EARTH OBSERVATION AND GEOMATICS ENGINEERING
website: https://eoge.ut.ac.ir

3D Convolutional Autoencoder for Tree Extraction from Mobile and Airborne Lidar Point Clouds

Mahdieh Zaboli[1*], Danesh Shokri[1], Fariba Dolati[1], Saeid Homayouni[2]

1- Department of Photogrammetry and Remote Sensing, School of Surveying and Geospatial Engineering, College of Engineering, University of Tehran, Tehran, Iran, (m_zaboli, daneshshokri72, fariba.dolati) @ut.ac.ir

2- Centre Eau Terre Environnement Research, Institut National de la Recherche Scientifique (INRS), Quebec G1K 9A9, Canada, saeid.homayouni@inrs.ca

Article history:
Received: 28 March 2023, Revised: 16 June 2023, Accepted: 17 July 2023

ABSTRACT

Urban planning and green space resource management rely heavily on accurate tree information to combat air pollution, preserve biodiversity, and support wildlife. Conventional tree inventory techniques involve time-consuming, on-site data collection, limiting spatial coverage despite their accuracy. Recent advancements in remote sensing offer promising solutions for precise and scalable tree mapping. This study introduces a novel approach using a 3D convolutional autoencoder (3D-CAE) directly on LiDAR point clouds for tree extraction. Ground point filtering effectively removes extraneous data, and the 3D-CAE model automatically encodes deep features. The Support Vector Machine (SVM) classification then identifies tree-related point clouds. The method was evaluated on diverse datasets, including mobile and airborne laser scanner data (MLS and ALS), demonstrating exceptional efficiency. Results indicate a 95% Precision, showcasing the high performance of the proposed method in individual tree detection and identification. Furthermore, the algorithm computes essential geometrical information for classified trees, including 2D coordinate space, height, stem diameter, and 3D boundary tree locations. In summary, the utilization of 3D-CAE directly on LiDAR point clouds presents a significant advancement in tree inventory, offering accurate and cost-effective urban tree mapping with wide-ranging applications in urban planning and green space management.

KEYWORDS

LiDAR Point Clouds

Mobile Laser Scanner (MLS)

Convolutional Autoencoder

Machine learning algorithms

Tree Extraction

Deep learning

1. Introduction

As an essential part of urban green infrastructures, trees provide beneficial economic services, including ameliorating air and water pollution, thermal regulation, and reducing soil contamination (Jombo et al., 2021). Specifically, urban trees can improve the quality of living and local climate through shading, mitigating the temperature of the land surface that, leads to energy saving, maintaining biodiversity, and reducing air pollution and natural stresses (Schmohl et al., 2022). However, increasing urbanization and deforestation can be an important concern over urban ecology (Yan et al., 2018). Consequently, automatic vegetation recognition in urban and forest environments has attracted particular attention in urban planning and forestry management.

Recent advances in remote sensing technology such as Light Detection and Ranging (LiDAR) point cloud, hyperspectral images, and Synthetic Aperture Radar (SAR) data present the detailed mapping of trees in vast areas, particularly in boreal and temperate ecosystems (Fassnacht et al., 2016; Mäyrä et al., 2021). LiDAR data, including Mobile Laser Scanner (MLS) and Airborne Laser Scanner (ALS) by providing high-point density, is considered a good platform for single tree extraction and related parameter estimation (Fassnacht et al., 2016; Gupta et al., 2010; Schmohl et al., 2022). An MLS system can collect 3D surface information accurately along the driving paths (up to millimeter-level with a few thousand points/m2 density) (Wang et al., 2019). This data has been widely used in urban applications, including autonomous vehicle driving (Jombo et al., 2021), 3D city modeling

(Zaboli et al., 2019a; Zaboli et al., 2023), and environment monitoring (Shokri et al., 2023; Wang et al., 2019) .

In recent decades, aerial, stationary, and mobile laser scanners have provided georeferenced information for various applications, including mobile mapping, cultural heritage documentation, reverse engineering, building reconstruction, digital elevation model production, and urban digital and smart modeling. Aerial and terrestrial laser scanners differ in acquisition modes, typical project sizes, acquisition mechanisms, and achievable accuracy and resolution. An airborne laser scanner is a measurement system in which light pulses (usually generated by a laser) are emitted from an instrument mounted on an aircraft and directed to the ground in a scanning pattern. These systems are equipped with a positioning and orientation system based on the global navigation satellite system and inertial measurements to ensure the calculation of the coordinates of the points. This makes acquiring point clouds of reflection points with high density and spatial coordinates possible .

Besides these main advantages of ALS data, recording LiDAR point clouds from a nadir view results in a lack of other essential observations, such as the stem of trees. To overcome this limitation, mobile laser scanner systems can provide complementary observations from a terrestrial point of view. To this end, ground laser scanners on various platforms and vehicles have been widely used to record LiDAR point clouds .

Although the rapid developments in the development of mobile laser scanner systems, automatic information extraction from LiDAR point clouds has relatively slowly progressed. This is because the processing of hundreds of millions of points is time-consuming. Therefore, new algorithms should be developed to accelerate the computation time without negatively affecting accuracy results. Recently various methods have been proposed for tree extraction from MLS LiDAR point clouds (Dai et al., 2018; Shokri et al., 2021). They can be categorized into three groups 1) mathematical-based methods, 2) rule-based descriptors, and 3) deep learning algorithms .

Regarding the mathematical methods, Canopy Height Model (CHM) and Hough Transform (HT) are popular in tree detection, which converts the 3-dimensional point clouds into 2-dimensional raster images. For example, Safaie et al. (2021) initially filter ground points in a preprocessing step to accelerate the computation time. Next, the tree trunks were extracted by applying the HT algorithm to the raster images. In a novelty way, Dai et al. (2018) suggested a multispectral ALS analysis to measure tree characteristics. A mean shift segmentation procedure on various feature spaces was initially used, and spatial-multispectral domains refined the segmentation process. This study gained accuracy between 82% and 88% in tree segmentation. Burt et al. (2019) suggested a Tree-Seg procedure for forest tree segmentation. This procedure used a combination of Euclidean distance, principal component analysis (PCA), and surface normal parameters in tree segmentation, gaining around 96% accuracy. Fan et al. (2020) considered trees as cylinder shapes to classify them. In this case, the method used was the Ad-Tree algorithm that estimates the geometry of tree stem and branches .

Parameters of every individual tree, like height, stem volume, and diameter at breast height (DBH), can be obtained. Since trees are located on the ground surface and have at least four meters from the ground surface, Zhang et al. (2015) used the height elevation to find tree points inside low-density ALS LiDAR point clouds. In summary, mathematical-based methodologies can approximate tree structures with high acceptable accuracy. Moreover, these algorithms can measure tree parameters like height in a fast computation time. On the other hand, converting coordinate space from 3D into 2D, which eliminates contextual information, and needs numerous sensitive parameters, are the main drawback of these algorithms.

Concerning the rule-based methods, the handicraft descriptors, like Linearity, which can assign a meaningful value to irregular point clouds, can be implied and fed to the classifiers of machine learning procedures such as Support Vector Machine (SVM). These descriptors apply values to each point, meaning that a cable point with a linear structure would get higher than other non-linear objects, such as building facades. Zaboli et al. (2019b) classified the MLS point clouds like roads by feeding ten descriptors to classifiers of SVM, Random Forest (RF), K Nearest Neighbor (KNN), and Multi-Layer Perceptron (MLP). They finally concluded that the RF performed better in object extraction, like trees. Yu et al. (2011) initially made a CHM model on the ALS point clouds, then smoothed it with a Gaussian method. Tree attributes of height and DBH were measured with a trainable RF model, which gained about 10% accuracy in Root Square Mean Error (RMSE). Instead of using laser scanner platforms for obtaining LiDAR point clouds, Chen et al. (2020) suggested using high-quality 3D meshes for tree and building extraction. The 3D meshed was generated from the contextual photogrammetry images inputted in machine learning classifiers. In summary, the primary positive side of these methods is the high computation time and describing objects' geometrical structure. But these methodologies' main disadvantages are needing the huge training data and hand-crafted descriptors.

Regarding the deep learning methodologies, PointNet and PointNet++ have commonly used procedures in point cloud classification and segmentation. A PointNet++ algorithm was proposed by Ma et al. (2022) for urban area object extraction, including a tree which gained an accuracy of around 84.7%. The most beneficial of these algorithms are directly consuming 3D LiDAR point clouds without needing any coordinate space transportation. But, Zou et al. (2017) initially generated regular 3D voxels for making 2D images. Then, a deep learning layer was evaluated on the images, which acquired an accuracy of about 93% in tree extraction. Hui et al. (2021) suggested a transfer learning procedure that uses weights of other developed deep learning algorithms in tree detection. They first detected the canopy of trees by PCA transformation, kernel density estimation,

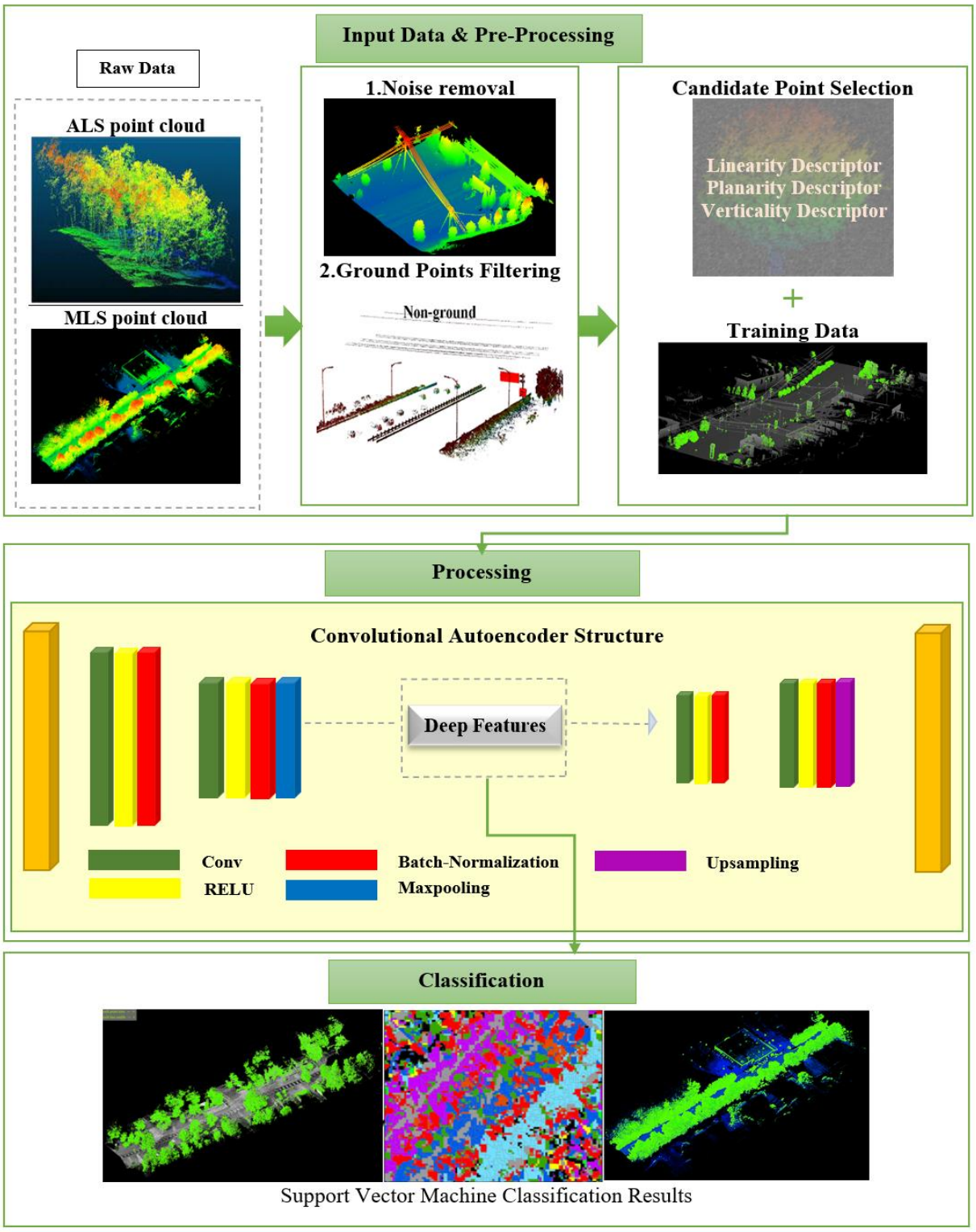


Figure. 1. The flowchart of the proposed method

Gaussian mixture model, and transfer learning weights. Afterward, the tree stems were estimated with an up-to-down manner on the extracted canopies. Shokri et al. (2023) extracted urban trees that collected by a MLS LiDAR point clouds They developed a PointNet++ structure which gained the F1-accuracy more than 95%. In summary, unlike the mathematical and rule-based methods, deep learning algorithms would not need any conversion from 3-dimensional space to a lower one. Also, they automatically measure thousands of descriptors regardless of any human supervision. The considerable disadvantage of these methods is needing big training data and, more importantly, the high computational time. Using the transfer learning procedures means no longer massive training data is required.

In this research, we present an efficient and rapid deep-learning algorithm designed for tree extraction. The algorithm's evaluation encompasses challenging urban and non-urban environments, encompassing scenarios with

numerous large trees, tall buildings, and abundant vehicles. Importantly, our algorithm is rigorously assessed on two distinct systems, namely Mobile Laser Scanner (MLS) and Airborne Laser Scanner (ALS), to ensure its efficiency performance across various setups. To handle datasets with millions of points, we employ a digital surface model (DSM) generated from LiDAR point clouds, significantly reducing computation time. Notably, our proposed algorithm accurately measures essential tree features such as elevation and stem diameter. The primary objectives and contributions of our work are outlined as follows:

• Proposing a fast and robust autoencoder deep learning structure specifically tailored for tree extraction. This architecture is thoroughly tested on five challenging urban and suburban areas, demonstrating its effectiveness in tree analysis.

• Calculating crucial characteristic features of trees, including 2-dimensional coordinate space, elevation, stem diameter, and maximum foliage diameter, thereby providing comprehensive tree information for further analysis.

• Identifying the most suitable rule-based descriptors to efficiently reduce the volume of LiDAR point clouds, optimizing data processing while preserving essential information.

Our research aims to advance the field of tree extraction using deep learning techniques, providing practical solutions for accurate and comprehensive tree analysis in complex environments.

2. Data and methodology

The proposed algorithm consists of four steps (1) preprocessing to overcome the immense volume of MLS LiDAR point clouds, (2) candidate point selection to limit the search area to find tree points, (3) tree extraction by an efficient and fast autoencoder and lastly (4) tree inventory measurement to measure characteristic features and encroachment analyzing. Fig. 1 shows the flowchart of our proposed algorithm.

Our model performs a preprocessing step, including noise removal and ground point filtering on the raw point cloud. After preprocessing, we can cluster important objects using beneficial descriptors in the candidate point selection step. For extraction of efficient and useful features in the input data, encoding part of autoencoders aids in learning important hidden features in the reconstruction error reduction process. In the encoding step, a new set of combinations of original features is generated. These features are input components to enter a Support Vector Machine (SVM) to classify tree points.

2.1 Preprocessing

The preprocessing step enhances the proposed algorithm in terms of computation time and robustness on noisy points. It includes two parts of Noise Removal (NR) and Ground Points Filtering (GPF). NR and GPF enhance the computation time process, while NR reduces the negative impact of noisy points on the final outputs, each of which is discussed as follows:

Thanks to the capability of Robot Operating System (ROS) technology, saving the MLS LiDAR point clouds as separate LAS files would be simultaneously implemented. In other words, the real data is recorded with various small-size files, not heavy files. The LAS file is a popular format designed for archiving LiDAR point clouds. This step needs a conditional parameter to save LAS files where the trajectory data here would play a positive role. The conditional parameter is when the vehicle passes a specific pass way length (L). The recorded point clouds are saved as a separate LAS file archive. The parameter of L is calculated with the help of trajectory data and Euclidean distance measuring like the equation below (Eq. 1).

$$S_k = sum\left(sqrt(\left(X_{i,k} - X_{i+1,k}\right)^2 + \left(Y_{i,k} - Y_{i+1,k}\right)^2)\right) = L, \quad (1)$$
$$1 \le i \le j, 1 \le k \le n$$

where X, Y = trajectory data positions,
L = the sum of calculated distances
n = the final number of created sections

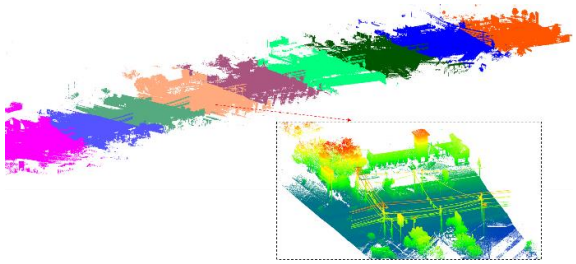


Figure. 2. Displaying samples of LiDAR point clouds with multicolour.

Five steps are considered to detect noisy points inside each crested section. Initially, every point (seed) finds its adjacent points using the K Nearest Neighbour (KNN) algorithm. Then, the Euclidean distance between the seed point and their k neighbourhood points would be computed, called local measuring. For each seed point, k distances are calculated. Afterward, two parameters of local mean and local standard deviation are measured for calculated distances in the local measuring, meaning that the mean and standard deviation of the computed distances is just estimated. Assuming that the MLS point clouds distribution is Gaussian, global mean and global standard deviation parameters are calculated from the total local means and local standard deviations. Each created section from the sectioning only has a unique global mean and global standard deviation. Since the number of noisy points would not be more than one percent of each section volume, the measured global mean and standard deviations are far away from the noisy points. Consequently, those neighbourhood points with a distance from the points more than the sum of the global mean and global standard deviation are considered noisy and eliminated.

The difference between trees and other natural vegetation like bushes is the elevation parameter, meaning vegetation above 4m from the ground surface is considered a tree

(www.frontiersin.org). This vegetation classification is vital in removing unneeded points such as ground surface ones. The popular ground extraction method like pseudo nDSM generation or Cloth Simulation Filter (CSF) needs an elevation parameter as an inputted value (Yang et al., 2020). The CSF was considered for ground point extraction because it robustly simulates the ground surface structure ranging from flat to steep. Moreover, if a place of collected LiDAR point clouds did not have points, it would also simulate the ground surface properly. CSF needs two parameters of grid size and elevation where selected, 0.5m and 0.3, respectively. As seen in Fig. 3, CSF models the ground surface points correctly. With removing the ground points (red ones in Fig. 3), about 90% volume reduction happens, meaning that 4 million points in each section with around 5 million points are correctly removed.

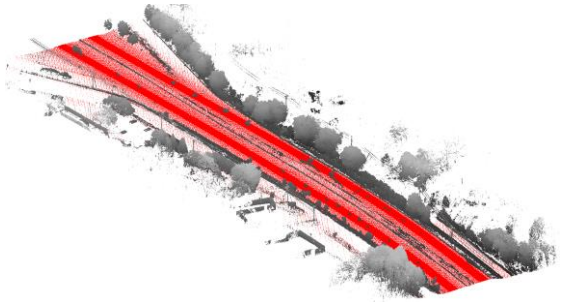


Figure 3. Generation of nDSM for reducing the massive volume of MLS data. The red points indicate the ground points that are eliminated.

2.2. Candidate Points Selection

After eliminating the ground points, various challengeable objects are still available, like buildings, pole-shaped objects, and traffic signs beside the trees. Therefore, this step aims to identify candidate points for limiting the search area to tree recognition. Thanks to the descriptors of rule-based descriptors, which the Principal Component Analysis calculates (PCA) algorithm, they can assign values between [0,1] to the remained points. For instance, by applying the descriptor of Linearity, those objects like power line cables following a linear structure would get a value near one, and other non-linear objects much lower than one. Likewise, the descriptor of planarity also detects the building facades and flat structures like traffic signs. This trend also goes through the pole-shaped objects, meaning that the Verticality descriptor detects them. The Linearity, Planarity, and Verticality are measured based on the eigenvalues of PCA parameters as follows:

$$A = [E_1 \;\; E_2 \;\; E_3] \begin{bmatrix} e_1 & 0 & 0 \\ 0 & e_2 & 0 \\ 0 & 0 & e_3 \end{bmatrix} [E_1 \;\; E_2 \;\; E_3] \quad (2)$$

$$Linearity = \frac{e_1 - e2}{e_1} \quad (3)$$

$$Planarity = \frac{e_2 - e_3}{e_1} \quad (4)$$

$$Verticality = 1 - N_z \quad (5)$$

where $(E_1 \;\; E_2 \;\; E_3)$ *= the PCA eigenvectors* $(e_1 \;\; e_2 \;\; e_3)$ *= the PCA eigenvalues.*
Since trees are natural and do not follow a specific pattern, unlike manufactured objects, points that obtain a descriptor close to one are considered non-tree and eliminated. Figure 4 shows the linearity feature on a sample of points where the cable points received a value close to one while the tree points are around 0.5.

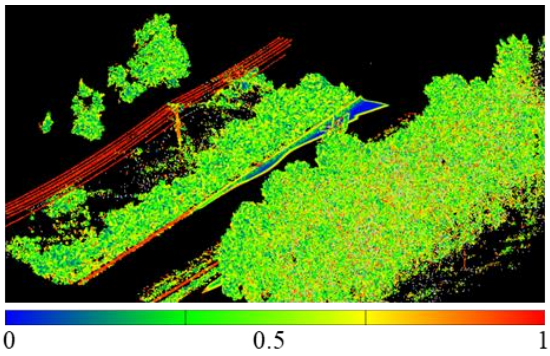


Figure 4. Displaying linearity descriptor on a point cloud sample with the linearity colour bar.

2.3 Tree Extraction

Autoencoders are unsupervised learning algorithms that extract descriptors from a compressed version of the original data. Extraction of such descriptors might be helpful in scientific discovering and characterizing essential factors of variation in data, compressing data for efficient storage and calculation, and using in a preprocessing step before entering the supervised learning algorithm (Bank et al., 2020). Autoencoders can be used for dimensionality reduction, denoising data, generative modelling, and even pretraining deep learning neural networks (Pinaya et al., 2020). An autoencoder is a specific type of neural network trained to reconstruct input data and consists of three components: encoder, code, and decoder. The encoder and decoder are neural networks, and code is a single layer of an ANN with desired dimensionality (www.towardsdatascience.com). Four hyperparameters must be set before training the autoencoder, including code size, number of layers, number of nodes per layer, and loss function (Pinaya et al., 2020). In addition, the stochastic gradient descent of a multi-layer neural network can be used similarly to supervised learning to minimize a loss function. The structure of a simple autoencoder is shown in Fig. 5.

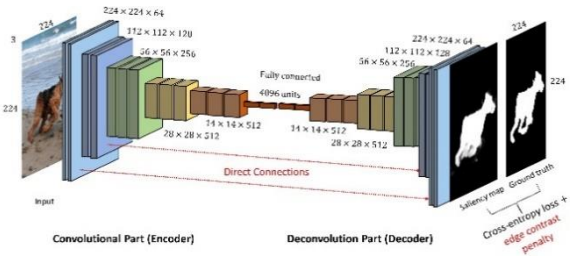


Figure 5. The structure of an autoencoder algorithm (www.gabormelli.com) for learning the weights in an autoencoder.

For example, for x with continuous-valued, the parameters of h and g must be optimized to minimize the reconstruction error, measured according to this loss function (squared loss):

$$\mathcal{L}_{SE}(\tilde{x}, x) = \sum_{j=1}^{d}(x_j - \tilde{x}_j)^2 \quad (6)$$

$$\min_{w^1, w_0^1, w^2, w_0^2} \sum_{i=1}^{n} \mathcal{L}_{SE}(h(g(x^{(i)}; W^1, W_0^1); W^2, W_0^2), x^{(i)}) \quad (7)$$

where $x^{(i)} \in R^d$, encoder g: $R^d \rightarrow R^k$, and decoder h: $R^k \rightarrow R^d$. Linear autoencoder can be powerful in minimizing the objective function using principal components analysis (PCA) to obtain a closed-form solution. To optimize the problem, a singular value decomposition (SVD) can be applied in a multi-layer neural network with non-linear activations for regression (Kunin et al., 2019).

There are five popular autoencoder structures for feature learning, anomaly detection, and dimension reduction: under-complete autoencoders, sparse autoencoders, contractive autoencoders, and denoising autoencoders, and variational autoencoders for generative modelling (Bank et al., 2020). The proposed 3D-Convolutional autoencoder (3D-CAE) is a supervised feature learning method that recently attracted much scientific attention. 3D-CAE is a multi-level feature extraction model to discover the structural information of input data (Rahimzad et al., 2021). As shown in Fig. 5, the designed 3D-CAE contains two main blocks: encoder and decoder. This network has been designed to transform the input point clouds into a compressed and meaningful representation (encoding). The extracted feature maps from input data can be used in classification and segmentation. Finally, the constructive features are transformed into the initial data, and the reconstructed output is as similar as possible to the original input (decoding) (Pinaya et al., 2020).

In the presented 3D-CAE, input point cloud data is represented by a 3D cube which contains a 3-dimensions context spatial. To explore the inner information of data, only 3D or element-wise operations are adopted in the proposed 3D-CAE for 3D point cloud, including 3D convolution, 3D pooling, 3D batch normalization, ReLU function, and 3D deconvolution (Mei et al., 2019).

١) 3D convolution

For input $I \in R^3$ of size $D_x^{(I)} * D_y^{(I)} * D_z^{(I)}$, with a kernel $\boldsymbol{W} \in R^3$ of size $D_x^{(\boldsymbol{w})} * D_y^{(\boldsymbol{w})} * D_z^{(\boldsymbol{w})}$ ($D_x^{(\boldsymbol{w})} \leq D_x^{(\boldsymbol{I})}$, $D_y^{(\boldsymbol{w})} \leq D_y^{(\boldsymbol{I})}$, and $D_z^{(\boldsymbol{w})} \leq D_z^{(\boldsymbol{I})}$), the output is defined as

$$O^{x,y,z} = b + \sum_{p=0}^{D_x^{(w)}-1} \sum_{q=0}^{D_y^{(w)}-1} \sum_{r=0}^{D_z^{(w)}-1} W^{p,q,r} I^{x.s_x+p, y.s_y+q, z.s_z+r} \quad (8)$$

$x = 1, 2, \ldots D_x^{\boldsymbol{O}}$, $y = 1, 2, \ldots D_y^{\boldsymbol{O}}$ and $z = 1, 2, \ldots D_z^{\boldsymbol{O}}$ where $o^{x,y,z}$ illustrates the (x,y,z) elements of output $\boldsymbol{O} \in R^3$, (s_x, s_y, s_z) are related to the size of the stride in 3-dimensions, b represents the bias, $D_x^{\boldsymbol{O}}$, $D_y^{\boldsymbol{O}}$, $D_z^{\boldsymbol{O}}$ denote the output's sizes ($\boldsymbol{O}$) and are defined as

$$D_x^{\boldsymbol{O}} = \left[\frac{D_x^{\boldsymbol{I}} - D_x^{(\boldsymbol{W})}}{s_x}\right] + 1$$
$$D_y^{\boldsymbol{O}} = \left[\frac{D_y^{\boldsymbol{I}} - D_y^{(\boldsymbol{W})}}{s_y}\right] + 1 \quad (9)$$
$$D_z^{\boldsymbol{O}} = \left[\frac{D_z^{\boldsymbol{I}} - D_z^{(\boldsymbol{W})}}{s_z}\right] + 1$$

where $[.]$ represents the round-to-zero process.

Applying such 3D convolution to a point cloud cube can extract useful features because 3D convolution kernels are connected to discover various useful features. When many 3D convolution layers are stacked sequentially, an extra dimension will be considered to handle these extracted feature cubes. Therefore, in the proposed network, the 3D convolution kernel is denoted as $W \in R^4$ of size $D^x \times D^y \times D^z \times D$, where D as the extra fourth dimension demonstrates the number of 3D feature cubes input to the convolutional layer.

The input to the ith convolution layer is determined as $I_i \in R^4$ of size $D_x^{(I)i} \times D_y^{(I)i} \times D_z^{(I)i} \times D_i$ and the 3D convolution in the ith convolution layer is defined as

$$O_{i,j}^{x,y,z} = b_{i,j} + \sum_{k=0}^{D_i-1} \sum_{p=0}^{D_x-1} \sum_{q=0}^{D_y-1} \sum_{r=0}^{D_z-1} W_{j,k}^{p,q,r} I_{i,k}^{x.s_x+p, y.s_y+q, z.s_z+r} \quad (10)$$

Where 'i' and 'j' indexs are the convolutional layer and the convolutional kernels in a layer, respectively.

٢) 3D deconvolution

The deconvolution or transposed convolution can be considered the reverse of the convolutional layer. Transferring low-dimensional to high-dimensional space is useful in many applications such as image semantic segmentation, style transfer, and image inpainting (Gatys et al., 2015; Long et al., 2015; Mei et al., 2019).

٣) 3D Batch Normalization

By supposing $X_i(i = 1,2, \ldots, M_i) \in R^3$ as a minibatch of inputs, $Y_i(i = 1,2, \ldots, M_i) \in R^3$ is defined as the output of 3D batch normalization with the same size as the input and M_i is the number of feature maps in minibatch.

The 3D batch normalization is described as

$$Y_i = \frac{X_i - mean^{M_i}[X]}{\sqrt{Var^{(M_i)}[X] + \epsilon}} * \gamma + \beta, \quad i = 1,2, \ldots, M_i \quad (11)$$

where $mean^{M_i}[X]$ and $Var^{(M_i)}[X]$ denote the mean and standard deviation of X_i respectively,

γ and β are the learnable parameters, and ϵ was set to $1e^{-5}$ as default (Mei et al., 2019).

٤) 3D pooling

This layer is applied to convolved layers to reduce the number of calculated parameters in the training step in a CNN. 3D max pooling can be applied to summarize the explored features using T 3D convolutional kernels W_t, t =1, 2, …, T. The 3D max pooling is defined as

$$O^{x,y,z} = \max F_t^{x,y,z} \quad (12)$$

Where $F_t^{x,y,z}$ illustrates the calculated features through 3D convolution kernels W_t and $O^{x,y,z}$ is a feature at the position (x, y, z) after applying 3D max pooling without flattening (Mei et al., 2019).

After deep feature extraction, the calculated features are fed to Support Vector Machine (SVM) algorithm to classify tree points. SVM is a classification method that generates input-

output descriptor functions from a group of labelled training data. This algorithm finds a plane in an N-dimensional space (descriptor space) that specifically classifies the data. Descriptor functions often transform the input data into a multi-dimensional descriptive space using non-linear kernel functions so that the data in the new space can be separated more easily than the input space. The kernel function of SVM accepts the data as input and converts it into the required form.

- ***Tree characteristics measurement***

After detecting tree points, individual trees from the extracted trees are segmented based on the methodology proposed by Safaie et al. (2021). Next, this step aims to measure the characteristic feature of each tree as follow:

- ***Characteristic measuring***

Characteristic measuring means a set of tree parameters is going to be estimated. Elevation, stem diameter, stem elevation, and, more importantly, the 3D boundary of leaves is measured from this stage.

Tree elevation parameter. assume $T_i = \{X_i, Y_i, Z_i\}$, $0 \leq i \leq n$ *(n is the number of individual tree points) is the points of an individual tree. Initially, the two points which have the maximum* (max_Z) *and minimum elevation* (min_Z) *in T are found. The elevation of an individual tree is equal to the subtraction of* max_Z *and* min_Z.

Stem Diameter (SD) parameter. Thanks to the methods of circle fitting procedures with the help of Random Sample Consensus (RANSAC), the diameter of the tree stem will be estimated. As stems are located bottom part of trees, firstly, those points placed in the lowest part of T with elevation ranging from min_Z *and* $min_Z + 1$ *are detected. Afterward, a circle equation (Eq. 12) is fitted to the extracted points.*

$$(X_i - x_0)^2 + (Y_i - y_0)^2 = R^2 \qquad (13)$$

where R *= the estimated stem radius*

(x_0, y_0) *=2-dimensional location of the tree,*

(X_i, Y_i) *= the extracted points as the tree stem.*

3D Boundary extraction parameter. Recently, the procedure of convex hull has shown positive performance in object analysis from radiometric imageries or LiDAR point clouds (Yan et al., 2019). Regarding tree points, the convex hull remains the 3D boundary points and eliminates interior ones that are useless in parameter measuring. Indeed, the convex hull is the smallest shape that contains the tree, reducing each tree's volume.

Foliage height (FH), maximum foliage diameter (MFD), and trunk height (TH) are three other parameters that will be measured from each tree (Fig. 3).

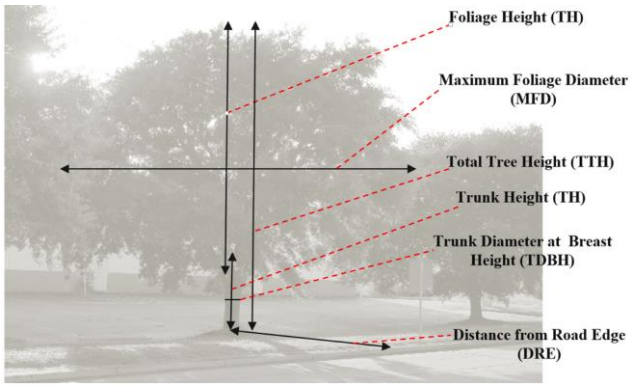


Figure 5. The characteristic feature of an individual tree (Safaie et al., 2021).

Nevertheless, the trajectory data is considered instead of road surface points. Road boundary extraction would not run rapidly due to millions of points processing located on the road structure. Also, the trajectory points are placed on the road surface and have a much lower volume of implementation. The distance condition here is 3.6m.

2.3 Tree Extraction

Generally, object extraction methods, including the proposed algorithm, are evaluated by Precision and recall accuracies. Three variables of True Positive (TP), False Positive (FP), and False Negative (FN) would be needed to calculate the accuracies.

$$precision = \frac{true\ positive}{true\ positive\ and\ false\ positive} = \frac{TP}{TP + FP} \qquad (14)$$

$$recall = \frac{true\ positives}{true\ positive\ and\ false\ negative} = \frac{TP}{TP + FN} \qquad (15)$$

$$F1score = 2 \times \frac{precision \times recall}{precision + recall} \qquad (16)$$

where, TP *= the extracted tree points*

FP *= not extracted tree points*

FN *= extracted non-tree points as the tree*

۳. *Experiments and Results*

3.1 Study Area and Results

In all tested datasets, 20% was used for training, while 80% was used for evaluation. The training and test data were selected randomly at the beginning of the classification process. The efficiency and performance of the proposed algorithm are going to be assessed in four urban and sub-urban environments as follows:

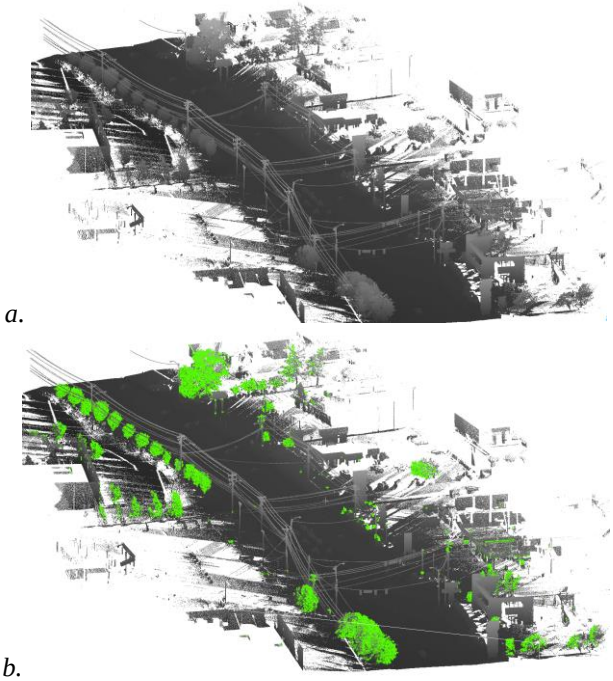
- ***USA urban environment***

Our algorithm underwent thorough evaluation in a complex urban environment, encompassing a diverse range of challenging objects, including dense trees and tall buildings. For the research assessment, we selected an urban roadway test section located in Anderson, South Carolina, USA.

The chosen MLS dataset covers a 600-meter stretch of US Highway Route 76, also known as Clemson Blvd. This road is a 4-lane urban arterial, starting from Forest Hill Drive and extending all the way to the intersection with East-West Parkway.

This carefully selected urban area provided a suitable and demanding setting to test the robustness and effectiveness of our algorithm in accurately identifying and extracting various objects, including dense vegetation and towering

buildings, essential for urban planning and management purposes.



a.

b.

Figure 6. USA urban environment; (a) Collected LiDAR point clouds and (b) Extracted trees by our method (green ones).

Table 1 presents the characteristics of the point clouds collected using an MLS system, as illustrated in Fig. 6. The MLS system showcased an impressive capability of recording 1100 points per second. In total, approximately 18 million points were collected from five sections spanning a 250-meter road surface.

Upon removing the ground and noisy points within each section, the volume of each section significantly decreased, with processing times averaging around 14 seconds. This substantial reduction in volume proved to be a noteworthy advantage in efficiently handling the vast amount of MLS LiDAR point clouds.

Notably, our algorithm, as depicted in Fig. 6-b, accurately identified approximately 983 thousand points as trees. This outcome highlights the efficacy of our approach in successfully detecting trees amidst the collected MLS point clouds, further showcasing the potential of our algorithm in urban tree assessment and inventory.

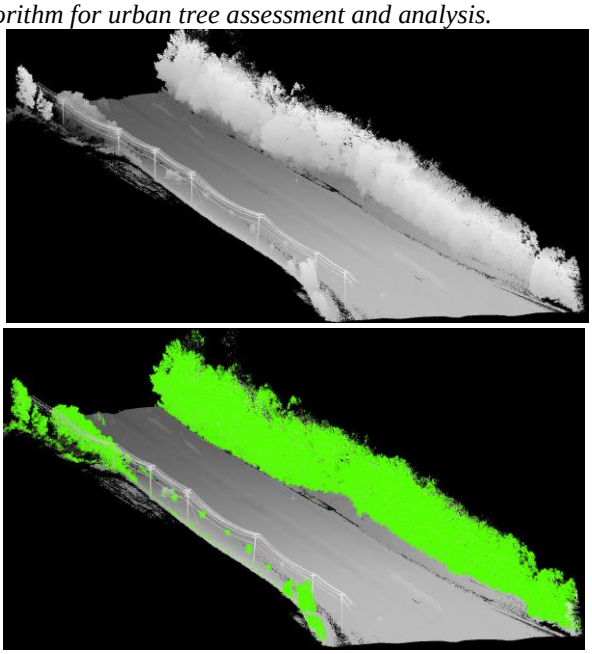
Table 1. Specification of MLS LiDAR pint clouds.

	Scanner Parameter	***USA dataset***	***China dataset***
LiDAR	*Brand*	*Optech*	*RIEGL*
	Model	*SG1*	*VQ-450*
	Single / Dual laser	*Dual*	*Dual*
	Measurement rate	*600 kHz/sensor*	*1100 kHz/sensor*
DMI	*Brand*	*Applanix*	*Riegl*
	Model	*HS35F*	*N/A*
IMU	*Brand*	*Applanix*	*Riegl*
	Model	*FMU P301*	*N/A*
	Roll/pitch accuracy	*0.005°*	*0.005°*
	Heading Accuracy	*0.015°*	*0.015°*
Camera	*Type*	*Point Grey 360°*	*VMX-450-CS6*
	No. of Cameras	*6*	*6*
	Focal Points of Cameras	*N/A*	*N/A*
	Resolution	*5 MP*	*5 MP*
Vehicle Mounted GPS/GNSS	*Brand*	*Trimble*	*RIEGL*
	Model	*AT1675-540TS*	*VMX-450-MH*
	Accuracy	*0.02' H;0.04' V*	*20mm abs, 10mm relative*

USA suburban environment:

The data collection for this study took place in South Carolina, USA, as shown in Fig. 5, resembling a typical urban environment in the United States. The MLS system employed for data acquisition adheres to the specifications outlined in Table 1. In total, approximately 25 million points were recorded by the MLS system, out of which 4.85 million points (comprising both true positives (TP) and false positives (FP)) were identified as tree points.

The specific region in South Carolina was selected for evaluation due to its dense and compacted tree presence, particularly along road infrastructures, as depicted in Fig. 7-a. Leveraging our autoencoder algorithm, we achieved remarkable results, as illustrated in Fig. 7-b, attaining an accuracy of more than 99%. This remarkable accuracy signifies the effectiveness of our approach in accurately identifying and extracting tree points in this challenging urban environment, underscoring the potential of our algorithm for urban tree assessment and analysis.



a.

b.

Figure 7. USA suburban environment; (a) collected LiDAR point clouds, (b) the output of the proposed algorithm.

- ***Lidarusa Urban dataset***

Lidarusa, an organization specializing in civil engineering and heritage mapping projects (www.lidarusa.com), has played a significant role in providing diverse LiDAR point cloud datasets to the public. Notably, one of their datasets, Fig. 8, was chosen to evaluate the efficiency performance of our proposed algorithm. This dataset comprises approximately 14.5 million points along a 330-meter roadside, featuring various challenging tree structures.
Our algorithm demonstrated remarkable accuracy in the evaluation, achieving Precision and Recall accuracies of 96.5% and 99.9%, respectively. These impressive results affirm the effectiveness and reliability of our approach in accurately extracting trees from the challenging Lidarusa dataset.

- ***Lidarusa railway environment***

Lidarusa, known for its open-access datasets, provides another valuable dataset featuring a railway environment densely covered with massive trees (Fig. 9-a). This dataset comprises approximately 22.5 million points spanning a 500-meter railway stretch. Due to the high density of trees compared to other non-tree points, this area serves as a suitable test environment for evaluating precision accuracy.
In this challenging context, traditional accuracies like recall and F1-score might not hold significant meaning. However, in terms of acquiring accuracy, our algorithm demonstrated exceptional performance, achieving a precision accuracy of approximately 98.8%, as depicted in Fig. 8-b. These results further validate the effectiveness of our approach in accurately identifying trees amidst the dense foliage in the Lidarusa railway dataset.

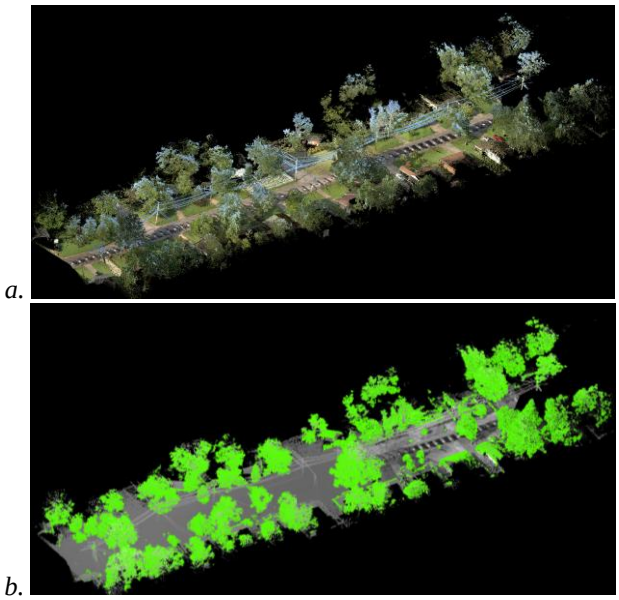


Figure 8. Lidarusa urban dataset; (a) Recorded LiDAR point clouds, (b) Output of the proposed algorithm in tree extraction.

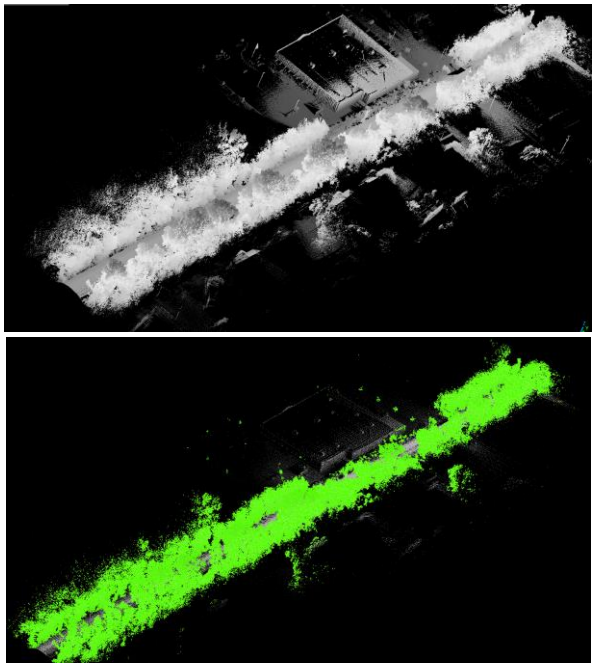
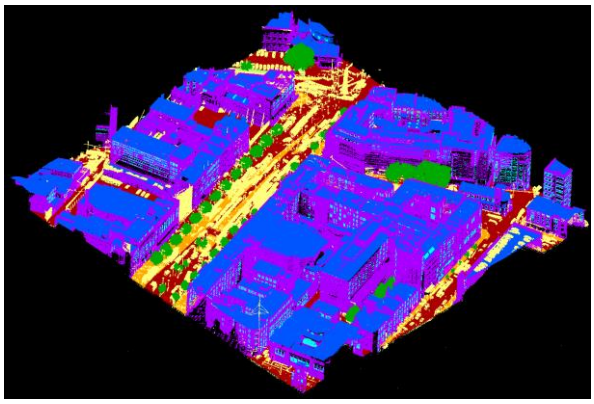


Figure 9. Lidarusa railway dataset; (a) collected LiDAR point clouds; (b) Output of the proposed algorithm.

- ***ALS Urban Area***

As mentioned in the Introduction section, the ALS system is another popular format of LiDAR point clouds; therefore, our algorithm will be assessed on free-access datasets of "DublinCity: Annotated LiDAR Point Cloud and its Applications." (Zolanvari et al., 2019). This dataset (Fig. 10) has been collected from an urban region with ALS equipment. It contains numerous dense trees, tall buildings (walls and roofs), and vehicles that, overall, 250 million points recorded. Fig. 10-b shows the output of our algorithm. The precision accuracy of these results is around 98.6%, while the recall measures about 56.38%.



Evaluation Dataset	*Volume (million)*	*Precision (%)*	*Recall (%)*	*F1-Score (%)*
USA Urban	*18*	*93.7*	*98.1*	*95.8*
USA Suburban	*25*	*98.8*	*100.0*	*99.4*
Lidarusa Urban	*14.5*	*96.5*	*99.9*	*98.2*
Lidarusa Railway	*22.5*	*99.5*	*98.2*	*98.8*
ALS Urban Area	*250*	*98.6*	*56.4*	*71.8*

b.

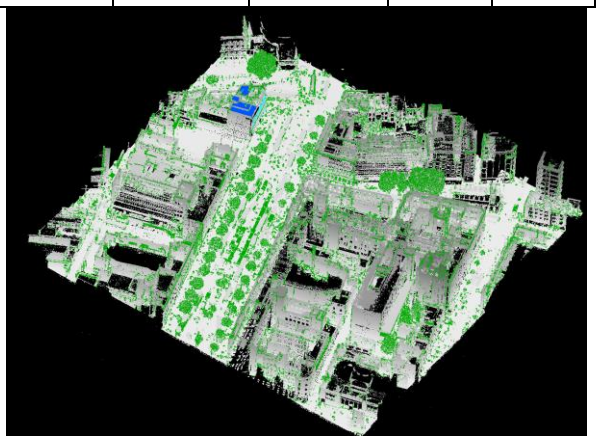


Figure 10. Assessment of our algorithm on ALS point clouds; (a) Dublin dataset; (b) acquired results

Table 2 presents the accuracies obtained by our autoencoder algorithm on 5 MLS and ALS LiDAR point clouds. The algorithm has exhibited commendable performance in tree extraction. Across the MLS point clouds, the algorithm achieved an impressive F1 score accuracy ranging from 95.8% to 99.4%, highlighting its efficiency and feasibility. However, the ALS dataset posed a notable challenge as it incorrectly identified numerous non-tree points as tree points, although it correctly detected over 98%.

The architecture of our deep learning algorithm is illustrated in Table 3, and Table 4 showcases the acquired characteristic parameters for three tree points. These parameters include the precise georeferenced location (x, y) represented by boundary points (indicated by red points). As each extracted tree point comprises thousands of points, identifying their boundary locations proves highly valuable in determining occupied space volume. Additionally, we have measured seven other parameters for each tree, such as tree height, further enhancing the comprehensiveness of our tree analysis.

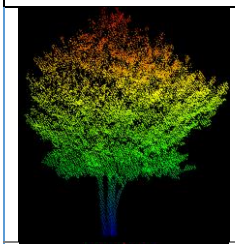
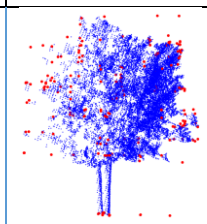
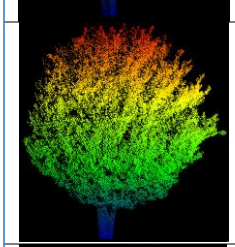
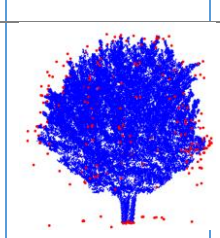
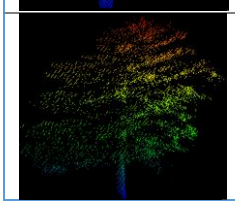
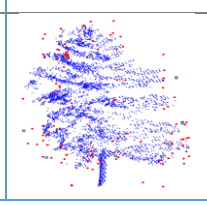
In summary, Table 2 reflects the favorable performance of our autoencoder algorithm in tree extraction from MLS and ALS LiDAR point clouds. Table 3 and Table 4 provide insights into the network architecture and crucial characteristic parameters, showcasing the advancements and potential applications of our deep learning-based approach in tree assessment and inventory.

Table 2. Summarizing the acquired accuracies.

Table 3. Parametric settings of the proposed 3D-CAE when applied to the datasets.

	Input size	***Kernel size***	***strides***	***Output size***

Table 4. Characteristics measurement of three sample trees. The measurement features are at the unit of the meter.

Tree	***3B Boundary***	x_0	y_0	***Height***	***TH***	***FH***	***MFD***	***SD***	***ERC***	***EPL***
		*1496710.58*	*991489.53*	*5.18*	*1.32*	*3.86*	*4.86*	*0.13*	*4.9*	*1.7*
		*1496546.53*	*991667.44*	*6.36*	*0.60*	*5.76*	*5.80*	*0.47*	*7.1*	*5.6*
		*1496686.28*	*191841.72*	*10.49*	*1.67*	*8.82*	*8.11*	*0.24*	*10.23*	*25.5*

Conv1	200×5×5×1	24×3×3×24	1×1×1×1	180×3×3×24
BN1	180×3×3×24	-	-	180×3×3×24
Conv2	180×3×3×24	24×3×3×48	1×1×1×1	160×1×1×48
BN2	160×1×1×48	-	-	160×1×1×48
Pool2	160×1×1×48	18×1×1	18×1×1	9×1×1×48
DConv3	9×1×1×48	9×3×3×24	22×1×1×1	175×3×3×24
BN3	175×3×3×24	-	-	175×3×3×24
DConv4	175×3×3×24	27×3×3×24	1×1×1×1	200×5×5×1
BN4	200×5×5×1	-	-	200×5×5×1

3.2 Discussion

The proposed algorithm was implemented on an ordinary computer system with the specification of Intel (R) Core (TM) i5-3210M CPU @2.50GHz, 12GB RAM, DDR 3, NVidia GeForce 2.630 GB, unlike Yadav, Lohani et al. (2016),Yadav et al. (2016) study which used a cloud computer system. Python was selected as the programming environment to test and run our algorithm.

It is worth mentioning that criteria of selected datasets, space transformation, acquired accuracies, sensitivity analysis, and tree characteristic parameter measuring should be considered to make a clear and comprehensive comparison between our methodology and previous ones. For example, we have proposed a method that gained more than 95% Precision accuracy on five MLS and ALS LiDAR point clouds. While previous ones such as (Fan et al., 2020; Pérez-Martín et al., 2021; Zolanvari et al., 2019) rarely considered diverse study areas. This means the previous studies may not work properly on other environments for tree extraction. In addition, our algorithm has acquired such high accuracy in both urban and suburban regions, while the previous ones have been only tested on one of them like (Dai et al., 2018; Yu et al., 2011). As can be seen in these survey works (Che et al., 2019; Dong et al., 2020; Wang et al., 2020), another positive side of our work is proposing a methodology tested on simultaneously MLS and ALS LiDAR point clouds.

Regarding the space transformation, there is no need to coordinate space transformation for our algorithm compared to (Safaie et al., 2021; Zou et al., 2017) studies where a 3D-to-2D transformation was wanted. Our deep algorithm extracted the tree points and measured the characteristic parameters of the trees. Mainly parameter extraction was ignored, particularly in urban areas (Dai et al., 2018; Wang et al., 2008; Yu et al., 2011; Zhang et al., 2015)

A similar paper for tree extraction has been proposed by (Shokri et al., 2023) study based on a transfer learning procedure. They used a popular deep-learning structure called PointNet++ for tree extraction in MLS LiDAR point clouds. Our algorithm has gained better F1-score accuracy at 4 MLS datasets than in this study. For example, our algorithm acquired F1-score accuracy at 95.8% while they gained 93.8% in the US Urban dataset.

4. Conclusion

Over the past decade, the Mobile Laser Scanner (MLS) system has garnered significant attention from scientists for tree assessment due to its ability to capture hundreds of millions of points from a side-view perspective. This platform offers sub-centimeter accuracy in recording leaves and tree trunks, enabling precise measurement of tree parameters. Our algorithm comprises several key steps, including preprocessing, candidate selection, tree extraction, and tree inventory measurement. To handle the massive volume of MLS data, we have implemented a Cloth Simulation Filter (CSF) that efficiently eliminates 80% of unnecessary points. Additionally, challenging objects like buildings, traffic signs, and pole-shaped structures have been successfully removed using three descriptors: Linearity, Planarity, and Verticality. Subsequently, a 3D convolutional autoencoder was trained on five MLS and ALS LiDAR point clouds for accurate tree extraction, achieving an impressive F1-Score exceeding 95%, affirming its feasibility and high performance. Furthermore, our algorithm accurately measures essential tree parameters, including georeferenced location, tree height, and foliage height. These results underscore the positive contributions of our work, which encompass. i) Proposing a rapid and accurate deep neural learning network structure tailored specifically for tree recognition. ii) Measuring tree inventory parameters, such as planarity and coordinate space, providing comprehensive data for tree analysis. iii) Identifying the most effective descriptors for volume reduction, optimizing data processing without compromising crucial information.

In future endeavours, we highly recommend specifying tree types. As MLS LiDAR point clouds are georeferenced, researchers can leverage satellite datasets like Landsat to determine tree types, further enhancing the depth and applicability of tree assessment.

Acknowledgement

Authors would like to thanks Prof. Wayne Sarasua for sharing their MLS dataset.

References

Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. arXiv preprint arXiv:2003.05991.

Burt, A., Disney, M., & Calders, K. (2019). Extracting individual trees from lidar point clouds using treeseg. Methods in Ecology and Evolution, 10(3), 438-445.

Che, E., Jung, J., & Olsen, M. J. (2019). Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review. Sensors, 19(4), 810.

Chen, M., Feng, A., McAlinden, R., & Soibelman, L. (2020). Photogrammetric point cloud segmentation and object information extraction for creating virtual environments and simulations. Journal of Management in Engineering, 36(2), 04019046.

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.

Dong, Z., Liang, F., Yang, B., Xu, Y., Zang, Y., Li, J., Wang, Y., Dai, W., Fan, H., Hyyppä, J., & Stilla, U. (2020). Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark. ISPRS Journal of Photogrammetry and Remote Sensing, 163, 327-342.

Fan, G., Nan, L., Chen, F., Dong, Y., Wang, Z., Li, H., & Chen, D. (2020). A New Quantitative Approach to Tree Attributes Estimation Based on LiDAR Point Clouds. Remote Sensing, 12(11).

Fassnacht, F. E., Latifi, H., Stereńczak, K., Modzelewska, A., Lefsky, M., Waser, L. T., Straub, C., & Ghosh, A. (2016). Review of studies on tree species classification from remotely sensed data. Remote Sensing of Environment, 186, 64-87.

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576.

Gupta, S., Koch, B., & Weinacker, H. (2010). Tree species detection using full waveform lidar data in a complex forest. na.

Hui, Z., Jin, S., Li, D., Ziggah, Y. Y., & Liu, B. (2021). Individual Tree Extraction from Terrestrial LiDAR Point Clouds Based on Transfer Learning and Gaussian Mixture Model Separation. Remote Sensing, 13(2), 223.

Jombo, S., Adam, E., & Odindi, J. (2021). Classification of tree species in a heterogeneous urban environment using object-based ensemble analysis and World View-2 satellite imagery. Applied Geomatics, 13(3), 373-387.

Kunin, D., Bloom, J., Goeva, A., & Seed, C. (2019). Loss landscapes of regularized linear autoencoders. International conference on machine learning,

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition,

Ma, H., Ma, H., Zhang, L., Liu, K., & Luo, W. (2022). Extracting Urban Road Footprints from Airborne LiDAR Point Clouds with PointNet++ and Two-Step Post-Processing. Remote Sensing, 14(3).

Mäyrä, J., Keski-Saari, S., Kivinen, S., Tanhuanpää, T., Hurskainen, P., Kullberg, P., Poikolainen, L., Viinikka, A., Tuominen, S., & Kumpula, T. (2021). Tree species classification from airborne hyperspectral and LiDAR data using 3D convolutional neural networks. Remote Sensing of Environment, 256, 112322.

Mei, S., Ji, J., Geng, Y., Zhang, Z., Li, X., & Du, Q. (2019). Unsupervised spatial–spectral feature learning by 3D convolutional autoencoder for hyperspectral classification. IEEE Transactions on Geoscience and Remote Sensing, 57(9), 6808-6820.

Pérez-Martín, E., López-Cuervo Medina, S., Herrero-Tejedor, T., Pérez-Souza, M. A., Aguirre de Mata, J., & Ezquerra-Canalejo, A. (2021). Assessment of Tree Diameter Estimation Methods from Mobile Laser Scanning in a Historic Garden. Forests, 12(8), 1013.

Pinaya, W. H. L., Vieira, S., Garcia-Dias, R., & Mechelli, A. (2020). Autoencoders. In Machine learning (pp. 193-208). Elsevier.

Rahimzad, M., Homayouni, S., Alizadeh Naeini, A., & Nadi, S. (2021). An efficient multi-sensor remote sensing image clustering in urban areas via boosted convolutional autoencoder (BCAE). Remote Sensing, 13(13), 2501.

Safaie, A. H., Rastiveis, H., Shams, A., Sarasua, W. A., & Li, J. (2021). Automated street tree inventory using mobile LiDAR point clouds based on Hough transform and active contours. ISPRS Journal of Photogrammetry and Remote Sensing, 174, 19-34.

Schmohl, S., Narváez Vallejo, A., & Soergel, U. (2022). Individual tree detection in urban ALS point clouds with 3D convolutional networks. Remote Sensing, 14(6), 1317.

Shokri, D., Rastiveis, H., Sheikholeslami, S. M., Shahhoseini, R., & Li, J. (2021). Fast extraction of power lines from mobile LiDAR point clouds based on SVM classification in non-urban area. Earth Observation and Geomatics Engineering, 5(2), 63-73.

Shokri, D., Zaboli, M., Dolati, F., & Homayouni, S. (2023). POINTNET++ Transfer Learning for Tree Extraction from Mobile LIDAR Point Clouds. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 10, 721-727.

Wang, C., Wen, C., Dai, Y., Yu, S., & Liu, M. (2020). Urban 3D modeling with mobile laser scanning: a review. Virtual Reality & Intelligent Hardware, 2(3), 175-212.

Wang, Y., Chen, Q., Zhu, Q., Liu, L., Li, C., & Zheng, D. (2019). A Survey of Mobile Laser Scanning Applications and Key Techniques over Urban Areas. Remote Sensing, 11(13), 1540.

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).

Yadav, M., Lohani, B., Singh, A. K., & Husain, A. (2016). Identification of pole-like structures from mobile lidar data of complex road environment. International Journal of Remote Sensing, 37(20), 4748-4777.

Yan, J., Zhou, W., Han, L., & Qian, Y. (2018). Mapping vegetation functional types in urban areas with WorldView-2 imagery: Integrating object-based classification with phenology. Urban Forestry & Urban Greening, 31, 230-240.

Yan, Z., Liu, R., Cheng, L., Zhou, X., Ruan, X., & Xiao, Y. (2019). A Concave Hull Methodology for Calculating the Crown Volume of Individual Trees Based on Vehicle-Borne LiDAR Data. Remote Sensing, 11(6).

Yang, A., Wu, Z., Yang, F., Su, D., Ma, Y., Zhao, D., & Qi, C. (2020). Filtering of airborne LiDAR bathymetry based on bidirectional cloth simulation. ISPRS Journal of Photogrammetry and Remote Sensing, 163, 49-61.

Yu, X., Hyyppä, J., Vastaranta, M., Holopainen, M., & Viitala, R. (2011). Predicting individual tree attributes from airborne laser point clouds based on the random forests technique. ISPRS Journal of Photogrammetry and Remote Sensing, 66(1), 28-37.

Zaboli, M., Rastiveis, H., Hosseiny, B., Shokri, D., Sarasua, W. A., & Homayouni, S. (2023). D-Net: A Density-Based Convolutional Neural Network for Mobile LiDAR Point Clouds Classification in Urban Areas. Remote Sensing, 15(9), 2317.

Zaboli, M., Rastiveis, H., Shams, A., Hosseiny, B., & Sarasua, W. (2019). Classification of mobile terrestrial Lidar point cloud in urban area using local descriptors. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 42, 1117-1122.

Zhang, C., Zhou, Y., & Qiu, F. (2015). Individual Tree Segmentation from LiDAR Point Clouds for Urban Forest Inventory. Remote Sensing, 7(6), 7892-7913.

Zolanvari, S., Ruano, S., Rana, A., Cummins, A., da Silva, R. E., Rahbar, M., & Smolic, A. (2019). DublinCity: Annotated LiDAR point cloud and its applications. arXiv preprint arXiv:1909.03613.

Zou, X., Cheng, M., Wang, C., Xia, Y., & Li, J. (2017). Tree Classification in Complex Forest Point Clouds Based on Deep Learning. IEEE Geoscience and Remote Sensing Letters, 14(12), 2360-2364.