applied sciences

MDPI

Article

Path Segmentation from Point Cloud Data for Autonomous Navigation

Krishnamoorthi Rajathi [1,*], Nandhagopal Gomathi [1], Miroslav Mahdal [2,*] and Radek Guras [2]

[1] Department of Computer Science & Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi 600 062, India; gomathin@veltech.edu.in
[2] Department of Control Systems and Instrumentation, Faculty of Mechanical Engineering, VSB-Technical University of Ostrava, 17. Listopadu 2172/15, 708 00 Ostrava, Czech Republic; radek.guras@vsb.cz
* Correspondence: rajathi@veltech.edu.in (K.R.); miroslav.mahdal@vsb.cz (M.M.)

Abstract: Autonomous vehicles require in-depth knowledge of their surroundings, making path segmentation and object detection crucial for determining the feasible region for path planning. Uniform characteristics of a road portion can be denoted by segmentations. Currently, road segmentation techniques mostly depend on the quality of camera images under different lighting conditions. However, Light Detection and Ranging (LiDAR) sensors can provide extremely precise 3D geometry information about the surroundings, leading to increased accuracy with increased memory consumption and computational overhead. This paper introduces a novel methodology which combines LiDAR and camera data for road detection, bridging the gap between 3D LiDAR Point Clouds (PCs). The assignment of semantic labels to 3D points is essential in various fields, including remote sensing, autonomous vehicles, and computer vision. This research discusses how to select the most relevant geometric features for path planning and improve autonomous navigation. An automatic framework for Semantic Segmentation (SS) is introduced, consisting of four processes: selecting neighborhoods, extracting classification features, and selecting features. The aim is to make the various components usable for end users without specialized knowledge by considering simplicity, effectiveness, and reproducibility. Through an extensive evaluation of different neighborhoods, geometric features, feature selection methods, classifiers, and benchmark datasets, the outcomes show that selecting the appropriate neighborhoods significantly develops 3D path segmentation. Additionally, selecting the right feature subsets can reduce computation time, memory usage, and enhance the quality of the results.

Keywords: autonomous vehicles; path segmentation; LiDAR sensors; 3D geometry information; semantic labels; feature selection; quality of results

check for updates

Citation: Rajathi, K.; Gomathi, N.; Mahdal, M.; Guras, R. Path Segmentation from Point Cloud Data for Autonomous Navigation. *Appl. Sci.* **2023**, *13*, 3977. https://doi.org/10.3390/app13063977

Academic Editor: Antonella Petrillo

Received: 23 February 2023
Revised: 16 March 2023
Accepted: 17 March 2023
Published: 21 March 2023

1. Introduction

One of the countries with the highest traffic in the world is India. In 2018, there were approximately 467,000 motor vehicle accidents in India, with 70% of them being attributed to human error. Autonomous vehicles, with their ability to sense their surroundings, can decrease the number of accidents on the road, making transportation safer and more convenient for seniors and others who do not have to constantly rely on drivers. Level 0 (No Automation) to Level 5 are the different levels of autonomy for vehicles (Full Automation). Level 0 describes the typical non-automated vehicle, while Level 1 and 2 are driver-assisted vehicles. Level 3 and 4 represent conditional and high automation, respectively, where the driver is only required in certain circumstances. Level 5 vehicles are completely autonomous, requiring no human intervention [1]. As the automotive industry continues to advance, intelligent automobiles are becoming increasingly automated, moving from driver-assisted vehicles to fully autonomous ones. These advancements bring incremental improvements, such as structures that help humans maintain constant speed, stay in their lane, and recognize obstacles.

As a result, one of the most significant challenges facing autonomous or self-driving vehicles is road detection. The study into self-driving cars heavily depends on road segmentation, which can be accomplished in several ways. Camera images, using RGB data, vanishing points, and edges, can perform road segmentation, but this method is susceptible to errors due to weather, light, and shadow conditions. LiDAR sensors can still detect road areas using 3D PC images from top, front, and distance views, but it cannot utilize color data. Hence, even a small height difference could result in inaccurate distinction between the road and the roadside. To overcome these limitations, the LiDAR and camera data fusion technique was developed by calibrating both the camera and LiDAR, and by displaying the LiDAR-detected cloud on the camera-captured image. This combination of exceptional data sets facilitates accurate road area division using LiDAR and camera information [2]. A comprehensive evaluation, using two standard benchmark datasets, provides insights into the suitability of various approaches.

Autonomous navigation in modern driving systems relies on sensors' abilities to detect and classify both fixed and moving objects, which in turn aids in forecasting the future state of the environment, preventing accidents, and organizing routes and schedules. By separating moving from static objects in 3D LiDAR PC data, moving object segmentation (MOS) techniques enhance environment awareness, localization, and future state prediction. While the network model is sophisticated and the MOS job is computationally demanding, it is crucial to fulfil the real-time processing needs of autonomous vehicle applications [3].

One of the areas of robotics that is expanding the quickest is automation. Giving robots the capacity to navigate on their own is crucial given the ever-increasing interest in automating procedures for both performance and privacy. For the autonomous navigation of mobile robots, a variety of sensor technologies, navigational system types, and data fusion techniques are available. The capacity to recognize things in their path and enable them to move about freely with a minimal issue of collision is a crucial feature of mobile autonomous robots. Other vehicles (such as forklifts and truck trailers) may not be spotted, even when using laser scanners designed for outdoor usage, potentially resulting in crashes. Under these challenging circumstances, 3D PCs recorded using a depth camera might be utilized to lessen this issue. In a brief streaming sequence, 3D coordinates and reflection intensities may be produced from more than 10,000 points of data using 3D PCs. PCs have clear 3D spatial structures that also contain reflection elements on the surface of objects, unlike RGB pictures, which are their main benefits [4].

We have developed a framework that can be used universally to interpret 3D PC data captured from different sources, including airborne laser scanning (ALS), mobile laser scanning (MLS), and terrestrial laser scanning (TLS). This is because it only relies on the spatial 3D geometry, represented as appropriate object surface measurements, as input from the real world.

Some applications, such as smart cities, agriculture, autonomous vehicles (AVs), and mobile rover navigation, require a framework to tackle environmental obstacles. Examples of automatic systems that observe the environment include reverse engineering, AVs and robots, modeling, concurrent positioning, and navigation vision. In automated systems, sensors play a crucial role, and LiDAR is widely used in research projects for remote sensing of the earth's surface. This active sensor has improved object detection accuracy and provides a real-time, error-free environment, resulting in the generation of a local map of the area.

Despite the widespread use of camera sensors in surveillance cameras, traffic signals, and similar devices, they are not appropriate for applications that require distance and lighting-based measurements, and their accuracy is limited. In the medical field, robots utilize 3D scanners for body scans, which require a high resolution. An accurate laser scanner, based on the triangulation principle, is used for this purpose. To prevent harm to human eyes during face scanning, these precise scanners use laser emitters classified as Class 1 or Class 1M. The motivation of this study is to make the different components useable for end users who do not have specialist expertise by taking into consideration their ease of use, their efficiency, and their ability to be reproduced.

The paper's primary contributions include,

- The creation of a connection among 3D LiDAR PCs and cameras data for road recognition by using a unique approach.
- The enhancement of autonomous navigation by choosing the most important geometric elements for route planning.
- The use of four steps to pick neighborhoods, obtain classifying characteristics, and select features for an automated architecture in SS.

The other phases of the studies are as follows: Section 2 offers a related review; Section 3 offers a proposed technique; Section 4 offers findings and perspectives; and Section 5 offers the conclusions.

2. Literature Survey

A method for detecting pedestrians has been proposed that utilizes a Bayesian policy that integrates semantic information with data from both the camera and LiDAR for 3D terrestrial data. It employs heuristics approaches to assign probabilities to each of the semantic classes included in the labelled pictures in the first phase of the four-step process of transferring semantic information from the labelled image to the LiDAR point cloud. Finally, taking into account the change in timestamps between each LiDAR scanning and camera picture, the LiDAR points are adjusted to account for the motion of the vehicle. The pixel coordinate for the accompanying camera picture is determined in a third stage. In the last stage, we remove the LiDAR data that are hidden from view by the camera while transferring semantic data from the heuristics likelihood pictures to the LiDAR frames [5]. This method employs classification and segmentation techniques.

The new segment data are trained using semantic class objects and a LiDAR test dataset from a hierarchical segment learning scheme. The final product is the creation of an SS class label for each point, which eliminates the main physical elements of the environment and calculates the residual point using a feature vector. Feature vectors are used to identify regions, classify information from SS with reduced data, and improve contrast to aid in object identification in a 3D position tracing scheme for Wireless Sensor Networks (WSNs) [6].

PC data are created using images in the form of spheres and important network characteristics [7]. The authors have presented a new structure for PC named pointSeg, consisting of three layers. The first layer emphasizes on creating a lightweight layer that resembles Alexnet. The second layer, called Squeeze reweight layer, generates the feature representation vector. Third layer, referred to as the extension layer, employs the pooling layer to identify the background data. The RANSAC method is used to improve the results, but the proposed scheme does not deliver the expected outcomes for low-level objects [8].

The Range-Image (RI) UNet is used to perform SS on 3D PC data using quasi-uniform angular strides. This technique transforms the raw 3D PC data into a compact image known as a range image [9]. The UNet framework starts with three-to-three convolutional operations, followed by two-to-two max-pooling and down-sampling of attributes. The latter half of the framework consists of both one-to-one convolutional and two-to-two up-convolutional operations. The loss function of the system is determined using cross-entropy, which is a simple and fast method compared to other schemes [10].

Another researcher proposed using the Gaussian Mixture Model (GMM) to create a collection of vectors known as the Fisher Vector representation. The improved Fisher Vector model is stable, but it may not always yield better results. The learning rate can be increased by using the PC scheme's part segmentation in the DmFV framework for similar objects, thus reducing misclassification rates.

The loss function of a network can be calculated using cross entropy. It has been found that the RI UNet is relatively simple and efficient when compared to other methods. To further improve performance, the focal loss function is employed. The Fisher Vector representation is dependent on a set of vectors associated with the GMM. Although the improved Fisher Vector model remains unchanged, it does not perform well enough to provide a significant advantage. The learning process is occurring at a faster rate [11]. The DmFV framework handles the part segmentation of the per PC method. Similar items are subject to misclassification. Table 1 illustrates the various architectures used in path planning for autonomous navigation.

Table 1. Architectures used in path planning.

Technique	Architecture	Work Proposed
Cityscapes test	HRNetV2 + OCR [12]	SS using Object-Contextual Illustrations
PASCAL VOC	DeepLabv3+ [13]	Semantic Image Segmentation using Encoder-Decoder with Atrous divisible convolution
PASCAL Context	OCR (HRNetV2-W48) [12]	SS using Object-Contextual Illustrations
ADE20K val	ACNet (ResNet-101) [14]	Scene Parsing using adaptive context network
ScanNet	MinkowskiNet [15]	4D spatio-temporal ConvNets: Minkowski Convolutional Neural Network (CNN)
Semantic3D	RandLA-Net [16]	Large-Scale PCs based effective SS using RandLA-Net
ADE20K	LaU-regression-loss [12]	SS using Positon-based upsampling
LIP val	SCHP (ResNet-101) [12]	Human parsing using self-improvement
Cityscapes val	HRNetV2 (HRNetV2-W48) [17]	Visual acknowledgment using deep high-resolution illustration learning
CamVid	DeepLabV3Plus + SDCNetAug [18]	Video propagation and label reduction based improved SS
COCO-Stuff test	OCR (HRNetV2-W48) [19]	SS using object-contextual Illustrations
PASCAL VOC 2012 val	ExFuse (ResNeXt-131) [20]	SS using ExFuse, an improved feature fusion
Freiburg Forest	SSMA [21]	Multi-modal SS using self-supervised model
ScanNetV2	SSMA [22]	Multi-modal SS using self-supervised Model
SUN-RGBD	SSMA [23]	Multi-modal SS using self-supervised Model
SYNTHIA- CVPR'16	SSMA [24]	Multi-modal SS using self-supervised Model
ShapeNet	SGPN [25]	3D PC Instance segmentation using Similarity Group Proposal Network (SGPN)
KITTI SS	DeepLabV3Plus + SDCNetAug [26]	Video propagation and label reduction based improved SS
NYU Depth v2	Dilated FCN-2s RGB [27]	SS using effective Deep CNN

Our findings indicate that the alignment of the LiDAR sensor is consistent with that of the camera, GPS, and IMU sensors as observed in prior studies. Our aim was to prove that the LiDAR sensor can be used alone for environment reconstruction. To achieve accurate results in our work, we are utilizing state-of-the-art techniques within the suggested framework. To enhance the performance, we introduce a pre-learning module that allows the framework to learn prior information about the world, which is referred to as prior knowledge (e) by applying basic division and ordering calculations. The data used in our approach will be based on prior experience. A new feature with labels known, desired, and unknown for both structured and unstructured environments has been added to this experience. In the study [28], they create and evaluate methods for autonomous ground vehicle off-road navigating using the simulation design and annotated LiDAR data. For off-road areas including paths and trees, the Mississippi State University Autonomous Vehicle Simulator (MAVS) purpose of advancing 3D point clouds. In the paper [29], the authors suggest an OpenStreetMap (OSM)-based autonomous robotic technique that combines road system data with local perception information in order to address the issue that the map inaccuracy OSM would result in the global route being incompatible with the actual environment.

Problem Statement

The majority of high-mix, low-volume manufacturing operations are still carried out by hand, which is expensive and ineffective. Moreover, LiDAR PC could not be reliable or accessible to support production. The user may utilize the scanner to obtain an accurate representation of the sensor as scanning technology develops. If LiDAR and camera data for road detection is developed directly on the scan data before being translated to cloud scripts for execution, the production process may be greatly streamlined. In conclusion, this work proposes unique scanned data that utilize geometric aspects for route planning and enhancing autonomous navigation. This technique may aid manufacturers in automating their production processes and enhancing their ability to respond quickly to individual client demands.

3. Proposed Work

The framework of the suggested technique is depicted in Figure 1.

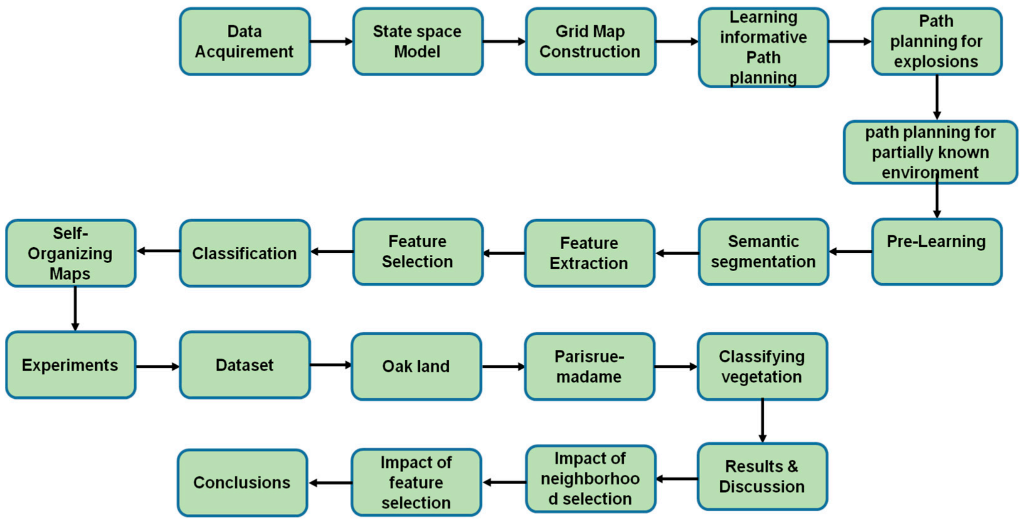

Figure 1. Proposed structure.

3.1. Data Acquirement

Since LiDAR sensor can identify items in its immediate area, self-driving cars are able to make thoughtful judgements. It is the key element needed to make self-driving cars an actuality and may be seen as the "set of eyes" for a car. The most expensive sensor available is Velodyne's Puck lidar sensor (formerly known as the VLP-16). The PuckTM is the best choice for economical low-speed autonomous and driving assistance systems owing to its dependability, power economy, and surrounding vision capabilities, in addition to its 100 m-range and small form factor. Raw data are collected using the VLP16 PUCK sensor. Table 2 lists the characteristics and specifics of the LiDAR sensor. For the purpose of this research, a smaller section of the university lane has been taken and stored in the Packet Capture (PCAP) format. Wire Shark is commonly used as a packet analyzer to open the PCAP file and examine the frames. The frames contain information such as the X, Y, and Z coordinates intensity, timestamp, and the start and end of each frame. MATLAB R2019b is utilized to process these frames. The Velodyne LiDAR object includes the model, timeout for the port terminal, calibration file, IP address, number of available PCs, and streaming capabilities. All simulation were carried out on Windows 10 system with Intel® Core™ i7 CPU, 16 GB RAM, and NVIDIA graphics card.

Table 2. LiDAR Configuration.

Features	Values
Make	Velodyne LiDAR
Model	Puck LITE
Weight	590 g
Output	Dual reoccurrence and 16 Channels
Range	100 M
Capacity	Up to ~600,000 points/s
FOV	360° Horizontal, ±15° Vertical

3.2. State Space Model

The environmental condition cannot be directly measured. Through a set of observables, the state measures the effect on the outside world. The state space model, which consists of estimating and measuring the environment, describes the character of a dynamic system. Equation (1) depicts the estimation of the environment's state.

$$X_{n+1} = A_n(x_{n+1,}W_n) \tag{1}$$

where

- X_n denotes the current PC data from the environment;
- X_{n+1} denotes the subsequent PC data from the environment;
- W_n is the noise value in PC like NaN;
- A_n is the estimation function.

Measurement of environment state is represented as Equation (2),

$$Y_n = b_n(x_n,V_n) \tag{2}$$

where

- Y_n the set of observable form LiDAR;
- b_n is measurement function between points;
- V_n is the measured noise.

Let us assume the initial state X_o is uncorrelated with the dynamic noise W, and the two sources of noise (W_n, V_n) are statically independent. The Extended Kalman Filter (EKF) accommodate linear model. The main advantage of EKF is the small number of parameters it can object.

In this work, the environment is captured using a LiDAR sensor, and the PC data are generated from it. To model the state of the LiDAR PC data, the EKF algorithm is employed. The EKF for LiDAR PC data is discussed in Algorithm 1. Three different states are identified: Free State, Occupied State, and Unknown State. An octree data design is utilized to store the sensor information and is effective for representing unknown features. The features are then extracted and stored in a voxel representation. The EKF algorithm is applied in both positioning systems and perception systems.

Algorithm 1: EKF for LiDAR PC Data

```
Input: observations $\{y_1, y_2 \ldots, y_n\}$
Output: State estimation and state prediction using PC data
1.  The linearized matrix $A_{n+1}, n$ and $B_n$ are computed from the nonlinear parameter
    $a_n(x)$ and $b_n(x)$.
2.  The values $a_n(x_{n/m})$ and $b_n(x_{n/m})$ are obtained by substituting the filter state
    estimate and predict the statement estimate.
3.  Examine the order of iteration.
```

3.3. Grid Map Construction

An advanced motion planning system has been developed for autonomous vehicles, allowing them to navigate through dynamic and changing environments. The procedure of searching a secure and collision-free path from an initial location to a final destination is referred to as motion planning for autonomous vehicles. The path-planning problem is solved using a Non-Parametric (NP) method. The grid map of the sensed environment is generated using the Bayesian filtering method. Equation (3) represents the posterior function of the Bayesian filter, which can be calculated using Equations (1) and (2).

$$\mathrm{P}(m/z_{1:t}) = \frac{p(\mathrm{m}/\mathrm{z}_{1:\mathrm{t}-1})\mathrm{p}(Z_t/m)}{P(Z_t/Z_{1:t-1})} \tag{3}$$

where

- $\mathrm{P}(m/z_{1:t})$ is the Posterior Points Occupy;
- $p(\mathrm{m}/\mathrm{z}_{1:\mathrm{t}-1})$ is prior probability and $\frac{\mathrm{p}(Z_t/m)}{P(Z_t/Z_{1:t-1})}$ is likelihood.

The log odds for each side of equation are found after simplifying Equation (3) by assuming a uniform prior distribution threshold of 0.5 for the unknown state.

$$\mathrm{L}(m/z_{1:t}) = L(\mathrm{m}/\mathrm{z}_{1:\mathrm{t}-1}) + \mathrm{L}(Z_t/m) \tag{4}$$

Equation (4) is recursive, and $\mathrm{L}(m/z_{1:t})$ is defined as the inverse of the sensor model. Thus, the grid map was incrementally updated while reading the new sensor data. The size of the voxel representation is assumed to be a × a × a, the normal distribution points are defined as N_{ijk} in Equation (5), and x, y, z is calculated with respect to i, j, k.

$$N_{ijk}^{f} = N\left(\frac{a}{d}\left(\frac{1}{\theta_v} + \frac{1}{\theta_h}\right), a^2\right. \tag{5}$$

where

- d is Sensor distance from the grid;
- θ_v is vertical resolution of 3D LiDAR;
- θ_h is horizontal resolution of 3D LiDAR.

Hence, the probability of occupancy for each grid is:

$$P(n) = \frac{N_{ijk}}{N_{ijk}^{f}} \tag{6}$$

Because each state in the grid space is either "Free", "Occupied", or "Unknown", the grid state is defined as "F, O, U". The minor object in the unknown state may not be detected by the sensor due to its small size. The "Free" or "Occupied" or "Unknown" or "Conflict" elements of sensor data in grid space are represented by the set F, O, U, C. Conflict refers to data that are redundant, such as an image or some noise. The following mass functions are shown for the grid space: $m(F)$, $m(O)$, $m(U)$, $m(C)$. All states have masses that satisfy $m = 1$. The occupancy probability which is in Equation 6 shows grid timing and the probability of occupancy for each grid, as well as the data fusion formula derived to satisfy the Dezert Smarandache Theory (DSmT).

3.4. Learning Informative Path Planning

The researchers have introduced a new algorithm, named Learning Informative Path Planning (Algorithm 2), to optimize a path in a continuous space with certain constraints that pertain to path budgets and the motion capabilities of a robot designed for plant phenotyping. The utility of the location set A is assigned by the sensing quality function F(A).

Given: finite set V of locations

Want: $A^* \subseteq V$ such that

$$A^* = argmaxF(A)$$

$$|A| \le k$$

We now describe LIPP (Learning Informative Path Planning algorithm) follows recursive Bayesian for updating and replanning the path periodically. An upper bound of B time steps on the path cost and specified starting and ending locations (s_1) are used to initialize the algorithm. The utility function which represents the "conditional utility" conditioned on observation Y_s applies a nonadaptive algorithm in the first time step. After that, the algorithm moves the robot to the next location on the chosen nonadaptive path and iteratively uses update in subsequent time steps with an updated starting location of s_1 and a budget of B_1, while maintaining the finishing location at sB. Selecting the sensing location will be computed by,

$$S^* := argmaxF(A \cup \{S\})A := A \cup \{S^*\}$$

Algorithm 2: Learning Informative Path Planning algorithm

```
Input: $s_L, s_u, u, B$
Output: Sequence of selected locations $\pi(xV)$
Begin
        $s \leftarrow s_1; \pi[1] = s; Bpp \leftarrow B; obs \leftarrow \{\};$
    for $1 \le t \le B$ do
        $y_s \leftarrow observe(s); obs \leftarrow obs \cup \{y_s = y_s\};$
        $P_t \leftarrow update(s, sB, Bpp, F(|obs));$
        $s \leftarrow P_t[2]; \pi[t+1] \leftarrow s; Bpp \leftarrow Bpp$ –1
    return $\pi$
end
```

Latombe's path planning techniques [30] and earlier work assumed that the world was fully known in advance. However, particularly in outdoor settings, planning is often initiated with information that is incorrect, incomplete, lacking sufficient resolution, or has changed since it was obtained. This type of planning is based on the "free space assumption", which states that until it is proven otherwise, the unknown world can be completely traversed. To allow replanning based on sensor data gathered during the mission, the algorithms must be modified. This refers not to replanning around small obstacles, which is the responsibility of the reactive local navigator, but to replanning to obtain information critical to the robot's ability to achieve its goal according to the previous plan. The common approach is as follows: the system creates a global path using the information available and uses obstacle avoidance to navigate around obstacles. A replan is necessary if the route is completely blocked or if crucial information has been uncovered. Planning and execution must go hand in hand. Replanning methods must be fast because the robot cannot move toward its goal while replanning. Their design prioritizes speed of operation, sometimes at the cost of optimality and completeness.

3.4.1. Path Planning for Exploring

An unmanned ground vehicle (UGV) can be beneficial for exploring and creating maps. Thanks to various algorithms, robots are able to explore uncharted territories. By computing the optimal route until a map discrepancy is found, updating the map, and then replanning the entire path, they can exhibit optimal behavior. However, this can be challenging on large maps, where replanning can be inefficient, especially if the map contains limited information and requires frequent replanning. To achieve speed, some algorithms prioritize coverage over optimality. The robot is guided [31] until it reaches its objective. To prevent repeated visits to the same location, the robot moves to the next location with the lowest cost, and the cost of that location is increased each time. Author [32] calculates the cost of reaching the goal for each state based on the initial data, and updates it with backtracking costs as the robot progresses.

3.4.2. Path Planning for Partially Known Environments

The goal is to provide more goal-directed behavior than can be achieved with exploration algorithms when working with partial maps, while still prioritizing planning speed. When there are no obstacles, Lumelsky's bug algorithms [33] steer the robot toward the objective. If an obstacle is encountered, it moves around it until it reaches the point closest to the goal, and then moves directly toward the goal. In the past, when incorrect information was discovered, some early approaches to planning in partially known environments required a complete replan [34]. However, since then, more efficient methods have been developed to expedite the replanning process. These typically involve refined graph search algorithms, such as those mentioned earlier. Using quad-trees [35] to indicate traversability improves efficiency and speeds up the search by adjusting the heuristic using A* search and past terrain navigation experience. There are also additional methods for accelerating the search [36].

Some fundamental tools that speed up replanning include:

- Heuristics.
- Using limited look-ahead to limit the planning area (Real-time Heuristic Search).
- Replacing only the altered portion of the path (incremental search).
- Only changing the portion of the path that is necessary to reach the objective (incremental search).

In this research, a LiDAR sensor is combined with a camera, and the current study shows that no additional sensors are necessary in a LiDAR sensor-based environment. Well-established benchmarked methods are used to obtain accurate results. The current approach uses supervised data to handle image-based data, while unsupervised algorithms are being developed and researched to address the shortage of image-based data. Pretraining is suggested for SS as it provides the system with experience and prior knowledge of the real world. Figure 2 shows the PC data captured by the LiDAR sensor and the unstructured PC data. A PC is a set of data points in three-dimensional space that represent the shape and form of an object or environment. These data points are typically collected using 3D laser scanners or other surveying instruments, and they can be utilized to develop detailed 3D designs of objects and environments.

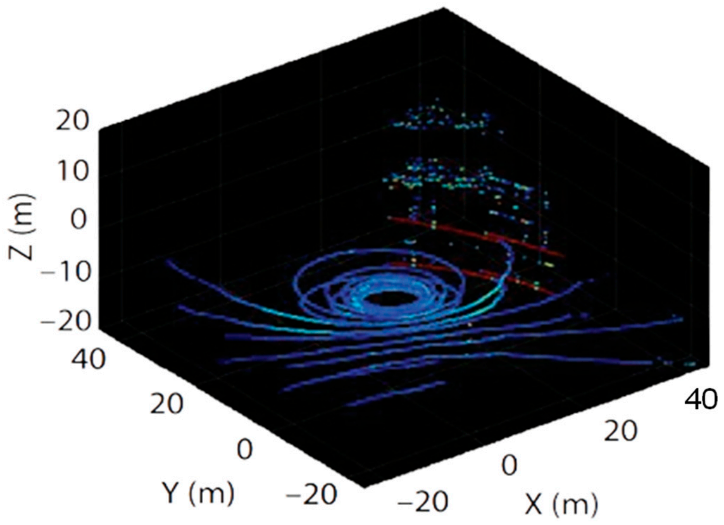

Figure 2. LiDAR PC data from PCAP file.

3.5. Pre-Learning

The pre-learning stage is critical for setting up the LiDAR sensor in a partially unknown environment. It focuses on using raw LiDAR data with mono-model sensors. To start, the Velodyne-FileReader function is used with parameters that take into account the file and device names. This creates a Velodyne PC object from the disorganized PC data. The input and ground regions of the PC are then segmented and avoided using this object. The distance threshold is set at 0.5. The ground plans are then labeled. The output is the number of clusters and their labels, which are obtained using M $\times$ N integer matrices. Every point within a cluster is assigned a label that falls within a range of 1 to the total number of clusters. Points that are deemed unacceptable, such as those labeled as "Not a Number" (NaN) or "inf", are allocated a label of "0" based on the IEEE754 2008 standard for maintaining floating point values. The count of valid clusters can be represented by looking at the total number of positive labels assigned to the points. Finally, the PC data are segmented based on color and prepared for use. Image labelling that focuses on recognizing and labelling certain aspects in a picture is known as image labelling. Pre-learning applications for categorizing three-dimensional point clouds are becoming more popular as a result of it.

The choice of either a single scale or many scales can be the primary consideration when choosing a neighborhood. By postponing the decision on the appropriateness of a neighborhood for the predictor, one can assess its characteristics across various scales. However, numerous additional factors must also be considered, such as the architecture of the scale space, the number of scales involved, and the technique used to calculate the distance between scales. These decisions, which are often based on heuristics or empirical selection [37], are unique to each dataset. In the context of the preceding to characterize the local 3D design. Finding the 3D points in the vicinity of the specified 3D point $X = (x, y, z)^T$ belonging to R is the primary challenge. This necessitates (i) a suitable definition of the neighbourhood, (ii) an efficient method for recovering nearby 3D locations, and (iii) a sufficient parameterization based on the neighbourhood's size. We give these three issues careful consideration.

The properties of the relevant PC data will undoubtedly affect the choice of the appropriate neighborhood definition and scale parameter. In comparison to the cylindrical as well as spherical definitions of neighborhoods N_s and N_C, which, as discussed in Section 2, require direct empirical or heuristic knowledge of the scene to determine an appropriate radius, N_k, the neighbourhood description, allows for greater flexibility in the face of changing point densities. We chose the N_k definition, which is dependent on the KNN of a provided 3D point X, as it provides a flexible framework that is not limited to a specific dataset. We can generate a spherical neighborhood with a radius that can vary by selecting the nearest neighbors dependent on their 3D distances [38].

When choosing a suitable number of neighbors k to consider, the simplest method would be to choose a fixed value of k for each point in the PC [39]. In this sense, the existing heuristic or empirical knowledge still plays a role in the selection of k. However, it is intuitive that a more adaptive and dynamic approach, where the variables k (also referred to as the scale) can differ in the dataset, would be preferable. This is because k is clearly influenced by the 3D structures and local point density in the dataset. Therefore, it would be ideal to use a standardized approach to determine the optimal local neighborhoods. The Pre-learning algorithm is discussed in Algorithm 3.

The proposed plan offers improved results, including increased space, speed, and average space efficiency in LiDAR data. The pre-learning process is outlined in Figure 3, and the framework is designed to work in both structured and unstructured environments. The next crucial step in the pre-learning process is to find an open ROI. Ground truth applications provide a more sophisticated approach to identifying rectangular ROIs.

Algorithm 3: Pre-learning.

```
Input: LiDAR PC raw data
Output: Labelled Point-wise segmented 2D
1.  Capture the surroundings
2.  Alter the PCAP file into PC object
3.  Eliminate ground segment from PC data
4.  Eliminate the points that are not valid from PC
5.  Divide image frames and follow steps for frames numbered from 1 to n
    a.  Initialize the point weights
    b.  Create ground truth vector as input for the building
    c.  Examine the point to find the weights that are similar to the input vector.
    d.  Compute the neighborhood similarity (The quantity of neighbors lessens with time)
    e.  Reward the winning weight by resembling the sample vector (neighbors
        resemble the sample vector)
    f.  Repeat the step b for N times
End for
6.  Allocate Color Index (CI) weight to every clustered point
7.  Perform PC registration for successive frames
```

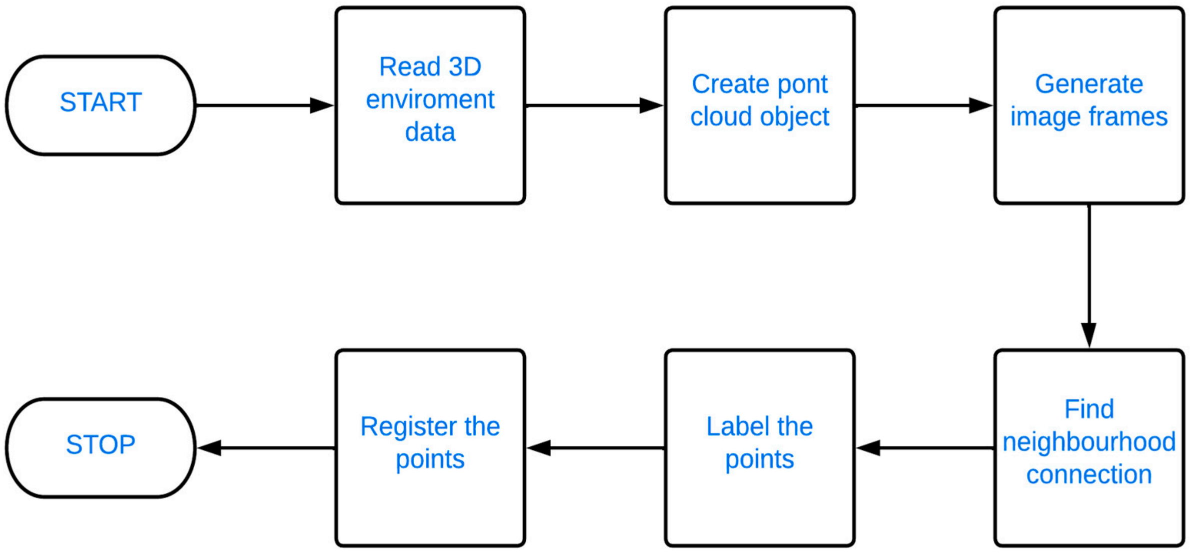

Figure 3. Workflow of Pre-Learning.

In this work, physical ground truth values are employed. The unlabeled data are first introduced into the workspace, then labels are assigned to the designated objects and moved to the workspace. Label assignment is performed based on point similarity using the Color Index (CI). The CI is an RGB-based vector with values ranging from 0 to 255. These elements are deemed crucial as LiDAR sensor manufacturers assign labels, which are a collection of CI-related values. Using a bounding box that represents an object from an unstructured smart environment, these inputs are used to assign labels. The label "1 to N" is assigned with a CI value of "255, 255, 255". The PC's image frames are read to register the arrangement of images, and the Normal Distributions Transform (NDT) scheme is used for image registration. The successive image frames of the PC are combined, resulting in a two-dimensional mono-model with labels. The coordinate points and labels are included as features in the vector representation.

4. SS System

Segmentation involves dividing the points of a PC into small, understandable, and connected subsets. It is a divisive approach. The points are then labeled based on their similarity, and each subset of points has distinctive characteristics. The PC process involves three pipelined processes. Figure 4 shows the proposed architecture of the SS system. Understanding semantics is a crucial aspect in autonomous cars and reverse engineering and is a part of the positioning and vision-dependent navigation model for independent robotic systems. Segmentations are of different types. The local segmentation of PCs categorizes data points based on geometric factors in cluster-based schemes.

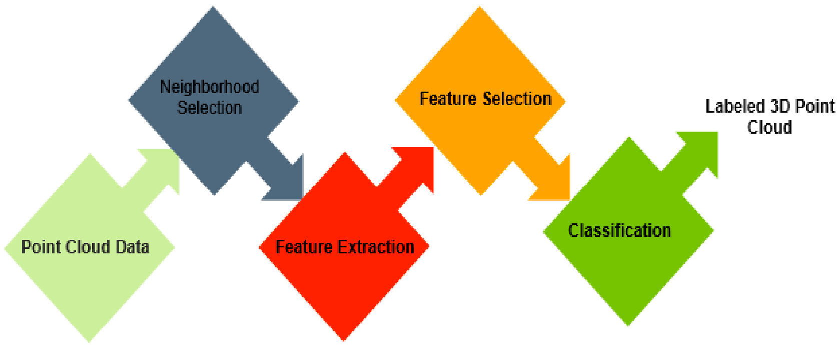

Figure 4. SS Process flow.

Graph-dependent segmentation schemes are motivated by the neighborhood graph structure. However, these schemes have limitations such as difficulties in identifying borders and interpolation in the case of similar closed objects. To address these problems, the researchers have introduced a local descriptor that can be used to extract the relevant features and information from the dataset. The Fast Point Fast Histogram (FPFH) [40] scheme is utilized to draw out local features.

4.1. Feature Extraction

There have been various proposals on how to extract discriminative features from PCs that have varying densities of points [41]. Spin image descriptors are formed by rotating a, and can be used to characterize the local 3D area surrounding a point X. An effective alternative to spin image descriptors is the use of shape distributions, which involve randomly sampling geometric region properties such as intensity, color, texture, contours, regions, angles, and lengths. To determine the parameters of a color transformation that best separates the two groups, color data from the vehicle and nonroad zones are employed. After that, a line fitting method is used to express the contour as a series of parallel lines. Calculating the distances from the vehicle at each place x_i, where the range is $d_{i,j} = \| x_i \|$ for point i^{th} allows us to describe the texture of an area. To eliminate the mean and magnify the noise, the distance data are next treated by obtaining the exact number of the first derivatives of the data. As a result, they are suitable for describing the immediate vicinity of a 3D point X [42]. Point Feature Histograms (PFHs) is another method that follows a similar approach. These relations are then encoded into histograms to construct the PFHs [43]. The "Signature of Histograms of Orientations (SHOT)" is another type of descriptor that is dependent on a spherical grid centered at point X, which is used to construct the descriptor. However, it is rare to understand a single entry in the resulting feature vector for any of these approaches.

4.2. Feature Selection

To handle a large number of features or training cases, it is often desirable to use simpler and more efficient procedures. Therefore, filter-based approaches are commonly employed to choose a subset of related features [44]. These filter-based techniques focus on relationships between features and classes that are clearer and easier to understand, as well as possible relationships within features (such as relationships depending on better known concepts such as distance, consistency, dependency, or information), while embedded approaches use more complicated relationships that are harder to understand [45]. Because each score function focuses on reducing the classification error, it could be argued that embedding methods are more appropriate for statistical learning or machine learning, as they present these interactions as score functions. Further, embedded methodologies would immediately debrief a dependence between the settings of a classifier and selected features, such as the number and type of weak learners used and the number of (preferably high) choices taken into account for each variable. To avoid extensive classifier adjustment and maintain their applicability for non-expert users, we concentrate on filter-based approaches and are willing to agree that these approaches generally produce slightly lower performance. We will briefly discuss the fundamental ideas behind filter-based techniques for both univariate and multivariate analysis in the following sections. It is significant to specify that when using these methods, the data should be trained with an equal number of training samples for each class to avoid bias in feature selection.

4.3. Classification

Most approaches tend to concentrate on supervised classification schemes for various applications. The basic idea is to train a classifier with previously provided training data so that it can adapt to unseen data. As a result, the testing data are typically depicted by a set of testing samples, X, where each sample contains a class label and a feature vector in a d-dimensional feature space. On the other hand, the test set, Y, typically consists of new data that need to be classified and feature vectors in the d-dimensional feature space. We focus on individual point categorization due to its simplicity, efficiency, and reproducibility, as well as the availability of numerous suitable algorithms in different software packages. In order to determine the relationship between feature vectors and class labels during the training phase, multiple learning principles may be involved. Therefore, we use various classifiers and provide a concise overview of the most significant concepts in the following subsections. Here, we consider the possibility that the training procedure may frequently be negatively impacted by an uneven splitting of training examples by class in the training set. To prevent this, we provide class re-balancing, which lowers error rates by randomly choosing the similar number of training samples for each class. End users will, as a result, not only understand how each approach works, but they will also be able to compare everything.

A classifier Is provided with either all or just the selected features in the final step, and it assigns them to one of the specified (semantic) classes. Most approaches concentrate on supervised classification schemes for a diverse range of applications. The basic idea is to teach a classifier by utilizing existing training data, so that it can classify new information. The training data comprise a set X of training samples, each consisting of a class label and a feature vector in a feature space with d dimensions. Only feature vectors in the d-dimensional feature space can be used in the test set Y, which contains new data that need to be categorized.

We prioritize individual point classification because it offers simplicity, efficiency, and reproducibility, as well as the availability of numerous suitable algorithms in different software. In order to establish a correlation between feature vectors and class labels during the training phase, we use various classifiers and provide a brief overview of the most important concepts in the following sections. On the other hand, an uneven distribution of training examples by class in the training set can frequently be detrimental to the training procedure. We use class rebalancing to address this, which entails randomly choosing the same number of testing samples for each class, which results in a smaller testing set. In this manner, end users will not only comprehend how each method functions but will also be able to compare them in a knowledgeable manner.

Self-Organizing Maps (SOM)

In "Self-Organizing Maps (SOM)", the pre-learned principal component data are taken into consideration and trained. The input is chosen from the similar workspace. First, the building is selected as the input of the ROI. Then, ten classes are considered, and the specific data are trained.

In the building class, two distinct sub-labels are referred to as "Block 4" and "Block 6". The Weight Distance (WD) and the number of hits for the building are utilized. SOM differs from other Artificial Neural Networks (ANN) as it implements an efficient learning mechanism for error correction. The Weight Vectors (WVs) are initialized. From a randomly selected sample vector, the WV is analyzed to identify the weight that represents the sample. Each WV has neighboring weights that are close. The weight is adapted to resemble the selected vector. Figure 5 illustrates the SOM framework. The modification of weights and adjustments in the surrounding area are illustrated below.

$$W_{ij}(t+1) = W_{ij}(t) + \alpha_{i(t)}\left[x(t) - W_{ij}(t)\right] \tag{7}$$

where W is Weight vector, $W_{ij}(t)$ is Connection Weight, t is Current iteration, $\alpha_{i(t)}$ is Learning Rate, $x(t)$ is Input Vector.

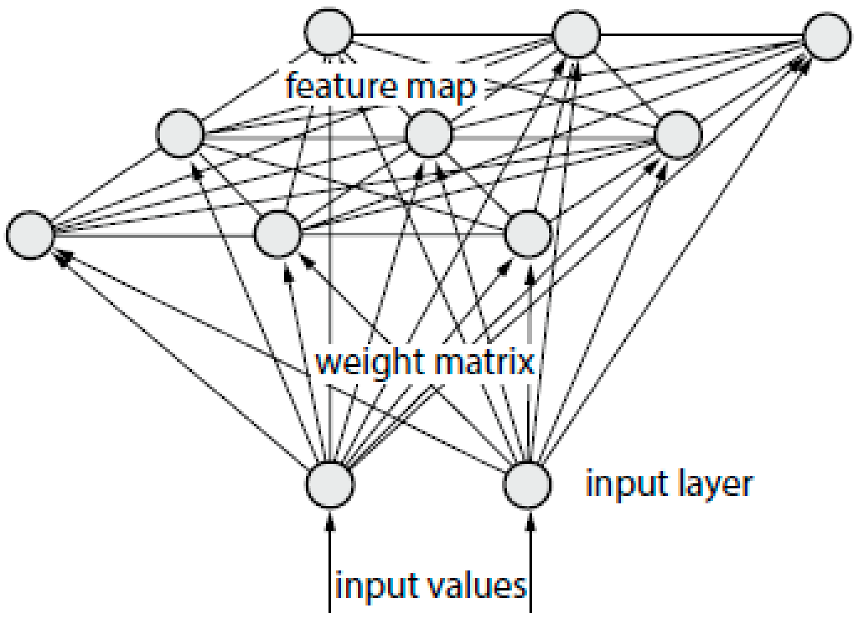

Figure 5. SOM Framework.

5. Experiments

This section includes a description of the experiment's methodology and a data summary. The equipment and sensors used in this section are normally described. Here, we explain the steps used to gather the data. If the experiment is complicated, a separate section could be dedicated to the technique. We employed a challenges being faced that was very comparable to the two datasets in order to test our claim. In the upcoming sections, we concentrate on evaluating the efficiency of our framework. Initially, in Section 5.1, we introduce two pertinent benchmark datasets, which are publicly accessible, and we describe their real-time application. We then elaborate on the experiments conducted in Section 5.2. Later, in Section 6, we provide the outcomes of the analysis of selecting the ideal neighborhood size, comparing them with conventional neighborhood definitions and focusing on evaluating individual techniques.

5.1. Data Set

We aim to facilitate an unbiased comparison with other approaches and prioritize the ease of applying the procedures and reproducibility of results. As a result, we assess the effectiveness of our framework by utilizing two labeled 3D PC datasets that are publicly available. These datasets are elaborated on in the subsequent sections. The implementation of these procedures is also performed in real-time data.

5.1.1. Oakland (3D PC Dataset)

The Oakland 3D PC Dataset is among the most extensively employed MLS datasets. Table 3 displays the quantity of samples for training and testing data in the Oakland dataset. This dataset was gathered using a mobile platform that was equipped with side-looking "SICK LMS laser scanners operating in push-broom mode", and it portrays an urban region. "To every 3D point is allocated one of the five semantic labels, namely wire, pole/trunk, façade, ground, and vegetation". The dataset is separated into the training set X, validation set V, and test set Y. Table 3 provides the number of samples for each class to give readers an idea of the dataset's distribution. Following class rebalancing, the training set is reduced to 1000 training instances for each class.

Table 3. For the Oakland 3D PC Dataset, the number of samples per class.

S.No.	Class	Training Set	Test Set
1	Wire	2572	3795
2	Pole/trunk	1087	7934
3	Façade	4714	222,113
4	Ground	14,122	934,147
5	Vegetation	14,442	267,326
Total		36,933	1,324,311

5.1.2. Paris-rue-Madame Database

We consider the Paris-rue-Madame database [46] as our second dataset. This database was collected in the French city of Paris, with a street stretch measuring approximately 160 m in length representing the 20 million 3D points in this collection. The Velodyne HDL32-equipped Mobile Laser Scanning system L3D2 was used for data collection, and annotation was carried out with the help of a manual process. Point labels as well as segmented objects are included in the annotated dataset, with 642 objects in the database divided into 26 classes. Our tests are limited to 3D points that fall into one of the six core semantic classifications—façade, ground, autos, motorbikes, traffic signs, and pedestrians—because the remaining classes only make up less than 0.05% of the whole dataset. Table 4 shows the Paris-rue-Madame database, with the number of samples per class.

Table 4. For the Paris-rue-Madame database, the number of samples per class.

S.No.	Class	Training Set	Test Set
1	Façade	1000	9,977,435
2	Ground	1000	8,023,295
3	Cars	1000	1,834,383
4	Motorcycles	1000	97,867
5	Traffic signs	1000	14,480
6	Pedestrians	1000	9048
Total		6000	19,956,508

Once more, we perform a class rebalancing and choose at random a training sample X including 1000 training instances for each class. The remaining data are then used as the test set Y.

5.2. Classifying Vegetation

We acquired a minor portion of our institute's facilities, including the building, road, ground, open stage, trees, stone benches, lane, pillar, flag, rainwater channel, and other features in the evening, for the purpose of processing as a Packet Capture Data (PCAP) file. Our LiDAR data contain approximately 28 unknown classes and 25 known classes. The Velo viewer tool was used to open the raw data and convert it to the CSV format for further processing. After loading the data in CSV format, a PC object was created. Each PC in our dataset contains approximately 100 frames. The distribution of LiDAR PC occupancy is depicted in Figure 6, where the x-axis denotes the current position of the sensor, and the *y*-axis denotes the number of points from the sensor.

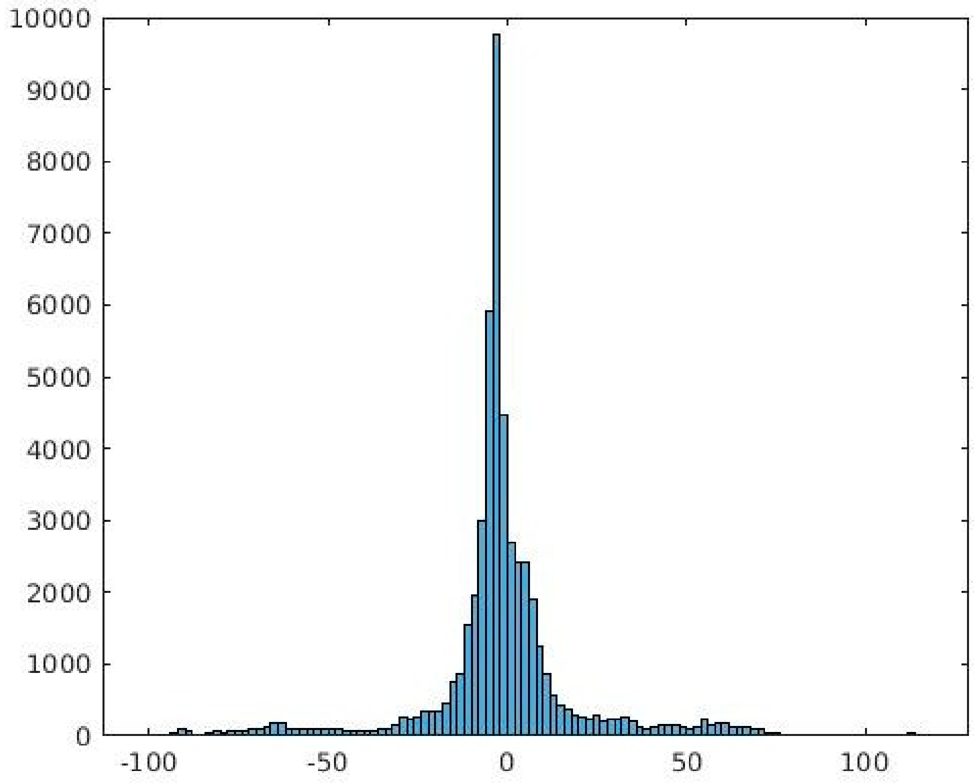

Figure 6. PC data distribution.

We are conducting experiments in MATLAB R2019b. The sequence of images used to generate labels is loaded following the selection of the truth. Labels are made for regions of interest, which are then imported and exported from the workspace for future use. This object serves as the foundation for subsequent processing. The k-nearest neighbor algorithm is utilized to determine if two points are connected. Using this method, 53 distinct segmented objects are categorized, and 9 of these 53 objects have pre-assigned labels, with one class named "unknown". The organized PCs facilitate the identification of clusters, and each cluster is assigned a distinctive color index value. Buildings are assigned the color red, trees the color green, concrete the color yellow, and unknowns the color purple. The remaining clusters are not included in the pre-learning process. The PC data are organized in a specific order, with buildings, trees, lanes, concrete benches, an open terrace, and other features clearly visible in this frame. The ground truth value is utilized to extract the building points from the organized LiDAR PC. The Self-Organizing Map (SOM) clustering topology is created by passing the ground truth value to SOM as an input variable. Approximately 700 features are distinguished from the PC, with the main 200 elements being the target. Although the Mono model has high accuracy, it needs an important amount of time complexity. The process of detecting features using the Mono model requires a maximum of 100 iterations, a relaxation factor of 0.5, and a pyramid level of 3.

6. Results and Discussion

Python and C++ were used to create the point cloud categorization system. The point cloud library (PCL) was modified to conduct point cloud downsampling and feature extractions in order to process the point clouds quickly. The LiDAR and camera data algorithms were implemented in the Scikit-learn framework for the classification procedure. The implementation of both classification methods used parallel computing. Consequently, both the training and testing procedures were sped up using a multicore processor. Python version 2.7 was used to put the suggested tree location recognition and building footprint extraction techniques into practice.

Our framework has the benefit of being made up of four separate components, which can be individually evaluated, giving us the ability to thoroughly examine any possible configuration. The free software OpenDroneMap was then used to process the photos that had been gathered. Using the gathered photos, this software reassembles point clouds and detailed 3D models of the survey objects. CloudCompare technique was used to analyze the 3D models' and point clouds' degree of accuracy. Moreover, Jupiter Notebook's implementation of all the phases of the technique we describe here for LiDAR data uses the Python programming language and open-source tools, making it free to use on any machine. In this part, we briefly explain the key findings from our research for each component. The use of individual, optimally sized neighborhoods provides a generic strategy for the first part of neighborhood selection, eliminating the need for prior knowledge of the scene that would be required when specifying neighborhoods using traditional methods. The concept that the most suitable neighborhood size may differ between different classes and also rely on the correct point density is reinforced by the use of each 3D neighborhoods. According on the results of our classifying particular groups, neighborhood definitions with set scale parameters may not be applicable across all groups [47].

Many tests have been run to improve this method throughout this effort. These tests were carried out on the grounds of Active Space Technologies and involved recording bags utilizing the depth with multiple sorts of obstacles placed at varied distances. We evaluated the method for both stationary and moving obstructions in both indoor and outdoor settings.

On the other hand, the eigen entropy-based scale selection method that has been suggested is capable of immediately adjusting for the provided 3D PC data, resulting in enhanced classification results, particularly with regard to the mean class recall values. This enhancement has been observed across a broad range of classifiers. The substantial impact on the smaller classes' "wire" and "pole/trunk" is due to the positive effect of the optimal size of individual neighborhoods on mean class recall values. As a result, the use of individual, optimally sized neighborhoods reduce the likelihood of overfitting for classifiers. This is evident from the unbalanced test set in the Oakland 3D PC Dataset, where an overall accuracy of 70.5% could be achieved by correctly categorizing instances of the "ground" class, but a trend toward overfitting of 20.0% is observed when considering the mean class recall. Therefore, in our experiments, although the overall accuracy may not be adequate, the mean class recall is a more reliable indicator of the quality of the outcomes.

For the investigation of the proposed system, a PCAP file was taken in the evening of a minor part of the institute's building, roads, trees, benches, lanes, pillars, etc. The LiDAR data consist of 25 recognized classes and 28 unidentified classes. The Veloviewer tool is used to convert it into a CSV file for processing after opening the raw data. The CSV file is loaded and a PC object is created. The dataset includes nearly 100 frames per PC, which are transformed and stored in separate locations.

The labeling process involves selecting a ground truth and uploading the image structure to generate the labels. New labels are added to the ROIs and exported to the workspace for additional reference. This object can be used as input for further processing. The connection between each point is generated using the K-Nearest Neighbor (KNN) algorithm, which categorizes 53 varieties of segmented objects. Of these, 9 are labeled with pre-defined terms.

Among them, one class is labeled as "unknown". Clusters are identified from the prearranged PC. Each cluster has a unique CI. Different colors are assigned for each label, with red assigned to buildings, yellow to concrete, green to trees, and purple to unknown objects. The remaining objects are not considered for the pre-learning process. Figure 7 shows a color-based classified image frame, which clearly indicates that Block 4 and 6 consist of buildings, trees, benches, lanes, etc. Figure 7, on the right, shows the PC's contour image. The labeling process begins with the selection of a ground truth, followed by the uploading of the picture structure so that the labels may be generated. Updated with new labels, and the labels are then exported to the workspace for further reference. Figure 8 depicts the SOM hit for a feature building.

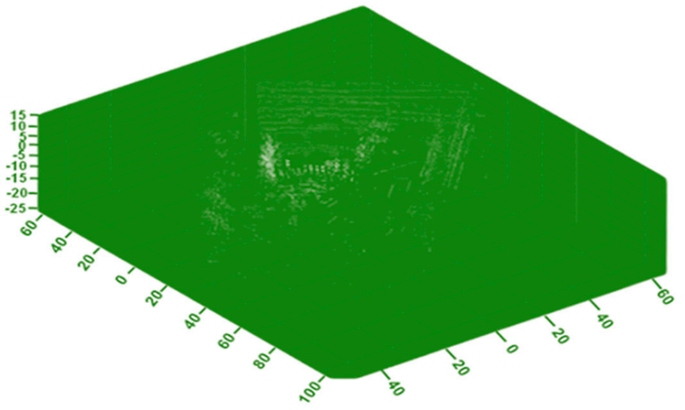

Figure 7. Segmented PC for 3D Object.

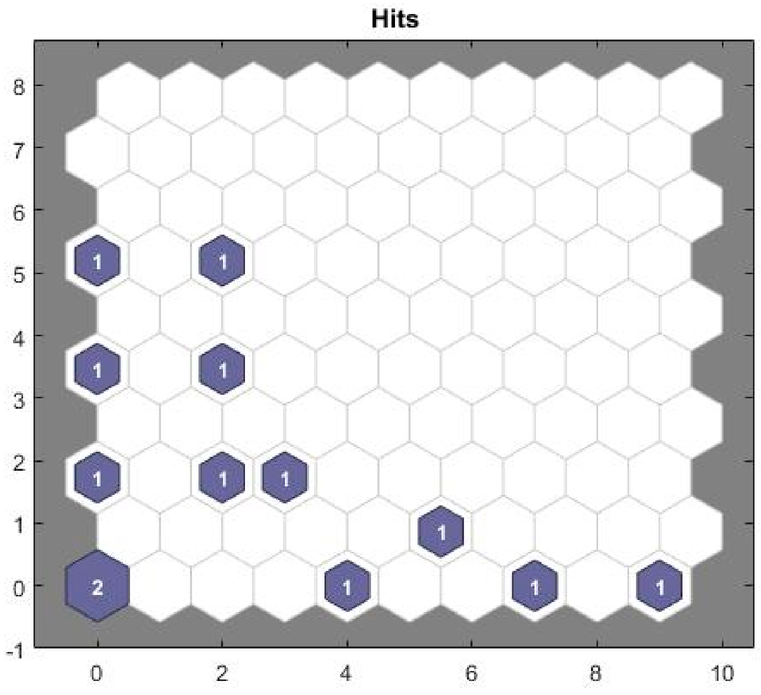

Figure 8. Amount of Hits for Building.

Figure 9 depicts SOM neighbor weight distance. "An unsupervised machine learning method called a self-organizing map (SOM) or self-organizing feature map (SOFM)" is used to create a low-dimensional (usually 2D) representations of a higher-dimensional data collection while maintaining the topological architecture of the data. A data collection containing parameters calculated in assumptions, for instance, could be described as groups of findings with related variable values. A two-dimensional "map" of these clusters might then be used to represent them, with study in proximal clusters having more similar values than research in distal clusters. High-dimensional data may be easier to display and interpret as a result.

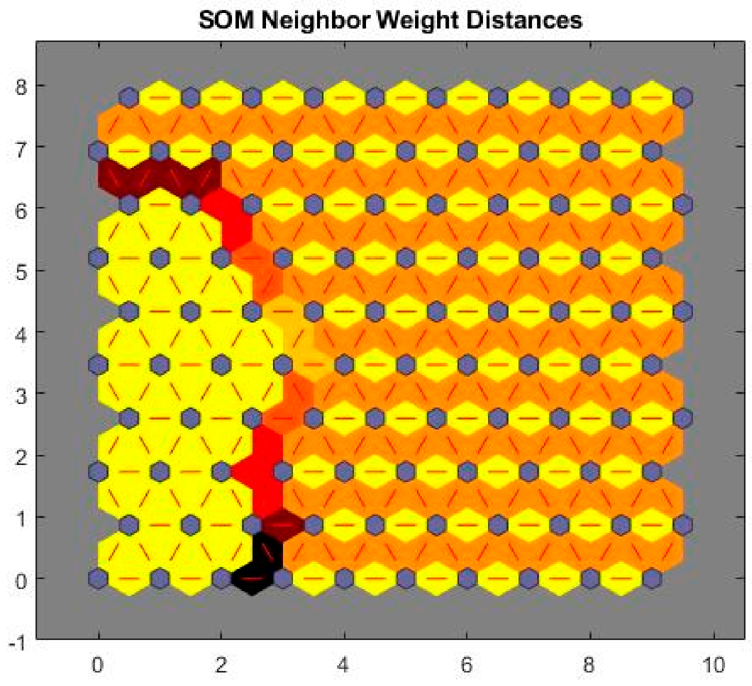

Figure 9. Weight Distance.

In Figure 10, the refurbished 2D image of the PC data are displayed. It is noticeable that the image is not well defined and lacks complete information, but the absent information is reconstructed to enable further analysis. The image frames are incorporated into the training set for testing, and the testing period lasts for 3 days or more to verify the outcomes. Some objects are not accurately classified, and the errors observed from the results are eliminated from the image of interest. The error rate of the samples is illustrated in Figure 11.

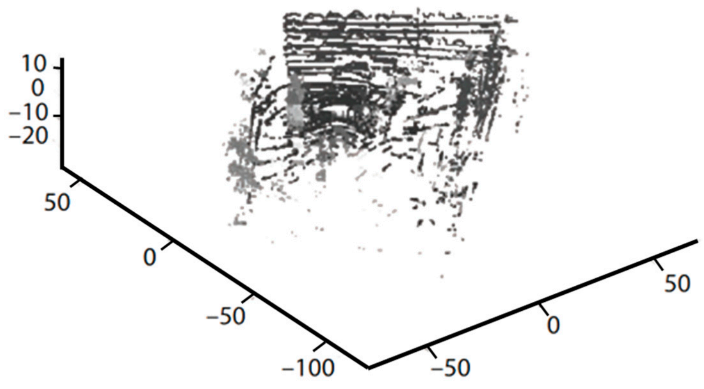

Figure 10. Reconstructed 2D Image from PC data.

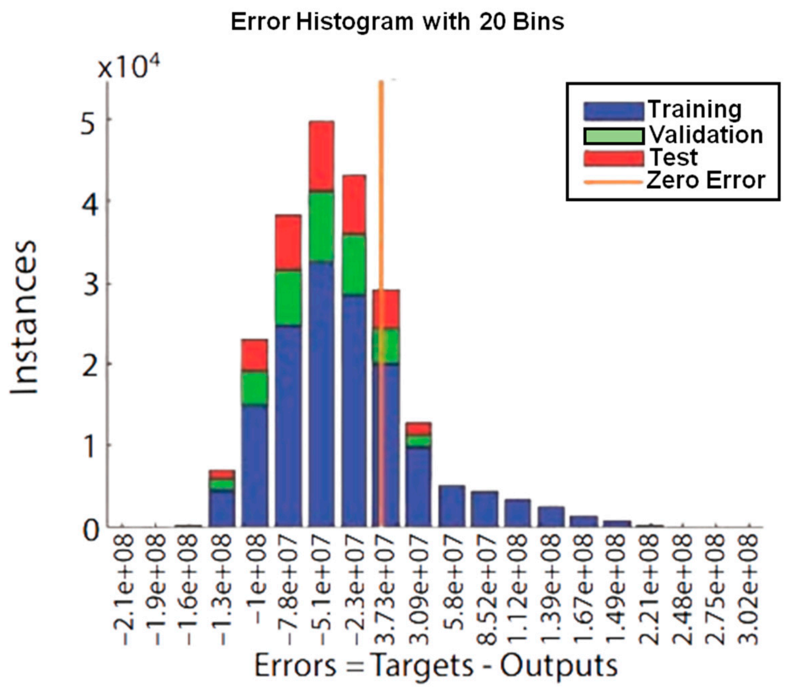

Figure 11. Rate of misclassification.

Table 5 shows the feature recognition using various models in image registration. Approximately 700 features were recognized from the PC, and 200 of these features were predicted. The mono-model provides higher accuracy, but requires more time. Feature detection takes a maximum of 100 iterations, with a relaxation factor of 0.5 and a pyramid level of 3. Figure 12 compares various feature identification methods, including the proposed work, with minEigen, FAST, BRICK, and SURF. MimEigan's accuracy is 86.5%, FAST accuracy is 89.2%, BRISK accuracy is 86.2%, SURF accuracy is 91.7%, and Mono-model accuracy is 92%. MimEigan's Timing is 60 ms, FAST Timing is 30 ms, BRISK Timing is 50 ms, SURF Timing is 50 ms, and Mono-model Timing is 20 ms.

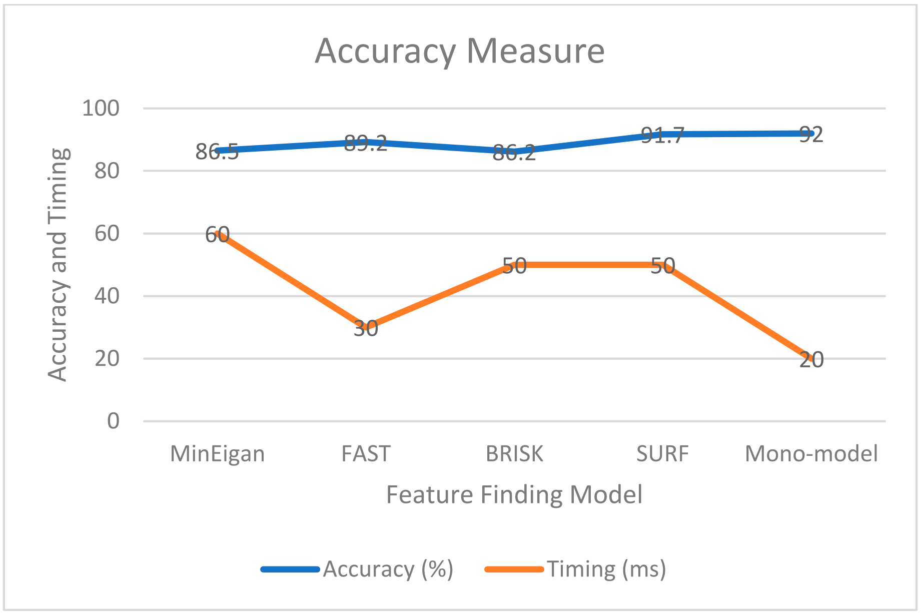

Figure 12. Accuracy Measure for feature identification.

Table 5. Feature Identification Schemes.

Feature Finding Models	Accuracy (%)	Timing (ms)
MinEigan	86.5	60
FAST	89.2	30
BRISK	86.2	50
SURF	91.7	50
Mono-model	92	20

6.1. Impact of Neighborhood Selection

To investigate the influence of neighborhood selection on the classification of individual points, we evaluate 21 geometric attributes for each of the 7 alternative neighborhood definitions using 10 classifiers from various categories. Tables 6 and 7 provide a thorough investigation of the effect of neighborhood selection on classification outcomes by displaying the recall and accuracy values for the various neighborhood descriptions and classifiers. Additionally, Table 6 provides the absolute and relative processing times for the training and testing phases, enabling an assessment of the efficiency of the classifiers involved.

Table 6. Overall accuracy (in %) for different neighborhood definitions and different classifiers.

Oakland	NN [48]	DT [48]	NB [48]	LDA [48]	QDA [48]	SVM [48]	MLP [48]	SOM
N 10	73.86	65.64	78.88	87.38	78.93	85.69	80.54	82.54
N 25	86.25	69.3	83.64	90.08	83.62	88.88	78.59	85.62
N 50	88.89	75.47	85.03	92.83	84.95	92	85.68	89.03
N 75	89.97	76.87	85	93.05	84.99	91.99	87.07	91.87
N 100	89.9	84.45	84.33	92.6	84.43	91.76	84.39	88.32

Table 7. Mean class recall values (in %) for different neighborhood definitions and different classifiers with the respective classifier.

Oakland	NN	DT	NB	LDA	QDA	SVM	MLP	SOM
N 10	64.4	55.20	63.30	71.69	63.34	59.77	64.2	66.87
N 25	70.01	57.41	68.46	75.54	68.47	68.5	68.03	70.5
N 50	69.47	59.99	67.12	72.76	66.98	68.47	69.13	71.13
N 75	68.29	57.82	65.49	73.05	65.44	68	70.47	73.05
N 100	66.66	57.96	63.44	72.35	63.46	64.76	68.98	73.04

Probabilistic learning involves inferring the most likely class label for each observed feature vector from a specified probabilistic model. One example of a probabilistic classifier is the Naive Bayes (NB) classifier, which relies on Bayes' theorem and assumes that all features are conditionally independent (John and Langley, 1995). In classification, a feature vector from the experimental set Y is given a class label that is most likely to accurately reflect its content by utilizing the class probabilities and conditional probabilities for the presence of a class provided a particular class label, both of which are determined by reference to the training set X. However, this method cannot accurately model correlated features as it assumes conditional independence. An alternative approach is to use distribution-based Bayesian Discriminant Analysis to construct a traditional maximum likelihood (ML) classifier. During the training phase of a Gaussian-based classifier, the variables of a Gaussian distribution are assessed for each class using either multivariate Gaussian distribution or parameter fitting. The Linear Discriminant Analysis (LDA) classifier assumes that each class has the same covariance matrix, with only the means varying. On the other hand, the "Quadratic Discriminant Analysis (QDA) classifier" allows the covariance matrix for each individual class to differ. The likelihood of a feature vector belonging to each class is calculated, and the class with the greatest likelihood of accuracy is selected when classifying a new feature vector. Figure 13 compares various classifiers and their accuracy levels for different neighborhood definitions.

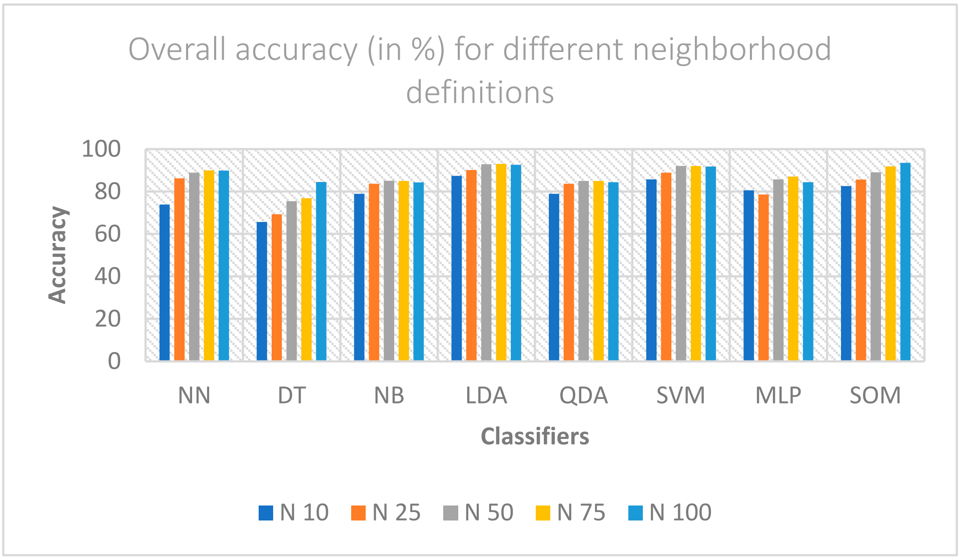

Figure 13. Neighborhood definitions for various classifiers.

Precision is the level of a classifier's conformance and accuracy when measured against a real or absolute value. The accuracy levels of different classifiers for various neighborhood configurations are compared in Figure 14. Comparing the SOM approach to others, we find it to be very accurate.

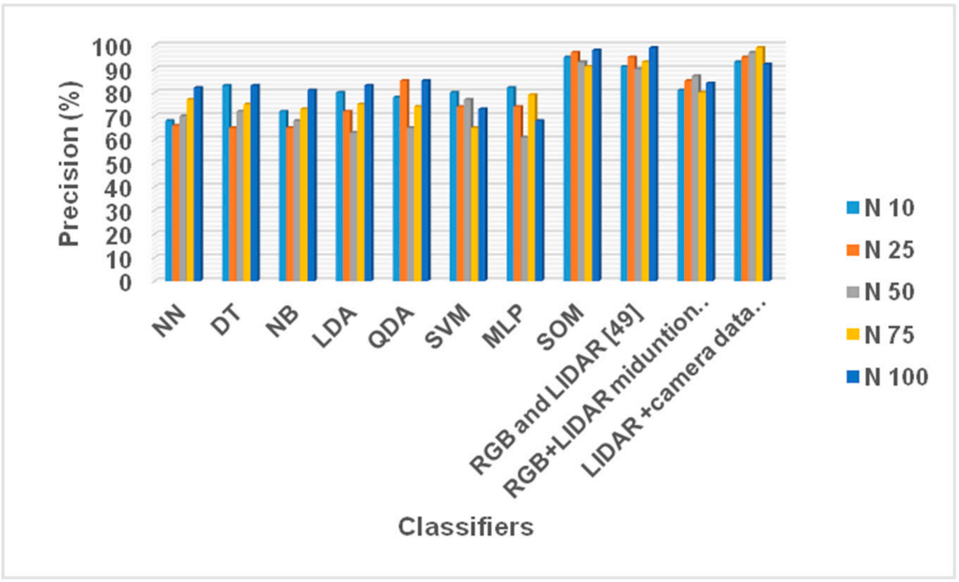

Figure 14. Comparison of Precision.

6.2. Impact of Feature Selection

Additionally, we aim to introduce a method for selecting significant features while disregarding insignificant ones to minimize the computational load in terms of processing time as well as memory utilization. Our emphasis is on the ability to analyze large-scale 3D scenes with much huge datasets. To achieve this, we once again utilize a SOM classifier and present the outcomes attained by utilizing the seven various neighborhood descriptions and feature sets, resulting in 49 different possible combinations. The mean class recall values (in percentage) for various neighborhood descriptions and classifiers are presented in Table 7.

Decision trees (DTs) use a hierarchical structure of basic tests to make predictions. The construction of a decision tree usually follows a top-down approach. At each step, the best parameter for splitting the training examples is selected. The description of the splitting function and the stopping criterion for the recursive partitioning process is crucial to building an accurate decision tree. In this case, standard settings are used for both criteria.

The complete feature set achieved a mean class recall of 83.56% and an overall accuracy of 88.76%. The overall accuracy and mean class recall for Feature Set 1 were 88.98% and 84.66%, respectively; for Feature Set 2, these figures were 89.16% and 83.83%. The recall values for different classifiers are displayed in Figure 15. By measuring the natural logarithm of a classifier's accuracy and recall, the F1-score integrates both into a single statistic. It is utilized to compare the performance of two classifiers. For various neighborhood definitions, Figure 16 compares several classifications and their F1-scores. Comparing our SOM approach to others, it has a strong F1-score.

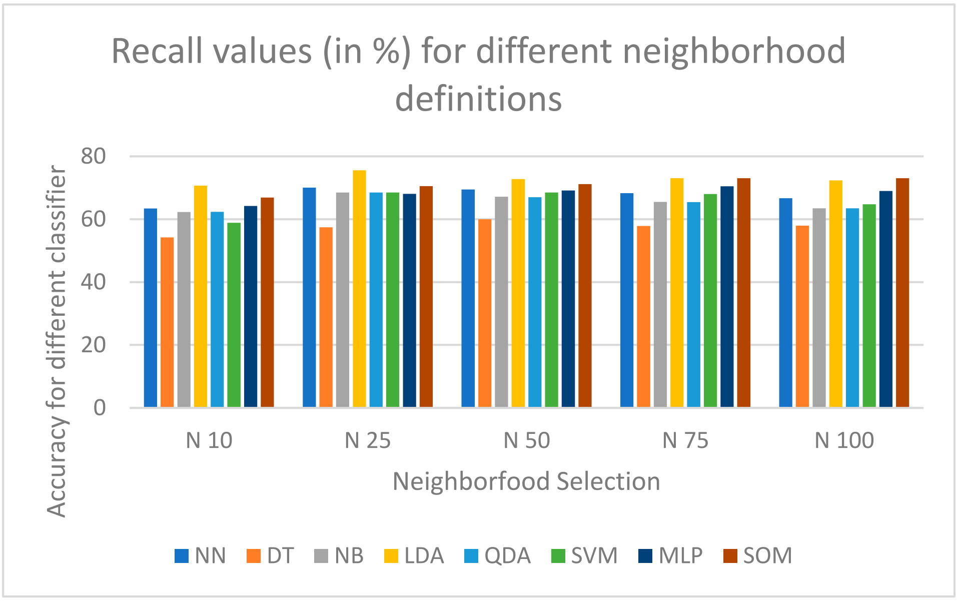

Figure 15. Recall Value for different neighborhood definitions.

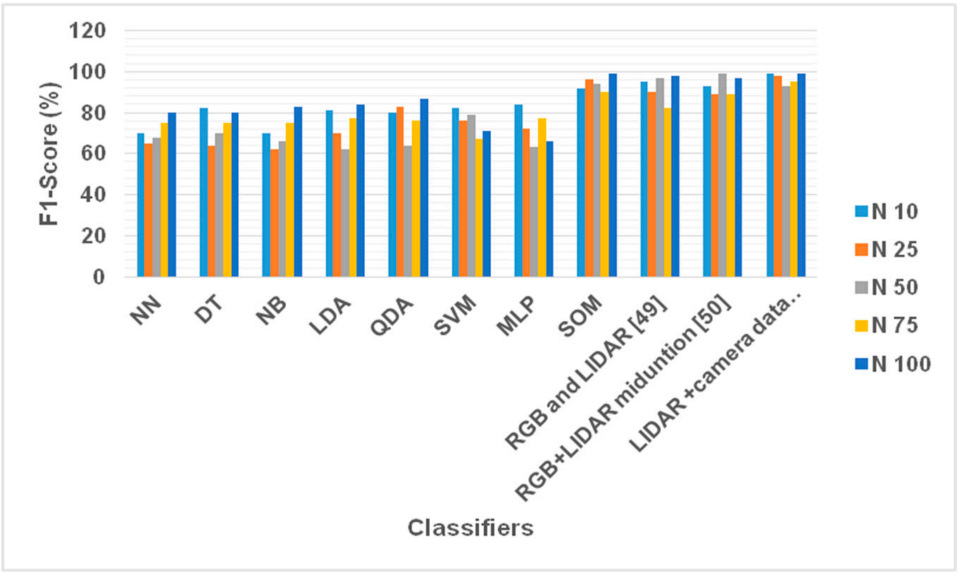

Figure 16. Comparison of F1-score.

7. Conclusions

In this paper, a mono-model pre-learning-based SS scheme is proposed for unstructured data and smart environments. The proposed scheme gains insight from PCs through the incorporation of a pre-learning process, which includes the use of Self-Organizing Maps (SOM). The PCs are transformed into 2D frames of images using a mono-model based image registration scheme. We present a novel, fully automated, and adaptable architecture consisting of four sequential components, each equipped with a range of methods, designed to meet the requirements of simplicity, efficiency, and reproducibility.

Our work addressed two interconnected issues: selecting the optimal size of individual neighborhoods for feature extraction and choosing a subset of the most applicable characteristics. We evaluated seven different neighborhood definitions, twenty-one different features, seven different feature selection strategies, and ten different classifiers in order to show how using appropriate neighborhood sizes and the benefits of feature selection can significantly improve classification accuracy while decreasing the complexity of computation. We found that the neighborhood selection method depends on decreasing the eigen entropy evaluation over a range of scales, which had a positive effect on classification, regardless of the classifier being used. We also found that feature subsets produced by feature selection methods based on the symmetrical uncertainty measure were the most appropriate as they removed unnecessary features and reduced feature redundancy.

Our long-term goal is to employ well-spaced, smooth labelings to dramatically enhance the outcomes of 3D scene analysis in subsequent projects. This can be achieved through the use of smoothing techniques or approaches that utilize contextual data. These choices rely on the outcomes of individual point categorization, making the suggested approach a necessary precondition. Nevertheless, because this research addresses such an issue, it can be anticipated that the LiDAR sensor's usage would rise in the future. Moreover, it is expected that this trend will progressively pick up speed when LiDAR sensor quality is enhanced to catch up to image sensor sensitivity. Further study will include extending the findings to real-time autonomous vehicles and validating the LiDAR object identification ability using filtering point cloud data. In our future study, we want to investigate more complex fusion structures using network architecture search approaches and make use of all of the color pixels that are now accessible. Conventional scanners are not designed to pick up on this trait, and thus, they do not normally look for it.

Author Contributions: Conceptualization, K.R., N.G. and M.M.; methodology, K.R., N.G. and M.M.; validation, K.R. and N.G.; software, K.R., N.G., M.M. and R.G.; formal analysis, K.R., M.M. and R.G.; investigation, K.R., N.G., M.M. and R.G.; resources, K.R. and N.G.; writing—original draft, K.R., N.G. and M.M.; writing—review and editing, M.M. and R.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the project SP2023/074 Application of Machine and Process Control Advanced Methods supported by the Ministry of Education, Youth and Sports, Czech Republic.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available through email upon request to the corresponding author.

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

References

1. Schwarting, W.; Alonso-Mora, J.; Rus, D. Planning and Decision-Making for Autonomous Vehicles. *Annu. Rev. Control. Robot. Auton. Syst.* **2018**, *1*, 187–210. [CrossRef]
2. Lee, J.-S.; Park, T.-H. Fast Road Detection by CNN-Based Camera-Lidar Fusion and Spherical Coordinate Transformation. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 5802–5810. [CrossRef]
3. Xie, X.; Wei, H.; Yang, Y. Real-Time LiDAR Point-Cloud Moving Object Segmentation for Autonomous Driving. *Sensors* **2023**, *23*, 547. [CrossRef] [PubMed]

4. Pires, M.; Couto, P.; Santos, A.; Filipe, V. Obstacle detection for autonomous guided vehicles through point cloud clustering using depth data. *Machines* **2022**, *10*, 332. [CrossRef]
5. Akai, N.; Morales, Y.; Murase, H. Simultaneous pose and reliability estimation using convolutional neural network and Rao–Blackwellized particle filter. *Adv. Robot.* **2018**, *32*, 930–944. [CrossRef]
6. Chromy, A.; Zalud, L. Robotic 3D scanner as an alternative to standard modalities of medical imaging. *SpringerPlus* **2014**, *3*, 13. [CrossRef] [PubMed]
7. Abellan, A.; Derron, M.-H.; Jaboyedoff, M. "use of 3D point clouds in geohazards" special issue: Current challenges and future trends. *Remote Sens.* **2016**, *8*, 130. [CrossRef]
8. Albano, R. Investigation on roof segmentation for 3D building reconstruction from aerial LIDAR point clouds. *Appl. Sci.* **2019**, *9*, 4674. [CrossRef]
9. Biasutti, P.; Bugeau, A.; Aujol, J.-F.; Bredif, M. RIU-Net: Embarrassingly simple semantic segmentation of 3D LiDAR point cloud. *arXiv* **2019**, arXiv:1905.08748.
10. Wang, X.; Lyu, H.; Mao, T.; He, W.; Chen, Q. Point cloud segmentation from iPhone-based LiDAR sensors using the tensor feature. *Appl. Sci.* **2022**, *12*, 1817. [CrossRef]
11. Ben-Shabat, Y.; Lindenbaum, M.; Fischer, A. 3D Point Cloud Classification and Segmentation using 3D Modified Fisher Vector Representation for Convolutional Neural Networks. *arXiv* **2017**, arXiv:1711.08241.
12. Wang, W.; Zhou, T.; Yu, F.; Dai, J.; Konukoglu, E.; Van Gool, L. Exploring cross-image pixel contrast for semantic segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 10–17 October 2021; pp. 7303–7313.
13. Libiao, J.; Wenchao, Z.; Changyu, L.; Zheng, W. Semantic segmentation based on DeeplabV3+ with multiple fusions of low-level features. In Proceedings of the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 12–14 March 2021; pp. 1957–1963.
14. Huang, Z.; Wei, Y.; Wang, X.; Liu, W.; Huang, T.S.; Shi, H. Alignseg: Feature-aligned segmentation networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 550–557. [CrossRef] [PubMed]
15. Nekrasov, A.; Schult, J.; Litany, O.; Leibe, B.; Engelmann, F. Mix3d: Out-of-context data augmentation for 3d scenes. In Proceedings of the 2021 International Conference on 3D Vision (3DV), London, UK, 1–3 December 2021; pp. 116–125.
16. Xie, B.; Li, S.; Li, M.; Liu, C.H.; Huang, G.; Wang, G. SePiCo: Semantic-Guided Pixel Contrast for Domain Adaptive Semantic Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, 1–17. [CrossRef]
17. Li, P.; Xu, Y.; Wei, Y.; Yang, Y. Self-correction for human parsing. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *44*, 3260–3271. [CrossRef]
18. Borse, S.; Cai, H.; Zhang, Y.; Porikli, F. Hs3: Learning with proper task complexity in hierarchically supervised semantic segmentation. *arXiv* **2021**, arXiv:2111.02333 2021.
19. Yuan, F.; Zhu, Y.; Li, K.; Fang, Z.; Shi, J. An anisotropic non-local attention network for image segmentation. *Mach. Vis. Appl.* **2022**, *33*, 23. [CrossRef]
20. Zhang, Z.; Zhang, X.; Peng, C.; Xue, X.; Sun, J. Exfuse: Enhancing feature fusion for semantic segmentation. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 269–284.
21. Valada, A.; Mohan, R.; Burgard, W. Self-supervised model adaptation for multimodal semantic segmentation. *Int. J. Comput. Vis.* **2020**, *128*, 1239–1285. [CrossRef]
22. Huang, S.-S.; Ma, Z.-Y.; Mu, T.-J.; Fu, H.; Hu, S.-M. Supervoxel convolution for online 3d semantic segmentation. *ACM Trans. Graph. (TOG)* **2021**, *40*, 1–15. [CrossRef]
23. Gao, T.; Wei, W.; Cai, Z.; Fan, Z.; Xie, S.; Wang, X.; Yu, Q. CI-Net: Contextual information for joint semantic segmentation and depth estimation. *arXiv* **2021**, arXiv:2107.13800. [CrossRef]
24. Shikishima, J.; Tasaki, T. Dynamic 3D-Obstacles Detection by a Monocular Camera and a 3D Map. In Proceedings of the 2021 IEEE/SICE International Symposium on System Integration (SII), Fukushima, Japan, 11–14 January 2021; pp. 704–705.
25. Wang, X.; Liu, S.; Shen, X.; Shen, C.; Jia, J. Associatively segmenting instances and semantics in point clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4096–4105.
26. Cao, B.; Sun, Z.; Zhang, J.; Gu, Y. Resource Allocation in 5G IoV Architecture Based on SDN and Fog-Cloud Computing. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3832–3840. [CrossRef]
27. Rajathi, K.; Sarasu, P. Pre-Learning-Based Semantic Segmentation for LiDAR Point Cloud Data Using Self-Organized Map. In *Role of Edge Analytics in Sustainable Smart City Development: Challenges and Solutions*; Wiley: Hoboken, NJ, USA, 2020; pp. 171–188.
28. Chen, P.; Pei, J.; Lu, W.; Li, M. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing* **2022**, *497*, 64–75. [CrossRef]
29. Li, J.; Qin, H.; Wang, J.; Li, J. OpenStreetMap-based autonomous navigation for the four wheel-legged robot via 3D-lidar and CCD camera. *IEEE Trans. Ind. Electron.* **2022**, *69*, 2708–2717. [CrossRef]
30. Triharminto, H.H.; Wahyunggoro, O.; Adji, T.B.; Cahyadi, A.I. An integrated artificial potential field path planning with kinematic control for nonholonomic mobile robot. *Int. J. Adv. Sci. Eng. Inf. Technol.* **2016**, *6*, 410–418. [CrossRef]
31. Cabreira, T.M.; Brisolara, L.B.; Paulo, R.F.J. Survey on coverage path planning with unmanned aerial vehicles. *Drones* **2019**, *3*, 4. [CrossRef]
32. Yu, J.; LaValle, S.M. Optimal multi-robot path planning on graphs: Structure and computational complexity. *arXiv* **2015**, arXiv:1507.03289 2015.

33. Khaksar, W.; Tang, S.H.; Khaksar, M. Improved Bug Algorithm for Online Path Planning: Utilization of Vision Sensor. *Sci. Res. Essays* **2012**, *7*, 2744–2753. [CrossRef]
34. Ayawli, B.B.K.; Mei, X.; Shen, M.; Appiah, A.Y.; Kyeremeh, F. Mobile Robot Path Planning in Dynamic Environment using Voronoi Diagram and Computation Geometry Technique. *IEEE Access* **2019**, *7*, 86026–86040. [CrossRef]
35. LaSalle, D.; Karypis, G. A parallel hill-climbing refinement algorithm for graph partitioning. In Proceedings of the 2016 45th International Conference on Parallel Processing (ICPP), Philadelphia, PA, USA, 16–19 August 2016; pp. 236–241.
36. Fankhauser, P.; Hutter, M. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In *Robot Operating System (ROS) The Complete Reference (Volume 1)*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 99–120.
37. Boyko, A.; Funkhouser, T. Extracting roads from dense point clouds in large scale urban environment. *ISPRS J. Photogramm. Remote Sens.* **2011**, *66*, S02–S12. [CrossRef]
38. Zhao, Z.; Morstatter, F.; Sharma, S.; Alelyani, S.; Anand, A.; Liu, H. *Advancing Feature Selection Research—ASU Feature Selection Repository*; Tech. Rep.; School of Computing, Informatics, and Decision Systems Engineering, Arizona State University: Tempe, AZ, USA, 2010.
39. Weinmann, M.; Jutzi, B.; Mallet, C. Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features. In Proceedings of the ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, Zurich, Switzerland, 5–7 September 2014; pp. 181–188.
40. Rusu, R.B.; Marton, Z.C.; Blodow, N.; Beetz, M. Persistent point feature histograms for 3d point clouds. In Proceedings of the International Conference on Intelligent Autonomous Systems, Zagreb, Croatia, 13–16 June 2008; pp. 119–128.
41. Criminisi, A.; Shotton, J. (Eds.) *Decision Forests for Computer Vision and Medical Image Analysis*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
42. Blomley, R.; Weinmann, M.; Leitloff, J.; Jutzi, B. Shape distribution features for point cloud analysis—A geometric histogram approach on multiple scales. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2014**, *II-3*, 9–16. [CrossRef]
43. Monnier, F.; Vallet, B.; Soheilian, B. Trees Detection From Laser Point Clouds Acquired In Dense Urban Areas By A Mobile Mapping System. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2012**, *I-3*, 245–250. [CrossRef]
44. Priyadarshini, J.; Premalatha, M.; Čep, R.; Jayasudha, M.; Kalita, K. Analyzing Physics-Inspired Metaheuristic Algorithms in Feature Selection with K-Nearest-Neighbor. *Appl. Sci.* **2023**, *13*, 906. [CrossRef]
45. Ganesh, N.; Shankar, R.; Čep, R.; Chakraborty, S.; Kalita, K. Efficient Feature Selection Using Weighted Superposition Attraction Optimization Algorithm. *Appl. Sci.* **2023**, *13*, 3223. [CrossRef]
46. Munoz, D.; Bagnell, J.A.; Vandapel, N.; Hebert, M. Contextual classification with functional max-margin Markov networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 975–982.
47. Yang, S.; Xu, S.; Huang, W. 3D point cloud for cultural heritage: A scientometric survey. *Remote Sens.* **2022**, *14*, 5542. [CrossRef]
48. Wang, Y.; Chen, Q.; Liu, L.; Li, X.; Sangaiah, A.K.; Li, K. Systematic comparison of power line classification methods from ALS and MLS point cloud data. *Remote Sens.* **2018**, *10*, 1222. [CrossRef]