INDIVIDUAL TREE SEGMENTATION FROM BLS DATA BASED ON GRAPH AUTOENCODER

Reda Fekry [1], Wei Yao [2] *, Abubakar Sani-Mohammed[2], Doha Amr[2]

[1]Department of Geomatics, Faculty of Engineering at Shoubra, Benha University, Cairo, Egypt.
[2]Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong.

KEY WORDS: LiDAR, Individual Tree Segmentation, Backpack Laser Scanning, Graph Neural Network, Graph Autoencoder.

ABSTRACT:

In the last two decades, Light detection and ranging (LiDAR) has been widely employed in forestry applications. Individual tree segmentation is essential to forest management because it is a prerequisite to tree reconstruction and biomass estimation. This paper introduces a general framework to extract individual trees from the LiDAR point cloud based on a graph link prediction problem. First, an undirected graph is generated from the point cloud based on K-nearest neighbors (KNN). Then, this graph is used to train a convolutional autoencoder that extracts the node embeddings to reconstruct the graph. Finally, the individual trees are defined by the separate sets of connected nodes of the reconstructed graph. A key advantage of the proposed method is that no further knowledge about tree or forest structure is required. Seven sample plots from a plantation forest with poplar and dawn redwood species have been employed in the experiments. Though the precision of the experimental results is up to 95 % for poplar species and 92 % for dawn redwood trees, the method still requires more investigations on natural forest types with mixed tree species.

1. INTRODUCTION

Accurate forest inventory is decisive to forest management. Advanced remote sensing techniques particularly LiDAR have proven potential in forest mapping. The variety of LiDAR systems from airborne to personal laser scanners enables comprehensive forest interpretation due to the huge amount of gained information (Fekry et al., 2021). A key process for precise biomass estimation is individual tree segmentation. In general, tree segmentation from LiDAR data is divided into raster-based methods (Xu et al., 2021), point-based methods (Comesaña-Cebral et al., 2021; Yao et al., 2012), and multi-source data fusion methods (Duncanson et al., 2014).

The basic principle of raster-based methods is to locate the tree location based on the local maximum from the canopy height model (CHM). Then, the tree crowns are delineated based on the region's growth. The main limitation of these methods is the information loss due to the projection of a three-dimensional (3D) point cloud and interpolation while creating the CHM. This means that under-story trees are not segmented particularly in a multi-layer forest. Moreover, the segmentation is initiated at predefined markers. Several tree segmentation methods were developed based on CHM generation. For instance, Zhen et al. (2014) developed a marker-controlled approach for tree crown delineation based on region growing with the treetops as seed points. Yang et al. (2020) segmented the individual trees by improving the watershed algorithm results using the 3D information from the point cloud. Xu et al. (2021) proposed a crown morphology-based tree detection approach from unmanned aerial vehicle laser scanning (ULS) data in an urban forest of mixed broadleaf trees.

In contrast, the point-based methods operate directly on the 3D point cloud then, the information loss is mitigated. The point-based methods involve different implementations including converting point clouds to voxels (Wang et al., 2008), clustering such as mean shift (Hui et al., 2021) and density-based spatial clustering of applications with noise (DBSCAN) (Fu et al., 2022), conditional euclidean distance (Shendryk et al., 2016) and graph-based segmentation (Yao et al., 2012). There are numerous methods depending on point-based tree segmentation approaches. Li et al. (2012) introduced a distance-based approach in a conifer forest. Hosoi et al. (2013) generated the 3D tree model in a mixed Japanese plantation based on point cloud voxelization. Moreover, a bottom-to-top method (Lu et al., 2014) was developed for deciduous tree segmentation based on intensity. Dai et al. (2018) used mean shift segmentation to extract individual trees from multi-spectral ALS data. Hui et al. (2021) presented a self-adaptive bandwidth of mean shift algorithm such that the single trees were gradually extracted based on hierarchy mean shift. Fu et al. (2022) proposed single-segmented trees from terrestrial laser scanning (TLS) data using a bottom-up approach based on DBSCAN by first detecting the trunks. Yao et al. (2012) performed tree segmentation by employing the normalized cut of a graph composed of voxels for less computational cost. However promising results of point-based methods, the segmentation results are significantly affected in the case of forests with multiple tree species. In addition, they demand high requirements due to the LiDAR point density.

Moreover, advanced machine learning models such as deep learning have been utilized to detect individual trees from LiDAR data. For example, Jin et al. (2018) utilized Faster R-CNN to detect maize trees from TLS data, while Chen et al. (2021) used PointNet to detect individual tree crowns. Luo et al. (2022) employed a multi-channel representation to detect individual trees from ULS, and Ying et al. (2021) introduced a pointCNN-based approach for tree detection using a CHM from LiDAR to generate rough seed points for extracting the detection samples. However, deep learning-based methods often require converting 3D point clouds to 2D raster images, which may result in the loss of important 3D information. Additionally, annotating the data can be challenging.

* Corresponding author

In summary, the limitations of raster-based methods for tree segmentation include (1) 3D information loss, which can result in the misidentification or omission of understory trees; and (2) segmentation initiation at predefined markers, which can lead to errors in tree location and segmentation. Point-based methods also have drawbacks, including high computational requirements, and sensitivity to noise and outliers. Both raster-based and point-based can face difficulties in situations where tree crowns are complex or overlap with each other, especially in mixed forest environments. Furthermore, existing tree segmentation methods that use deep learning algorithms necessitate significant amounts of labeled training data, which can be both costly and time-consuming to obtain. These approaches may also be affected by variations in the training data and the selection of model architecture and hyperparameters, which can impact their accuracy.

This study presents a new approach for segmenting individual trees from LiDAR data using convolutional graph autoencoder. The proposed method uses raw point cloud data without any additional information about forest structure or predefined markers, which can overcome the limitations of raster-based methods such as 3D information loss, understory misinterpretation, and errors caused by inaccurate tree locations. Unlike deep learning-based point-based methods, the proposed method does not require labeled data, which can overcome the need for data annotation. The proposed method can be broken down into several main steps. Firstly, a LiDAR point cloud is used to create a graph by connecting each node to its K nearest neighbors. Secondly, the generated graph is defined by its adjacency matrix and node features, and is then reconstructed by the autoencoder. During this stage, the autoencoder predicts the existence of an edge between any pair of unconnected nodes. Lastly, a single tree is identified by finding a set of connected nodes from the reconstructed graph. The main contributions of this research can be summarized as:

- The development of a convolutional graph neural network that can extract single trees from LiDAR data.
- The ability to perform individual tree extraction without requiring any additional information about tree parameters or forest structure.

The rest of this paper is structured as follows: Section 2 describes the methodology used in the research. In Section 3, the study area, dataset, and experimental design are discussed. Section 4 presents the research findings and discussion. Finally, Section 5 provides a summary of the conclusions and future outlook of the research.

2. METHODOLOGY

An end-to-end graph neural network is proposed to perform tree segmentation from backpack laser scanning (BLS) data in four stages as shown in Figure 1. First, the data preparation and graph formation. Then, the graph is fed to a convolutional graph autoencoder which reconstructs the graph to make predictions. Finally, the reconstructed graph analysis to extract the single trees.

2.1 Data Preparation

In order to develop an end-to-end neural network, point cloud pre-processing has been integrated with the framework. The pre-processing includes outlier removal, ground-filtering, downsampling, and graph formation. In the subsequent subsections, the detailed steps of these processes are outlined.

2.1.1 Outlier Removal: It is an important preprocessing step in LiDAR data processing. Outliers can be caused by a variety of factors, such as sensor noise, environmental conditions, and measurement errors, and can lead to errors in downstream processing tasks such as object detection and 3D reconstruction. As a result, it is necessary to remove these outliers from the point cloud before proceeding with any further processing. To achieve this, the distances between each point and its neighboring points within a certain radius r are calculated. The average distance μ_d and standard deviation σ_d of these distances are then computed. The maximum distance $D_{max} = \mu_d + n \times \sigma_d$ is determined as the sum of μ_d and a multiple of σ_d, where the multiple n is a predefined constant. Points that have $\mu_d > D_{max}$ are considered outliers and are marked for removal from the point cloud.

2.1.2 Ground Filtering: A point cloud $\mathbf{PC} \in \mathbb{R}^3$ was classified into the ground and non-ground points based on the cloth simulation filter (CSF) developed by Zhang et al. (2016). First, **PC** was inverted and covered with a rigid cloth. Then, the interactions between the nodes of the cloth and their counterparts from LiDAR points were analyzed. Accordingly, the positions of the cloth nodes were identified to form the ground surface. Finally, the formed surface was compared with the original LiDAR data to extract the ground points. The non-ground points were most important because they served as input for the various phases of this study.

2.1.3 Point Cloud Downsampling: A 3D grid of a point cloud $\mathbf{PC} \in \mathbb{R}^3$ is generated by subdividing **PC** into small cubes with side lengths d_x, d_y, and d_z in X, Y, and Z directions, respectively. A single cube (voxel) contains m points whose centroid is represented by Equation(1). Voxel downsampling was performed to only reduce the point density for computational reasons, and it had no effect on the point distribution or object shapes. It is worth noting that the voxelized point cloud $\bar{\mathbf{PC}} = \bar{x}_i \in \mathbb{R}^f, i = 1, 2, 3, ..., \bar{N}$ will be deployed in further steps of the proposed method.

$$\bar{x} = \frac{\sum_{i=1}^{m} x_i}{m} \tag{1}$$

2.1.4 Graph Formation A graph $\boldsymbol{G}$ is defined by a set of edges (links) $\mathscr{E}$ which connect a set of vertices (nodes) $\mathscr{V}$. Graph $\boldsymbol{G}$ is represented by its adjacency matrix $\boldsymbol{A}$ and the nodes feature matrix $\boldsymbol{X}$. According to Equation (1), a voxelized point cloud $\bar{\mathbf{PC}}$ is used to generate graph $\boldsymbol{G}$ with $f = 3$ denotes the XYZ-dimensionality or more when additional features (e.g., RGB or normal features) are present. To define the graph edges, the nearest neighbor search is employed to identify the K neighbors to every node based on 2D distances. Then, every node of $\boldsymbol{G}$ is connected to its K neighbors, therefore, the adjacency matrix $\boldsymbol{A} \in \{1, 0\}^{\bar{N} \times \bar{N}}$ is formed, where 1 denotes a positive (present) edge while 0 denotes a negative (absent) edge. Moreover, the nodes feature matrix $\boldsymbol{X} \in \mathbb{R}^{\bar{N} \times f}$ thus a single row in $\boldsymbol{X}$ involves the positional coordinates of a

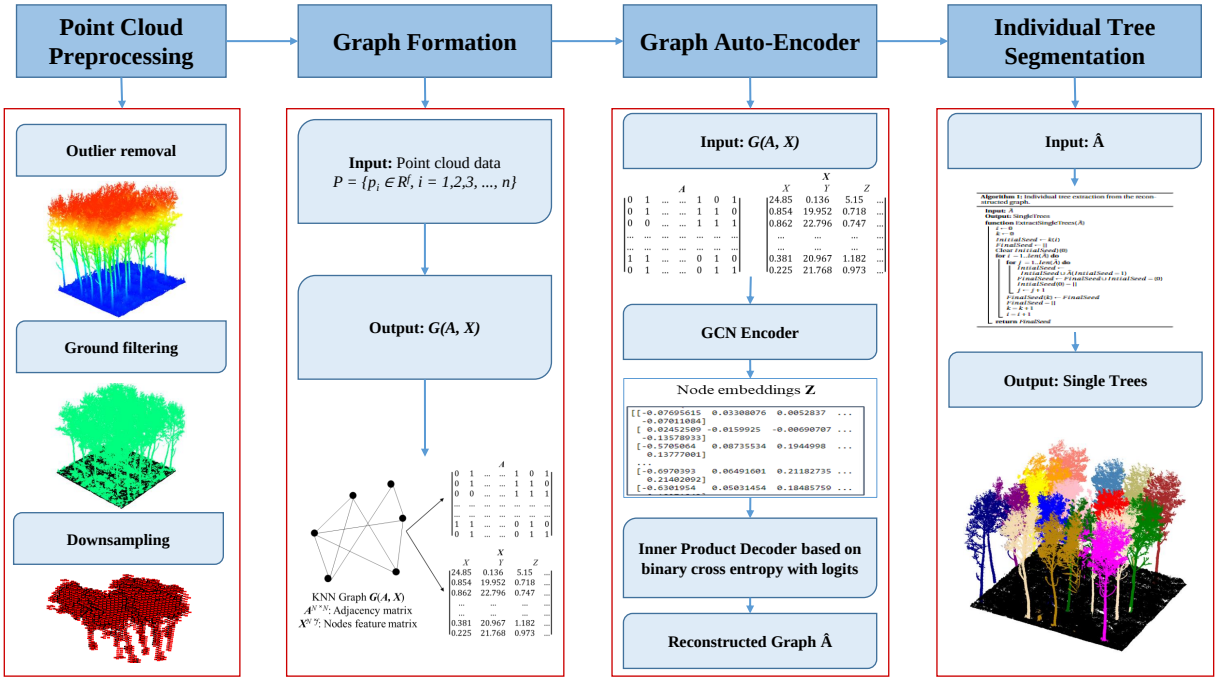

Figure 1. Research framework.

single node in $\boldsymbol{G}$.

2.2 Model Architecture

The proposed graph neural network consists mainly of a convolution autoencoder. This autoencoder encompasses two parts: encoder and decoder. A convolution encoder is a set of convolution layers that can learn low-dimensional representations of the nodes in the graph. The autoencoder consists of an encoder that maps the adjacency matrix to a low-dimensional latent space, and a decoder that maps the latent space back to the adjacency matrix. A detailed explanation of the model architecture is presented in the next subsections.

2.2.1 Graph Autoencoder: The principal question of a link prediction problem: if there are any two entities, should they be connected? The structural link prediction problem involves estimating the probability of the presence of a missing or absent connection between two nodes in a partially observed graph (Wang et al., 2014). The likelihood of an unobserved edge in a graph can be computed using various approaches, such as statistical models, machine learning algorithms, or graph-based methods. Graph autoencoders are a powerful tool for predicting missing edges in graphs because they can capture complex patterns and dependencies between nodes, and can handle graphs of arbitrary size and structure (Tran, 2018).

The structure of a graph $\boldsymbol{G}$ is defined by its adjacency matrix $\boldsymbol{A}$ and the features of its nodes $\boldsymbol{X}$. The goal of the autoencoder model $h(A, X)$ is to learn a set of low-dimensional latent variables $\boldsymbol{Z} \in R^{\tilde{N} \times D}$ that represent the nodes, such that the model can produce an output $\hat{\boldsymbol{A}}$ that closely approximates the original adjacency matrix $\boldsymbol{A}$. By minimizing the error between $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$, the autoencoder preserves the overall structure of the graph.

The autoencoder architecture involves transforming the adjacency vector a_i of the i^{th} node in graph $\boldsymbol{A}$ using non-linear transformations. The autoencoder architecture consists of two parts: an encoder $g(a_i)$ that maps a_i from the original space $\tilde{N}$ to a low-dimensional latent space R^D, and a decoder $f(z_i)$ that maps the latent variables z_i back to the original space $\tilde{N}$. The resulting autoencoder model produces an approximate reconstruction output $\widehat{a_i}$ that closely approximates the original adjacency vector a_i for the i^{th} node (Kipf and Welling, 2016). The computation of the hidden representations for the encoder and decoder parts is performed in the following manner:

$$Z = g(a_i) = GCN(\boldsymbol{X}, \boldsymbol{A}) = \boldsymbol{A}ReLU\big(\boldsymbol{H}(l)\boldsymbol{W}(l)\big), \tag{2}$$

where $\boldsymbol{H}(l)$ and $\boldsymbol{W}(l)$ represent the output of a convolution layer l and the weight matrix, respectively. Given the low-dimensional representations of nodes i and j, denoted by z_i and z_j respectively, the probability of an edge between nodes i and j is given by:

$$\hat{\boldsymbol{A}}_{i,j} = f(z_i) = \sigma(\boldsymbol{z}_i^T \boldsymbol{z}_j), \tag{3}$$

where σ is the sigmoid activation function, which maps the output to a probability value between 0 and 1.

2.2.2 Loss Function: In link prediction for graphs, the goal is to predict the presence or absence of edges between pairs of nodes in a graph. Because link prediction of graphs is a binary classification problem, cross-entropy with logits is commonly utilized. In addition, logits provide a stable numerical representation of the model's output. Also, cross-entropy with logits can handle imbalanced datasets where the number of edges is much smaller than the number of possible edges. Moreover, it is a differentiable function, which means that it can be used to update the model parameters through backpropagation during training. This allows the GCN to learn the features that

are most relevant for link prediction. The cross-entropy with logits loss function is defined as:

$$L = -\frac{1}{N_e}\sum_{i=1}^{N_e}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right], \quad (4)$$

where N_e is the total number of edges, y_i is the true label for the i^{th} edge (either 0 or 1), p_i is the predicted probability of the i^{th} edge being present, and log represents the natural logarithm. The p_i values are obtained from the output of the model before the sigmoid function is applied, also known as the logits. The sigmoid function is then applied to the logits to obtain the predicted probabilities $p_i = \sigma(z_i) = \frac{1}{1+e^{-z_i}}$. The loss function can be broken down into two terms: the first term penalizes the model for predicting a probability that is too small when the true label y_i is 1, and the second term penalizes the model for predicting a probability that is too large when the true label y_i is 0. The overall loss L is the average of the per-edge losses. The goal of training the model is to minimize this loss by optimizing the model parameters. By minimizing the loss, the model learns to predict the existence of edges in the graph more accurately.

The loss function you provided is a binary cross-entropy loss commonly used for training binary classification models such as link prediction models. In the context of a link prediction model, the encoder and decoder are responsible for learning representations of nodes and relationships between them, respectively.

During training, the encoder takes in a pair of nodes and produces their corresponding embeddings. The decoder then takes these embeddings and produces a probability score indicating the likelihood of a relationship existing between the two nodes. The loss function is used to measure the difference between the predicted probability score and the true label (i.e., whether a relationship actually exists between the two nodes).

The loss function is computed for each training example (i.e., each pair of nodes) and averaged over all examples to obtain a scalar value representing the overall loss for that iteration of training. This loss value is then used to update the parameters of the encoder and decoder using backpropagation.

By minimizing the loss function during training, the encoder and decoder are encouraged to learn representations of nodes and relationships that accurately predict whether a relationship exists between two nodes. In other words, the loss function serves as a measure of how well the model is able to perform the task of link prediction, and guides the learning process towards better performance.

2.3 Single Tree Segmentation

After the graph reconstruction using the graph autoencoder, a single tree is identified as a set of connected nodes in the reconstructed graph. Therefore, the individual trees are extracted from the reconstructed adjacency matrix $\hat{A}$ of the graph $\boldsymbol{G}$ based on Algorithm 1.

The algorithm takes an adjacency matrix $\hat{A}$ as input and returns a list of lists (i.e., trees), where each inner list (i.e., single tree) contains the indices of all nodes that are connected to each other. The algorithm starts by initializing an empty set for visited nodes and an empty list for individual trees. It then defines a recursive *DFS* function that visits all unvisited neighbors of a given node and adds them to the same tree. The visited set is used to keep track of which nodes have already been visited. The algorithm then iterates over all nodes in the graph, and for each unvisited node, it initializes a new tree and calls the *DFS* function to add all connected nodes to the same tree. The resulting tree is then added to the list of individual trees. Finally, the algorithm returns the list of individual trees.

Algorithm 1: Individual tree extraction from the reconstructed graph.

Input: Reconstructed adjacency matrix $\hat{A}$
Output: List of individual trees
function
  group$_c onnected_n odes()$
$visited \leftarrow \emptyset$; $Extracted\_trees \leftarrow []$;
function `dfs`$(node, current\_tree)$**:**
  $visited$.add$(node)$
  $current\_tree$.append$(node)$
  for $neighbor \leftarrow 0$ **to** $len(\hat{A}) - 1$ **do**
    if $\hat{A}(node, neighbor) = 1$ ***and*** $neighbor \notin visited$ **then**
      `dfs`$(neighbor, current\_tree)$
for $node \leftarrow 0$ **to** $len(\hat{A}) - 1$ **do**
  if $node \notin visited$ **then**
    $current\_tree \leftarrow []$
    `dfs`$(node, current\_tree)$
    $Extracted\_trees$.append$(current_tree)$
return $Extracted\_trees$

3. EXPERIMENTS

3.1 Study Area and Dataset

The national forest park on the Yellow Sea coast in southeast China (120.82° E, 32.87°N) is the study area of this research. It is a managed forest with a total area of 2240 ha while the planted tree species involve dawn redwood (*Metasequoia glyptostroboides*), poplar (*Populus deltoids*), and ginkgo (*Ginkgo biloba*) (Fekry et al., 2022). Seven square test plots (30 m side length) were sensed using backpack laser scanning. The test sites comprise two tree species with different tree parameters including diameters at breast height (DBH), heights H, and stand densities SD as listed in Table 1.

Table 1. Forest parameters of the test sites.

Plot	Species	DBH(cm)	H(m)	SD (1/ha)
1	Poplar	36.4	31.4	256
2	dawn redwood	30.6	29.4	489
3	dawn redwood	30.7	26.7	233
4	Poplar	39.7	36.1	222
5	dawn redwood	27.6	22.8	578
6	Poplar	36.4	26.9	156
7	Poplar	22.1	21.3	444

The GreenValley BLS system consists of a Velodyne Puck VLP-16 scanner, POS, and a touchpad to manage data capture was used for LiDAR data collection. The point cloud was produced using SLAM technology, which combines scan data with POS information. Table 2 lists the sensor parameters during the data acquisition.

Table 2. Scanner parameters used for data collection.

Parameter	Value
Sensor	Velodyne VLP-16
Max. range (m)	100
Ranging accuracy (cm)	± 3
Max. footprint (°)	0.1-0.4
Wavelength (nm)	905
Horizontal FOV (°)	360
Vertical FOV (°)	30
Point density (m^{-2})	$(5-10)\times10^3$
Rotation rate (Hz)	5-20

3.2 Evaluation Scheme

To evaluate the performance of the tree segmentation, three metrics were employed. The first metric is called recall $\boldsymbol{R}$, which is computed by dividing the number of correctly segmented trees by the total number of trees. The second metric is precision P, which is calculated by dividing the number of correctly segmented trees by the total number of segmented trees. The third metric is the $\boldsymbol{F1}score$, which provides an overall measure of accuracy based on both recall and precision. The formula for computing these metrics is provided in Equation 5, where the true positive $\boldsymbol{TP}$ represents the number of correctly segmented trees, false negative $\boldsymbol{FN}$ represents the number of omitted trees, and false positive $\boldsymbol{FP}$ represents the number of incorrectly segmented trees.

$$\begin{aligned} \boldsymbol{R} &= \left(\frac{\boldsymbol{TP}}{\boldsymbol{TP}+\boldsymbol{FN}}\right)\% \\ \boldsymbol{P} &= \left(\frac{\boldsymbol{TP}}{\boldsymbol{TP}+\boldsymbol{FP}}\right)\% \\ \boldsymbol{F1}score &= \left(\frac{2\times\boldsymbol{P}\times\boldsymbol{R}}{\boldsymbol{P}+\boldsymbol{R}}\right)\% \end{aligned} \tag{5}$$

It is worth noting that the number of trees in each plot is identified based on field observations which also provide tree locations. Therefore, the total number of segmented trees is compared to the reference number from fieldwork. If the reference tree location corresponds to a segmented tree, then the detected tree belongs to the $\boldsymbol{TP}$. The $\boldsymbol{FN}$ occurs when the tree location does not represent a segmented tree (i.e., the omitted trees) while the $\boldsymbol{FP}$ represents the over-segmented and under-segmented trees.

4. RESULTS

4.1 Model Training

The ADAM optimizer Kingma (2015) was used for model training and the learning rate was 0.01. For nonlinearity, the rectified linear unit (ReLU) activation function was employed. The number of epochs was 250 with a step of 25, and the number of neighbors was set to 250 (i.e., K = 250 in KNN). All experiments were performed using an NVIDIA GeForce GTX 1080 GPU and the model was fully implemented based on the PyTorch library Fey and Lenssen (2019). Figure 2 shows the minimization of the reconstruction loss during training.

4.2 Single Tree Segmentation

Individual tree segmentation was performed on seven sample plots as shown in Table 1. The evaluation metrics were computed for single plots and the results were summarized in Table 3. Segmentation results were promising for all plots. Detection accuracy ranged from 87.5 % to 92.5 % for the dawn redwood plots, while its counterpart from poplar plots was 84.6 % to 95.2 %. Figure 3 shows the segmentation results for the point cloud of a poplar sample plot based on the proposed approach.

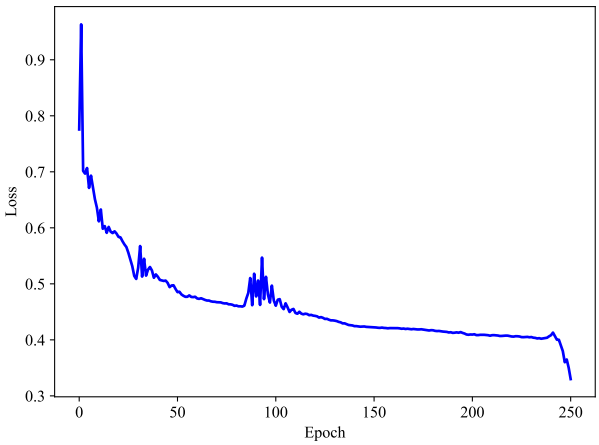

Figure 2. Reconstruction loss.

Table 3. Individual tree segmentation from BLS point cloud.

Plot	$\boldsymbol{TP}$	Total segmented trees	$\boldsymbol{R}$ (%)	$\boldsymbol{P}$ (%)	$\boldsymbol{F1}Score$ (%)
1	20	23	86.9	95.2	91.2
2	37	44	84.1	92.5	88.1
3	21	29	72.4	87.50	79.2
4	18	20	90	94.7	92.3
5	50	52	96.2	92.5	94.3
6	11	14	78.6	84.6	81.5
7	36	40	90	90	90

The segmentation results were analyzed by comparing the results of the individual plots as shown in Table 3. Considering the tree species, the precision range of the poplar species was 84.6 % to 95.2 % while this range was 87.5% to 92.5 % for dawn redwood species. On the one hand, the largest precision of poplar species occurred at Plot 1 (stand density: 256 tree/ha) while the precision recorded its minimum at Plot 6 with the lowest stand density. This could be attributed to the irregular tree shapes of Plot 6 where some trees are co-dominant. Although Plot 7 comprised the highest stand density among poplar plots, it reported a precision of 90% (5.4 % greater than the precision of Plot 6). The reason also could be the co-dominant trees in the case of Plot 6. On the other hand, sample Plot 5 had the highest stand for the dawn redwood plots and recorded the largest precision among the dawn redwood plots. This can be attributed to the simple understory structure with low shrubs of Plot 5. The precision gained at sample Plot 3 (87.5 %) was lower than that of Plot 5; however, the stand density of Plot 5 was greater than that of Plot 3. This also could be attributed to the complex understory structure of Plot 3 where there were a large number of irrigated grasses and shrubs. Similar findings were gained when comparing the precision of Plot 1 (poplar species) and Plot 3 (dawn redwood species). The precision of Plot 1 was more remarkable than that of Plot 3; however, Plot 3 was planted with poplar trees. Thus, the proposed approach has potentially performed for poplar and dawn redwood species that involve forest parameters.

It is worth noting that however the stems are well sampled from BLS data, the proposed methodology has segmented single trees from BLS data without stem detection. Moreover, our segmentation approach can be generalized to airborne

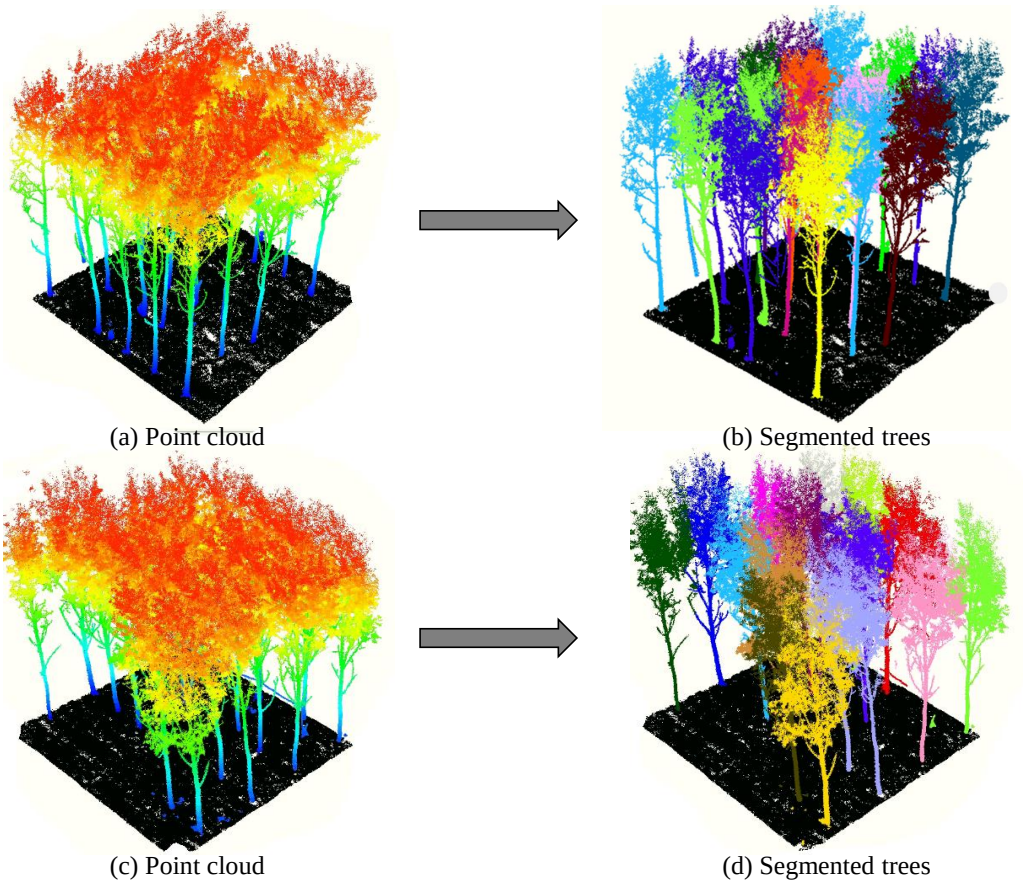

Figure 3. Individual tree segmentation of two poplar plots based on the proposed approach. Unique colors represent single trees.

(e.g., ALS or UAV) LiDAR data where the stems are not well-sampled without treetop detection on a CHM. Thus, the loss of 3D information will be mitigated and the understory will be segmented. Unfortunately, the experiments have been conducted on planted sample plots with regular tree shapes. Therefore, we are willing to extend the experiments to other forest types with high complexity and mixed species. Further, performing tree segmentation from ground-based (e.g., TLS) and airborne LiDAR systems.

5. CONCLUSIONS

This research proposes a novel approach to perform single tree segmentation from LiDAR data. The developed approach is inspired by link prediction problems from recommendation systems. First, a point cloud is seen as an undirected graph. Then, this graph is fed to a graph convolution autoencoder to learn the node features for graph reconstruction. Finally, a single tree is extracted as the set of connected graph nodes based on link prediction. One main advantage of the proposed approach is that it does not require any information about trees or forest structures. In addition, it does not depend upon stem detection or treetop detection which mitigates the limitations of CHM-based approaches. The experiments involved seven forest plots from BLS data. The results showed a potential performance of the proposed approach in tree segmentation however, the experiments were conducted on a planted forest with monotype tree species at the plot level. Accordingly, one important aspect of future work is testing the proposed methodology against LiDAR data of natural forests (e.g., temperate, boreal) with complex plots. Also, extending the experiments to other types of LiDAR data including ground-based (e.g., TLS, MLS) and airborne (e.g., ALS, and UAV-LiDAR).

ACKNOWLEDGEMENTS

The authors are very appreciative of the support of researchers at the Nanjing Forestry University, particularly Lin CAO for data provision. Also, the authors acknowledge the foresters in the Dongtai forests for the data acquisition and for sharing their knowledge of the regional forest ecosystems.

References

Chen, X., Jiang, K., Zhu, Y., Wang, X., Yun, T., 2021. Individual Tree Crown Segmentation Directly from UAV-Borne LiDAR Data Using the PointNet of Deep Learning. *Forests*, 12(2). https://www.mdpi.com/1999-4907/12/2/131.

Comesaña-Cebral, L., Martínez-Sánchez, J., Lorenzo, H., Arias, P., 2021. Individual Tree Segmentation Method Based on Mobile Backpack LiDAR Point Clouds. *Sensors*, 21(18). https://www.mdpi.com/1424-8220/21/18/6007.

Dai, W., Yang, B., Dong, Z., Shaker, A., 2018. A new method for 3D individual tree extraction using multispectral airborne LiDAR point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 144, 400-411. https://doi.org/10.1016/j.isprsjprs.2018.08.010.

Duncanson, L., Cook, B., Hurtt, G., Dubayah, R., 2014. An efficient, multi-layered crown delineation algorithm for

mapping individual tree structure across multiple ecosystems. *Remote Sensing of Environment*, 154, 378-386. https://doi.org/10.1016/j.rse.2013.07.044.

Fekry, R., Yao, W., Cao, L., Shen, X., 2021. Marker-Less UAV-LiDAR Strip Alignment in Plantation Forests Based on Topological Persistence Analysis of Clustered Canopy Cover. *ISPRS International Journal of Geo-Information*, 10(5). https://www.mdpi.com/2220-9964/10/5/284.

Fekry, R., Yao, W., Cao, L., Shen, X., 2022. Ground-based/UAV-LiDAR data fusion for quantitative structure modeling and tree parameter retrieval in subtropical planted forest. *Forest Ecosystems*, 9, 100065.

Fey, M., Lenssen, J. E., 2019. Fast graph representation learning with PyTorch Geometric. *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Fu, H., Li, H., Dong, Y., Xu, F., Chen, F., 2022. Segmenting Individual Tree from TLS Point Clouds Using Improved DBSCAN. *Forests*, 13(4). https://www.mdpi.com/1999-4907/13/4/566.

Hosoi, F., Nakai, Y., Omasa, K., 2013. 3-D voxel-based solid modeling of a broad-leaved tree for accurate volume estimation using portable scanning lidar. *ISPRS Journal of Photogrammetry and Remote Sensing*, 82, 41-48. https://doi.org/10.1016/j.isprsjprs.2013.04.011.

Hui, Z., Li, N., Xia, Y., Cheng, P., He, Y., 2021. INDIVIDUAL TREE EXTRACTION FROM UAV LIDAR POINT CLOUDS BASED ON SELF-ADAPTIVE MEAN SHIFT SEGMENTATION. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-1-2021, 25–30. https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-1-2021/25/2021/.

Jin, S., Su, Y., Gao, S., Wu, F., Hu, T., Liu, J., Li, W., Wang, D., Chen, S., Jiang, Y., Pang, S., Guo, Q., 2018. Deep Learning: Individual Maize Segmentation From Terrestrial Lidar Data Using Faster R-CNN and Regional Growth Algorithms. *Frontiers in Plant Science*, 9. https://www.frontiersin.org/article/10.3389/fpls.2018.00866.

Kingma, D. P., 2015. &Ba J.(2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., Welling, M., 2016. Variational graph auto-encoders.

Li, W., Guo, Q., Jakubowski, M. K., Kelly, M., 2012. A New Method for Segmenting Individual Trees from the Lidar Point Cloud. *Photogramm. Eng. Remote Sens.*, 78(1), 75–84.

Lu, X., Guo, Q., Li, W., Flanagan, J., 2014. A bottom-up approach to segment individual deciduous trees using leaf-off lidar point cloud data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 94, 1-12. https://doi.org/10.1016/j.isprsjprs.2014.03.014.

Luo, Z., Zhang, Z., Li, W., Chen, Y., Wang, C., Nurunnabi, A. A. M., Li, J., 2022. Detection of Individual Trees in UAV LiDAR Point Clouds Using a Deep Learning Framework Based on Multichannel Representation. *IEEE Transactions on Geoscience and Remote Sensing*, 60, 1-15.

Shendryk, I., Broich, M., Tulbure, M. G., Alexandrov, S. V., 2016. Bottom-up delineation of individual trees from full-waveform airborne laser scans in a structurally complex eucalypt forest. *Remote Sensing of Environment*, 173, 69-83. https://doi.org/10.1016/j.rse.2015.11.008.

Tran, P. V., 2018. Learning to Make Predictions on Graphs with Autoencoders. *CoRR*, abs/1802.08352. http://arxiv.org/abs/1802.08352.

Wang, P., Xu, B., Wu, Y., Zhou, X., 2014. Link Prediction in Social Networks: the State-of-the-Art. *CoRR*, abs/1411.5118. http://arxiv.org/abs/1411.5118.

Wang, Y., Weinacker, H., Koch, B., 2008. A Lidar Point Cloud Based Procedure for Vertical Canopy Structure Analysis And 3D Single Tree Modelling in Forest. *Sensors*, 8(6), 3938–3951. https://www.mdpi.com/1424-8220/8/6/3938.

Xu, W., Deng, S., Liang, D., Cheng, X., 2021. A Crown Morphology-Based Approach to Individual Tree Detection in Subtropical Mixed Broadleaf Urban Forests Using UAV LiDAR Data. *Remote Sensing*, 13(7). https://www.mdpi.com/2072-4292/13/7/1278.

Yang, J., Kang, Z., Cheng, S., Yang, Z., Akwensi, P. H., 2020. An Individual Tree Segmentation Method Based on Watershed Algorithm and Three-Dimensional Spatial Distribution Analysis From Airborne LiDAR Point Clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 1055-1067.

Yao, W., Krzystek, P., Heurich, M., 2012. Tree species classification and estimation of stem volume and DBH based on single tree extraction by exploiting airborne full-waveform LiDAR data. *Remote Sensing of Environment*, 123, 368-380. https://doi.org/10.1016/j.rse.2012.03.027.

Ying, W., Dong, T., Ding, Z., Zhang, X., 2021. Pointcnn-based individual tree detection using lidar point clouds. N. Magnenat-Thalmann, V. Interrante, D. Thalmann, G. Papagiannakis, B. Sheng, J. Kim, M. Gavrilova (eds), *Advances in Computer Graphics*, Springer International Publishing, Cham, 89–100.

Zhang, W., Qi, J., Wan, P., Wang, H., Xie, D., Wang, X., Yan, G., 2016. An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation. *Remote Sensing*, 8(6).

Zhen, Z., Quackenbush, L. J., Zhang, L., 2014. Impact of Tree-Oriented Growth Order in Marker-Controlled Region Growing for Individual Tree Crown Delineation Using Airborne Laser Scanner (ALS) Data. *Remote Sensing*, 6(1), 555–579. https://www.mdpi.com/2072-4292/6/1/555.