LiDAR Point Cloud Terrain Segmentation

***Abstract*—Forests present a highly heterogeneous environment composed of terrain, vegetation, tree trunks, tree stumps, and other natural elements. The irregularity of Earth's topography, combined with dense vegetation and occlusions, poses a significant challenge to distinguish terrain from other environmental components, particularly when using traditional machine learning approaches. A major bottleneck in this task lies in the feature extraction process from noisy and partially labeled 3D point-cloud datasets, which often lack semantic consistency and spatial clarity. This report proposes a robust and adaptable pipeline for semantic segmentation of 3D point clouds, using 2D annotations to guide learning. Our method integrates state-of-the-art feature extraction techniques and clustering algorithms to isolate tree stumps and terrain as independent semantic clusters. The proposed approach achieves a classification accuracy of 59.73% for terrain and 86.77% for segmentation in a forest UAV dataset of benchmark containing more than 2.5 million points. The results of this work are highly applicable to forest monitoring based on UAVs, with potential extensions to vegetation indexing, biomass estimation, and land use analysis in both forested and urban settings. The pipeline not only addresses the limitations of existing methods, but also sets a foundation for scalable, real-time environmental perception in ecological research and smart forestry initiatives.**

***Index Terms*—Forest segmentation, Point cloud processing, Tree stump detection, Terrain classification, UAV-based monitoring, Machine vision, Terrain Cluster, Object segmentation, 2D labeled point clouds, Machine learning in forestry, PointNet, PointNet2.**

I. Introduction

A. Motivation

Imagine standing inside a dense forest, looking at a vast expanse of trees, vegetation, and natural beauty. Now, imagine the challenge of understanding and analyzing the complex terrain hidden within that forest. This is the inspiration for this R&D project. My mission is to develop a robust and efficient method to segment the forest terrain from the point-cloud data obtained through LiDAR technology. This will involve accurately identifying and excluding objects on the forest surface, such as tree stumps, tree trunks, and dense vegetation. In doing so, My team can gain valuable insight into the terrain and its characteristics, paving the way for effective forest management and planning. The forest terrain is rough and uneven, with different height-varying objects scattered throughout. This complexity makes it difficult to distinguish between the actual terrain and the objects on the forest floor. It requires advanced algorithms and techniques to accurately filter and process the LiDAR data, ensuring that the sensor captures only the relevant information. My goal is to develop such an algorithm.

To overcome these challenges, I am going to dive into the latest challenges and methods in LiDAR data analysis. I am also exploring various filtering, processing, and visualization techniques to make sense of the vast amount of data. I aim to automatically detect and segment the forest terrain without obstacles, providing a clear understanding of the forest's topography. This endeavor will significantly contribute to forest management, ecological research, and land-use planning by offering precise and actionable insights into the underlying terrain.

B. Problem Statement

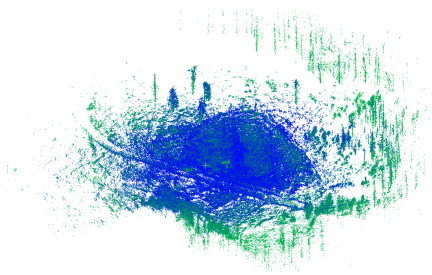

Fig. 1: Original LiDAR Data.

The figure1 illustrates a 3D point cloud representation of a forested area, acquired via UAV-based LiDAR or photogrammetry. The color-coded points—green and blue—indicate different structural features within the scene, where green points are suggestive of vertical elements such as tree trunks or vegetation, and blue points represent the denser ground surface or terrain. This visualization facilitates spatial understanding of forest composition and is instrumental for tasks such as terrain modeling, vegetation mapping, and object segmentation in forest environments.

The project addresses the challenge of accurately segmenting terrain from LiDAR point cloud data in complex forest environments. Raw LiDAR scans capture a dense collection of 3D points representing not just the ground but also a variety of objects, such as tree trunks, stumps, and dense vegetation. The primary problem lies in distinguishing the actual terrain from these overlying objects, given the ruggedness and heterogeneity of forest floors. Existing methods often struggle with transformation invariance, scalability, high computational demands, and the need for large labeled datasets, resulting in suboptimal segmentation and classification performance. This project aims to develop a potent and efficient classification system that can automatically clean, process, and segment LiDAR point cloud data to accurately identify and separate terrain from other objects, thereby enabling improved ecological analysis and forest management.

C. Proposed Approach

The project adopts a multi-stage, unsupervised learning pipeline for point cloud terrain segmentation, leveraging both classical and state-of-the-art machine learning techniques:

1) Data Acquisition and Preprocessing: Raw LiDAR point cloud data is collected, providing 3D coordinates for each scanned point. The data undergoes cleaning to remove outliers and normalization to standardize features, ensuring consistency and reliability for downstream analysis. The Open3D library is utilized for efficient point cloud processing.

2) Feature Extraction: Key geometric features—density, curvature, and height—are extracted from the cleaned data. These features are selected for their effectiveness in differentiating terrain from objects like tree trunks and stumps. Tools such as Open3D [**?**] and PointNet [**?**] are used for detailed feature analysis.

3) Unsupervised Clustering and Segmentation: Clustering algorithms, including DBScan [**?**] and hierarchical clustering, are applied to group points based on feature similarity. This step segments the point cloud into clusters corresponding to different physical entities within the terrain, such as ground, stumps, and trunks [**?**].

4) Cluster Classification: Further refinement is achieved through K-Means clustering, categorizing the segmented clusters into distinct classes. Advanced methods like Cloth Simulation Filtering (CSF) [**?**] for ground extraction and PointNet++ [**?**] for deep feature learning are employed to enhance classification accuracy and scalability. PyTorch serves as the primary framework for implementing these models.

5) Validation and Evaluation: The effectiveness of the segmentation and classification is assessed using multiple validation metrics, including the silhouette coefficient, Davies-Bouldin index [**?**], and Calinski-Harabasz index [**?**]. These metrics evaluate cluster cohesion and separation. Spatial validation ensures consistency across different regions, supporting practical ecological applications. This code pipeline is designed to overcome the limitations of existing methods by ensuring transformation invariance, reducing reliance on labeled data through unsupervised learning, promoting scalability via architecture-agnostic design, and optimizing performance for large-scale, real-world forest datasets. The outcome is a validated segmentation model that provides actionable insights for forest management, ecological research, and land-use planning.

II. Related Work

A. PointNet

Charles et al. [**?**] introduces a novel deep learning architecture specifically designed to handle unordered point directly sets for tasks such as 3D shape classification and segmentation. PointNet's architecture is invariant to input permutation and robust to input transformations, utilizing a combination of multi-layer perceptrons (MLPs) and max pooling to learn global point cloud features. However, the only limitation of PointNet is its inability to capture local structures within the point cloud, as it primarily focuses on global feature learning, which can lead to less accurate segmentation in complex scenes where local geometric details are crucial. Despite this, PointNet is pioneering work directly applying deep learning to point clouds, eliminating the need for voxelization or other preprocessing steps. It demonstrates strong performance across various 3D recognition tasks and opens up new possibilities for deep learning applications in 3D vision.

B. PointClustering

Long et al. [**?**] proposes an unsupervised pre-training method for 3D point cloud representation learning that leverages transformation invariance in clustering. This method learns a generic feature representation by exploiting the inherent clustering structure of point cloud data. However, it primarily focuses on pre-training the representation and does not explore the performance of these representations on downstream tasks. Despite this, the contributions include introducing an unsupervised pre-training method for 3D point cloud representation learning and effectively utilizing the inherent clustering structure of point cloud data to learn a generic feature representation.

C. PointGLR

Rao et al. [**?**] propose a new framework for learning point cloud representations without supervision. Their method uses bidirectional reasoning, which means it connects and compares features from both small local parts and the overall shape of a 3D object. This helps the model learn the shared semantic patterns between local details and global structures. While the approach is innovative, the paper does not discuss how well it works with large-scale 3D data. Still, PointGLR is a valuable contribution, as it introduces a new way to learn meaningful 3D features and expands the method to more complex scenes by using intermediate structures called structural proxies.

D. PointSeg

PointSeg [**?**] is a real-time semantic segmentation network designed for 3D LiDAR point cloud data, particularly in autonomous driving scenarios. Unlike PointNet or PointNet++, which are general-purpose, PointSeg specifically optimizes for real-time performance on mobile platforms by utilizing an efficient convolutional architecture that processes spherical projections of the point cloud. The network begins by projecting the 3D point cloud onto a 2D spherical image based on range and azimuth, effectively transforming the irregular and unordered 3D data into a dense 2D representation. This allows PointSeg to leverage well-established 2D convolutional neural networks (CNNs) for feature extraction. The architecture adopts an encoder-decoder structure, where the encoder learns rich hierarchical features and the decoder refines spatial resolution for pixel-wise segmentation. To maintain contextual awareness and localization precision, PointSeg integrates squeeze-and-excitation (SE) modules and dilated convolutions. These modules enhance feature representation by modeling interdependencies between channels and expanding receptive fields without losing resolution. In their experiments, PointSeg demonstrated competitive segmentation performance on large-scale LiDAR datasets such as KITTI. The network achieved a favorable trade-off between accuracy and inference speed, with real-time throughput on embedded GPUs, making it suitable for robotics and autonomous vehicle applications. The authors

concluded that spherical projection combined with lightweight convolutional designs enables efficient and effective semantic segmentation of 3D point clouds, especially in scenarios where computational resources are limited.

III. Background

This section briefly overviews the key tools, techniques, and algorithms employed in this research, which are integral to point cloud processing, analysis, and classification.

A. Software Libraries

1) ***ICP*** *(Iterative Closest Point):* ICP [**?**] is a major algorithm used to align two-point clouds. It iteratively minimizes the difference between the source and target point clouds by finding the optimal rotation and translation. The algorithm assumes that an initial rough alignment is available and progressively refines the pose by minimizing a distance metric (typically the Euclidean distance) between corresponding points. ICP is fundamental in applications like 3D registration, SLAM (Simultaneous Localization and Mapping), and scene reconstruction.

2) ***CloudCompare****:* CloudCompare [**?**] is an open-source 3D point cloud processing software known for its robust support for point cloud and mesh comparison, registration, and visualization. It provides various manual and automatic processing tools, including filtering, segmentation, distance computation, and statistical analysis. CloudCompare supports various file formats and includes plugins like CANUPO for advanced classification. It is a powerful visualization and preprocessing platform in many LiDAR and photogrammetry-based workflows.

3) ***Scalar Fields****:* Scalar fields in point cloud processing refer to assigning scalar values to individual points based on computed attributes such as curvature, density, height, or intensity. These values provide additional semantic information that can be used for segmentation, classification, or feature extraction. Visualization of scalar fields allows for straightforward interpretation of spatial patterns, while numerical thresholds help filter or cluster based on specific properties.

4) ***CANUPO*** *(Classifier of Advanced Numerics for Unstructured Pointclouds):* CANUPO [**?**] is a supervised machine learning plugin integrated into CloudCompare, designed specifically for classifying 3D point clouds. It leverages multi-scale dimensionality features, which describe the local geometric structure of a point neighborhood across different spatial scales. These features capture whether a group of points behaves more like a line (1D), a surface (2D), or a volume (3D) depending on the scale of observation. By analyzing these patterns at multiple scales, CANUPO [**?**]can effectively distinguish between natural elements like vegetation and terrain, and artificial structures such as buildings. Once trained on labeled data, the model can accurately classify new point cloud data, making it especially useful in complex natural environments like forests or mountainous areas.

5) ***CSF****(Cloth Simulation Filter:* The Cloth Simulation Filter [**?**] is a ground segmentation algorithm that simulates draping a virtual cloth over an inverted point cloud. As the cloth settles over the lowest points (representing the ground), it forms a surface separating ground and non-ground points. CSF is beneficial in outdoor and unstructured environments where traditional height-based filters fail due to terrain irregularities. It is effective for preprocessing in terrain modeling and object isolation tasks.

6) ***RANSAC*** *(Random Sample Consensus):* RANSAC is a robust model-fitting algorithm used to estimate the parameters of a mathematical model from a dataset containing outliers [**?**]. It works by repeatedly selecting random subsets of the data, fitting a model, and evaluating how many points fit this model within a predefined threshold. RANSAC is commonly used in plane fitting, line detection, and geometric shape recognition within point clouds, as it effectively handles noisy data and partial observations.

7) ***Open3D****:* Open3D [**?**] is an open-source library developed for 3D data processing, particularly focused on point cloud, mesh, and RGB-D image processing. It provides tools for I/O operations, visualization, filtering, transformation, feature extraction, and deep learning integration. Open3D supports modern algorithms such as ICP, RANSAC, and voxel-based downsampling and integrates well with machine learning frameworks like PyTorch. It is widely used in academic research and industry due to its flexibility, performance, and ease of use in Python and C++ environments.

8) ***PCL*** *(Point Cloud Library):* The Point Cloud Library [**?**] (PCL) is a large-scale, open-source project designed for 2D/3D image and point cloud processing. It offers a comprehensive collection of algorithms for filtering, feature estimation, surface reconstruction, registration, segmentation, and visualization of point clouds. PCL supports multiple file formats and is optimized for performance, making it suitable for real-time applications such as autonomous navigation, object detection, and environment reconstruction. Its modular architecture includes key components like the filters module for preprocessing, the segmentation module for geometry-based classification, and the registration module for aligning multiple scans. Integration with ROS (Robot Operating System) further extends its utility in robotics and perception systems.

B. Methods

1) ***PointNet****:* PointNet [**?**] is a pioneering deep learning architecture specifically designed for directly consuming raw 3D point cloud data without the need for voxelization or meshing. It introduces a symmetric function (e.g., max pooling) to handle the unordered nature of point clouds and ensure permutation invariance. The architecture consists of a shared multilayer perceptron (MLP) that learns point-wise features, followed by a global feature aggregation step that encodes the overall shape. PointNet is capable of performing classification and segmentation tasks, making it effective for tasks like object recognition and semantic labeling in 3D scenes. However, it primarily captures global features and lacks sensitivity to local

geometric structures, which limits its performance in fine-grained segmentation.

*2) **PointNet++**:* PointNet++ [**?**] extends the original PointNet by incorporating hierarchical feature learning to capture both local and global context. It partitions the point cloud into overlapping local regions using sampling and grouping techniques, and applies PointNet recursively within each region to learn localized features. This approach enables the model to handle varying point densities and complex geometric variations more effectively than its predecessor. PointNet++ supports tasks such as semantic segmentation and instance segmentation with higher spatial precision. It is particularly well-suited for natural environments like forests or urban scenes where structural complexity requires both global awareness and local discrimination.

IV. Experimental Approach

This section describes the experimental approach to developing and modeling a training pipeline for terrain segmentation from the 3-D point cloud LiDAR dataset. This approach consists of several steps, including data collection, preprocessing, evaluation, validation, and finally, the conclusion based on the experiments.

A. Tools and Environment Setup

The project environment was established using Anaconda to ensure reproducibility and consistency. A dedicated environment with all dependencies was maintained and shared via GitHub [**?**]. Core tools included CloudCompare [**?**] for 3D point cloud visualization and preprocessing, Open3D for Python-based processing, and MATLAB [**?**] for supplementary analysis. SciKit-learn supported clustering and data preprocessing, while the PCL (Point Cloud Library) [**?**] was employed for specialized point cloud operations during experimentation.

B. Data Pre-Processing

The initial dataset comprised 3D LiDAR point clouds of Field-D, captured by the Garrulus team's UAV in a local coordinate system, without georeferencing or RGB data. It included point clouds, polygon meshes, and VTK files. A 100×100×100 m Region of Interest (ROI) was selected using CloudCompare for focused analysis, and exported as a polygon mesh for better cross-platform compatibility.

The dataset contained several invalid points (NaN and infinite), which were retained but ignored during processing in Open3D to avoid loss of potentially useful information. Further cropping was done based on visible fencing boundaries, and the cleaned polygon mesh was finalized as a benchmark for subsequent processing and experiments.

C. Segmentation Clustering Techniques

To segment meaningful structures from the filtered point cloud data, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [**?**] algorithm was employed. DBSCAN is particularly well-suited for 3D point cloud segmentation due to its ability to identify arbitrarily shaped clusters and its robustness to noise. The input point cloud, stored in .ply format, was loaded using the Open3D library and converted into a NumPy array to facilitate clustering using the scikit-learn implementation of DBSCAN.

Clustering was performed using a rich set of features: in addition to the basic spatial coordinates (x, y, z), geometric descriptors such as surface normals (nx, ny, nz), curvature, and local point density were included. These additional features, which are critical for distinguishing between different natural elements like tree stumps, terrain, and undergrowth, added contextual information beyond mere location. This multi-dimensional feature space allowed DBSCAN to more accurately group points with similar geometric and structural characteristics.

The DBSCAN parameters were tuned to the nature of the dataset, with an epsilon [**?**] (EPS) value of 0.05 and a minimum samples threshold of 800, balancing sensitivity to dense clusters and resistance to outliers. Points were assigned cluster labels, with label -1 [**?**] indicating noise. Clusters were visualized using the jet colormap from Matplotlib, with each cluster assigned a unique color and noise points shown in black, providing an intuitive and informative 3D visualization using Open3D's interactive viewer.

This technique is especially beneficial in natural, unstructured environments—such as the forests familiar from UAV scans—where distinguishing fine features like tree stumps or ground depressions can be vital. Individual clusters, such as the largest one (often corresponding to terrain), could be isolated for deeper analysis, enabling a targeted examination of ecological or structural elements within the scene.

D. Challenges Faced

One of the significant challenges faced during this project was the absence of labeled data in the LiDAR point cloud. Manual annotation of the point cloud data became a significantly time-consuming task without prior labeling and no direct reference to photogrammetry or RGB values. Additionally, directly labeling the point cloud would have resulted in a compromise of precision and accuracy due to the complicatedness in visually distinguishing between objects using only geometric features.

Two auxiliary methods were introduced to address this issue. During data acquisition, the UAV system simultaneously captured both LiDAR and RGB photogrammetry data. The RGB data, being more intuitive for human interpretation, was manually annotated using QGIS. These annotations were then exported in the GeoPackage [**?**] (.gpkg) format, which allowed for high-resolution spatial referencing and integration with other datasets.

A secondary challenge involved the disparity in coordinate systems. The RGB photogrammetry data was geo-referenced, aligning with a global spatial reference system, whereas the LiDAR data was captured using a local coordinate system. However, due to the availability of metadata and known reference points, it was feasible to transform both datasets into a

unified, standardized coordinate system. This enabled accurate overlay and cross-referencing between the manually labeled photogrammetry data and the LiDAR point cloud, ultimately improving the efficiency and reliability of the labeling process.

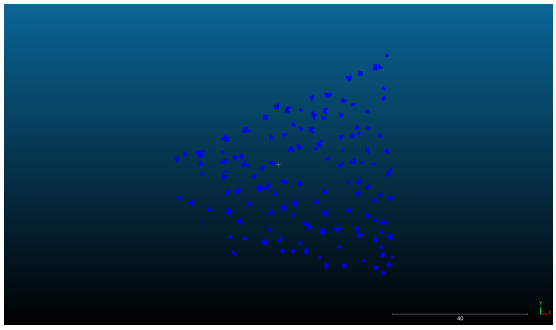

Fig. 2: Stumps

E. Data Preparation

Following the collection of the LiDAR datasets and their corresponding RGB photogrammetry data, along with the associated GeoPackage (.gpkg) annotations, the next critical step was extracting and aligning semantic labels. Although the GeoPackage format provided precise 2D polygonal annotations—particularly for features like tree stumps—aligning these 2D labels2 with the 3D LiDAR data posed a significant challenge. The process required accurately mapping flat polygonal data to volumetric point clouds, a task complicated by the lack of direct height information.

To bridge this dimensional gap, an approximate height of 0.3 meters was assumed for stump structures based on field observations and average stump dimensions. This approximation enabled the transformation of 2D annotations into 3D labeled regions, which were then converted into .ply format to ensure compatibility with LiDAR processing tools. Subsequent alignment of the converted 3D labels with the point cloud was performed using reference-based registration and Iterative Closest Point (ICP) algorithms available in CloudCompare. This step ensured spatial accuracy between labeled regions and the actual 3D geometry of the environment.

The labeled dataset was segmented into square grid tiles upon successful alignment to facilitate model training and evaluation. Each grid was assigned one of three class labels based on the contents of the points within it: 0 for terrain, 1 for stumps, and 2 for all other elements. Several permutations were tested to determine the optimal grid size. After extensive trials, a 7×7 meter grid size was selected as it provided a balanced distribution of points across categories while maintaining spatial resolution sufficient for effective classification and segmentation tasks.II

F. Model Experiments

The prepared dataset, originally in CSV format, contained key geometric and structural attributes, including the x, y, and z coordinates, surface normals, curvature values, and class labels. To facilitate deep learning experiments using PointNet and PointNet++ architectures, the dataset was converted into formats compatible with each model's input requirements.

For the classification task using PointNet, the dataset was transformed into NumPy (.npy) arrays. The data was then organized into labeled directories corresponding to each class. A stratified split was performed to divide the dataset into training and testing subsets, allocating 70% of the samples for training and the remaining 30% for testing. These structured inputs were then fed into the PointNet classification model to evaluate its ability to distinguish between terrain, stumps, and other features based on point cloud geometry.

In the case of PointNet++ for segmentation tasks, the dataset was formatted as 6×N matrices, where each column represented a single point consisting of x, y, and z coordinates referring to labels, curvature, and density. These were also stored in .npy format for compatibility with the model architecture. This configuration enabled the network to perform point-wise classification across the entire spatial domain, allowing fine-grained segmentation of the forest environment.

V. Evaluation

The figure1 represents a visualization of the original LiDAR point cloud data captured from a UAV-mounted sensor system over a forested area. The data is visualized using a color-coded scheme to represent different spatial distributions, where shades of blue and green indicate variations in surface elevation and structure density.

The dense central region of the point cloud signifies the forest floor and lower vegetation layers, while the vertically extended, less dense structures—predominantly colored green—correspond to taller vegetation such as trees. The presence of vertical lines and scattered points further suggests canopy-level vegetation and possibly sparse outliers from surrounding terrain features or minor alignment inaccuracies.

For the scope of this project, the upper vegetative regions, including the tall tree canopies and vertical outliers, were filtered out figure3 to reduce complexity and focus analysis on the forest floor and stump structures. This preprocessing step improved segmentation precision and ensured computational efficiency in the downstream tasks. The raw dataset, after filtering, served as the foundational input for subsequent preprocessing, alignment, labeling, and segmentation operations conducted throughout the study.

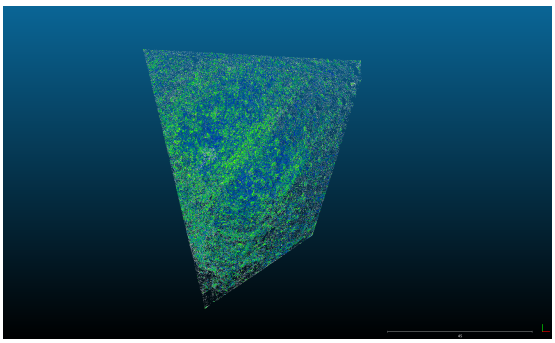

Fig. 3: Filtered Point Cloud after Cropping.

A. DBScan Trials

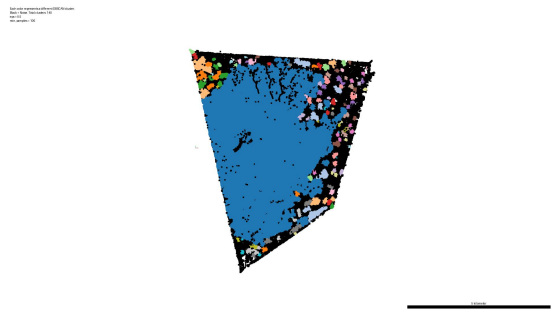

Fig. 4: DBScan Trial.

DBSCAN clustering was applied to the filtered point cloud to identify spatially coherent regions that may correspond to individual stumps or other ground-level structures. The algorithm was configured with an epsilon value of 0.5 and a minimum sample threshold of 100, chosen empirically to balance between noise reduction and meaningful cluster formation. These parameters allowed for the effective segmentation of densely packed structures while filtering out sparse and isolated points as noise. The resulting cluster labels were visualized using a colormap, facilitating an intuitive assessment of spatial distribution and clustering quality. To aid spatial orientation, a reference coordinate axis was included in the visualization, enhancing interpretability within the three-dimensional scene.

The clustering process resulted in 140 distinct clusters figure4, not accounting for noise. These clusters captured the fine-grained structural diversity of the forest floor, highlighting the natural complexity present in such environments. The segmentation provided by DBSCAN enabled the targeted extraction of relevant subregions, such as the largest or most densely populated clusters. This output was essential for downstream analytical tasks, including object-specific feature extraction, supervised classification, and point-wise segmentation. By isolating coherent spatial groups, the DBSCAN results significantly contributed to the robustness and interpretability of subsequent modeling stages.

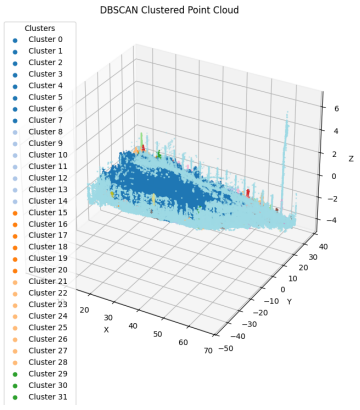

Fig. 5: DBScan Cylinder-like Clusters.

Following the initial DBSCAN clustering, a geometric model-based filtering approach was employed to further refine the segmentation results. A reference cylindrical model with a radius of 150 mm and a height of 150 mm—corresponding to typical stump dimensions—was defined as a geometric constraint figure5. Each identified cluster from the DBSCAN output was evaluated against this reference model to determine similarity based on shape and scale. Only those clusters that exhibited geometric compatibility with the reference cylinder were retained for further analysis.

This selective filtering process yielded a total of 140 clusters that conformed to the specified cylindrical profile. These clusters are considered potential stump candidates, exhibiting structural characteristics consistent with the expected dimensions. The application of this model-based filtering step significantly reduced false positives and contributed to a more precise identification of relevant ground-level structures, thereby enhancing the accuracy of subsequent classification and segmentation tasks.

B. RANSAC

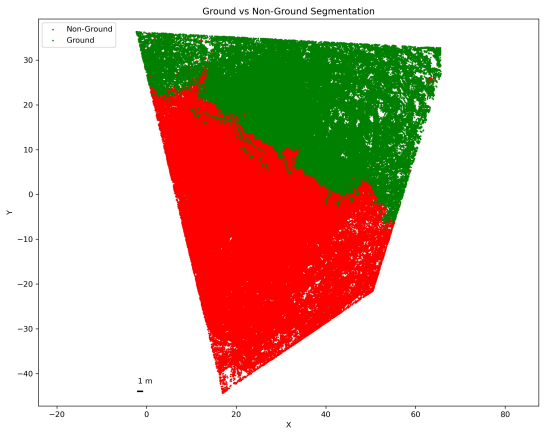

Fig. 6: RANSAC Result - Top View.

The segmentation of ground and non-ground points within the forest point cloud was conducted using two complementary methodologies: a supervised learning strategy employing a Random Forest Classifier and a model-based approach utilizing RANSAC plane fitting.

In the supervised classification method, a Random Forest algorithm was employed to distinguish between terrain and non-terrain features based on geometric attributes including spatial coordinates and curvature figure6. Ground-truth labels were derived using the 25% of total data points with lowest Z-axis value, providing a robust criterion for terrain estimation. Prior to training, the dataset underwent normalization to ensure consistent feature scaling. The classifier processed a total of 145,134 points, successfully identifying 36,413 as ground and 108,721 as non-ground. The segmentation results were visualized, with ground points represented in green and non-ground points in red, effectively delineating terrain surfaces from above-ground vegetation and structures.

Simultaneously, a model-based technique employing RANSAC (Random Sample Consensus) was implemented

to estimate the underlying ground surface. This algorithm iteratively fits a plane to the point cloud data by selecting subsets of points and evaluating inlier consensus within a defined distance threshold. Points lying within this threshold from the estimated plane were classified as ground (green), while remaining points were categorized as non-ground (red).

Both segmentation strategies yielded high-quality results that are visually interpretable and suitable for further analysis. The Random Forest-based approach provided reliable and quantitatively robust visual performance. In contrast, the RANSAC approach served as a geometric validation technique, offering a model-driven alternative that effectively captured the planar characteristics of the forest floor. Together, these methods present a comprehensive framework for terrain extraction from LiDAR-based point cloud data in forested environments.

C. PointNet Classification

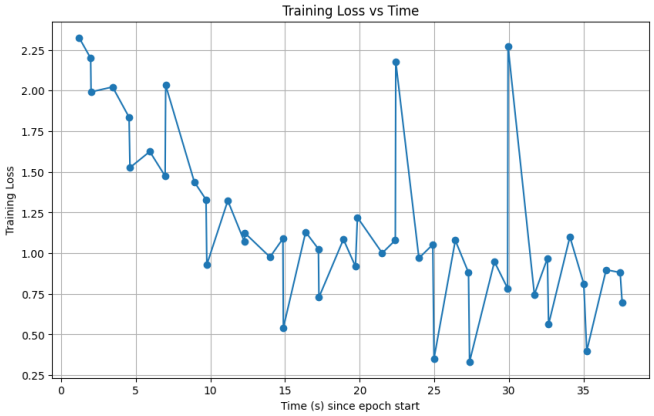

Fig. 7: PointNetClass output.

Test accuracy: 59.73%

In the classification task, the dataset utilized was in the .off file format, which encapsulates mesh information comprising vertices and faces for each sample representing classes such as terrain, stumps, and miscellaneous objects. While the inclusion of face data alongside vertices provides a richer geometric context, integrating face features directly into the PointNet [**?**] architecture introduced challenges. PointNet is inherently designed to process unordered point sets, and the structured nature of mesh faces conflicted with this requirement, leading to inconsistencies during training and a potential reduction in classification precision.

To address this, synthetic faces were generated based on the vertices alone, enabling a more uniform and consistent feature set across the dataset. This approach allowed the classifier to leverage structural information derived from the spatial configuration of points without introducing irregularities associated with original mesh face data. As a result, the training process was stabilized figure7 and model performance improved in terms of classification accuracy and generalization. This refinement enhanced the effectiveness of PointNet in distinguishing between different structural categories present in the forest environment.

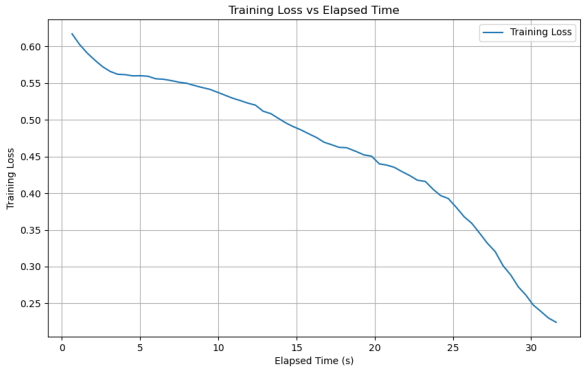

Fig. 8: PointNet++ Segmentation Batch Size - 128, Epochs - 64.

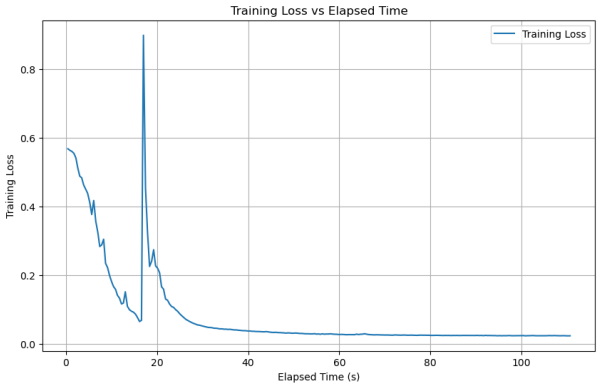

Fig. 9: PointNet++ Segmentation Batch Size - 128, Epochs - 251.

D. PointNet++ Segmentation

In the segmentation phase, the dataset was spatially divided into grid-based subregions to facilitate localized processing and annotation. Each grid segment was labeled into one of three predefined classes: terrain, stump, and non-terrain or everything else. Two distinct terrain extraction approaches were employed to generate ground truth for segmentation.

The first method utilized RANSAC-based plane fitting to identify terrain points. This approach estimated the ground

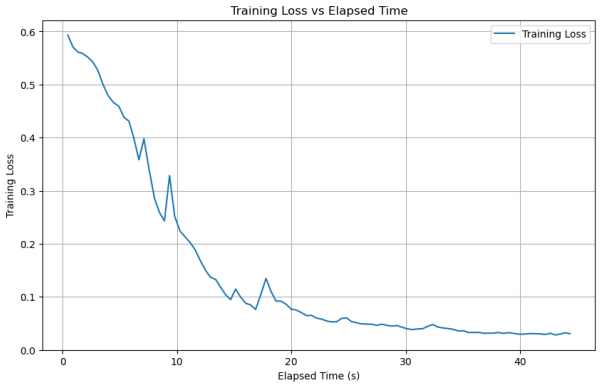

Fig. 10: PointNet++ Segmentation Batch Size - 128.

TABLE I: Test Accuracy of PointNet++ Segmentation Models

Trial	Batch Size	Epochs	Test Accuracy	Reference Graph
CSF	128	64	11.77%	Figure 8 CSF
128	251	34.14%	Figure 9	
CSF	128	100	36.15%	Figure 10
Grids	128	64	86.77%	Figure 11

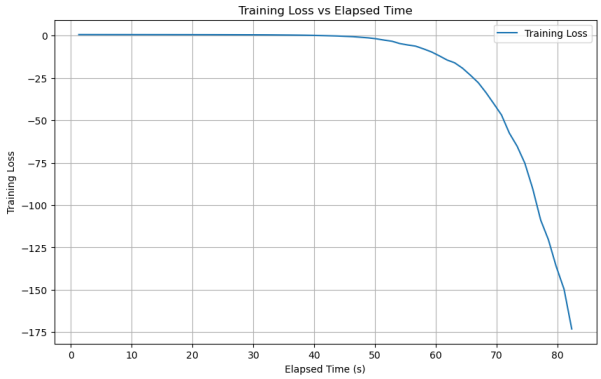

Fig. 11: PointNetSeg grids.

surface by fitting a planar model and labeling points within a certain proximity to this surface as terrain. All remaining points above the threshold were considered non-terrain. The second approach employed the CSF, a model inspired by cloth physics, to simulate a draped surface over the point cloud. Points below this simulated cloth were classified as terrain, while the rest were treated as non-terrain.

Following terrain extraction, the segmented point clouds were further refined by identifying stump-like structures from previously extracted .ply files. These were algorithmically labeled and assigned a class index of 1, while terrain points were labeled as 0 [**?**]. All remaining non-terrain points were grouped under the class label 2. This structured labeling enabled effective training and evaluation of segmentation models across diverse forest structures. figure:11

X (meter)	Y (meter)	Z (meter)	Label	Curvature	Density
16.3600	17.0471	-1.0025	1.0	0.0	0.0

TABLE II: Sample point cloud data with features

VI. Conclusions

The classification task using the PointNet model encountered significant limitations when trained on CSV file formats that included padded entries. The padding values, which were numerically represented as zero, conflicted with the actual class label for terrain, also encoded as zero. The masking technique was not applied beforehand so, this overlap introduced ambiguity during training, resulting in reduced model performance and learning instability. To address this, rather than applying masking, as it was better to have meshes and faces directly so, the dataset was reformatted into the .off file structure, which encapsulates both mesh and vertex information, thereby eliminating the need for zero-padding and improving the clarity and quality of input data. This adjustment facilitated more reliable model training and enhanced classification outcomes.

For the segmentation task, the PointNet++ architecture was employed; however, it yielded a suboptimal accuracy of 36.15%. This limited performance is likely attributed to the algorithmic approach adopted for dataset labeling. The labeling strategy, while efficient, may have introduced inconsistencies or inaccuracies, particularly in differentiating between complex forest elements such as stumps, terrain, and vegetation. These inaccuracies could have hindered the model's ability to generalize effectively across the various classes. The findings suggest that while deep learning-based segmentation has potential, its success is highly contingent upon the precision of input data and the reliability of the labeling process.

A. Future Work

To enhance the reliability and accuracy of segmentation results, future work should prioritize the manual labeling of the dataset to generate a fully validated ground truth. Although this process is time-consuming, it is feasible with the aid of reference data such as RGB imagery and photogrammetry outputs. Incorporating these supplementary data sources can significantly improve the precision of labels by providing contextual and visual information, which is otherwise absent in purely geometric datasets.

In addition, the MATLAB Deep Learning Toolbox presents a robust framework for tasks involving forest segmentation and can serve as a benchmark for evaluating the performance of alternative approaches. Its integrated workflows and support for a variety of deep learning architectures make it a valuable resource for both prototyping and large-scale experimentation.

Automation of the end-to-end pipeline also holds substantial promise for future development. Leveraging the Robot Operating System (ROS), it is possible to design nodes that automate data preprocessing, model training, and deployment using state-of-the-art algorithms. This would streamline the overall process, reduce manual intervention, and enable more consistent and scalable model training for real-time applications in forest environments.