. Eng. Remote Sens.* **2006**, *72*, 357–363. [CrossRef]
12. Li, W.; Guo, Q.; Jakubowski, M.K.; Kelly, M. A New Method for Segmenting Individual Trees from the Lidar Point Cloud. *Photogramm. Eng. Remote Sens.* **2012**, *78*, 75–84. [CrossRef]
13. Chen, W.; Hu, X.; Chen, W.; Hong, Y.; Yang, M. Airborne LiDAR Remote Sensing for Individual Tree Forest Inventory Using Trunk Detection-Aided Mean Shift Clustering Techniques. *Remote Sens.* **2018**, *10*, 1078. [CrossRef]
14. Ko, C.; Lee, S.; Yim, J.; Kim, D.; Kang, J. Comparison of Forest Inventory Methods at Plot-Level between a Backpack Personal Laser Scanning (BPLS) and Conventional Equipment in Jeju Island, South Korea. *Forests* **2021**, *12*, 308. [CrossRef]
15. Guo, Q.; Su, Y.; Hu, T.; Guan, H.; Jin, S.; Zhang, J.; Zhao, X.; Xu, K.; Wei, D.; Kelly, M.; et al. Lidar Boosts 3D Ecological Observations and Modelings: A Review and Perspective. *IEEE Geosci. Remote Sens. Mag.* **2021**, *9*, 232–257. [CrossRef]
16. Hu, T.; Wei, D.; Su, Y.; Wang, X.; Zhang, J.; Sun, X.; Liu, Y.; Guo, Q. Quantifying the Shape of Urban Street Trees and Evaluating Its Influence on Their Aesthetic Functions Based on Mobile Lidar Data. *ISPRS J. Photogramm. Remote Sens.* **2022**, *184*, 203–214. [CrossRef]
17. An, Z.; Froese, R.E. Tree Stem Volume Estimation from Terrestrial LiDAR Point Cloud by Unwrapping. *Can. J. For. Res.* **2023**, *53*, 60–70. [CrossRef]
18. Tao, S.; Wu, F.; Guo, Q.; Wang, Y.; Li, W.; Xue, B.; Hu, X.; Li, P.; Tian, D.; Li, C.; et al. Segmenting Tree Crowns from Terrestrial and Mobile LiDAR Data by Exploring Ecological Theories. *ISPRS J. Photogramm. Remote Sens.* **2015**, *110*, 66–76. [CrossRef]
19. Hyyppä, E.; Kukko, A.; Kaijaluoto, R.; White, J.C.; Wulder, M.A.; Pyörälä, J.; Liang, X.; Yu, X.; Wang, Y.; Kaartinen, H.; et al. Accurate Derivation of Stem Curve and Volume Using Backpack Mobile Laser Scanning. *ISPRS J. Photogramm. Remote Sens.* **2020**, *161*, 246–262. [CrossRef]

20. Comesaña-Cebral, L.; Martínez-Sánchez, J.; Lorenzo, H.; Arias, P. Individual Tree Segmentation Method Based on Mobile Backpack LiDAR Point Clouds. *Sensors* **2021**, *21*, 6007. [CrossRef]
21. Liu, L.; Zhang, A.; Xiao, S.; Hu, S.; He, N.; Pang, H.; Zhang, X.; Yang, S. Single Tree Segmentation and Diameter at Breast Height Estimation With Mobile LiDAR. *IEEE Access* **2021**, *9*, 24314–24325. [CrossRef]
22. Vitter, J.S. Faster Methods for Random Sampling. *Commun. ACM* **1984**, *27*, 703–718. [CrossRef]
23. Zhang, W.; Qi, J.; Wan, P.; Wang, H.; Xie, D.; Wang, X.; Yan, G. An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation. *Remote Sens.* **2016**, *8*, 501. [CrossRef]
24. Gu, Z.; Pei, F. A multi-layer K-means based algorithm for individual tree identification in forest point clouds. *For. Resour. Manag.* **2022**, *1*, 124–131.
25. Wu, J.; Cawse-Nicholson, K.; van Aardt, J. 3D Tree Reconstruction from Simulated Small Footprint Waveform Lidar. *Photogramm. Eng. Remote Sens.* **2013**, *79*, 1147–1157. [CrossRef]
26. Fischler, M.A.; Bolles, R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* **1981**, *24*, 381–395. [CrossRef]
27. Lu, J.; Wang, H.; Qin, S.; Cao, L.; Pu, R.; Li, G.; Sun, J. Estimation of Aboveground Biomass of Robinia Pseudoacacia Forest in the Yellow River Delta Based on UAV and Backpack LiDAR Point Clouds. *Int. J. Appl. Earth Obs. Geoinf.* **2020**, *86*, 102014. [CrossRef]
28. Yang, Q.; Su, Y.; Jin, S.; Kelly, M.; Hu, T.; Ma, Q.; Li, Y.; Song, S.; Zhang, J.; Xu, G.; et al. The Influence of Vegetation Characteristics on Individual Tree Segmentation Methods with Airborne LiDAR Data. *Remote Sens.* **2019**, *11*, 2880. [CrossRef]
29. Wang, Y.; Hyyppa, J.; Liang, X.; Kaartinen, H.; Yu, X.; Lindberg, E.; Holmgren, J.; Qin, Y.; Mallet, C.; Ferraz, A.; et al. International Benchmarking of the Individual Tree Detection Methods for Modeling 3-D Canopy Structure for Silviculture and Forest Ecology Using Airborne Laser Scanning. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 5011–5027. [CrossRef]
30. Burt, A.; Disney, M.; Calders, K. Extracting Individual Trees from Lidar Point Clouds Using Treeseg. *Methods Ecol. Evol.* **2019**, *10*, 438–445. [CrossRef]
31. Fan, G.; Liang, H.; Zhao, Y.; Li, Y. Automatic Reconstruction of Three-Dimensional Root System Architecture Based on Ground Penetrating Radar. *Comput. Electron. Agric.* **2022**, *197*, 106969. [CrossRef]
32. Cai, S.; Xing, Y.; Duanmu, J. Backpack Lidar Filtering Low Intensity Point Clouds to Extract Forest Tree Breast Diameter. *For. Eng.* **2021**, *37*, 12–19.
33. Xu, X.; Iuricich, F.; Calders, K.; Armston, J.; De Floriani, L. Topology-Based Individual Tree Segmentation for Automated Processing of Terrestrial Laser Scanning Point Clouds. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *116*, 103145. [CrossRef]
34. Ao, Z.; Wu, F.; Hu, S.; Sun, Y.; Su, Y.; Guo, Q.; Xin, Q. Automatic Segmentation of Stem and Leaf Components and Individual Maize Plants in Field Terrestrial LiDAR Data Using Convolutional Neural Networks. *Crop J.* **2022**, *10*, 1239–1250. [CrossRef]
35. Kim, D.-H.; Ko, C.-U.; Kim, D.-G.; Kang, J.-T.; Park, J.-M.; Cho, H.-J. Automated Segmentation of Individual Tree Structures Using Deep Learning over LiDAR Point Cloud Data. *Forests* **2023**, *14*, 1159. [CrossRef]
36. Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; Chen, B. PointCNN: Convolution on X-Transformed Points. *Adv. Neural Inf. Process. Syst.* **2018**, *31*.
37. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
38. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.