remote sensing

MDPI

Article

SK-TreePCN: Skeleton-Embedded Transformer Model for Point Cloud Completion of Individual Trees from Simulated to Real Data

Haifeng Xu [1,†], Yongjian Huai [1,2,†], Xun Zhao [3], Qingkuo Meng [1], Xiaoying Nie [1,2], Bowen Li [4] and Hao Lu [1,2,*]

[1] School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China; xuhaifeng@bjfu.edu.cn (H.X.); huaiyj@bjfu.edu.cn (Y.H.); mengqingkuo3110@bjfu.edu.cn (Q.M.); niexy@bjfu.edu.cn (X.N.)
[2] Engineering Research Center for Forestry-Oriented Intelligent Information Processing, National Forestry and Grassland Administration, Beijing 100083, China
[3] State Forestry and Grassland Administration Key Laboratory of Forest Resources and Environmental Management, Beijing Forestry University, Beijing 100083, China; xunzhao@bjfu.edu.cn
[4] Research Institute of Forest Resource Information Techniques, Chinese Academy of Forestry, Beijing 100091, China; libowen@caf.ac.cn
* Correspondence: luhao@bjfu.edu.cn
† These authors contributed equally to this work.

Abstract: Tree structural information is essential for studying forest ecosystem functions, driving mechanisms, and global change response mechanisms. Although current terrestrial laser scanning (TLS) can acquire high-precision 3D structural information of forests, mutual occlusion between trees, the scanner's field of view, and terrain changes make the point clouds captured by laser scanning sensors incomplete, further hindering downstream tasks. This study proposes a skeleton-embedded tree point cloud completion method, termed SK-TreePCN, which recovers complete individual tree point clouds from incomplete scanning data in the field. SK-TreePCN employs a transformer trained on simulated point clouds generated by a 3D radiative transfer model. Unlike existing point cloud completion algorithms designed for regular shapes and simple structures, the SK-TreePCN method addresses structurally heterogeneous trees. The 3D radiative transfer model LESS, which can simulate various TLS data over highly heterogeneous scenes, is employed to generate massive point clouds with training labels. Among the various point cloud completion methods evaluated, SK-TreePCN exhibits outstanding performance regarding the Chamfer distance (CD) and *F*1 Score. The generated point clouds display a more natural appearance and clearer branches. The accuracy of tree height and diameter at breast height extracted from the recovered point cloud achieved R^2 values of 0.929 and 0.904, respectively. SK-TreePCN demonstrates applicability and robustness in recovering individual tree point clouds. It demonstrated great potential for TLS-based field measurements of trees, refining point cloud 3D reconstruction and tree information extraction and reducing field data collection labor while retaining satisfactory data quality.

Keywords: point cloud completion; individual tree; tree skeleton; terrestrial laser scanning; simulation

check for updates

Academic Editor: Krzysztof Stereńczak

Received: 2 January 2025
Revised: 5 February 2025
Accepted: 12 February 2025
Published: 14 February 2025

Citation: Xu, H.; Huai, Y.; Zhao, X.; Meng, Q.; Nie, X.; Li, B.; Lu, H. SK-TreePCN: Skeleton-Embedded Transformer Model for Point Cloud Completion of Individual Trees from Simulated to Real Data. *Remote Sens.* **2025**, *17*, 656. https://doi.org/10.3390/rs17040656

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Forest structure refers to the three-dimensional (3D) arrangement of trees and other vegetation, which is essential in shaping forest productivity, biodiversity, and the overall

service quality of forest ecosystems [1]. Investigation and analysis of forest structure can help us understand the history, current state, and development of forest ecosystems [2]. Stand structure is the fundamental unit for studying forest structure and plays a crucial role in forest analysis. As specific subjects for studying stand structure, individual trees are essential components of forests. Accurate quantification of their structural parameters (e.g., tree height, diameter at breast height (DBH), crown spread, and leaf area index) is crucial for forest management and understanding ecosystem functions [3]. The conventional approach to assessing individual tree structures predominantly depends on field surveys to measure standing tree structural parameters directly. It is not only time-consuming and labor-intensive but also confronts the challenges of achieving comprehensive coverage across extensive forested areas. Moreover, measurements in complex stands are frequently obstructed by overlapping canopies and rugged terrain, leading to diminished accuracy [4].

Light detection and ranging (LiDAR) technology is widely used for forest inventory and assessments of forest structure because of its capacity to characterize the 3D vertical structure of forests precisely [5–8]. Among various LiDAR platforms, terrestrial laser scanning (TLS) is employed for detailed surveys of forest structure due to its ability to capture intricate details within the forest, such as tree trunks and branches. This facilitates the extraction of structural parameters, enabling the meticulous quantification of individual trees and overall forest structure [9,10]. Leveraging this feature of TLS data in conjunction with quantitative structural modeling (QSM) has propelled several studies, including inversion [4,11], radiative transfer [12], and tree species identification [13]. However, challenges such as point cloud registration, steep terrain, and inter-tree occlusion often result in incomplete point cloud data. These issues typically manifest as occlusion (where parts of the trees are hidden from view due to surrounding vegetation), low point density (leading to insufficient data to represent fine tree details), and scanning angle limitations (which can prevent some tree sections from being captured) [14,15]. These incompleteness issues can significantly affect the accuracy of quantifying tree structural parameters. Consequently, refining these incomplete tree point clouds is essential for accurate forest management and carbon sink assessment [15]. On the one hand, it reduces the workload of surveys by requiring only the establishment of reasonable scanning stations, subsequently complemented by point cloud completion technology to achieve a complete tree point cloud [16]. On the other hand, it can provide high-quality data for the downstream tasks [15].

Traditional tree point cloud completion methods can be categorized into priori knowledge-based and topological skeleton-based methods. The priori knowledge-based method integrates the tree's natural growth patterns and simple geometry features, then utilizes lightweight conventional algorithms to complete the point clouds. Representative prior knowledge-based methods include structure-aware optimization algorithms [17], the bottom-up growing algorithm [18], and the cylindrical shape prior [19]. These methods are effective for trees with simple structures, but reconstructing point clouds for trees with complex branches using only predefined rules or template matching is difficult [20]. Algorithms grounded in topological skeletons can overcome the difficulties above. They do so by assessing the dominant directions and density of points, and iteratively reconstructing the missing point clouds based on optimization algorithms [14,20]. Employing the topology of tree skeletons and iterative methods offers advantages in handling noise, yet calculating the geometric structure between point sets iteratively is time-consuming and may also converge to local optima.

With the advancements in deep learning technology, the point cloud-based learning frameworks PointNet [21] and PointNet++ [22] have garnered a lot of attention. PointNet and PointNet++ frameworks have introduced new opportunities for processing unstructured 3D data, with numerous studies being undertaken on point cloud completion. These

networks can be categorized into Point Features-based [23–25], Graph-based [26–28], Generative Adversarial Network (GAN)-based [29] and Transformer-based approaches [30–33]. Notably, the Transformer [34] technique has shown excellent results after being introduced into point cloud complementation. The advantage of the transformer model is that it lies on the unique structure (self-attention mechanism) without looping and convolution [30,33]. This is due to the inherent irregularity and density diversity of 3D point clouds, which are effectively encoded into the metric space through the self-attention mechanism [35]. Although the subjects of the aforementioned studies are not trees, their advancements offer the potential for completing tree point clouds.

Currently, deep learning is the most promising solution for point cloud completion. Even so, there are still some challenges for tree point cloud completion. Firstly, tree point clouds are very dense, so it is challenging to assign semantic information to each point cloud in high spatial resolution [25,30,33]. Compared to objects with regular or symmetric shapes, tree structures are more morphologically varied, and their shapes are mainly presented as fractal geometries. The irregular canopy and complex branching structure make model inference difficult. The topological skeleton structure of trees is very critical to expressing the connectivity of the point cloud, so it needs to be considered in the model design [10]. Secondly, in numerous point cloud completion studies, the samples of incomplete point clouds are primarily created by converting 2.5D depth images back into 3D [24,26,36,37]. This method does not accurately represent the incompleteness of point clouds due to natural limitations encountered during LIDAR data collection. The model's ability to generalize is enhanced only through the integration of logical prior knowledge. Thirdly, for the deep learning point cloud completion method, the complete ground truth point cloud is a significant input for supervising the model learning process. Nevertheless, securing accurate and adequate ground truth remains difficult, as the process of collecting point clouds with LiDAR inherently involves varying degrees of data incompleteness [26].

Virtual laser scanning (VLS) is an ideal tool to obtain near-realistic LiDAR data. LiDAR simulations within VLS primarily involve creating realistic 3D forest scenes, configuring LiDAR sensors on different platforms, and setting various scanning modes or positions to generate real-world LiDAR data. It was indicated that simulated point clouds could replace genuine data in certain studies [12,38–41]. Currently, an increasing number of LiDAR simulation platforms have been proposed, such as FLIGHT [42,43], DART [44], HELIOS [45], and LESS [46]. Previous studies have also utilized these models to conduct LiDAR data simulations on various platforms, yielding considerable outcomes in quantifying forest structure and facilitating novel LiDAR sensor development [12,39,40]. VLS enables the low-cost simulation of nearly infinite data volumes in various setups. Moreover, it reserves perfect ground truth from the original 3D tree models, which lays the foundation for developing high-quality tree point cloud completion models from virtual to real-world data.

Based on previous research, we propose a novel point cloud completion method, SK-TreePCN, dedicated to addressing the characteristics of tree structures. Within the 3D Transformer model, structural features of trees were added to predict the missing parts of the point cloud. Two modules based on the tree skeleton are designed: the Skeleton Proxy Point Sampler (SK→PS) and the Skeleton Topology Adjuster (SK→TA). The Skeleton Proxy Point Sampler connects to local point clouds by propagation and growth to ensure the connectivity of the global structure. The Topology Adjuster will guide point cloud recovery to rationalize the distribution of newly generated points around the tree skeleton. Finally, we validate the soundness of our scheme and the robustness of our method using tree point clouds from two field survey plots. Overall, the main contributions of our study are:

1. We propose a point cloud completion pipeline that transitions from simulation to reality. It is based on VLS simulation data to restore missing tree point clouds caused by limited scanning fields or branch and leaf occlusion during TLS.
2. We propose a Transformer-based point cloud completion model incorporating tree skeleton topology to address structural heterogeneity, effectively reducing issues like overlapping, discontinuity, and divergence through information propagation and bridging mechanisms.
3. Compared to various tree point cloud completion methods, SK-TreePCN achieves state-of-the-art (SOTA) performance.

2. Materials and Methods

We present a comprehensive overview of the entire point cloud completion approach, and the workflow is shown in Figure 1.

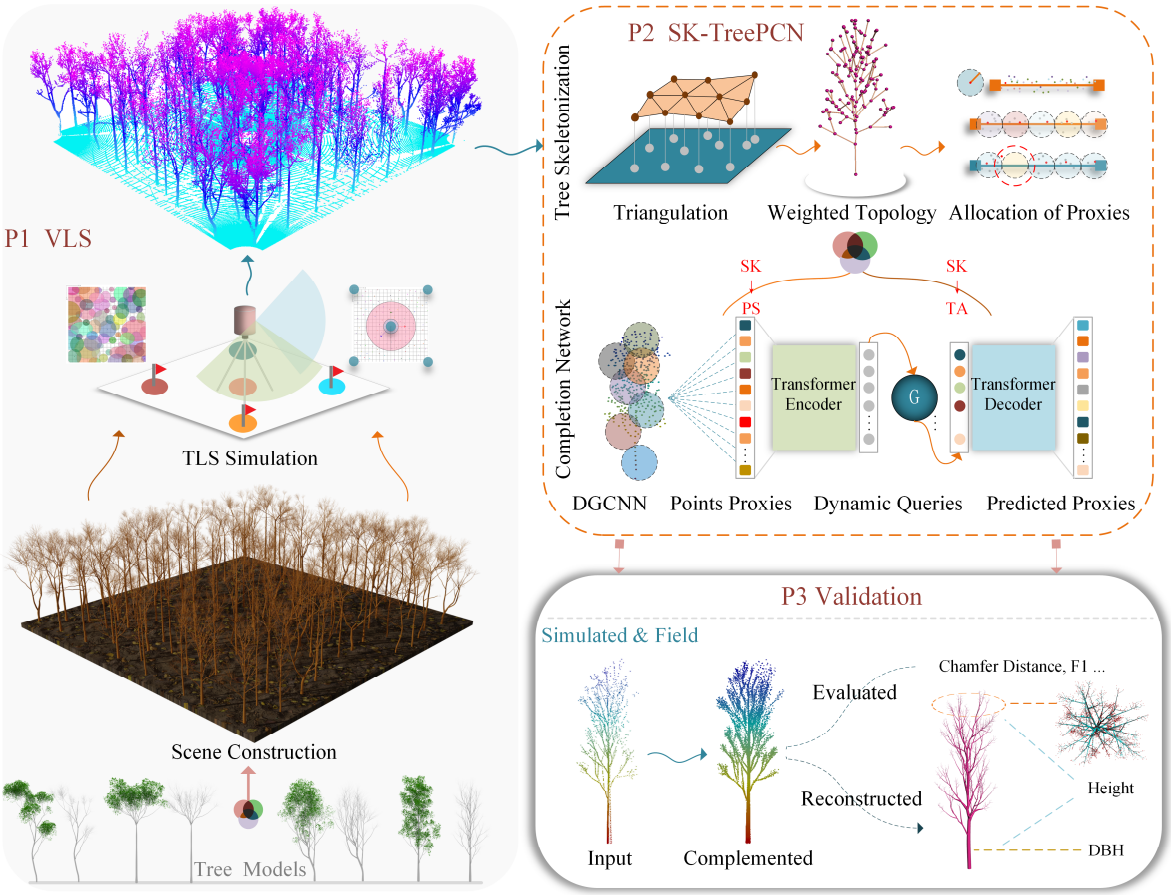


Figure 1. The workflow of the study. Part 1 involved the reconstruction of 3D scenes using a tree library and virtual laser scanning using LESS; Part 2 is SK-TreePCN. The point cloud is triangulated and simplified, and the skeleton is extracted. Then, the weighted proxy points are reassigned based on the open balls. These new proxy points are implanted into the transformer through the SK→PS and the SK→TA operations, and Part 3 addresses the simulation, field point cloud complemented, reconstructed, and evaluated.

Specifically, a 3D scene is first established based on a tree model dataset (Section 2.2.1). Tree point cloud data are generated through the simulation of TLS using LESS (Section 2.2.2). These data are applied to train the SK-TreePCN model (Section 2.3). After that, field and simulated data (Section 2.4) are employed to verify the generalization capability of the SK-TreePCN model. Upon completing the incomplete TLS point cloud using SK-TreePCN,

we conduct 3D reconstruction to extract structural parameters, which are then compared with field measured values to evaluate the contribution of point cloud completion to real point cloud data.

2.1. Area and TLS Data

2.1.1. Study Area

The study area is located in the eucalyptus plantation of Jiangxi Town, Jiangnan District, Nanning City, Guangxi Province (22°47′10.27″N–22°47′54.93″N, 108°6′7.39″E–108°6′58.43″E), as shown in Figure 2. Guangxi is the primary region in China for eucalyptus cultivation. As eucalyptus is a fast-growing tree species with high productivity, a short growth cycle, and strong competitiveness, it has become the region's leading advantageous industry and a critical economic pillar [47]. In addition, eucalyptus is also a critical timber harvesting species in China and plays an irreplaceable role in natural forest protection [48].

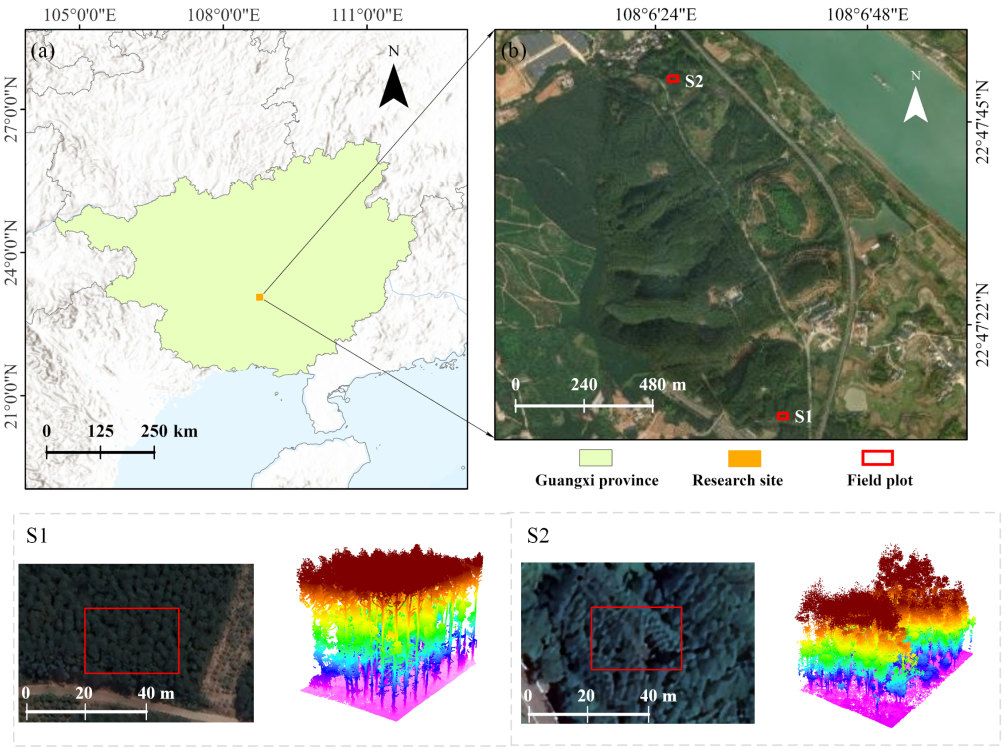


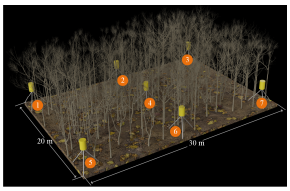
Figure 2. Location of the field plot in this study. (**a**) Geological Map of Guangxi Zhuang Autonomous Region. (**b**) Schematic diagram of plot distribution. Plots S1 (red box in the bottom left) and S2 (red box in the bottom right), featuring both satellite imagery and TLS point clouds.

2.1.2. TLS Data Acquisition and Pre-Processing

Two field plots (S1 and S2 in Figure 2) were surveyed in November 2022, with the size of 20 m × 30 m (20 m in the north–south and 30 m in the east–west direction). The acquisition equipment was a Trimble TX8 (Trimble Inc., Sunnyvale, CA, USA). The performance parameters of the equipment are shown in Table 1. To ensure the clarity of the TLS field of view (FOV), we evenly distributed the scanning stations throughout the plot, with seven stations set up in each plot (the scanning station layout is shown in Table 1). Manual targets were used for point cloud co-registration of individual scanning stations. Before scanning, eleven target balls were placed in the field plot, and at least three target balls were made sure to be visible intentionally within the FOV of each TLS station. These 3 target balls were not collinear, which can prevent a gimbal lock when calculating the transformation matrix during registration and alignment. In each plot, the scanning level of the central station was set to 3 (about 10 min of the scanning and 5.7 mm of the point spacing at 30 m). The

scanning level of other 6 stations was set to 2 (about 3 min of the scanning and 11.3 mm of the point spacing at 30 m), and the FOV of each station is 360° × 317°. The Trimble Realworks software (version 10.1) was used to process the raw point cloud data and detect the target balls for point cloud co-registration of each station.

Table 1. Plot scanning station setup and TLS parameters. Numbers 1–7 represent the scanning station locations.

Positions of the Scanning Stations in the Field Plot	Parameter	Value
	FOV	360° × 317°
	Scan distance	0.6–120 m
	Laser scan resolution	16″
	Scanning accuracy	2 mm
	Scan density	Level 1/2/3
	Laser wavelength	1.5 μm

Then, LiDAR360 (Green Valley International Co., Ltd., Beijing, China) was used for ground filtering and individual tree extraction. Specifically, the point cloud was first normalized, and ground points were then filtered using the Cloth Simulation Filter (CSF) algorithm [49]. The filtering process was performed with a grid size of 1, a classification threshold of 0.50, and a maximum of 500 iterations. Secondly, during the individual tree segmentation phase, vegetation points were first processed using the Point Cloud Segmentation (PCS) algorithm [50], with a distance threshold set to 2 m, a search radius of 3 m, and a minimum height above ground of 2 m. The initial segmentation results were then manually refined by adjusting the seed points. Based on the optimized seed points, a region-growing segmentation algorithm was applied to achieve fine-scale segmentation. Finally, edge or corner trees, which might have been partially scanned, were manually screened. Trees exhibiting more than 40% canopy loss, as visually inferred from the overhead view, were excluded from further analysis.

In addition, we also carried out measurements of structural parameters such as tree height, DBH, crown base height, and crown width of each tree. The statistical information of individual tree structure parameters is shown in Table 2. Plot S1 consisted of trees with uniform planting age and spacing, while plot S2 contained Eucalyptus trees of varying ages. Consequently, the structural parameters in plot S2 demonstrated greater variability than those in plot S1 (see Table 2). There were a total of 100 trees in plot S1 and 82 trees in plot S2.

Table 2. Tree structural parameters from field surveys.

Parameter Name/Plot No.	S1	S2
DBH (cm)	12.70 ± 1.50	9.53 ± 2.10
Height (m)	14.80 ± 1.25	12.12 ± 2.8
E-W crown spread (m)	3.04 ± 0.34	2.78 ± 1.39
N-S crown spread (m)	3.07 ± 0.16	2.81 ± 0.94
Number of trees	100	82

2.2. Virtual Laser Scanning

LESS is a 3D radiative transfer model that uses ray tracing techniques [46]. It stands out for its lightweight design, high compatibility, and user-friendly interface. At present, LESS has developed advanced simulation modules for both airborne laser scanning (ALS) and TLS. These modules enable the measurement of the distance and shape of target objects

by simulating emitted laser pulses, thereby generating high-resolution 3D point cloud data. The simulation technology based on the LESS platform finds extensive applications in algorithm development, sensitivity analysis, and forest parameter validation [12,38,39]. In this study, a well-structured 3D virtual forest scene is initially constructed. LESS is then employed to simulate TLS measurement of tree point clouds. This approach enables the generation of partial point clouds that conform to natural patterns while directly obtaining accurate ground truth information.

2.2.1. Tree Model Database and Scene Construction

We constructed a eucalyptus tree model library containing nine species to create realistic 3D scenes for the input of LESS for TLS simulations. These eucalyptus models were generated using the OnyxTREE (version 7.0 developed by Onyx Computing Inc., Cambridge, MA, USA) plant modeling software, with reference to the actual structure of eucalyptus trees (Figure 3). The model library consists of 222 individual trees, with detailed information for each species presented in Table 3.

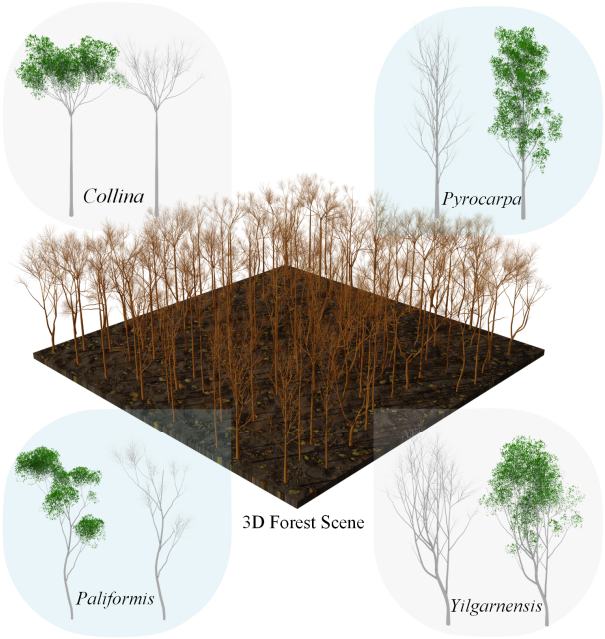


Figure 3. Tree 3D models and forest scene visualization. Each subgraph contains a rendering with foliage and a rendering with foliage removed.

Table 3. Species information (Eucalyptus).

ID	Latin Name	Height (m)	Number
T1	*Eucalyptus paliformis*	12.28 ± 2.32	31
T2	*Eucalyptus fraxinoides*	27.36 ± 3.41	22
T3	*Eucalyptus yilgarnensis*	10.45 ± 1.72	20
T4	*Eucalyptus sieberi*	25.55 ± 2.6	21
T5	*Eucalyptus ornata*	6.69 ± 1.25	31
T6	*Eucalyptus pyrocarpa*	28.31 ± 2.40	23
T7	*Eucalyptus kombolgiensis*	14.33 ± 2.82	30
T8	*Eucalyptus obtusiflora*	5.41 ± 1.40	24
T9	*Eucalyptus collina*	12.31 ± 3.15	20

Given that the focus of our point cloud completion algorithm is on the heterogeneous structure of tree branches, the leaves of each tree model were removed before constructing

the 3D scene, keeping only the branches and stems. During the virtual scene construction phase, we referred to critical structural features of the field plots to generate the forest scene with rich structure. Specifically, first of all, we combined the plant density in the field plot and used Poisson distribution [51] to generate tree positions within the 30 m × 30 m plot range (Figure 3). The number of trees was set between 110 and 130, determined based on the density of the S1 and S2 plots. Subsequently, 3D tree models were selected from individual tree database and placed at each generated position. In order to increase the diversity of the scene structure, we randomly rotated each 3D tree model from 0° to 360° along the Z axis.

2.2.2. TLS Simulation

During the TLS simulation phase, considering that point cloud data of trees in real-world environments are obtained from various sensors, we selected RIEGL VZ-1000 (RIEGL Laser Measurement Systems GmbH, Horn, Austria) scanner as the TLS simulation device in LESS to maintain the distinction between simulated and real-world sensor characteristics, which was helpful to augment the generalization of the trained model. Additionally, this setup allows us to assess the compatibility of the point cloud completion algorithm across different sensor types. The detailed parameters of the RIEGL VZ-1000 are provided in Table 4.

For the setup of TLS stations, we employed the five-stations scanning method (Figure 4) to obtain multi-level incomplete point cloud samples of varying densities. First, a central scanning station was established to ensure the scan field covered a large number of trees, followed by four corner scanning stations to imitate real TLS layout in forests (the impact of different scanning station configurations on completion results will be discussed in Section 4.3). Throughout the entire simulation process, single- ray scanning mode accelerates the generation of point cloud. Moreover, trees' labels are captured by each pulse echo and used for individual tree point clouds extraction.

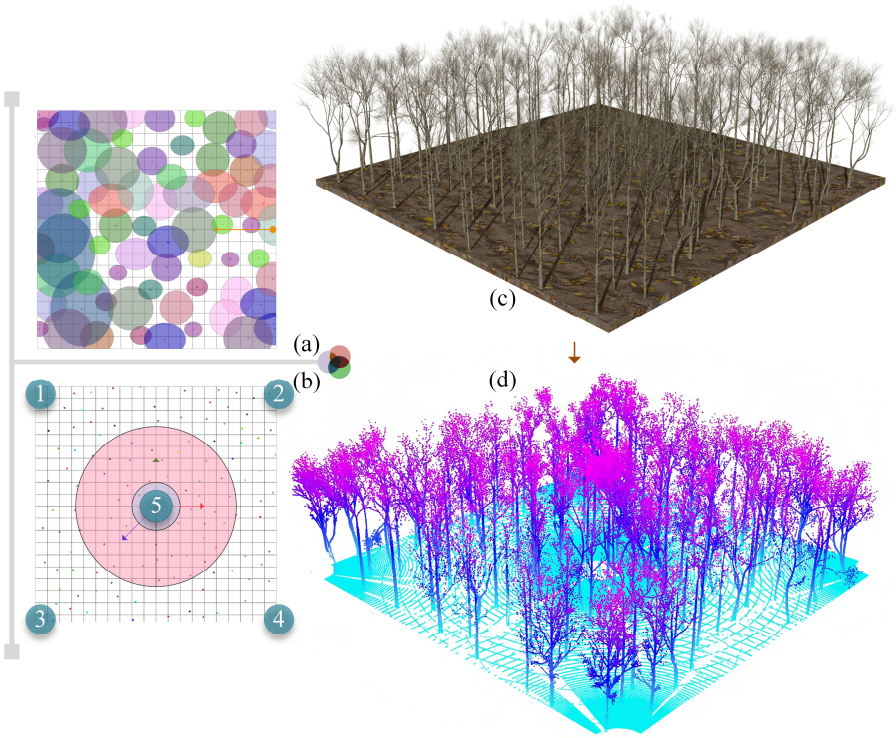


Figure 4. VLS components. (**a**) Tree locations generated using Poisson distribution; (**b**) TLS station setup, with numbers 1–5 representing the scanning station locations; (**c**) Virtual forest scene generated for simulation; (**d**) 3D point cloud data generated from TLS simulation.

Eighty plots were simulated in this study, resulting in a total of 9710 scanned individual trees. Figure 5a illustrates the distribution of the nine eucalyptus species. Figure 5b,c present the complete point clouds of various trees and point clouds with different degrees of missing data, respectively. The training process uses ground truth and incomplete point clouds.

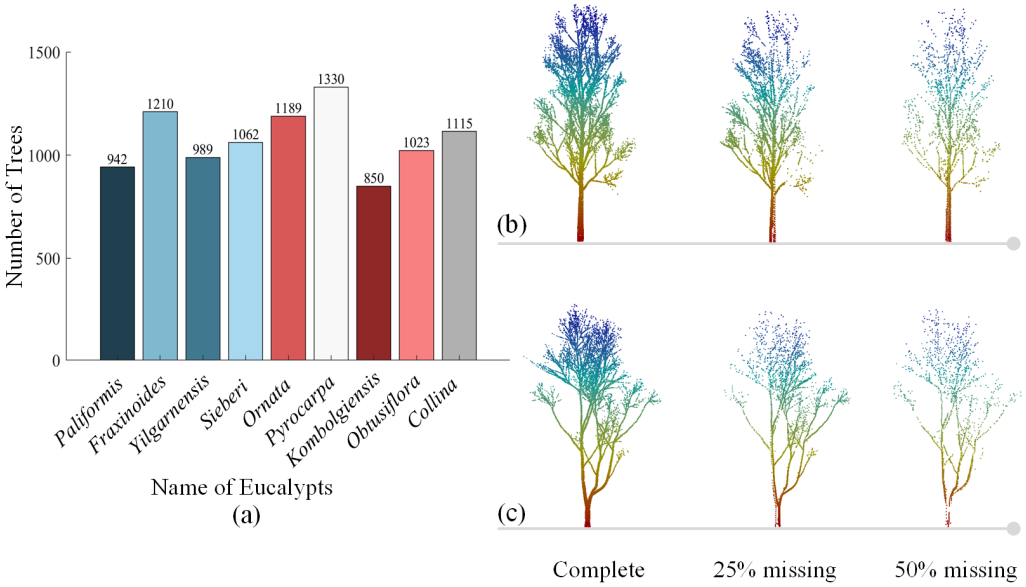


Figure 5. Training sample information. (**a**) illustrates the distribution of various tree species in terms of quantity, while (**b**,**c**) present the point clouds of complete trees, and trees with 25% and 50% missing points, respectively.

Table 4. TLS acquisition settings for each scenario.

Parameter Name	Value	Description
Delta zenith Delta azimuth	135° 360°	Range size of zenith angle or azimuth angle. Zenith angle range is generally set from 0 to 180. Azimuth angle range is generally set from 0 to 360.
Zenith resolution Azimuth resolution	0.3° 0.3°	The angle formed by two subsequent sampling points in zenith angle or azimuth is called angular resolution. The higher the angular resolution, the higher the point cloud density.
Center zenith Center azimuth	135° 360°	The position of the bisector of zenith angle or azimuth angle.
Acquisition rate	0.053 ns	The acquisition rate represents a certain sampling interval and is usually determined by the instrument itself.
Sensor Area	0.100 m^2	Sensor size (area) of LiDAR receiving energy, the sensor size (area) of LiDAR receiving energy, the larger the sensor area, the more energy it can receive.
Footprint half angle	1.5×10^{-4} rad	Field angle of LiDAR pulse. Note: This is half of the cone angle of the FOV.
Pulse energy	0.784 mJ	Emission energy of the pulse.

2.3. SK-TreePCN

The baseline model used in this study is AdaPoinTr [33] for point cloud completion. The novelty of AdaPoinTr lies in transforming the point cloud completion task into a set-to-set transformation problem. SK-TreePCN method was designed to address the heterogeneity of tree structures (Figure 6), which consists of two plug-and-play modules: SK→PS and SK→TA. Skeleton extraction and SK→PS will be introduced in Section 2.3.2, while the deep learning modules and SK→TA will be detailed in Section 2.3.3 in (d) SK→TA Cooperative Optimization.

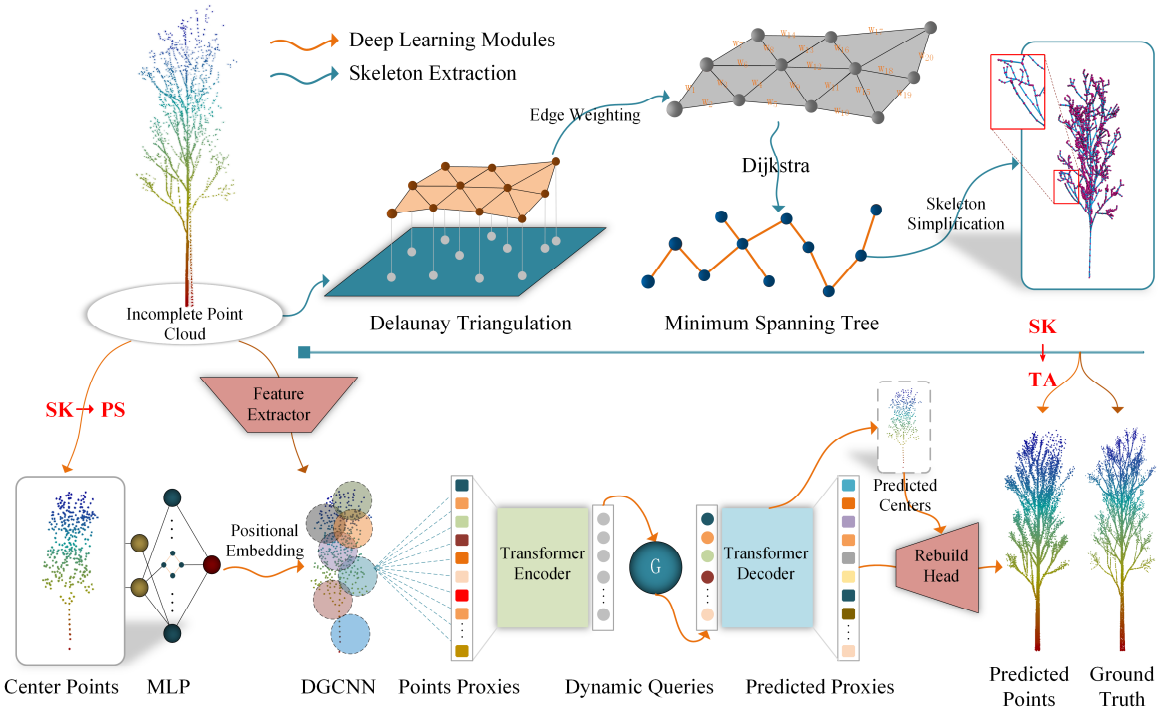


Figure 6. The pipeline of SK-TreePCN. The yellow connecting line is deep learning modules, including Dynamic Graph Convolutional Neural Network (DGCNN) feature extraction, Transformer encoder–decoder proxy prediction, and rebuild head. The blue connecting line is the tree skeleton extraction, where we decouple the extracted skeleton topology into SK→PS and SK→TA and implant them into the main module.

2.3.1. Tree Skeleton Extraction

Tree skeletons provide a simplified representation of tree structures, clarifying the relationships between branches and trunks. Incorporating tree skeletons into the tree's point cloud completion tasks enhances the point cloud connectivity and facilitates the inference of recovering missing point clouds. AdTree [10] is employed to extract the tree skeleton, followed by completing the proxy point sampler based on the skeleton topology. The detailed process is as follows:

1. Skeleton Initialization: First, the tree's point cloud is reconstructed into a triangular mesh using the Delaunay triangulation algorithm [52], followed by the computation of weights for all edges in the mesh to construct an adjacency matrix. Subsequently, the Minimum Spanning Tree (MST) is generated via Dijkstra's shortest path algorithm [53]. However, as shown in Figure 7a, when the tree's point cloud is highly dispersed, the MST alone cannot satisfactorily extract the skeleton (see Figure 7b). To address this issue, the AdTree algorithm applies the mean shift algorithm to simplify the set of points near the primary branching nodes, effectively removing incorrect branches in the skeleton. Moreover, this approach helps reduce noise from the key points, as shown in Figure 7c.
2. Skeleton Simplification: The preliminary skeleton derived from the MST may contain redundant vertices and edges, so it is necessary to simplify the MST. This refinement process unfolds in two phases: First, weights are assigned to the edges and vertices; second, neighbor points are merged through an iterative process. The tree skeleton topology is a rooted tree, which consists of a set of vertices $\boldsymbol{V}$ and edges $\boldsymbol{E}$, where the length of each point's subtree is assigned as the weight of that point. As shown in

Equation (1), for a given vertex $v \in \boldsymbol{V}$, the subtree S_v is defined as the collection of the vertex v, all its child nodes, and the edges connecting these child nodes.

$$S_v = v \cup \{v' \in \boldsymbol{V}_{\text{child}} \mid v' \text{ is a child node of } v\} \cup \{e' \in \boldsymbol{E}_{\text{child}} \mid e' \text{ is the edge of the two nodes}\}. \quad (1)$$

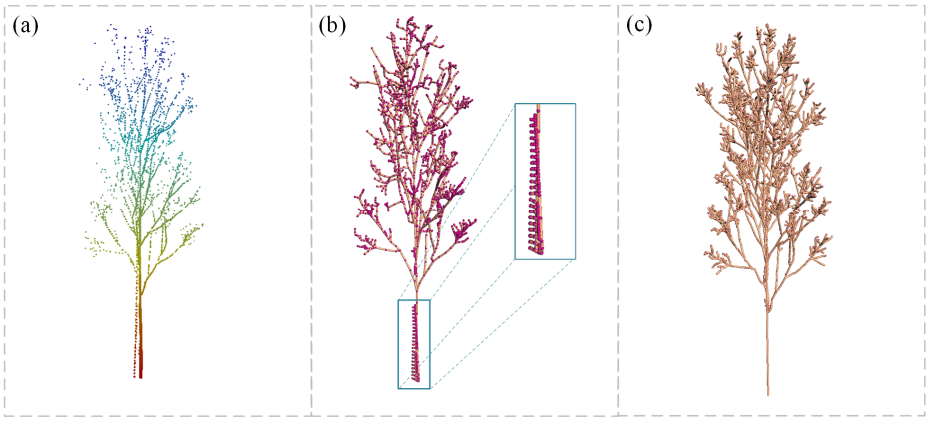


Figure 7. Schematic of skeleton initialization. (**a**) is the initial tree point cloud, (**b**) is based on the skeleton topology extracted from the MST, in which the trunk (blue box) has many redundant points, and (**c**) is the topology after mean shift noise removal.

The weights of vertices and edges are calculated based on the subtree structure: the weight of a vertex is the total sum of path lengths within its subtree, while the weight of an edge is the average weight of its endpoints. As the tree crown occupies the position of a leaf node in the tree structure, its weight is typically lower. In contrast, the root or branching nodes, which are regions of significant connectivity, possess relatively higher weights. Leveraging these features helps eliminate redundant small branches. The Douglas-Peucker algorithm [54] is then applied to remove excess vertices and edges, resulting in a simplified topological structure.

2.3.2. Skeleton Proxy Point Sampler (SK→PS)

SK→PS: Numerous point cloud completion methods utilize Farthest Point Sampling (FPS) to identify key points. However, for point clouds of trees with complex structures, the key points obtained through FPS cannot accurately represent the geometric structure [55]. Especially for some point clouds of trees that contain many tiny branches, using FPS for sampling can easily result in losing details. To address this issue, we propose the SK→PS method based on the topology of the tree skeleton. Figure 8 illustrates the processes of weight calculation and proxy point allocation in the SK→PS method.

Step 1: The edge weight of the tree's skeleton is computed using two components: the Euclidean distance between vertices and a weight based on the subtree length. For trees with straight trunks, utilizing only Euclidean distance as the edge weight may result in inadequate sampling of the trunk region. Since root and branch nodes are critical points in trees, we introduce a combination of subtree length and harmonic edge weight. For two key points, P and Q, their connecting weight is expressed as:

$$w_k^{pq} = w_e^{pq} \times w_d^{pq}, \quad (2)$$

where w_d^{pq} denotes the Euclidean distance between points P and Q, and w_e^{pq} represents the average length of the subtree between points P and Q. To ensure uniformity of measurement, the W_d and W_e values are normalized. This weighting approach ensures that regions near the tree root, which are relatively straight, maintain a higher weight, thereby

facilitating the allocation of more sampling points. The weight of smaller branches within the canopy is generally lower, thus preventing over-sampling that could arise from larger Euclidean distances calculated when leaf nodes are situated far from key points.

Step 2: The subsequent step involves the allocation of sample points. Assuming the total number of proxy points distributed throughout the tree is denoted by N, the number of points allocated between P and Q is determined by Equation (3):

$$k_{pq} = \left\lfloor \mathrm{N} \times \frac{w_k^{pq}}{\sum_{i=1}^{n} w_k^i} \right\rfloor, \tag{3}$$

where w_k^{pq} represents the joint weight between points P and Q, and $\sum_{i=1}^{n} w_k^i$ is the sum of all weights. To ensure the allocated number is an integer, the calculation is followed by applying the floor function. The location of each sample point is then identified after determining k_{pq}. As shown in Figure 9, numerous points are distributed around the key points $p\ (x_p, y_p, z_p)$ and $q\ (x_q, y_q, z_q)$. Assuming this set of points is denoted by $V = \{v_1, v_2, v_3, \ldots, v_n\}$, we divide V along the Z (height) axis into k open balls of equal radius.

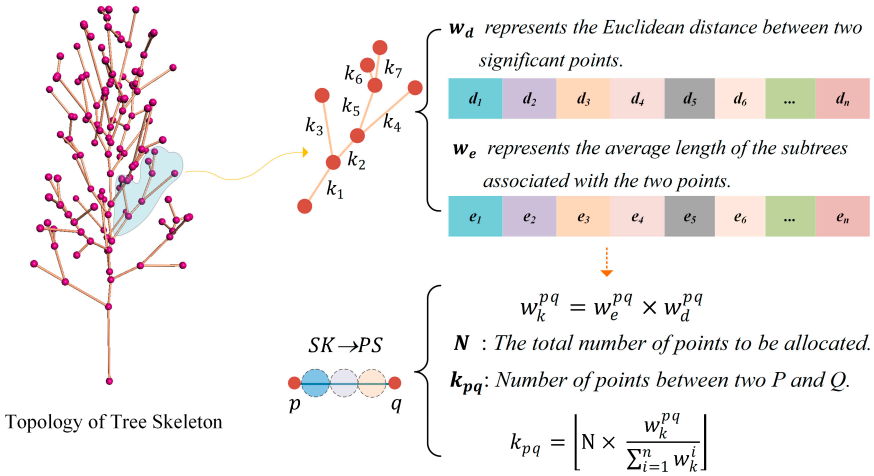


Figure 8. Illustration of skeleton proxy point sampling. The left part shows the tree skeleton topology visualization, and the right part illustrates the w_k^{pq} calculation and proxy point allocation.

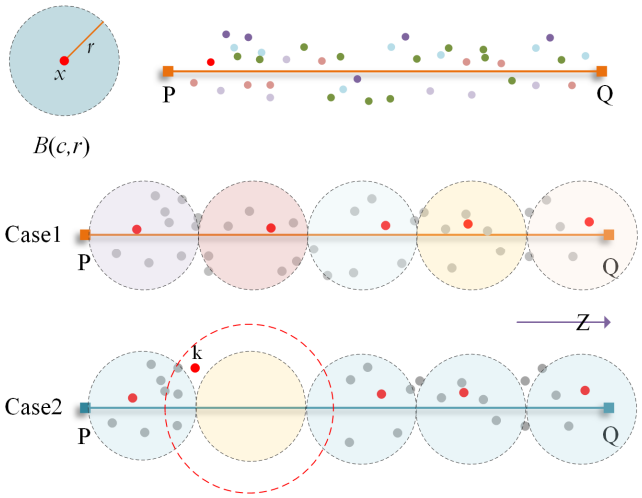


Figure 9. Calculation of the position of the proxy point between $\boldsymbol{P}$ and $\boldsymbol{Q}$. $B(x, r)$ is an open ball with center point x and radius r. The figure indicates how the location of the sampling point is determined for two different scenarios. The red points represent the allocated proxy points.

The definition of open ball $B(x, r)$ is shown in Equation (4), where c is the coordinate of the center of the ball, r is its radius, and $d(c, v)$ represents the distance between the c and point v.

$$B(c, r) = \{v | d(c, v) < r\}. \tag{4}$$

The centers and radii of the open balls are calculated as follows: Firstly, calculate the distance of P and Q along the Z axis, that is, $\Delta z = |z_p - z_q|$; Secondly, divide Δz into k equal intervals, and the length of each interval is $\delta = \frac{|z_p - z_q|}{k}$; Lastly, the radius of the open ball is $r = \frac{\delta}{2}$. The center coordinate of each open ball moves along the Z-axis direction and is represented as:

$$C_i = \left(\frac{x_p + x_q}{2}, \frac{y_p + y_q}{2}, z_p + i \times \delta \right), i = 0, 1, 2, \ldots, k_{pq}. \tag{5}$$

The point set contained in each open ball is calculated based on the Euclidean distance. There are two cases for these point sets. As shown in Figure 9, assuming that the tree point clouds are uniformly distributed. In Case 1, the points inside an open ball can be expressed as $V_{in} = \{v_i \in V | d(c, v) < r\}$. Thus, the point closest to the center of the ball is selected as the sampling point, and the calculation formula is defined as:

$$v_{closest} = \underset{v_i \in V_{in}}{argmin}\, d(c, v_i). \tag{6}$$

In Case 2 of Figure 9, highlighted by the yellow circles, if the tree point cloud are heterogeneous, it is possible that no points are found within certain open balls. This phenomenon is commonly attributed to substantial branch occlusion or constraints in the scanning field of view, which result in discontinuities within the point cloud. In this case, $V_{in} =$, we will gradually expand the search radius by 0.5 times, illustrated by the red dashed circle, until the closest point is found. $V_{closest}$ will be the final proxy point when all open ball searches are completed.

2.3.3. SK-TreePCN Architecture

The SK-TreePCN is presented in the section linked by the yellow connections within the deep learning module in Figure 6. It is primarily designed using a Transformer. A Transformer model was developed in recent years to process point clouds, with permutation invariance and the ability to capture local and global features. The approach of AdaPointTr abstracts the original point cloud into a collection of proxy point clouds, a technique that we adopted and enhanced by integrating two modules related to tree structures, thereby improving the ability to address the issue of tree point cloud completion. This section will delineate the SK-TreePCN through the following components: feature extraction, Transformer component, coarse-to-fine point cloud generation, and optimization.

(a) Feature Extraction

The point cloud format is generally in xyz form. If it is directly used as model input, it will lead to high computational complexity. AdaPoinTr transforms the initial point cloud set into a proxy point cloud set. Based on this approach, we incorporate the tree skeleton topology to guide the distribution of proxy points with the method described in Section 2.3.2. The proxy point sequence is denoted as $\{k_1, k_2, k_3, \ldots, k_n\}$, where n represents the total number of proxy points corresponding to the tree skeletons. For these N points, Dynamic Graph Convolutional Neural Network (DGCNN) [56] is first used to calculate the features of the point cloud in the field of these proxy points, denoted as f^{k_i}. A simple

Multi-Layer Perceptron (MLP) structure extracts position information and is denoted as $f(k_i)$. Then, the characteristics of the ith proxy point can be expressed as Equation (7):

$$F_k = f^{k_i} + f(k_i). \tag{7}$$

DGCNN is used to extract local semantic features around proxy points, while MLP captures the global location information of proxy points. The SK→PS method is then used to ensure that these sampling points are uniformly distributed on the tree structure to prevent the loss of small and compact branch details. In addition, it prevents the issue of under-sampling at the bifurcation points of trees in high-density canopy.

(b) Transformer Blocks

The encoder–decoder architecture in the Transformer is used to infer missing proxy points based on global features from existing proxy points. This architecture incorporates two vital modules: the Geometry-aware Transformer and Adaptive Denoising Queries. The Geometry-aware Transformer module primarily deciphers geometric connections within point clouds, maintaining the integrity of contextual links. The Adaptive Denoising Queries module is mainly engaged in addressing noise through queries.

The Geometry-aware Transformer is shown in Figure 10a. For the feature point of interest, $v \in V$, with the given query coordinate p_Q, a k-nearest neighbor search is conducted to identify k points in the surrounding neighborhood, denoted as $N_{p_k} = \{p_1, p_2, p_3, \ldots, p_k\}$. For the point set N_{p_k}, DGCNN is employed for feature extraction, denoted as f^d. This feature f^d is then processed through a linear layer, symbolized as $f^{d\to l}$, to highlight the significant features within the k neighborhood. The extracted geometric features are then aggregated through Max Pooling. Finally, the features f^{att} captured by the multi-head attention are concatenated, as shown in Equation (8).

$$F_v = maxP\left(f_{p_i}^{d\to l}\right) \oplus f_v^{att}, p_i \in N_{p_k}. \tag{8}$$

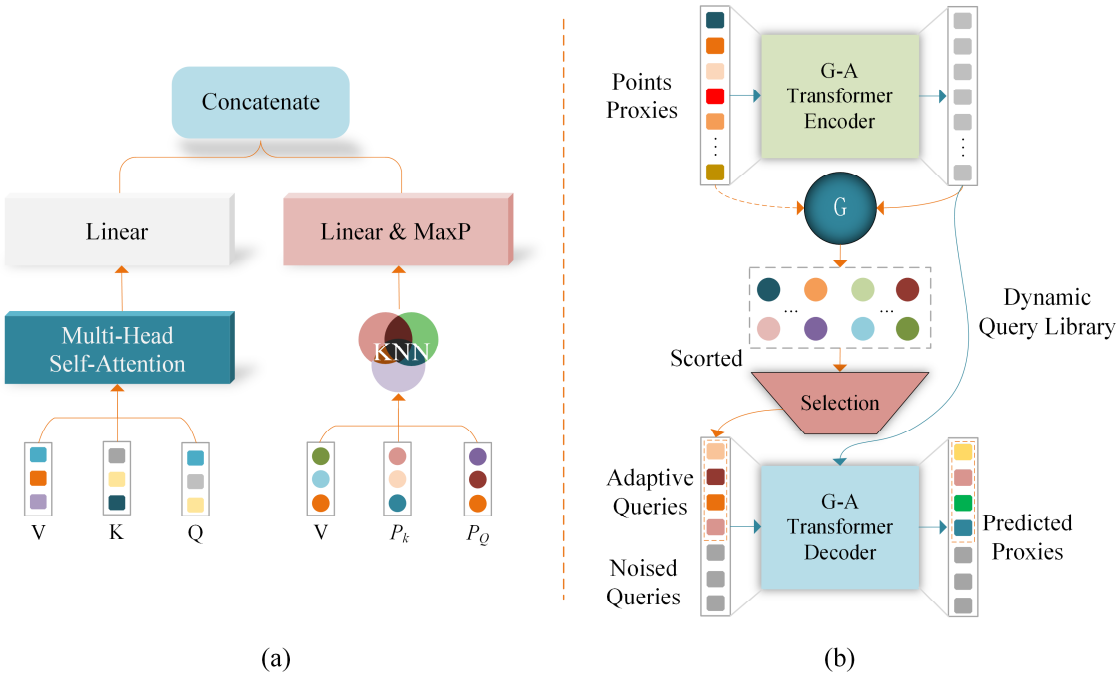


Figure 10. Transformer blocks. (**a**) Geometry-aware Transformer; (**b**) Adaptive Denoising Queries.

The Adaptive Denoising Queries, shown in Figure 10b, address prediction discontinuity by interconnecting two sets of proxy points. Initially, the outputs from the encoder

and the input proxy points are refined to establish a dynamic query library through linear projection, max pooling, and the adjustment of coordinates from both the encoder output and the input proxy points. A simplified scoring method is then employed to evaluate the queries based on their forecasted scores, selecting M queries with superior scores for coordinate documentation. This approach is specifically designed to mitigate problems such as noise.

(c) Point Cloud Generation

The encoder–decoder predicts only the central proxy points representing the missing point cloud. Subsequently, the rebuilt head (FoldingNet [37]) incrementally refines these proxy points to produce a high-resolution point cloud in a coarse-to-fine manner. Commonly, in various point cloud completion techniques, the reconstruction is achieved by a basic merging process, which involves combining the initially incomplete point cloud with the one produced by the model. Nonetheless, such initial merge processes can lead to distortions and discontinuities in the point cloud's appearance. AdaPoinTr employs a more straightforward approach, as presented in Equation (9).

$$P_{tree} = g(C_{in} \oplus C_{out}). \tag{9}$$

In Equation (9), P_{tree} represents the fully restored point cloud, and g denotes the FoldingNet operation. C_{in} is the proxy point extracted via the SK→PS method, and C_{out} is the predicted missing proxy point. Before applying FoldingNet for generation, the input proxy points are first queried and linked to the predicted missing proxy points. These proxy points are input into the rebuilt head as a complete structure for refined generation.

(d) SK→TA Cooperative Optimization

Throughout the training phase, the loss function for point cloud completion refines the model parameters to reduce the reconstruction error of the point cloud, serving as a crucial foundation for facilitating thorough completion of the point cloud. Given the inherent irregularity and randomness of the point cloud, we adopt the Chamfer distance (CD) as the metric for point cloud completion evaluation, as it shows comparatively modest computational demands [24,30,33,36,55]. Moreover, considering the complexity of tree structures, such as the intertwined nature of the tree crown, we incorporate the tree skeleton topology as a constraint in the optimization process. The formula for calculating the CD is as follows:

$$l_{(p,g)} = \frac{1}{n}\sum_{i=1}^{n} min_{j=1}^{m}\|p_i - g_j\| + \frac{1}{m}\sum_{j=1}^{m} min_{i=1}^{n}\|g_j - p_i\|, \tag{10}$$

where $l_{(p,g)}$ represents the CD between point set p and g, with p containing n points and g containing m points. The optimization process is illustrated in Figure 11.

As shown in Figure 11, the optimization computational graph consists of three parts: $l(P_{de}, G_{de})$, $l(P_{sk}, G_{sk})$, and $l(P_t, G_t)$. Firstly, the incomplete point cloud (the input in Figure 11) is processed by the Adaptive Denoising Queries module to identify noise proxy points. To enhance the robustness of the denoising process, $f(q)$ is applied to these proxy points to reconstruct the predicted point cloud. The CD between the predicted point cloud and the ground truth is then calculated as $l(P_{de}, G_{de})$. Secondly, to ensure the generated proxy points better align with the tree skeleton topology, we apply the $f(sk)$ operation to extract the skeletons from the predicted point cloud and ground truth, and calculate their CD, denoted as $l(P_{sk}, G_{sk})$. Thirdly, $l(P_t, G_t)$ calculates the CD between the predicted point cloud and the ground truth. Finally, the three optimization functions are combined into Equation (11). For the noise factor λ, we follow the design principles of AdaPoinTr, setting it to 0.5.

$$L = l(P_{sk}, G_{sk}) + l(P_t, G_t) + \lambda \times l(P_{de}, G_{de}). \tag{11}$$

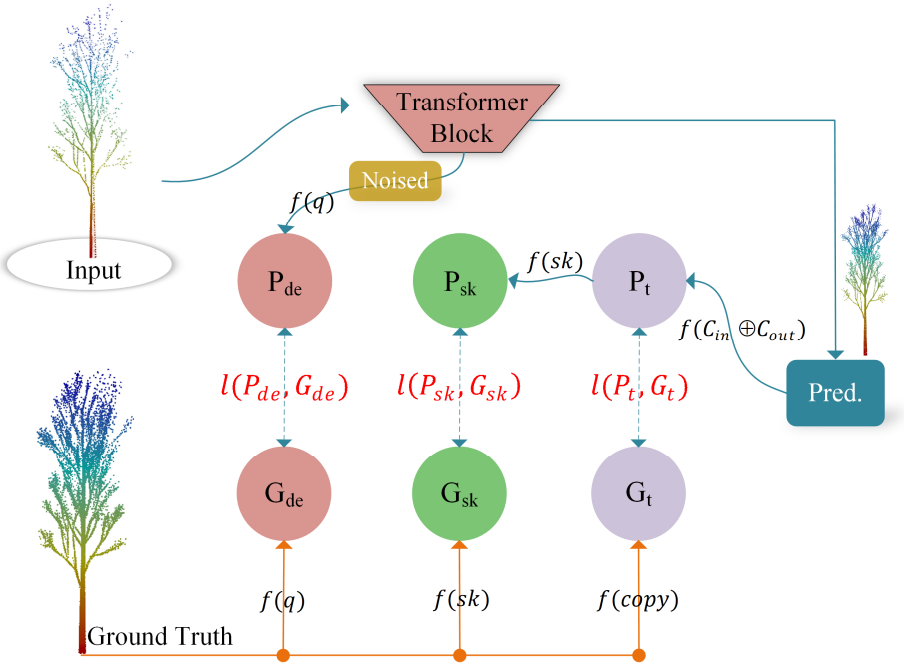


Figure 11. The optimization computational graph for SK-TreePCN. The upper branch comprises the computation of reconstructed noise P_{de}, completion of point cloud P_t, and extracted skeleton P_{sk}. The lower branch includes detected ground truth noise G_{de}, ground-truth skeleton G_{sk}, and complete point cloud G_t. $f(q)$ represents the reconstruction of local shapes by the noise proxy points, and $f(sk)$ represents the skeleton extraction. $f(copy)$ is the copying, and $f(C_{in} \oplus C_{out})$ represents the point cloud merge operation.

2.4. Validation

2.4.1. Dataset and Preparation

VLS Dataset: The VLS dataset of incomplete point cloud and ground truth is obtained using the LESS simulation in Section 2.2. The incomplete point cloud from 80 plots is split into 55 plots (6797 trees) as the training set and 25 plots (2913 trees) as the test set. To the sampling independence, the trees in the training and test sets come from trees of different shapes in the tree library. Ground truth is obtained following the production method in PCN, and each tree has 16,384 points. Both incomplete point cloud and ground truth are normalized to a unit ball.

Field Dataset: The field dataset (Section 2.1) is used to demonstrate the model's generalization and transfer capabilities. After the data preprocessing, the individual tree point cloud is first down-sampled with uniform sampling. The point cloud is then normalized before being completed by the point cloud completion model. Additionally, AdTree is used to reconstruct the 3D model and extract the tree structure parameters, which are used for comparison with the field survey data.

2.4.2. Implementation Details

During model training, each model's epoch is set to 600, the model is saved based on its optimal performance, and GPU graphics cards are used for acceleration. The hardware and software configurations for the computation are shown in Table 5.

Table 5. Hardware and software information.

	Component	Specification/Version
Hardware	Processor	AMD Ryzen 9 3900X 12-Core Processor 3.79 GHz
	Graphics Card	NVIDIA GeForce RTX 3080 TI2
	RAM	64 GB
	Disk	1 TB SSD+4 TB HDD
Software	Operating System	Ubuntu 22.04
	NVIDIA Driver	535.129.03
	CUDA Library	11.2
	PyTorch	1.13.1+CUDA11.7

2.4.3. Evaluation Metrics

We adopt CD and $F1$ Score as the evaluation metrics for the point cloud completion models, following the work of AdaPoinTr. Given a ground truth point cloud $\boldsymbol{P}$ and a predicted point cloud $\boldsymbol{G}$, the CD between $\boldsymbol{P}$ and $\boldsymbol{G}$ is defined as Equation (12), where CD_1 denotes the L1-norm, and CD_2 represents the L2-norm.

$$CD_{(\boldsymbol{P},\boldsymbol{G})} = \frac{1}{|p|}\sum_{p\in\boldsymbol{P}}\min_{g\in\boldsymbol{G}}\|p-g\| + \frac{1}{|g|}\sum_{g\in\boldsymbol{G}}\min_{p\in\boldsymbol{P}}\|g-p\|. \tag{12}$$

$$F1 = 2\times\frac{p_d\times r_d}{p_d+r_d}. \tag{13}$$

The $F1$ Score, as defined in Equation (13), utilizes a given threshold d, where $p_d = \frac{1}{|p|}\sum_{p\in\boldsymbol{P}}\left[\min_{g\in\boldsymbol{G}}\|p-g\| < d\right]$ represents the precision of point cloud completion at d, and $r_d = \frac{1}{|g|}\sum_{g\in\boldsymbol{P}}\left[\min_{p\in\boldsymbol{P}}\|g-p\| < d\right]$ represents the recall of point cloud completion at d, with d set to 1%. In this context, CD_1 and CD_2 approach 0, with their values in the range [0, +∞), and the $F1$ score approaches 1, with its value in the range [0, 1], indicating better performance.

3. Results

3.1. Comparison with State-of-the-Art Completion Methods

To evaluate the robustness and generalization of SK-TreePCN, we qualitatively and quantitatively compared several representative point cloud completion models, including PCN [24], GRNet [57], TopNet [36], FoldingNet, and AdaPoinTr, using their optimal parameter settings. The validation data consisted of the VLS dataset and the field dataset. The corresponding values of CD_1 and $F1$ for each approach were depicted in Figure 12, with exact numerical data provided in Tables 6 and 7. Additionally, Figures 13 and 14 presented the completion visualizations for the VLS and field datasets.

Table 6. CD_1 (↓) across different methods. Bold numbers indicate the optimal CD_1 for each tree species.

Type	Tree Species	PCN	GRNet	TopNet	FoldingNet	AdaPoinTr	Ours
VLS	*Paliformis*	14.568	9.703	14.494	25.804	6.966	**5.185**
	Fraxinoides	13.337	9.404	12.403	18.743	6.935	**5.109**
	Yilgarnensis	23.753	12.951	21.308	29.190	10.451	**8.324**
	Sieberi	20.564	12.131	16.013	28.347	8.434	**6.122**
	Ornata	15.337	12.756	15.672	24.937	10.155	**8.090**
	Pyrocarpa	26.088	12.655	23.410	34.868	8.592	**6.415**
	Kombolgiensis	24.170	13.467	22.239	35.017	9.918	**7.201**
	Obtusiflora	15.062	9.895	13.891	22.819	7.408	**5.309**
	Collina	10.179	9.503	10.251	16.187	5.848	**4.214**
	Avg.	18.118	11.385	16.631	26.212	8.301	**6.219**
Field	S1	17.593	17.281	18.271	18.641	8.641	**4.642**
	S2	16.384	15.036	19.305	22.814	7.905	**5.691**
	Avg.	16.989	16.159	17.788	20.728	8.273	**5.167**

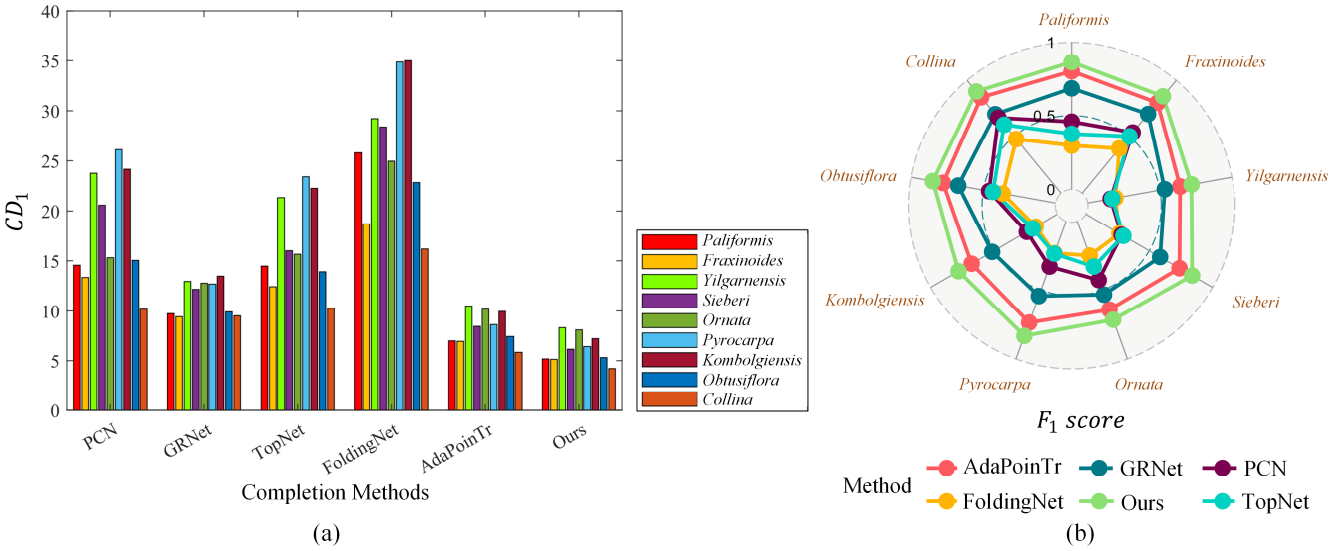


Figure 12. CD_1 (↓) and $F1$ (↑) of the results. In (**a**), each cluster of bars represents the CD_1 for different tree species using a specific completion method. In (**b**), each ring of the same color represents the $F1$ for different tree species using a specific completion method.

Table 7. $F1$ (↑) across different methods. Bold numbers indicate the optimal $F1$ for each tree species.

Type	Tree Species	PCN	GRNet	TopNet	FoldingNet	AdaPoinTr	Ours
VLS	*Paliformis*	0.458	0.686	0.375	0.296	0.803	**0.865**
	Fraxinoides	0.536	0.700	0.502	0.394	0.800	**0.858**
	Yilgarnensis	0.158	0.534	0.170	0.192	0.641	**0.720**
	Sieberi	0.282	0.586	0.296	0.264	0.739	**0.839**
	Ornata	0.427	0.533	0.329	0.249	0.642	**0.715**
	Pyrocarpa	0.329	0.543	0.235	0.231	0.736	**0.832**
	Kombolgiensis	0.242	0.515	0.195	0.172	0.677	**0.780**
	Obtusiflora	0.461	0.675	0.434	0.366	0.780	**0.851**
	Collina	0.666	0.699	0.605	0.477	0.849	**0.901**
	Avg.	0.395	0.608	0.349	0.293	0.714	**0.818**
Field	S1	0.428	0.392	0.225	0.388	0.764	**0.841**
	S2	0.486	0.456	0.267	0.332	0.785	**0.835**
	Avg.	0.457	0.424	0.246	0.360	0.776	**0.838**

Tables 6 and 7, and Figure 12 show that the SK-TreePCN method surpassed all other point cloud completion models and attained the highest CD_1 and $F1$ scores across both datasets and all tree species. Since SK-TreePCN evolved from AdaPoinTr, we particularly compared it with other non-transformer-based models and then conducted a focused comparison with AdaPoinTr. The comprehensive results are as follows:

In the VLS dataset, SK-TreePCN outperformed PCN, GRNet, TopNet, and FoldingNet by reducing CD_1 by 11.899, 5.166, 10.412, and 19.993, respectively, while increasing $F1$ by 0.423, 0.21, 0.469, and 0.525, respectively. In the field dataset, SK-TreePCN reduced CD_1 by 11.822, 10.992, 12.621, and 15.561, along with $F1$ improvements of 0.381, 0.424, 0.592, and 0.478. Furthermore, we noted a generally uniform performance improvement in SK-TreePCN over AdaPoinTr, with both achieving superior CD_1 and $F1$ scores compared to non-transformer-based approaches. Although AdaPoinTr is also based on a transformer architecture, SK-TreePCN reduced CD_1 in the VLS dataset by 2.091 and increased $F1$ by 7.7% in comparison. In the field dataset, CD_1 was decreased by 3.106, with an $F1$ increase of 6.2%.

Furthermore, we presented the visualization results of point cloud completion for trees in both the VLS and field datasets across various point cloud completion methods (Figures 13 and 14). These methods exhibited both consistency and variation, with consistency reflected in similar results across columns and variation observed in distinct differences across rows. In their completion results, TopNet (topology-based) and FoldingNet (autoencoder-based) focused on reconstructing the outer structural envelope of

trees. PCN (point-based) and GRNet (convolution-based) only captured the macroscopic shape of the trees without reconstructing branches and finer details. In contrast, AdaPoinTr and SK-TreePCN (Transformer-based) not only generated point clouds consistent with the natural structure of trees but also produced clear trunks and branches.

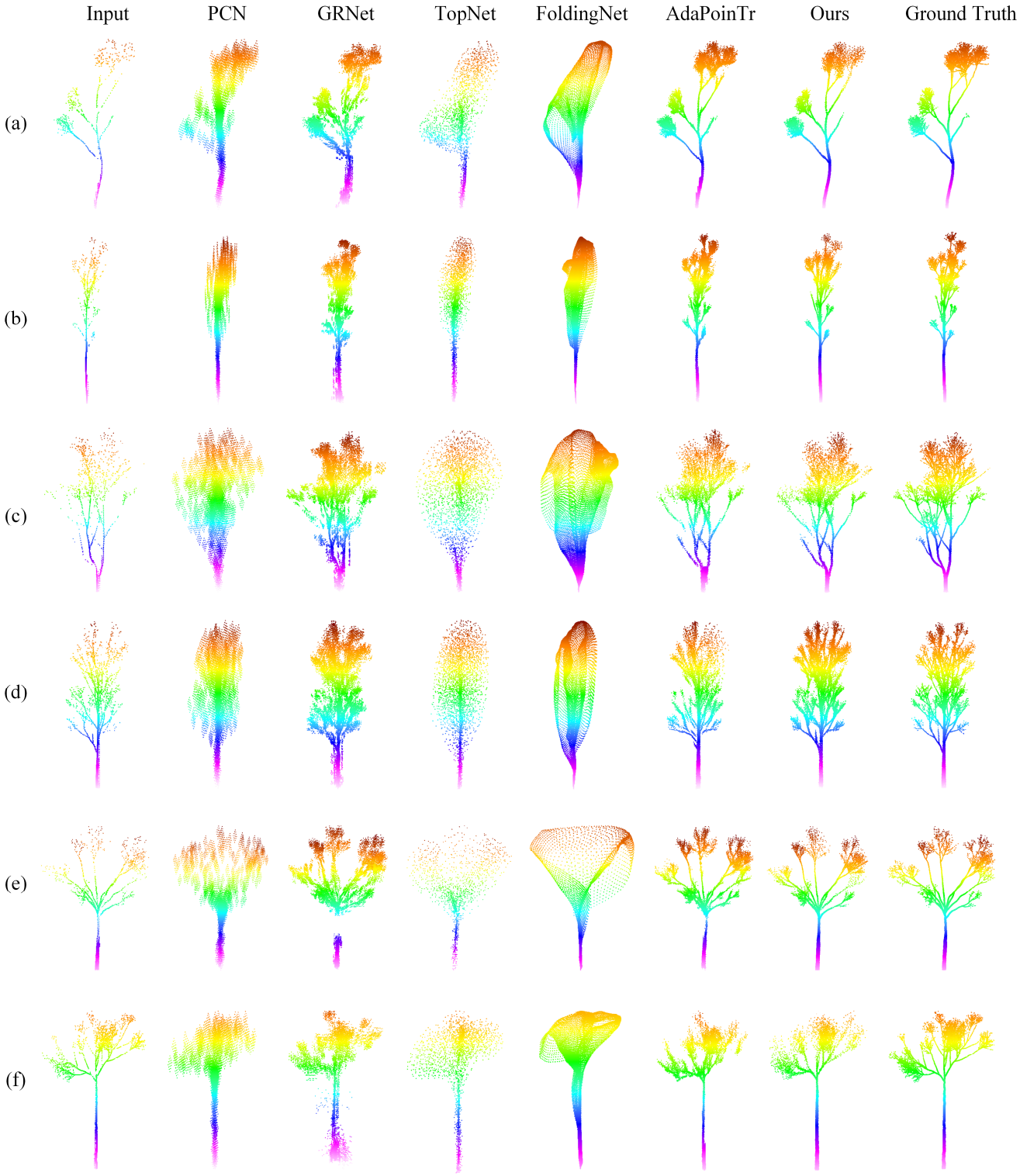


Figure 13. Qualitative results of various completion methods on the VLS dataset. The first column shows incomplete point clouds, and the last column displays the ground truth. (**a**–**f**) illustrate the visualization results for different trees using various point cloud completion methods.

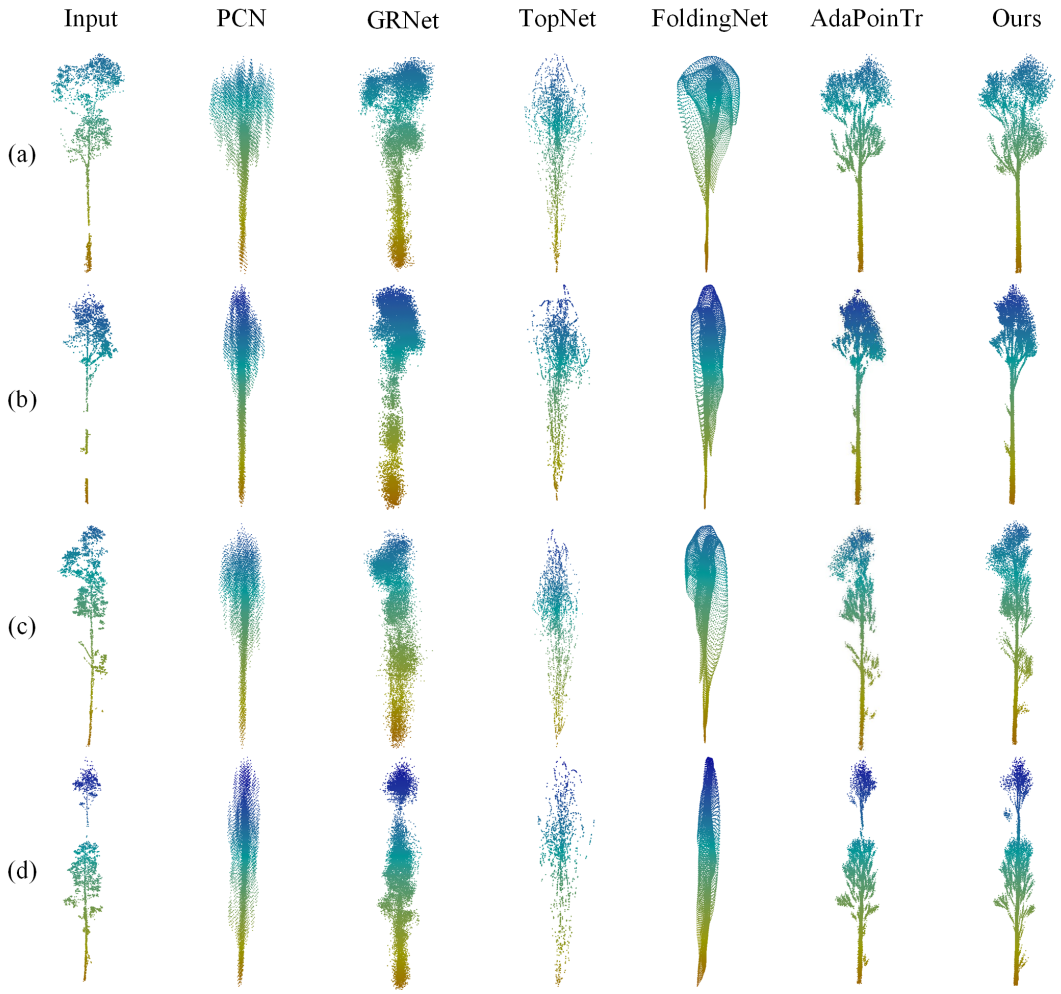


Figure 14. Qualitative results of various completion methods on the field dataset. The first column shows incomplete point clouds. (**a**–**d**) illustrate the visualization results for different trees using various point cloud completion methods.

In the meantime, Figures 13 and 14 illustrate notable distinctions between AdaPoinTr and SK-TreePCN. The point clouds generated by AdaPoinTr for the tree trunks, as illustrated in Figures 13a–f and 14b, exhibited issues such as jagged edges, overlapping, and underfilling. Similar issues were also apparent in the field dataset (Figure 14b–d). The point clouds generated by SK-TreePCN exhibited higher realism and smoothness. When looking at the tree branches, the point clouds by AdaPoinTr exhibited divergence and discontinuity, as shown in Figures 13d and 14a,b,d. In contrast, the tree point clouds completed by SK-TreePCN were geometrically continuous and had uniform density. Moreover, despite the varying levels of missing data in the TLS point cloud, SK-TreePCN consistently demonstrated superior performance.

3.2. Ablation Result of SK-TreePCN

Ablation: In Table 8, four models differed based on whether the SK→PS and SK→TA modules were enabled. Quantitative results were presented for each model. From the evaluation metrics, it was evident that performance was highest when both SK→PS and SK→TA were enabled. Compared to Model B, CD_1 and CD_2 decreased by 1.145 and 0.037, respectively, while $F1$ improved by 4.2%. Compared to Model C, CD_1 and CD_2 decreased by 1.509 and 0.098, respectively, with a 6.0% increase in $F1$. Compared to the baseline Model A, CD_1 and CD_2 declined by 2.082 and 0.106, respectively, while $F1$ improved by 7.7%.

Table 8. Ablation study of SK-TreePCN: incremental addition of skeleton proxy point sampler (SK→PS) and topology regulator (SK→TA).

Model	SK→PS	SK→TA	CD_1↓	CD_2↓	$F1$↑
A			8.301	0.285	0.741
B	√		7.364	0.216	0.776
C		√	7.728	0.277	0.758
D	√	√	6.219	0.179	0.818

Details: Figure 15 presents the results of completing the tree point cloud using AdaPoinTr (baseline model) and SK-TreePCN. There were different levels of incompleteness in the tree point clouds. For instance, with AdaPoinTr completion, the main trunks were difficult to discern in Tree1 and Tree2, and the canopy density appeared sparse in Tree3. In contrast, SK-TreePCN demonstrated no such deficiencies; the resulting point clouds were continuous, smooth, and uniformly distributed across the entire tree structures.

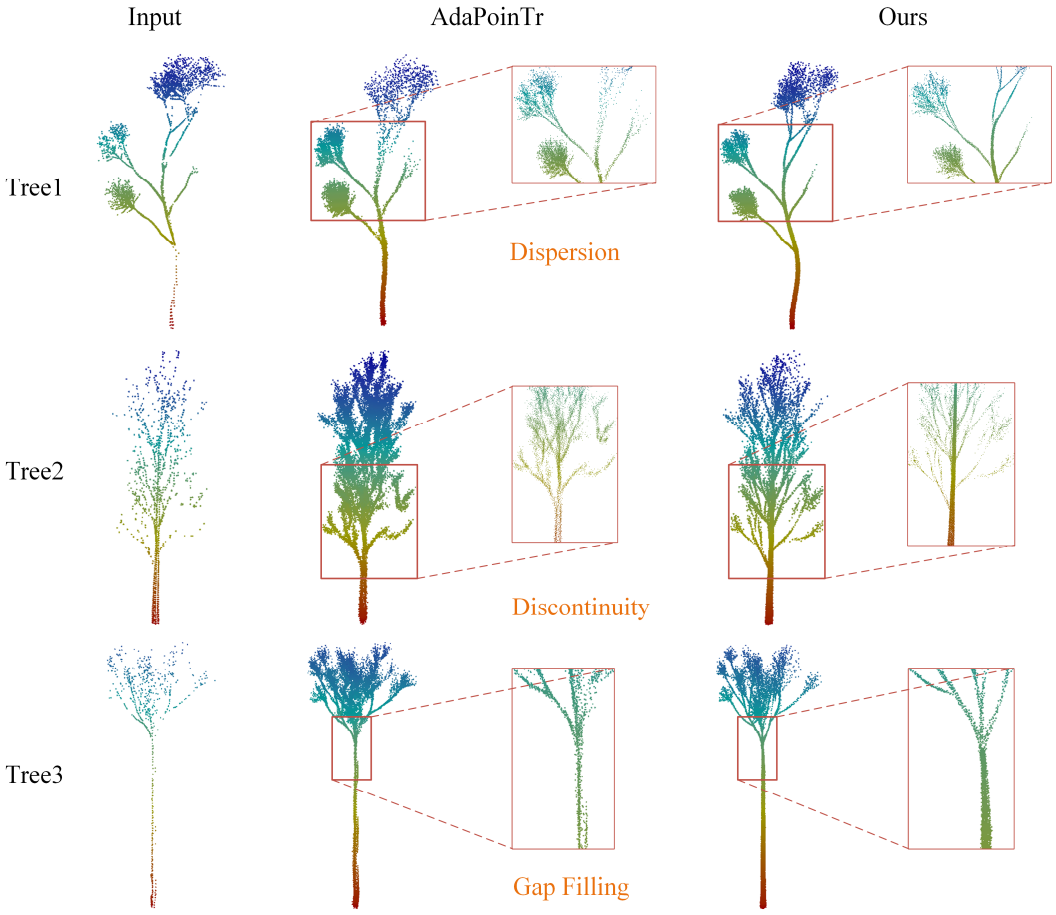


Figure 15. Tree point cloud completion results of AdaPoinTr and SK-TreePCN. Zoom in views of the details in the red box.

3.3. Validation Results of Reconstruction

Visualization validation: We used field data to show the 3D reconstruction results (Figure 16). These point clouds exhibited distinct characteristics, including low point density (T2), unconnected branches (T4, T5, and T6), and a complex canopy structure (T3). The results demonstrated that, across various point cloud conditions, the trees generated by our method exhibited visual consistency with actual tree structures. The branches in the completed point clouds were detailed, and the canopy features were well-defined.

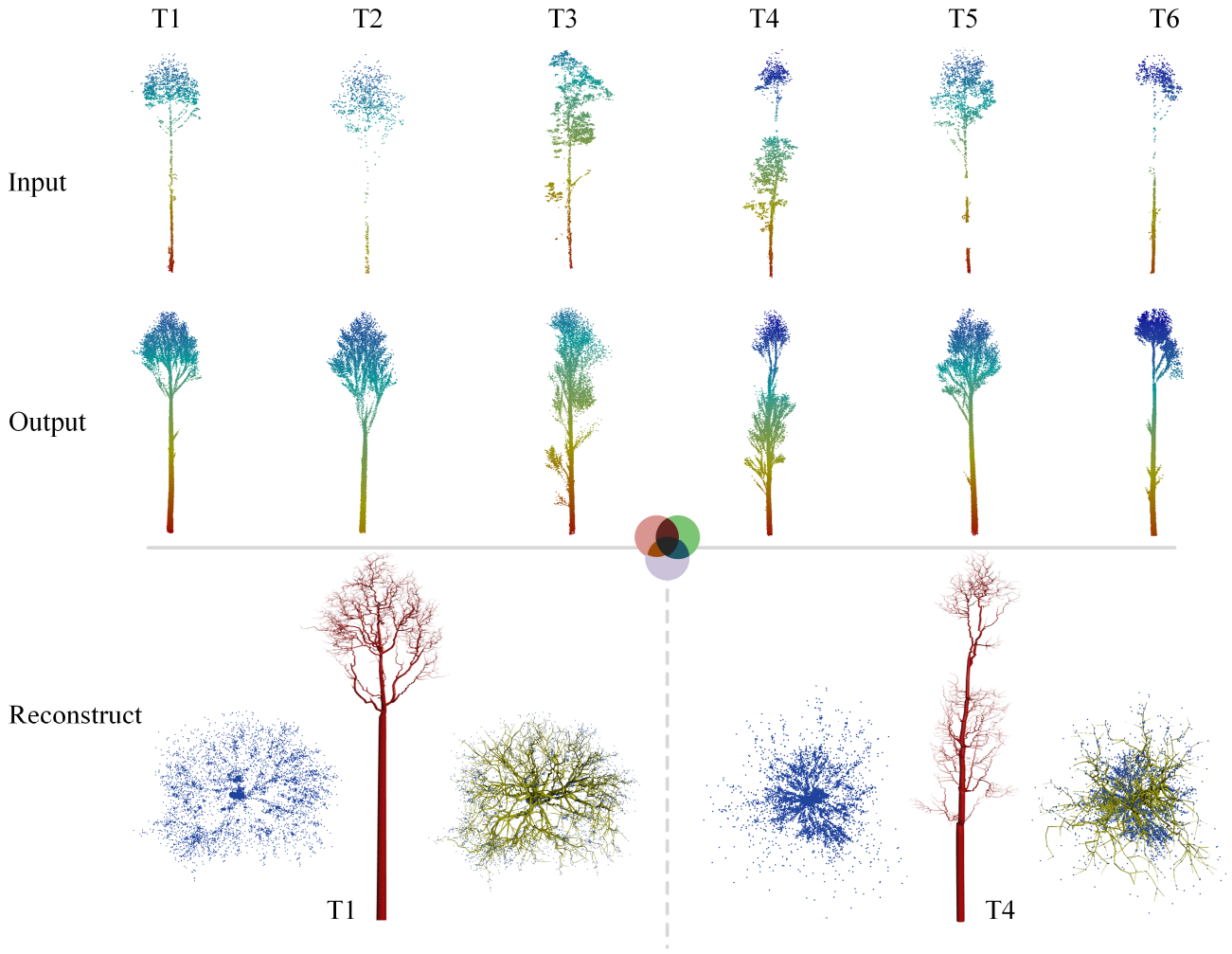


Figure 16. Completion results of field data. The last row visualizes the 3D models reconstructed by AdTree.

QSM validation: AdTree reconstruction was performed on the completed point cloud (shown at the bottom of Figure 16), after which tree structural parameters were extracted. Figure 17 illustrates the fitting results, where RD, RH, and RCW correspond to field-measured DBH, tree height, and average crown width, and ED, EH, and ECW represent estimated values obtained through QSM reconstruction.

In Figure 17a,c,e, the first column displayed the fitting results of estimated versus observed tree parameters from the incomplete point cloud, before point cloud completion. The second column, Figure 17b,d,f, showed the fitting results after the point cloud was completed. A detailed comparison was conducted between the two field plots (S1 and S2). In the two plots, the arithmetic mean R^2 values for DBH, tree height, and crown width were 0.836, 0.920, and 0.914, with corresponding arithmetic mean RMSE (Root Mean Square Error) values of 0.847, 0.579, and 0.184. After completing the point cloud using SK-TreePCN, the arithmetic mean R^2 values for DBH, tree height, and crown width increased to 0.904, 0.929, and 0.916, with arithmetic mean RMSE values of 0.588, 0.528, and 0.181. Upon completion, the R^2 for DBH improved by 0.068, and RMSE decreased by 0.259 cm; the R^2 for tree height rose by 0.009, while RMSE declined by 0.051 m; and the R^2 for crown width increased by 0.002, with RMSE decreasing by 0.003.

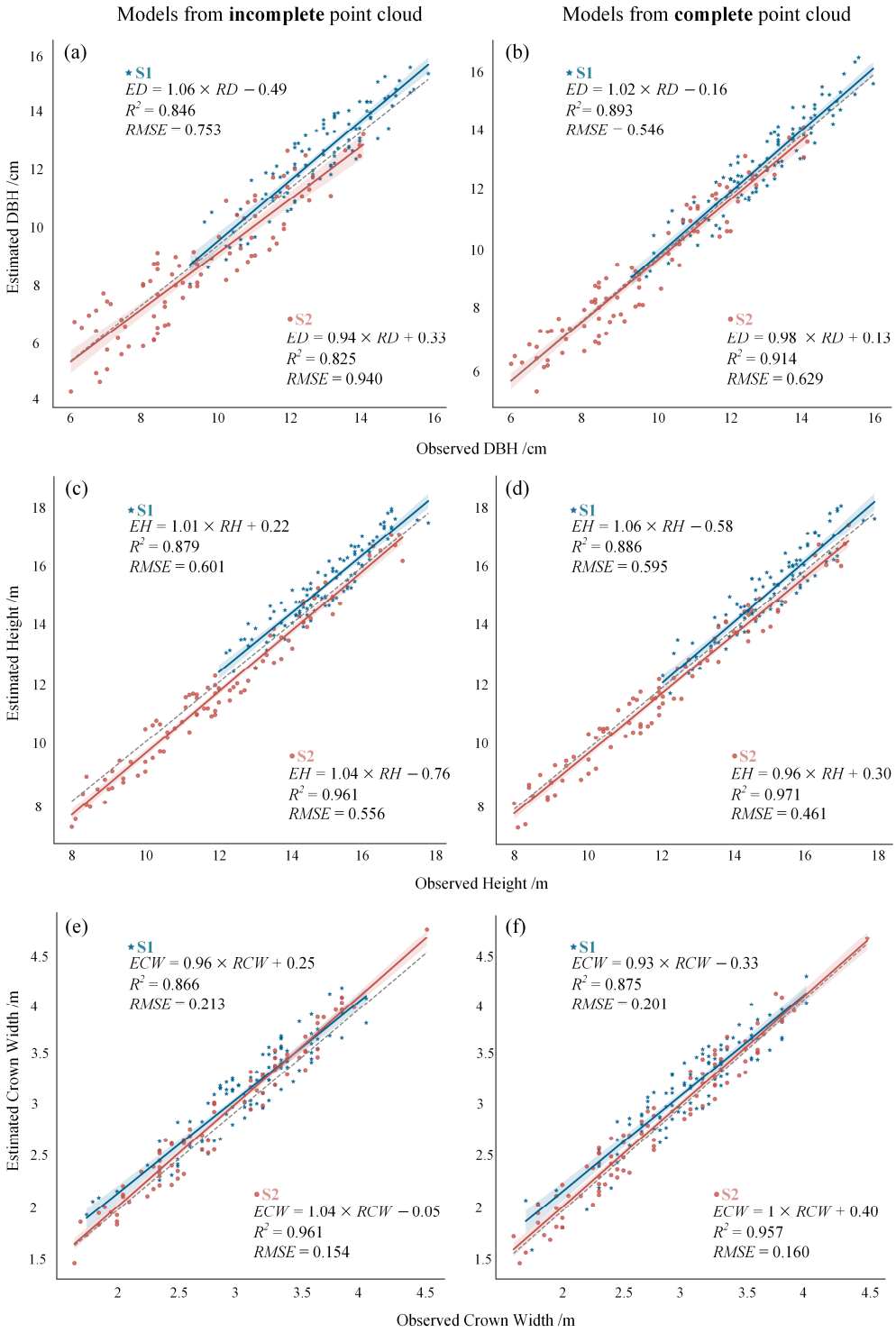


Figure 17. Comparative analysis of structural parameters. The left (**a**,**c**,**e**) and right (**b**,**d**,**f**) columns show the DBH, tree height, and crown width fitting results from the incomplete and completed point clouds, respectively. S1 (blue asterisks) and S2 (red dots) represent tree samples from two different plots.

4. Discussion

4.1. Quantitative and Qualitative Analysis

In Section 3.1, the quantitative results indicated that our model demonstrated superior performance in terms of CD_1 and $F1$ across the VLS and field datasets. Specifically, Figure 12 demonstrated that transformer-based designs (AdaPoinTr and SK-TreePCN) significantly

outperformed non-transformer models by effectively capturing the point cloud's local and global features. Compared to the AdaPoinTr model, the integration of tree topology significantly improved point cloud completion performance, illustrating that the topological structure enhanced the model's capacity to infer missing point clouds.

Additionally, it was noteworthy that the (CD_1, $F1$) values for *Collina*, *Fraxinoides*, and *Paliformis* were (5.518, 0.865), (5.109, 0.858), and (4.214, 0.901), respectively, while for *Yilgarnensis* and *Ornata*, the values were (8.324, 0.720) and (8.090, 0.715). This discrepancy was attributable to differences in tree morphology. *Collina*, *Fraxinoides*, and *Paliformis* had predominantly straight trunks and simple canopy formations, whereas *Yilgarnensis* and *Ornata* exhibited more intricate structures. This suggested that the point cloud completion performance for trees with complex structural characteristics was generally less effective than for trees with simpler structures.

In Figures 13 and 14, qualitative results showed differences and consistencies. The completion styles for different trees by the same model were analogous in each column, but the results by different models varied markedly across each row. Table 9 evaluated the potential reasons behind these trends from the perspective of model structures.

Table 9. Performance and causes analysis of point cloud completion models.

Method	Completion Performance	Causes
PCN	Composed of many 2D grid blocks with similar densities.	The coarse-to-fine process of PCN deformed the 2D grid into a 3D surface. The surface structure of the tree point cloud was a thin layer, making it difficult to deform into such a thin structure using 2D grid blocks. Is was also a limitation of PCN [24].
GRNet	Rough surface and uneven density. The point cloud was discontinuous (13e).	GRNet consisted of three parts: Gridding, Gridding Reverse, and Cubic Feature Sampling [57]. However, there were many fine branches in the canopy of trees. When performing Cubic Feature Sampling, multiple tree branches appeared in a single grid, and connecting these branches from different grids was error-prone.
TopNet	Uniformly diverging, lacking fine details.	TopNet used the Hierarchical Rooted Tree decoder to infer arbitrary topologies and structures from discrete point sets [36]. However, the topology of trees usually did not contain cycles; that is, the genus was 0. When using the Hierarchical Rooted Tree decoder to build a tree topology, loops were easily formed at the bifurcation of the trees.
FoldingNet	Approximating a manifold.	FoldingNet primarily used 2D grids to fold into 3D surfaces. This operation differed from PCN in that PCN was folded in multiple blocks, while FoldingNet was based on folding the entire surface. Consequently, the point cloud completion result was closer to a manifold.
AdaPoinTr	Similar overall structures, but many errors in the details.	It is similar in overall structure, but the details have many errors. Transformer can capture global features, but tree structures are highly heterogeneous, and tree morphology needs to be considered.
Ours	Accurate structures and apparent details.	Adding tree topology made the completion result approximate the actual distribution more closely.

In addition, we compared the training and inference times of the different models. Figure 18a illustrates the training time required for each model over 600 epochs, while Figure 18b shows the average inference time and variance for each model when inferring

100 point clouds. In terms of training time, the PCN model took the longest, with 11.5 h, while FoldingNet had the shortest training time of 7.5 h. This difference can primarily be attributed to the simpler architecture of the FoldingNet model. All models completed training within 12 h, using two 3080Ti GPUs. Notably, the inference time for all models did not exceed 150 milliseconds, indicating that the model complexity has negligible impact on inference time.

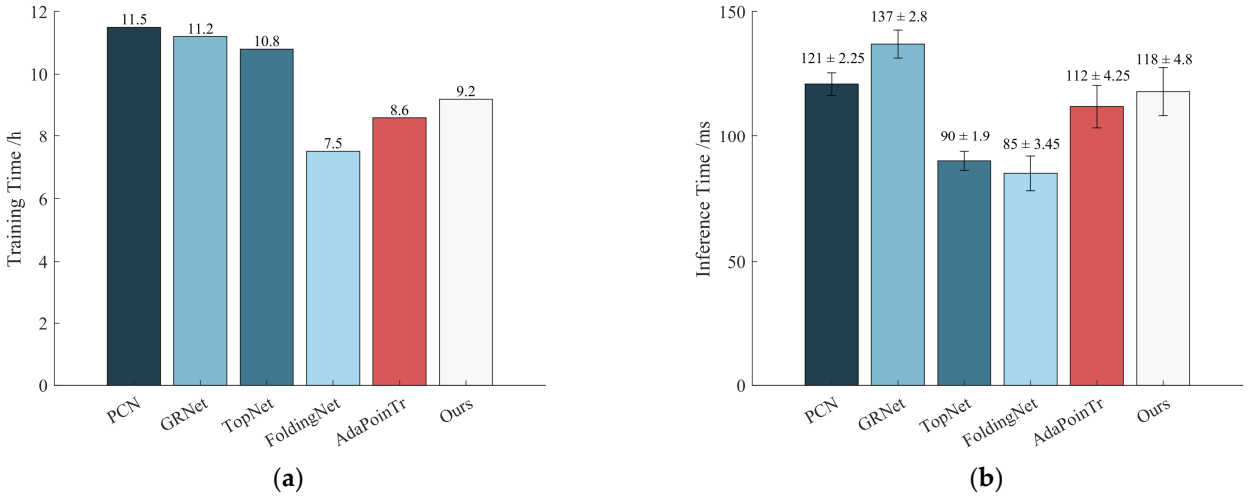


Figure 18. Speed Comparison of Different Models. (**a**) represents the training processing time, while (**b**) represents the inference time.

We also compared the training and inference times of different models. Figure 18a presents the training time required for each model over 600 epochs, while Figure 18b illustrates the average inference time and variance when processing 100 tree point clouds. Among the models, PCN exhibited the longest training time at 11.5 h, whereas FoldingNet required only 7.5 h, a difference primarily attributed to FoldingNet's simpler architecture. Our proposed SK-TreePCN model had a training duration second only to FoldingNet and AdaPoinTr. However, all models completed training within 12 h when utilizing two 3080Ti GPUs. In deep learning applications, inference speed is often a primary concern. Notably, the inference time for all models remained below 150 milliseconds, indicating that model complexity has a negligible impact on inference efficiency.

In conclusion, both qualitative and quantitative results indicate that the SK-TreePCN method delivers strong performance across the VLS and field datasets, demonstrating its effectiveness in addressing tree point cloud completion.

4.2. Model Design Analysis

The baseline model of SK-TreePCN first used FPS to identify the proxy point and employed the lightweight DGCNN model to extract local features around the proxy point. Then, it used the transformer encoder–decoder structure to infer the missing proxy points. Finally, MLP and Rebuild Head enhanced the point cloud resolution from coarse to fine. Based on the baseline model, we developed the SK→PS module to substitute FPS, enhance the baseline model, and incorporate SK→TA as a topological regulator in the final Rebuild Head process. The results in Table 8 confirm our expectation that both the SK→PS and SK→TA models contributed comparable improvements to the performance of the baseline.

In addition, we recognized multiple difficulties in point cloud completion associated with the baseline model, which highlighted SK-TreePCN's ability to tackle these challenges. Figure 15 highlights that AdaPoinTr presented the following issues:

- The completed tree point cloud canopy showed dispersion and insufficient structural clarity, with Tree1 and Tree3 being particularly noticeable.
- The enlarged point cloud in Tree2, highlighted by the red box, showed incomplete completion outcomes.
- Underfilling was evident in regions with significant occlusion, especially in the middle of Tree3.

These issues occurred because AdaPoinTr primarily addressed missing points within specified structures but could not infer connections. SK-TreePCN effectively resolved the problems of point cloud dispersion, discontinuity, and underfilling in the results. It employed the topological structure of the tree to facilitate completion, creating connections between disparate point cloud segments. The completed point cloud exhibited higher visual coherence due to connection inference and information transmission, showcasing improved details in the tree canopy.

In conclusion, the ablation experiments and detailed comparisons demonstrated that the SK→PS and SK→TA modules substantially enhanced the model's capacity to infer missing regions of the tree point cloud, particularly at the connections of branch segments.

4.3. Sensitivity of Point Density and Distribution to the Completion Effect

Section 2.2.2 described the simulation of the TLS process utilizing VLS, highlighting that scanning angle resolution and the number of scanning stations were critical factors affecting the distribution and density of the point cloud. We performed a sensitivity analysis on the model by varying the scanning resolution and the number of scanning stations. In the initial phase, we established five-scanning stations using the five-stations scanning method and determined the dynamic scanning resolution, with Azimuth and Zenith resolutions set sequentially to {0.8°, 0.7°, 0.6°, 0.5°, 0.4°, 0.3°, 0.2°, 0.1°, 0.05°, 0.025°}. The quantitative results shown in Figure 19 indicated that as the scanning resolution decreased (resulting in increased scanning density), the overall CD_2 decreased from 0.241 to 0.158, while the overall $F1$ score increased from 0.743 to 0.836. The trends in $F1$ and CD_2 demonstrated that as the scanning resolution decreased, the scanning precision improved, the point cloud density increased, and the completion effect became progressively more significant. When the resolution value reached 0.3°, the completion precision stabilized, and the variation significantly decreased.

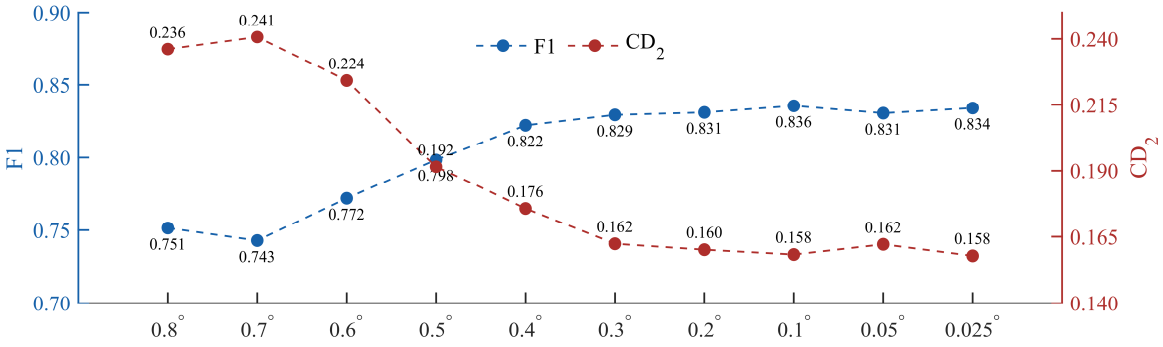


Figure 19. $F1$ and CD_2 results for different scanning resolutions. The values of the Azimuth and Zenith resolutions are set along the X-axis.

Subsequently, we fixed the scanning resolution (Azimuth and Zenith) to 0.3°. We varied the number of scanning stations in a plot to produce point clouds with varying degrees of absence and densities. Figure 20 illustrates five different scanning station setups and the quantitative point cloud completion evaluation results for five selected Eucalyptus species. Figure 20b indicates that the $F1$ score approximately reached a saturation point

after the number of stations was set to 5 and exhibited only minor variations when the number was increased to 6, 7, and 8. Similarly, for CD_1, once the number of stations was established at 5, further increasing the number of stations did not cause a significant reduction in CD_1.

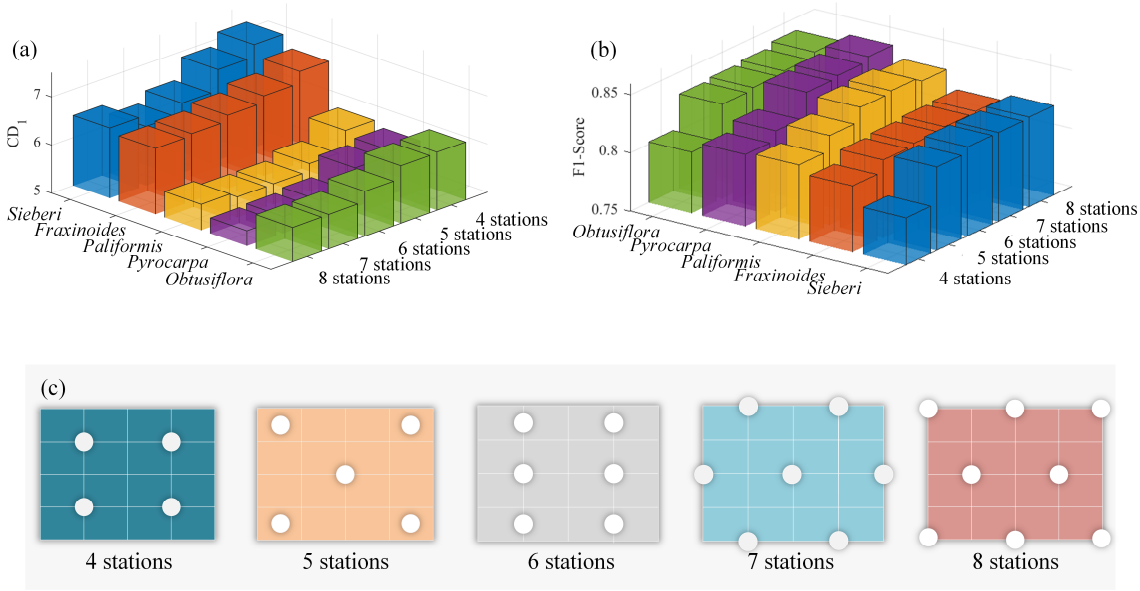


Figure 20. CD_1 and $F1$ across different scanning station numbers. (**a**) CD_1 and (**b**) $F1$ for point cloud completion. (**c**) Illustration of the setup for different scanning stations.

In conclusion, the experimental results underscored two key findings: (1) Adjusting the scanning resolution indicated that the performance of point cloud completion progressively improved with the increase in point cloud density and eventually stabilized. (2) Varying scan station number configurations demonstrated that the degree of point cloud incompleteness influenced completion accuracy. This was due to the difficulty in constructing an accurate tree topology when a substantial portion of the data were missing.

4.4. Model Transferability to Field Data

This study employed two field plots as research subjects to assess the model's transferability. The field data were analyzed quantitatively (Tables 6 and 7) and qualitatively (Figure 14). We reconstructed the point cloud and extracted QSM, subsequently comparing the results with the observed data. Figure 14 illustrates that the point cloud completion results demonstrated visual consistency with the tree structure across various scenarios. The branches of the completed TLS point cloud of trees exhibited detailed representation, and the canopy details were distinctly rendered. This indicated significant potential to address issues in practical applications, including 3D reconstructions with low-density point clouds for trees with discontinuous branches and complex canopies.

As shown in Figure 17, the fitting results before and after point cloud completion indicated minimal fluctuations in tree height and crown width. In contrast, the DBH was improved. This phenomenon was attributed to the regular plant arrangement and even spacing in the Eucalyptus plantation. These characteristics allowed the laser pulses emitted by the LiDAR scanner to effectively hit the tree tops, branches, and trunks. At this stage, the missing points in the tree point cloud were generally caused by occlusion from leaves and branches. Furthermore, based on the point cloud completion results shown in Figure 14, SK-TreePCN accurately filled the interior of the tree (i.e., gaps between the trunk and branches, as shown in Figure 14a–c) without generating external extensions (e.g.,

elongation at the tree top or trunk base). This finding demonstrated that SK-TreePCN could accurately identify and complete the missing parts of the tree point cloud.

On the other hand, the structural parameters in Figure 17 indicate that tree height, crown width, and DBH after point cloud completion were closely aligned with the measured values. The identified discrepancies could not be solely attributed to the point cloud completion process. Factors that influenced tree height and DBH estimation went beyond manual readout errors and equipment inaccuracies. Specifically, (1) the ground filtering algorithm might have altered the tree height values, and (2) the reconstruction algorithm might have impacted the DBH estimate. Since AdTree was based on a generalized cylinder, if the tree trunk surface was uneven or highly curved, the estimated DBH could have been lower than the measured value [15].

In conclusion, our model effectively addressed the issue of missing point cloud data in real-world scenarios, as demonstrated through visual and structural parameter validation. Additionally, the simulated and field data collected from various scanning devices further confirmed the compatibility of tree point cloud completion across diverse scanning devices and configurations.

4.5. Limitations and Outlook

Currently, tree point cloud completion approaches remain in their early stages. While this study has achieved considerable advancements, several issues require further investigation.

Tree species in nature exhibited a high degree of diversity, with each species adapting to its unique environment by developing distinct branch and foliage structures. In this study, we focused on leafless trees for two primary reasons. Firstly, leaves were typically distributed on smaller branches, which were difficult to capture during LiDAR scanning, leading to gaps between the leaf and trunk point clouds. Secondly, the shape and distribution of leaves differed significantly from those of branches. In most species, branches could be characterized with generalized cylinders, whereas leaves were typically flat with thin structures or complex curved surfaces, densely distributed within the canopy. The discontinuity of the point cloud, along with the complex shapes and diverse distribution, negatively impacted the robustness and generalization of the completion models.

Moreover, due to the complexity and uncertainty of real-world environments, VLS simulations had difficulty fully replicating all the conditions encountered during TLS. As a result, the simulated point cloud data exhibited discrepancies in distribution, density, and incompleteness characteristics compared to real TLS data. These differences negatively impacted the generalization ability and transferability of the completion models.

In future research, advancements in artificial intelligence and computer graphics are expected to bridge the gap between virtual and real-world data, effectively addressing these challenges. Additionally, incorporating specific evaluation metrics, such as tree topology structures, will facilitate a systematic assessment of point cloud incompleteness and completion methods, providing more accurate data support for forestry management and related applications.

5. Conclusions

This study proposed a point cloud completion scheme that progressed from virtual to real forest scenes. To address the challenges of complex domain adaptation and the strong heterogeneity of tree structures, we created forest scenes with clear spatial structures. These scenes were combined with the advanced 3D radiative transfer model LESS to simulate TLS and generate individual tree point clouds. The simulated data objectively reflected occlusions and limitations in scanning angles during the scanning process. Using these data as training inputs enhanced the point cloud completion model's capability to handle real-

world data. We also proposed a point cloud completion model named SK-TreePCN based on tree morphology to cope with the complexity of tree structures. Simulated and field data were also used to evaluate the method's contribution. The simulated data showed that this method had state-of-the-art (SOTA) performance compared with mainstream completion methods, and the completed tree structure details were more accurate and exhibited higher fidelity. At the same time, the field data further validated the transferability of our method and demonstrated the effectiveness of SK-TreePCN in enhancing the accuracy of extracting tree structural parameters. This study is expected to reduce data acquisition costs for future research on refined forest management and monitoring. Furthermore, it promises to improve the quality and usability of low-density or incomplete point cloud data.

Author Contributions: Conceptualization, H.X. and Y.H.; Methodology, H.X. and Y.H.; Software, H.X.; Validation, X.Z., Q.M. and B.L.; Formal Analysis, H.L.; Investigation, H.X. and X.Z.; Resources, H.L. and X.N.; Data Curation, H.X., B.L. and Y.H.; Writing—Original Draft Preparation, H.X.; Writing—Review and Editing, H.L. and Y.H.; Visualization, H.X.; Supervision, H.L.; Project Administration, Y.H.; Funding Acquisition, H.L. and X.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Xiong'an New Area Science and Technology Innovation Special Project of Ministry of Science and Technology of China (Grant No. 2023XAGG0065), the National Natural Science Foundation of China (Grant No. 42001376) and the National Key R & D Program of China (Grant No. 2023YFC3304000).

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to the permissible boundaries.

Acknowledgments: We are very grateful to all the reviewers, institutions, and researchers for their help and advice, which helped improve our work.

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

References

1. Bohn, F.J.; Huth, A. The Importance of Forest Structure to Biodiversity–Productivity Relationships. *R. Soc. Open Sci.* **2017**, *4*, 160521. [CrossRef]
2. Hui, G.; Zhang, G.; Zhao, Z.; Yang, A. Methods of Forest Structure Research: A Review. *Curr. For. Rep.* **2019**, *5*, 142–154. [CrossRef]
3. McElhinny, C.; Gibbons, P.; Brack, C.; Bauhus, J. Forest and Woodland Stand Structural Complexity: Its Definition and Measurement. *For. Ecol. Manag.* **2005**, *218*, 1–24. [CrossRef]
4. Disney, M.I.; Vicari, M.B.; Burt, A.; Calders, K.; Lewis, S.L.; Raumonen, P.; Wilkes, P. Weighing Trees with Lasers: Advances, Challenges and Opportunities. *Interface Focus* **2018**, *8*, 20170048. [CrossRef]
5. 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]
6. Lucas, R.; Lee, A.; Armston, J.; Breyer, J.; Bunting, P.; Carreiras, J. Advances in Forest Characterisation, Mapping and Monitoring through Integration of LiDAR and Other Remote Sensing Datasets. In Proceedings of the SilviLaser 2008, Edinburgh, UK, 17–19 September 2008.
7. Camarretta, N. Monitoring Forest Structure to Guide Adaptive Management of Forest Restoration: A Review of Remote Sensing Approaches. *New For.* **2020**, *51*, 573–596. [CrossRef]
8. Calders, K. Terrestrial Laser Scanning in Forest Ecology: Expanding the Horizon. *Remote Sens. Environ.* **2020**, *251*, 112102. [CrossRef]
9. Raumonen, P. Fast Automatic Precision Tree Models from Terrestrial Laser Scanner Data. *Remote Sens.* **2013**, *5*, 491–520. [CrossRef]
10. Du, S.; Lindenbergh, R.; Ledoux, H.; Stoter, J.; Nan, L. AdTree: Accurate, Detailed, and Automatic Modelling of Laser-Scanned Trees. *Remote Sens.* **2019**, *11*, 2074. [CrossRef]
11. Soma, M.; Pimont, F.; Allard, D.; Fournier, R.; Dupuy, J.-L. Mitigating Occlusion Effects in Leaf Area Density Estimates from Terrestrial LiDAR through a Specific Kriging Method. *Remote Sens. Environ.* **2020**, *245*, 111836. [CrossRef]
12. Zhao, X.; Qi, J.; Xu, H.; Yu, Z.; Yuan, L.; Chen, Y.; Huang, H. Evaluating the Potential of Airborne Hyperspectral LiDAR for Assessing Forest Insects and Diseases with 3D Radiative Transfer Modeling. *Remote Sens. Environ.* **2023**, *297*, 113759. [CrossRef]

13. Åkerblom, M. Automatic Tree Species Recognition with Quantitative Structure Models. *Remote Sens. Environ.* **2017**, *191*, 1–12. [CrossRef]
14. Cao, W.; Wu, J.; Shi, Y.; Chen, D. Restoration of Individual Tree Missing Point Cloud Based on Local Features of Point Cloud. *Remote Sens.* **2022**, *14*, 1346. [CrossRef]
15. Wang, W.; Li, Y.; Huang, H.; Hong, L.; Du, S.; Xie, L.; Li, X.; Guo, R.; Tang, S. Branching the Limits: Robust 3D Tree Reconstruction from Incomplete Laser Point Clouds. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *125*, 103557. [CrossRef]
16. Abegg, M.; Bösch, R.; Kükenbrink, D.; Morsdorf, F. Tree Volume Estimation with Terrestrial Laser Scanning—Testing for Bias in a 3D Virtual Environment. *Agric. For. Meteorol.* **2023**, *331*, 109348. [CrossRef]
17. Wang, Z.; Zhang, L.; Fang, T.; Mathiopoulos, P.T.; Qu, H.; Chen, D.; Wang, Y. A Structure-Aware Global Optimization Method for Reconstructing 3-D Tree Models from Terrestrial Laser Scanning Data. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 5653–5669. [CrossRef]
18. Hu, S.; Li, Z.; Zhang, Z.; He, D.; Wimmer, M. Efficient Tree Modeling from Airborne LiDAR Point Clouds. *Comput. Graph.* **2017**, *67*, 1–13. [CrossRef]
19. Fu, L.; Liu, J.; Zhou, J.; Zhang, M.; Lin, Y. Tree Skeletonization for Raw Point Cloud Exploiting Cylindrical Shape Prior. *IEEE Access* **2020**, *8*, 27327–27341. [CrossRef]
20. Mei, J.; Zhang, L.; Wu, S.; Wang, Z.; Zhang, L. 3D Tree Modeling from Incomplete Point Clouds via Optimization and L_1-MST. *Int. J. Geogr. Inf. Sci.* **2017**, *31*, 999–1021. [CrossRef]
21. 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 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; IEEE: Honolulu, HI, USA, 2017; pp. 652–660.
22. 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*, 5105–5114. [CrossRef]
23. Chang, Y.; Jung, C.; Xu, Y. FinerPCN: High Fidelity Point Cloud Completion Network Using Pointwise Convolution. *Neurocomputing* **2021**, *460*, 266–276. [CrossRef]
24. Yuan, W.; Khot, T.; Held, D.; Mertz, C.; Hebert, M. PCN: Point Completion Network. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; IEEE: Verona, Italy, 2018; pp. 728–737.
25. Nie, Y.; Lin, Y.; Han, X.; Guo, S.; Chang, J.; Cui, S.; Zhang, J.J. Skeleton-Bridged Point Completion: From Global Inference to Local Adjustment. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 16119–16130. [CrossRef]
26. Xu, D.; Chen, G.; Jing, W. A Single-Tree Point Cloud Completion Approach of Feature Fusion for Agricultural Robots. *Electronics* **2023**, *12*, 1296. [CrossRef]
27. Zhu, L.; Wang, B.; Tian, G.; Wang, W.; Li, C. Towards Point Cloud Completion: Point Rank Sampling and Cross-Cascade Graph Cnn. *Neurocomputing* **2021**, *461*, 1–16. [CrossRef]
28. Wu, H.; Miao, Y. *LRA-Net: Local Region Attention Network for 3D Point Cloud Completion*; SPIE: Bellingham, WA, USA, 2021; Volume 11605, pp. 357–367.
29. Huang, Z.; Yu, Y.; Xu, J.; Ni, F.; Le, X. PF-Net: Point Fractal Network for 3D Point Cloud Completion. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; IEEE: Seattle, WA, USA, 2020; pp. 7659–7667.
30. Li, S.; Gao, P.; Tan, X.; Wei, M. ProxyFormer: Proxy Alignment Assisted Point Cloud Completion with Missing Part Sensitive Transformer. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; IEEE: Vancouver, BC, Canada, 2023; pp. 9466–9475.
31. Yu, X.; Rao, Y.; Wang, Z.; Liu, Z.; Lu, J.; Zhou, J. PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 12498–12507.
32. Zhou, H.; Cao, Y.; Chu, W.; Zhu, J.; Lu, T.; Tai, Y.; Wang, C. SeedFormer: Patch Seeds Based Point Cloud Completion with Upsample Transformer. In Proceedings of the Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; pp. 416–432.
33. Yu, X.; Rao, Y.; Wang, Z.; Lu, J.; Zhou, J. AdaPoinTr: Diverse Point Cloud Completion with Adaptive Geometry-Aware Transformers. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 14114–14130. [CrossRef] [PubMed]
34. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010.
35. Zhao, H.; Jiang, L.; Jia, J.; Torr, P.H.; Koltun, V. Point Transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 16259–16268.
36. Tchapmi, L.P.; Kosaraju, V.; Rezatofighi, H.; Reid, I.; Savarese, S. TopNet: Structural Point Cloud Decoder. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; IEEE: Long Beach, CA, USA, 2019; pp. 383–392.

37. Yang, Y.; Feng, C.; Shen, Y.; Tian, D. FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; IEEE: Salt Lake City, UT, USA, 2018; pp. 206–215.
38. Qi, J.; Xie, D.; Jiang, J.; Huang, H. 3D Radiative Transfer Modeling of Structurally Complex Forest Canopies through a Lightweight Boundary-Based Description of Leaf Clusters. *Remote Sens. Environ.* **2022**, *283*, 113301. [CrossRef]
39. He, S.; Qi, J.; Wang, D.; Yan, K.; Huang, H. Estimation of Canopy Photon Recollision Probability from Airborne Laser Scanning. *Remote Sens. Environ.* **2024**, *311*, 114264. [CrossRef]
40. Gao, G.; Qi, J.; Lin, S.; Hu, R.; Huang, H. Estimating Plant Area Density of Individual Trees from Discrete Airborne Laser Scanning Data Using Intensity Information and Path Length Distribution. *Int. J. Appl. Earth Obs. Geoinf.* **2023**, *118*, 103281. [CrossRef]
41. Winiwarter, L.; Esmorís Pena, A.M.; Weiser, H.; Anders, K.; Martínez Sánchez, J.; Searle, M.; Höfle, B. Virtual Laser Scanning with HELIOS++: A Novel Take on Ray Tracing-Based Simulation of Topographic Full-Waveform 3D Laser Scanning. *Remote Sens. Environ.* **2022**, *269*, 112772. [CrossRef]
42. North, P.R. Three-Dimensional Forest Light Interaction Model Using a Monte Carlo Method. *IEEE Trans. Geosci. Remote Sens.* **1996**, *34*, 946–956. [CrossRef]
43. North, P.; Rosette, J.; Suárez, J.; Los, S. A Monte Carlo Radiative Transfer Model of Satellite Waveform LiDAR. *Int. J. Remote Sens.* **2010**, *31*, 1343–1358. [CrossRef]
44. Gastellu-Etchegorry, J.-P.; Yin, T.; Lauret, N.; Cajgfinger, T.; Gregoire, T.; Grau, E.; Feret, J.-B.; Lopes, M.; Guilleux, J.; Dedieu, G. Discrete Anisotropic Radiative Transfer (DART 5) for Modeling Airborne and Satellite Spectroradiometer and LIDAR Acquisitions of Natural and Urban Landscapes. *Remote Sens.* **2015**, *7*, 1667–1701. [CrossRef]
45. Bechtold, S.; Höfle, B. Helios: A Multi-Purpose Lidar Simulation Framework for Research, Planning and Training of Laser Scanning Operations with Airborne, Ground-Based Mobile and Stationary Platforms. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *3*, 161–168. [CrossRef]
46. Qi, J.; Xie, D.; Yin, T.; Yan, G.; Gastellu-Etchegorry, J.-P.; Li, L.; Zhang, W.; Mu, X.; Norford, L.K. LESS: LargE-Scale Remote Sensing Data and Image Simulation Framework over Heterogeneous 3D Scenes. *Remote Sens. Environ.* **2019**, *221*, 695–706. [CrossRef]
47. Huang, G.; Zhao, Q. The History, Status Quo, Ecological Problems and Countermeasures of *Eucalyptus* Plantations in Guangxi. *Acta Ecol. Sin.* **2014**, *34*, 5142–5152. [CrossRef]
48. Zhang, Y.; Wang, X. Geographical Spatial Distribution and Productivity Dynamic Change of Eucalyptus Plantations in China. *Sci. Rep.* **2021**, *11*, 19764. [CrossRef] [PubMed]
49. 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]
50. Tao, S.; Wu, F.; Guo, Q.; Wang, Y.; Li, W.; Xue, B.; Hu, X.; Li, P.; Tian, D.; Li, C. Segmenting Tree Crowns from Terrestrial and Mobile LiDAR Data by Exploring Ecological Theories. *ISPRS J. Photogramm. Remote Sens.* **2015**, *110*, 66–76. [CrossRef]
51. Pukkala, T.; Lähde, E.; Laiho, O. Which Trees Should Be Removed in Thinning Treatments? *For. Ecosyst.* **2015**, *2*, 32. [CrossRef]
52. Zhou, H.; Shenoy, N.; Nicholls, W. Efficient Minimum Spanning Tree Construction without Delaunay Triangulation. In Proceedings of the ASP-DAC 2001—Asia and South Pacific Design Automation Conference 2001, Yokohama, Japan, 2 February 2001; pp. 192–197.
53. Dijkstra, E.W. A Note on Two Problems in Connexion with Graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 287–290.
54. Wu, S.-T.; Marquez, M.R.G. A Non-Self-Intersection Douglas-Peucker Algorithm. In Proceedings of the 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003), Sao Carlos, Brazil, 12–15 October 2003; IEEE: Sao Carlos, Brazil, 2003; pp. 60–66.
55. Song, Z.; Wei, H.; Jia, C.; Xia, Y.; Li, X.; Zhang, C. VP-Net: Voxels as Points for 3-D Object Detection. *IEEE Trans. Geosci. Remote Sens.* **2023**, *61*, 5701912. [CrossRef]
56. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic Graph Cnn for Learning on Point Clouds. *ACM Trans. Graph. (TOG)* **2019**, *38*, 1–12. [CrossRef]
57. Xie, H.; Yao, H.; Zhou, S.; Mao, J.; Zhang, S.; Sun, W. *Grnet: Gridding Residual Network for Dense Point Cloud Completion*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 365–381.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.