Extraction of Road Markings from LIDAR Intensity for SLAM systems

Abdulbaasit Modebayo Sanusi[1]
Candidate Number: RHWT4

Degree: MSc. Robotics and Computation

Supervisor's name(s): Dr. Mohamed Abdellatif, Dr. Wallizada Mohib, Prof. Simon Julier

Submission date: 26 September 2022

[1]**Disclaimer:** This report is submitted as part requirement for the MSc in Robotics and Computation at University College London. It is substantially the result of my own work except where explicitly indicated in the text. The report will be distributed to the internal and external examiners, but thereafter may not be copied or distributed except with permission from the author.

Abstract

SLAM algorithms work with features as input. These features come in the form of points with varying intensity values in LIDAR-based SLAM. Prior literature points out that not all features are essential for SLAM. Reducing the processed features based on their relevance to SLAM can be essential. Road markings represent a high-order meaning among floor points, with arrows and lanes carrying intuitive information regarding driving. Especially with most of the remaining points corresponding to plain road surfaces. Finding a way to detect and extract these road features can help autonomous vehicles localise, map their environment and reduce the computational load that comes with processing all features within a scene. This project explores whether road-marking features can be extracted from LIDAR data. Using a modified OTSU thresholding method, the LIDAR intensity data on the road surface is normalised and segmented into road marking and non-road marking points. The HDL graph SLAM framework constructs a map of the environment by replacing the floor points with road marking points. This method of extracting road markings produced a better result than current fixed thresholding algorithms, with a precision of 91.62%, recall of 94.03% and F1-score of 92.81%, which are good indicators of accurate feature extraction with the tested dataset.

Acknowledgements

My sincere gratitude goes to my supervisors(Dr. Mohammed Abdellatif, Dr Wallizada Mohib and Prof Simon Julier) for their support and guidance while working on this project.

I also want to appreciate the entire team at AIDrivers for their valuable contribution while in the course of this project.

Lastly, I am especially thankful to my family for their unconditional and unwavering support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Road markings are basic components of road networks which are present on all roads and serve as important perceptual cues for all road users. Due to their distinct colours from the rest of the road surface, they provide strong visual guidance and information to drivers [1]. Thus, it is important for autonomous vehicles to learn how to detect these road features in order to localise and map their environments.

Due to the fact that road markings are quite distinct road features, with bright and different colours from the rest of the road, they can be differentiated from other road features by detecting them using a “LIDAR” sensor, which captures the intensity value of all road surfaces. The distinctiveness of the road-marking features is illustrated in Figure 1.1. Although LIDAR data contains intensity information about road markings that can be used to reliably detect them for mapping and localisation, little research has been conducted on how to use these intensity values to extract relevant road marking features for mapping and localisation [B11].

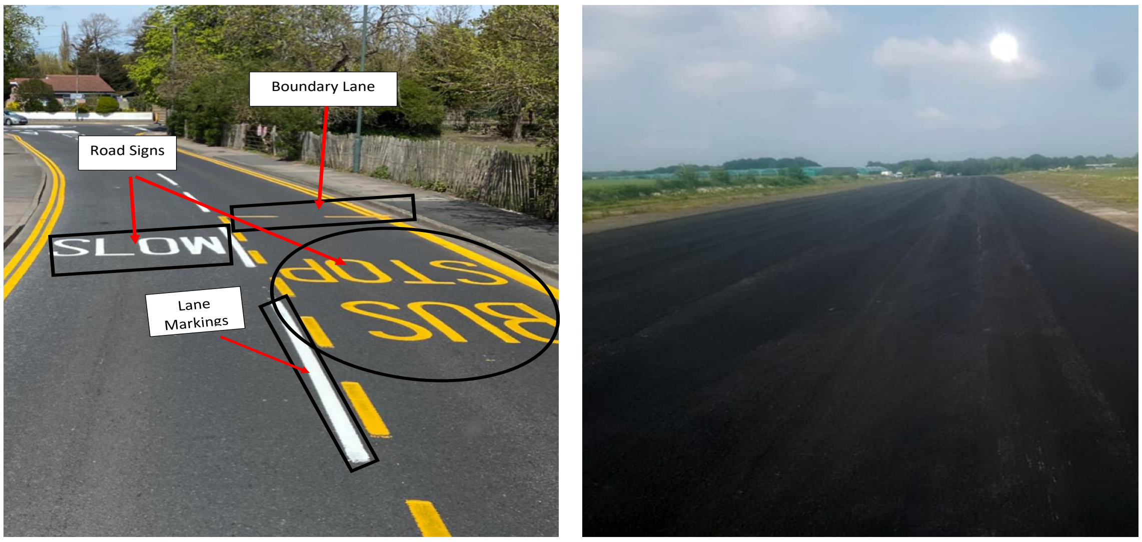


Figure 1.1: An image of a street: (a) with Road Markings [2] (b) without Road Markings [3]

1.2 Problem Definition

Attempts to extract road markings from LIDAR for mapping show that there is a huge number of points produced by a typical LIDAR sensor. The number of points produced can be on the order of 600,000 points per second, resulting in an excessive data volume for the SLAM system to process when recording for an extended period of time [4].

Most of these points are redundant and do not contain relevant information for building a map or localisation as can be visualised in Figure 1.1b., which are two fundamental processes of the SLAM system. They are also computationally expensive to process. These points make the map size too big and expensive to build. As road markings are the most relevant features on the road surface, it is, therefore, essential to find a means of detecting them. This will lead to building a less computationally expensive map without losing the most critical information- road markings.

The extracted road marking features are expected to contribute well to the accuracy of SLAM with decreased computational load. The work carried out in this report only describes accurate detection of the roadmarkings. The benefit of replacing the flooring with only these roadmarkings, to SLAM accuracy can be explored in subsequent works.

1.3 Aims and Objectives

In this research, a way of extracting road markings from these LIDAR scans based on intensity is explored. Firstly, the raw LIDAR scans are filtered to lower the number of points to be processed. Then, road surface points are detected from the LIDAR data by identifying the ground plane and detecting all points present on it. Lastly, the relevant road marking points are extracted by segmenting the road surface points into point clouds with road points and without road marking points.

Taking all these steps, redundant and non-essential road features are eliminated, thereby reducing the computational load of processing LIDAR data and reducing the map size of the road.

The specific contributions of this project are:

- Developing a road marking extraction algorithm based on fixed intensity value.
- Finding an optimal threshold value automatically that separates road marking points from non road-marking
- The extracted road marking points aim to reduce the total input points that may aid localisation and mapping.

1.4 Thesis Structure

The remaining chapters of this thesis is organised as follows:

- Chapter 2 describes the background and review literature relevant to this study. This gives all the background descriptions of the terminology and concepts used throughout this research. All the relevant literature describing the different processing stages is described here.
- Chapter 3 focuses on explaining the various stages of the SLAM system utilised in this study.
- Chapter 4 explains the proposed methodology and algorithm designed to extract road markings used in this project.
- Chapter 5 describes the various experiment carried out to ascertain the accuracy of the algorithms and also summarises the result obtained from this test.
- Chapter 6 concludes the study by discussing its limitations and future research directions.

Chapter 2

Background

In this chapter, the relevant background information relevent to this research is discussed. we also examine literature pertinent to the various phases of conducting this research. Section 2.1 describes the sensing system used for road marking extraction. Next, we discuss Simultaneous Localisation and Mapping (SLAM), and describe the two most prevalent SLAM algorithm types, one of which the package in use is based upon. In section 2.3, point cloud registration and the various point cloud registration techniques utilised in this study are described. In section 2.4, related work on road marking extraction is discussed, and finally, in the last section, the evaluation methods of road marking extraction used in literature are reviewed.

2.1 Sensing System used for road marking extraction

Researchers have utilised a variety of sensors in order to detect road markings. These sensors include cameras, Light Detection and Ranging (LIDAR), and Radio Detection and Ranging (RADAR) [5]. The most common sensors used, however, are LIDARs and cameras [5].

2.1.1 LIDAR

LIDAR is a time-of-flight device that measures the time it takes for a light pulse to be emitted, strike an object, and reflect back to the sensor [5]. The distance to the object is calculated by multiplying the speed of light, which is the speed of the light pulse as it leaves the source, by the amount of time it takes for the light pulse to return to the source. The LIDAR's light source makes it independent of natural light and gives it the advantage of being imperceptible to day and night, as well as weather conditions that may affect the sun's light, such as cloudiness' [5]. In addition, detecting road marking using LIDAR intensity is relatively straightforward as it only consider the reflectance of points that is invariable in scenes as compared to factors such as background lighting and height that may affect cameras [6, 7].

2.1.2 Cameras

Cameras used for road marking detection have either been used in a monocular vision camera set-up or a stereo vision camera set-up [5]. In a stereo vision camera, two cameras are used to obtain 3D information about the environment by reconstructing the 3D geometry of the environment based on the viewpoints of the images captured from the two cameras simultaneously [8]. In monocular camera, a single camera is used to obtain information about the environment. It does not give 3D information about the environment however 3D information can be computed by triangulation over a baseline [9].

Cameras are the most accessible and affordable sensors for detecting road markings and perceiving the road's surface [5]. Since monocular vision cameras lack 3D geometrical information about the environment, many assumptions must be made to detect road surfaces, especially road markings. Stereo vision cameras can be used to acquire 3D information about the environment and, by extension, road markings; however, they have a shorter range and are less precise than LIDAR for obtaining 3D information about the environment [5]. LIDAR is the best suited for road marking detection and extraction as they provide intensity value which is very valuable for separating road markings from non-road markings points on road surfaces [10]. In this research, the focus is more on further harnessing the intensity value of LIDAR for extracting road markings.

2.2 Simultaneous Localisation and Mapping - SLAM

SLAM is the problem in which a robot tries to build a map of its environment while simultaneously trying to localise itself within the environment using this map [11]. In SLAM, the robot neither knows its position and orientation in the environment nor has a prior map of the environment. The robot has to estimate its pose x_k at timestep k and map m given observations $z_{0:k}$ and control input $u_{0:k}$ together with any initial condition x_0. SLAM can be an online SLAM problem or full state SLAM based on the pose estimation measured [11]. Online SLAM is only interested in estimating the posterior location of the robot, x_{k+1} based on the current pose, x_k of the robot at timestep k together with the map; that is, previous measurements are not relevant when estimating the future pose of the robot. Figure 2.1 shows the graphical model of the online SLAM. Equation (2.1) offers the probabilistic interpretation of the online SLAM problem.

$$p\left(\mathbf{x}_k, \mathbf{m} \mid \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0\right) 1001[11] \tag{2.1}$$

where
k is current timestep
$\mathbf{m}$ is the map
$\mathbf{x}_k$ is the pose of the robot at timestep k
$\mathbf{z}_{0:k}$ are the observations at all timestep
$\mathbf{u}_{0:k}$ is the control input to the robot at all timesteps
$\mathbf{x}_0$ is any initial condition

Full-state SLAM, on the other hand, estimates the posterior pose of the robot x_{k+1} based on all previous poses $x_{0:k}$ of the robot till the current timestep k together with the map m. Figure 2.2 shows the graphical model of the full-state map. The equation (2.2) gives the probabilistic interpretation of including all prior states.

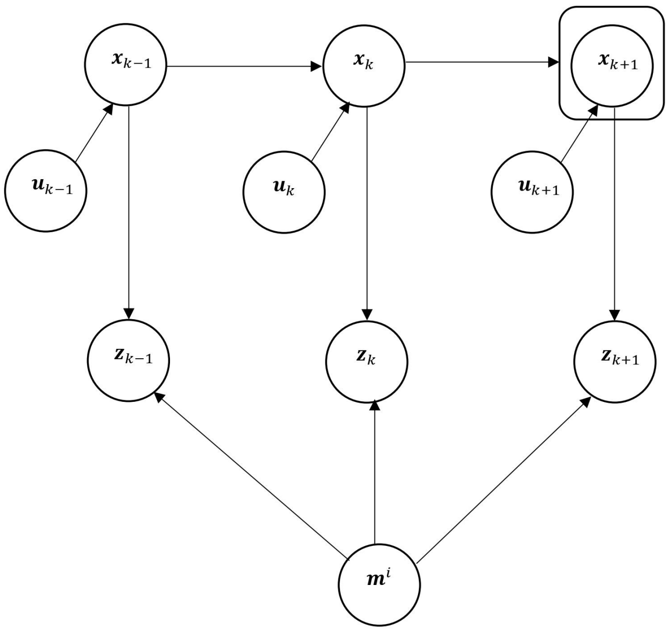


Figure 2.1: The graphical model of Online SLAM Problem [11]

$$p\left(\mathbf{x}_{0:k}, \mathbf{m} \mid \mathbf{z}_{0:k}, \mathbf{u}_{0:k}, \mathbf{x}_0\right) 1001[11] \tag{2.2}$$

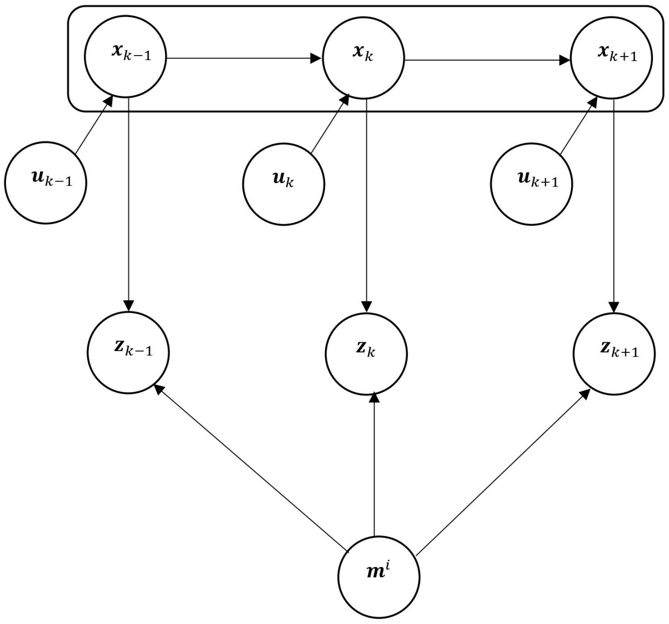


Figure 2.2: The graphical model of Full SLAM Problem [11]

To build a SLAM system, there are two interlinked steps: data association and estimation. Data association is discrete and is is concerned with ensuring the relationship between a detected object and a previously detected object. The SLAM estimation is continuous estimates object location on the map and the robot's pose. The two prevalent methods of dealing with the estimation problem are the Extended Kalman Filter technique and the Graph-Based technique.

2.3 Registration

A large number of point clouds are being processed in this study. A key problem we have to face is registration. Registration is the process of finding transformation between two coordinates such that points representing overlapping areas can align as perfectly as possible [12]. Given a 3D point cloud from a sensor, we want to align it with the 3D point cloud from the map. An overview of registration algorithm utilised i this study is discussed in this section

2.3.1 Iterative Closest Point (ICP)

ICP by is perhaps the most prevalent point cloud registration algorithm. Given two sets of point clouds, ICP works in such a way that it reduces the differences between these two sets of point clouds by finding similarities between these 2 point clouds [13]. This is achieved through the following steps.

- Let the two sets of point clouds to be matched be called Aligned($\mathbf{A}$) and Referenced($\mathbf{R}$). Let the set of points in the Referenced cloud $\mathbf{R} = \{\mathbf{r}_i : i \in [1, N]\}$where N is the total number of points in the cloud, $\mathbf{r}_i$ is a 3D point represented by $\mathbf{r}_i = [r_{xi}, r_{yi}, r_{zi}]$

T$andthepointsintheAlignedcloud\mathbf{A} = \{\mathbf{a}$i:i$\in [1, M]\}$ $A = a_i$, where M is the total number of points in the cloud, $\mathbf{a}_i$ is a 3D point represented by $\mathbf{a}_i = [a_{xi}, a_{yi}, a_{zi}]$

T.

- For each Iteration, the registration parameter(scaling, translation and rotation parameters) and registration error are initialised. The initial registration error is set to ∞. The registration error reflects the dissimilarity between the two point clouds.
- For each point, $\mathbf{a}_i$, in $\mathbf{A}$, the corresponding closest point $\mathbf{r}_j$ in $\mathbf{R}$ is found using the Euclidean Distance in equation (2.3) and (2.4):

$$d(\mathbf{a}_i, \mathbf{r}_j) = \|a_i - r_j\| \tag{2.3}$$

$$d(\mathbf{a}_i, \mathbf{r}_j) = \sqrt{(a_{xi} - r_{xj})^2 + (a_{yi} - r_{yj})^2 + (a_{zi} - r_{zj})^2} \tag{2.4}$$

The corresponding closest point to a_i is, therefore, the point r_j that gives the minimum distance after computing the distance to all points in R. This is given by equation (2.5) and (2.6):

$$d(\mathbf{a}_i, \mathbf{R}) = \arg \min_{j=1,\ldots N} d(a_i, r_j) \tag{2.5}$$

$$d(\mathbf{a}_i, \mathbf{R}) = \arg \min_{j=1,\ldots N} \|a_i - r_j\| \tag{2.6}$$

- Next, compute the registration parameters(scaling, rotation and translation) obtained after aligning the closest points. This is computed by calculating the transformation matrix between each point in the reference and aligned clouds and choosing the transformation matrix that minimises the error the 2 points. This is done by computing the centroid of corresponding points and shifting the points towards each other. This gives the translation component of the registration parameter; the optimal rotation of the aligned point cloud is computed using Singular Value Decomposition.
- The alignment is applied to the Aligned point cloud, and the registration error between the two point cloud(Aligned and Referenced) is recomputed using sum of squared errors as shown in equation (2.7):

  $$E_r = \sum_{i=1}^{M} \|e_i\|^2 \tag{2.7}$$

  where E_r is the registration error between the two point cloud e_i is the difference between the reference point cloud R and the transformed version of the aligned point cloud A.
- If this error is greater than an allowable threshold - an error threshold value that almost matches the two clouds (in this case, the ideal error value should be 0), then repeat the steps until the error is less than the threshold value.

The ICP algorithm does a fine registration of points using the steps outlined. The ICP algorithm is time-consuming and computationally expensive as it computes the closest points in every iteration. It requires a reasonable initial estimate or initialisation; otherwise, it gets stuck in local minima.

2.3.2 Generalised Iterative Closest Point (GICP)

The second type of registration to be discussed is the GICP. The GICP is a variant of the ICP developed by Segal, A.V et al.; it considers that the scanned objects are surfaces and not distinct individual points, which the standard ICP uses [14]. The standard ICP attempts to minimise the distance between sets of points. With the generalised ICP, however, the fact that the measured points lie somewhere on a surface implies that what is being minimised is not only the distance but the projection of the point-to-point error vector onto the surface normal. Hence, the Generalised ICP combines the point-to-point approach, point-to-plane approach, and plane-to-plane approach and represents points on surfaces locally by covariance matrices. The corresponding point pairing between the two point clouds then considers this covariance matrix's shape. Based on the shape of this covariance matrix, the cost function can be viewed as a point-to-point, point-to-plane or plane-to-plane association. GICP is a unifying metric for ICP that considers the shape of the surface before defining the cost function of the ICP. The GICP is utilised in this study for registering point cloud in two consecutive frames.

2.4 Related Works on Road Marking Extraction

This section explores related works on road marking extractions. Most of the current approaches have solely been on how to extract road markings without incorporating SLAM into the approaches.

Most of the studies on road markings extraction have either focused on extracting road features from cameras(image and video based) or LIDAR. In one of the works done on extracting from the camera, Hernandez, et al., developed a road marking extraction strategy from a vehicle-mounted camera; they extracted the road surface based on a model defined using a stopping distance approach. The lanes were then extracted by integrating edge and colour information. These lanes are grouped together and fitted to a line [15]. The disadvantage of their approach is that it focuses on road lanes and does not extract other valuable road features. Similarly, Chen, X. et al. extracted road markings from a camera using inverse perspective mapping [16] . The inverse perspective mapping converts an image from a camera view to an orthogonal view, and OTSU was used to segment the road markings. The disadvantage of image or video-based extraction techniques is that they do not have or capture geometrical information and are influenced greatly by environmental factors such as weather conditions.

Some works have been done on road marking extraction based on LIDAR, with a particular focus on taking advantage of LIDAR intensity data. LIDAR works better under different weather scenarios, and geometrical information is more easily obtainable; hence the LIDAR is better suited for road marking extraction.

Road Markings extraction based on LIDAR data can be further divided into two broad categories - 2D raster image extraction based on Intensity and 3D point cloud based road marking extraction.

Guan, H. et al. and Kumar, P. et al. used the 2D raster approach to extract road markings from LIDAR data [1,17]. Firstly, they extract road surfaces from LIDAR data using a curb-based approach. The curb-based approach detects curbs on the LIDAR data by using scan lines to find small height jumps on road surfaces. The point cloud data detected as road surfaces are converted into "geo-referenced" intensity images using an inverse distance weighted method. Road markings are obtained from these "geo-referenced" images using a "point-density-dependent multi-threshold" segmentation [17]. OTSU is used to improve the extraction process further while morphological operations are carried out to remove any remaining noise. Soilan, M. et al. also utilised the rasterisation approach to road marking extraction; however, they used a gaussian mixture model to model the intensity in 2 classes - curb class and road marking class [18]. The points belonging to the curb class were filtered out, and the road marking was extracted using OTSU. Cheng, M. et al. extracted road marking in a similar way to Guan, H. et al. and Kumar, P. et al., however, before generating the geo-referenced intensity image, they corrected the raw intensity data using a scan-angle-rank-based method [19]. This scan angle rank-based method corrects intensity variation caused by the varying angle of incidence. This was done by using a median filtering approach and region-growing segmentation.

As for 3D point cloud road marking extractions, Yu, Y. et al., detect road surfaces using the curb-based approach used by Guan, H. et al. and Kumar, P. et al [20] . Next, The road marking points are obtained by segmenting the road surface extraction based on OTSU. Jeong, J. et al. took a similar approach however they included in their approach intensity calibration before extracting road marking using OTSU [21].

Some researchers have presented new learning-based approaches such as He, B. et al., and Wen, C. et al [22,23]. He, B. et al., implemented a road marking extraction using Convolutional Neural

Network [22] . In their work, they preprocess a raw point cloud data, an orthographic projection of the road surface along the z-axis was used to acquire the topview reflectivity image [22]. Using the mean and standard deviation of all point clouds, they normalised the intensity values and rescaled them to a range of 0-255. The normalised intensity values are passed to a CNN model with five convolutional layers for road marking detection. The CNN model was trained on 2729 labelled intensity images and was tested on 1000 images for correctness and accuracy. The challenge with the learning-based approaches is that they require preprocessing of the raw intensity data, which may lead to sparsity of points. hence the neural network cannot learn the intended features from this sparse data. Also, the training data is insufficient to obtain a model that can learn the right features to extract adequately.

From all the existing literature, it is evident that all road marking extraction procedures require road surface detection from the LIDAR data. This may be done using a curb-detection different approach, as some researchers have done. Next, these road markings are extracted from the road surfaces utilising a segmentation approach, primarily applying OTSU to segment road marking points from non-road marking points. Additional steps can be taken to precisely extract the road marking points. These steps were taken into account when designing the methodology for this study.

2.5 Evaluation of Road Marking Extraction

Road Marking extractions are generally evaluated based on three accuracy metrics first suggested by Heipke C. et al. These accuracy measures are the completeness metric, the correctness metric, and the quality metric [24]. The completeness, or recall, measures how complete the extracted road markings are against a ground truth map and is computed using equation 2.8. The correctness metric, or precision, measures the number of points correctly classified compared to a ground truth map and is computed using equation 2.9. The quality metrics is a measure of the proportion of correctly classified point to the total number of points and is given by equation 2.11.

$$\text{Recall } = \frac{TP}{TP+FN} \tag{2.8}$$

$$\text{Precision } = \frac{TP}{TP+FP} \tag{2.9}$$

$$\text{Quality } = \frac{TP}{TP+FP+FN} \tag{2.10}$$

Where,
TP is the number of True Positives - points both in the map obtained using road marking extraction and the ground truth map.
FP is the number of False Positives - points detected in the map obtained using road marking extraction algorithm but does not exist on the ground truth map.
FN is the number of False Negatives - points in the ground truth map not found in the map obtained using road marking extraction algorithm

Pan Y., et al. and Fernández-Arango D. et al., have used the F1-Score given by equation 2.11 as their quality metric. The F1-score is the harmonic mean of the precision and recall [25, 26].

$$F1-\text{ score }=\frac{2\times\text{ precision }\times\text{ recall}}{\text{precision }+\text{ recall}} \tag{2.11}$$

These metrics will also be used to evaluate the road marking extraction algorithm developed as part of this study.

Chapter 3

HDL Graph SLAM

In this chapter, I describe in detail the HDL graph SLAM upon which this study primarily depends. Section 3.1 describes the HDL graph SLAM system and explains its map-building process. In section 3.3, the functionality of the HDL graph SLAM ROS package and the nodelets within the package that help with map building is elucidated.

3.1 HDL Graph SLAM Description

HDL graph SLAM, created by Koide, K. et al is a graph-based SLAM system for building a 3D environment map [27]. It reduces rotational error that comes with scan matching using a ground plane and GPS position constraints in outdoor and indoor applications. These constraints make its map building more precise than other SLAM frameworks like BLAM(Berkeley Localisation and Mapping) and LeGO LOAM which can occasionally fail to find observer trajectories due to accumulation of rotational error [27]. It was specifically designed to be a people tracking system; however, due to the robustness of its algorithmic design, it has become valuable within the research community for building 3D maps of environments. Figure 3.1 and 3.2 shows the HDL Graph SLAM System Overview and pose estimation system.

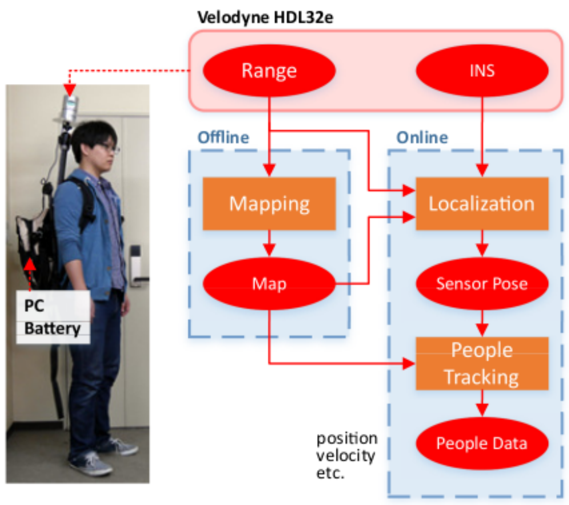


Figure 3.1: System Overview of HDL Graph SLAM [27]

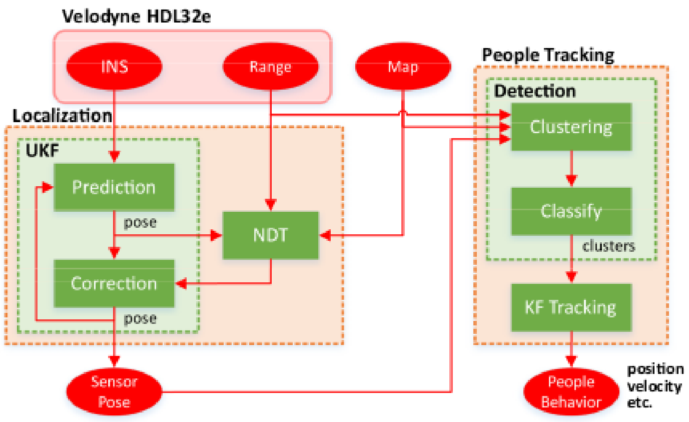


Figure 3.2: The Pose Estimation System of the HDL Graph SLAM Algorithm [27]

3.2 Map Building using HDL Graph SLAM

As mentioned in the previous section, HDL graph SLAM builds a 3D map of the environment using a graph optimisation-based approach. Graph-based SLAM specifies the graph nodes as the robots' poses during mapping and landmark locations. The edges between two nodes specify the constraints between the nodes, the landmark observation and odometry measurements. Accordingly, the nodes in the HDL graph SLAM are the sensors' poses, and landmark positions observed at different time intervals, while the edges depict constraints such as the relative pose between sensors and landmarks. A graph-based SLAM aims to build a graph with nodes that minimise the errors introduced by constraints. For HDL graph SLAM, this is done by minimising the objective function given by equation (3.1) [27] using Gauss-Newton or Levenberg-Marquardt algorithms.

$$F(x) = \sum e_k\left(\boldsymbol{x}_k, \boldsymbol{z}_k\right)^T \Omega_k \boldsymbol{e}_k\left(\boldsymbol{x}_k, \boldsymbol{z}_k\right) \tag{3.1}$$

Where

$\boldsymbol{x}_k$ is the robot pose at node k

$\boldsymbol{z}_k$ is the mean of the constraints

Ω_k is the information matrix of the constraints

$\boldsymbol{e}_k$ is error function between the current robot pose $\boldsymbol{x}_k$ and constraints $\boldsymbol{z}_k$

Registration algorithms are used to estimate the sensor trajectory between consecutive frames [13, 28]. Ground plane and GPS constraints are used to correct the rotational error that accrues as the map gets larger in indoor and outdoor environments, respectively.

Loops are detected based on the lateral distance between nodes and the trajectory length, as in BLAM [29]. A loop fitness score is used to check to see whether it is a loop or not. if the loop detected has a fitness score greater than 0.2, it is counted as a loop and added as an edge to the graph. Figure 3.3 shows the pose graph structure of the HDL graph SLAM. The g2o framework is used for pose graph optimisation [30].

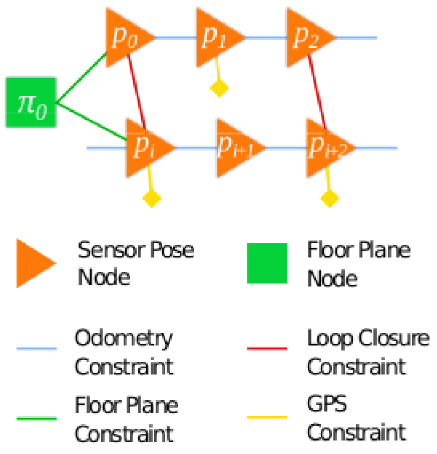


Figure 3.3: Graph Structure of HDL Graph SLAM [27]

3.3 HDL Graph SLAM Package

The HDL graph SLAM has been implemented as an open-source package and made publicly available for use and development on GitHub [31]. This package was developed using ROS which has a tremendous community that has ensured its continued growth and sustained relevance [32]. The package was extensively used in this research; hence it is necessary to describe the functionalities of the nodelets making up this package. Figure 3.4 shows the block diagram of these implementations. This package has four nodelets:

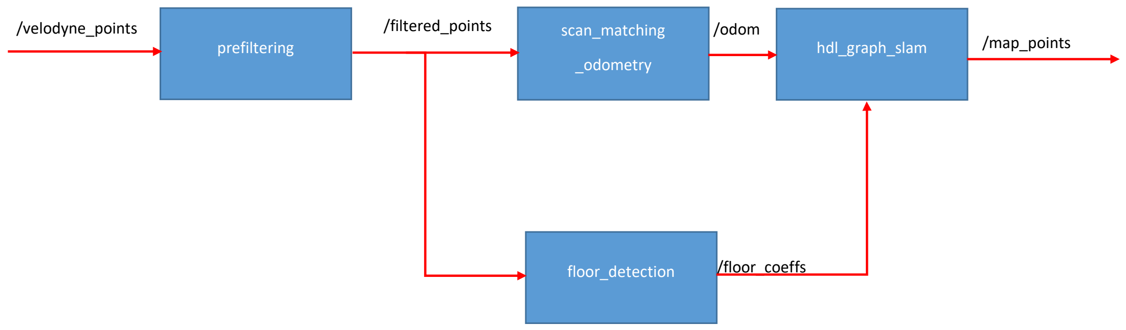


Figure 3.4: Block Diagram of the HDL Graph SLAM Package [31]

3.3.1 Prefiltering Nodelet

This nodelet reduces noisy points from the input point cloud data before other nodes process it for localisation and mapping. The nodelet uses three filters to prefilter the input point cloud:

Distance Filter

It is used to eliminates points that are either too close or too far from the LIDAR sensor since the measured values of these points are unreliable. It uses a near and far distance threshold to limit the scan range of the LIDAR sensor and processes points within a set range value. The near-distance threshold excludes points too close to the sensors and reduces saturation. Saturation is the condition where objects close to the LiDAR sensor surpass the detection threshold, thereby shortening the pulse of the incident light wave and resulting in an inaccurate estimate of the object's actual position. This occurs when the target object is highly reflective. The detection threshold is a threshold value within the sensor receiver used to determine whether a signal is real or noise [33]. The receiver is calibrated with sufficient sensitivity to identify weaker signals from less reflective targets. Objects close to a LIDAR sensor that are highly reflective can readily approach the detection threshold, resulting in saturation [34]. The far-distance threshold ensures that points far away are not captured, reducing blooming. Blooming occurs when highly reflective objects are situated far from the LIDAR sensor, thereby making returns from the neighbouring target larger than they are [35]. This is because the incident pulse gets weaker and splits with distance, hence resulting in the incident pulse, which is focused on a less reflective neighbour, returning higher reflectance because part of its pulse has fallen on the object with higher reflectivity and hence gives a higher than the actual value for this less reflective object [35].

Voxel Grid Filter

The purpose of this filter is to downsample or reduce the number of points at different distances; this is because the point cloud has more densely packed points for closer points and less densely packed points for farther points. Voxel grid filtering ensures that the number of points at each distance is balanced and proportional. This prevents large local weights and reduces the number of points without a loss of performance, increasing the computational speed of processing the point cloud.

The functionality of voxel grid filter from the PCL library is applied to do this [36]. The PCL voxel grid filter creates a 3D cube-like boxes called voxels over the input cloud data. The large voxel is further split into smaller voxels depending on the value of the voxel sampling size(leaf size). The centroids of all points in each small voxel is calculated. This centroid is used to downsample the points by approximating all the points within the voxels towards the coordinate of the centroids. The leaf size is essential in determining the number of points retained. If the leaf size is too large, the point cloud data loses too much information and may not represent the original point cloud data again. A smaller leaf size ensures that the voxelised cloud retains most of the information about the input cloud data.

Outlier Removal Filter

This is used to remove isolated points which may be left after filtering using the distance and voxel grid filters. The outlier removal can be done using either a statistical outlier removal or radius-based outlier removal. Depending on the application, the PCL Statistical Outlier Removal or Radius Outlier Filters are used for this application [37, 38]. The PCL Statistical outlier removal performs the k-Nearest Neighbour search for each point and calculates the distance of a point to its Neighbours(k-Neighbours) while computing the mean and standard deviation of all neighbour point distances [37]. It uses this mean value and standard deviation scaled by a factor as a criterion to determine whether to accept or reject a particular point. The equation representing this criteria is given in equation (3.2). If the average distance of a point is greater than τ, this point is rejected, classified as an outlier and removed from the dataset. This is done for all points in the input point cloud data.

$$\tau = \mu + \alpha \times \sigma \tag{3.2}$$

where,
τ is the criterion for determining whether a point is an outlier
μ is the mean of the distances of the K-Nearest Neighbour points
σ is the standard deviation of the K-Nearest Neighbour points
α is the scaling factor

The Radius Outlier Removal specifies a radius threshold for each point and a minimum number of points within its neighbourhood to avoid being classified as an outlier, The Euclidean distance of the point is computed, and the radius threshold and the number of points requirements are examined to determine if they are met.

3.3.2 Scan Matching Nodelet

The primary focus of the scan matching nodelet is to do point cloud registration of data of two consecutive frames in order to obtain the motion in the two frames using point cloud registration algorithms like the GICP [14].

3.3.3 Floor Detection Nodelet

After the prefiltering steps, the floor is detected using either the ground plane or GPS constraints. Defining a ground plane constraint that makes floor detection possible is necessary. For this step to be carried out, an assumption that a globally consistent ground exists has to be made. This ground has some parameters which make detecting ground feature possible. When there is a new LIDAR scan data, the ground features are detected, and the pose is corrected according to the detected ground features.

Ground Plane Detection

The floor detection in HDL graph SLAM detects the ground plane by segmenting the ground according to a height parameter using the PCL PlaneClipper3D function to detect the ground plane [39]. This library considers that the distance between the LIDAR and the ground is within a certain height threshold range and filters out points above and below the ground leaving points on the ground.

Normal Filtering

Following the ground point detection, the normal vector of each point on the ground is computed, and points with very large angular difference to the ground normal vector are filtered out. The normal vector to points on the ground should be in the same direction as the ground normal vector and should have little angle offset or difference from the ground normal vector. The normal vector computation is done using the PCL Normal Estimation [40], and this filter is applied to points that do not meet this normal filtering criterion.

3.3.4 Plane Fitting using RANSAC

Random Sample Consensus (RANSAC) is used to fit the remaining points to the ground plane parameters. The PCL RANSAC function is utilised for this purpose [41]. RANSAC distinguishes between data inliers and outliers [42]. It then computes the model parameter to determine how well the data separation matches the desired plane. The model is assessed based on the number of data points that support this plane. This procedure is repeated several times, and the model with the highest score is selected to determine which points are inliers and which are outliers [42]. To ensure the accuracy of the plane, it is also necessary to determine if the angle between the plane and the ground is large and if there are sufficient valid point clouds within the plane.

hdl_graph_slam Nodelet

The HDL graph SLAM nodelet takes in input from the other nodelets to construct the pose graph, detect loop and optimise the system. It uses the pose information of keyframe, which comes from the scan matching nodelet, and the floor plane coefficient, which comes from the floor detection nodelet, as inputs. IMU and GPS are also directly fed to this nodelet as optimised edges. This nodelet then uses the IMU, GPS, ground coefficient and closed-loop detection as optimisation conditions and the corresponding map after optimisation is the output of the nodelet. In order to avoid excessive overlapping of point clouds, octree filtering is used in the output, and the user defines the resolution.

Chapter 4

Road Marking Extraction Implementation

This chapter provides a complete description of the methodology and technique utilised in building a map and extracting road marking in this study. Sections 4.1-4.4 describe the techniques involved in extracting road markings from LIDAR intensity data for mapping and localisation by incorporating the road marking detection and extraction into the HDL graph SLAM framework. These procedures include preprocessing of input point cloud, road surface detection, road markings detection and extraction, floor replacement with road markings and map building with HDL graph SLAM. Figure 4.1 shows the block diagram of the implemented road marking extraction algorithm.

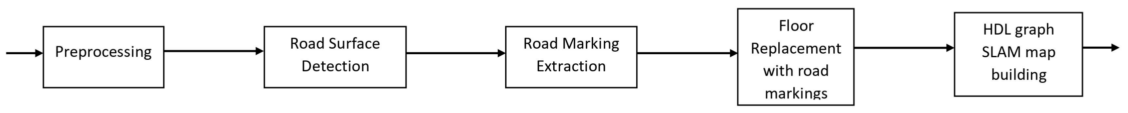

Preprocessing
Road Surface Detection
Road Marking Extraction
Floor Replacement with road markings
HDL graph SLAM map building

Figure 4.1: The Block Diagram of the Road Marking Extraction Algorithm

4.1 Preprocessing Input Point Cloud

Preprocessing the input point cloud data reduces noisy points as well as detects outliers which as a by-product also reduces the number of points in the point cloud. This is also useful to reduce the computational complexity of the SLAM algorithm due to the reduced number of points. The HDL graph SLAM package features were utilised. Three filters within the pre-filtering nodelet were used to reduce and downsample redundant point cloud data.

The distance filter eliminates points according to a distance threshold. A near and far distance threshold of 4m and 40m was set to ensure that only points located on the road and its boundaries are processed. This also prevents blooming and saturation and makes intensity correction easier as discussed in section 3.3.1.

The voxel grid filter downsamples the point cloud using a resolution of 0.1m. The resolution selection is ideal and ensures that input point cloud is further reduced while retaining most of the information contained in the input point cloud.

In the final preprocessing step, the radius outlier filter with a radius threshold of 0.5m, a mean and standard deviation of 30m and 1.2m respectively, was used to filter out isolated points in the raw point cloud. Figure 4.2 shows the point cloud before and after preprocessing operation.

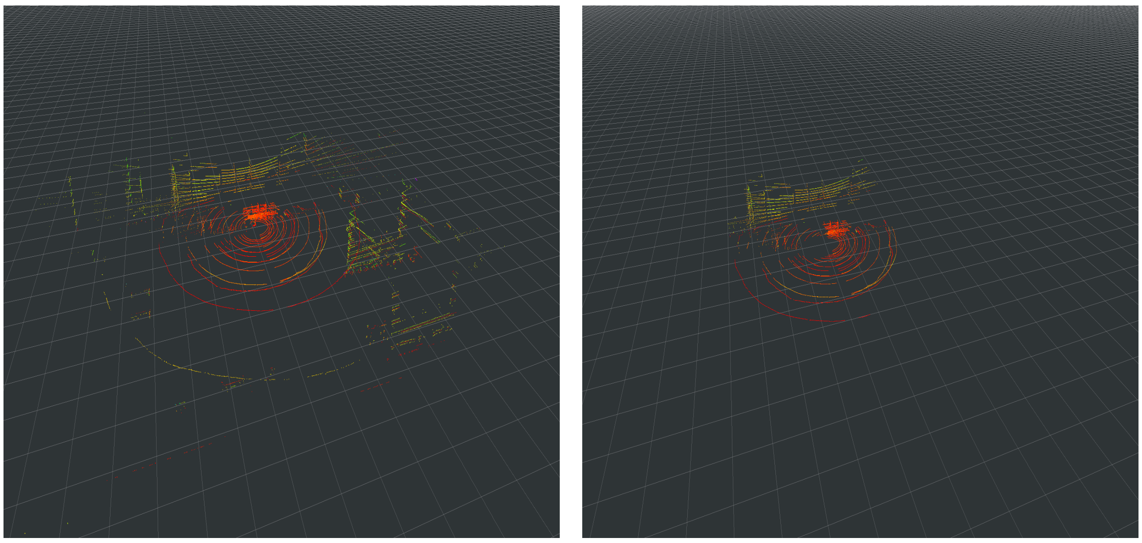


Figure 4.2: LIDAR scans: (a) Before Preprocessing (b) After Preprocessing

4.2 Road Surface Detection

This is an important step as road marking points are contained on road surfaces. The HDL graph SLAM floor detection was used to detect road surfaces. Firstly, a world coordinate frame XYZ called psa_map(or map) frame is assumed where the globally consistent ground is present. The current pose of the LIDAR is resolved to this world frame by using the transformation matrix to convert the pose from the LiDAR frame $x_L y_L z_L$ to the world frame. After which we detect the ground plane with a height parameter of 4m, this height parameter filters out 2m point above and below the ground plane. This height parameter was set to 4m as it is assumed that the LIDAR sensor is at an height of approximately 2m above the ground. The points on the detected ground plane are checked to see if they have angular difference greater than 20° when compared with the ground normal vector. This is to ensure that the floor points are in the same direction as the ground normal vector. The points left after using the normal filter were considered as ground points and were fitted to plane parameters using the PCL RANSAC library. Figure 4.3 shows the extracted road surface.

4.3 Road Marking extraction

The road marking takes in the point cloud after road surface has been detected and attempts to identify, among the floor points, those that can be classified as road marking points based on

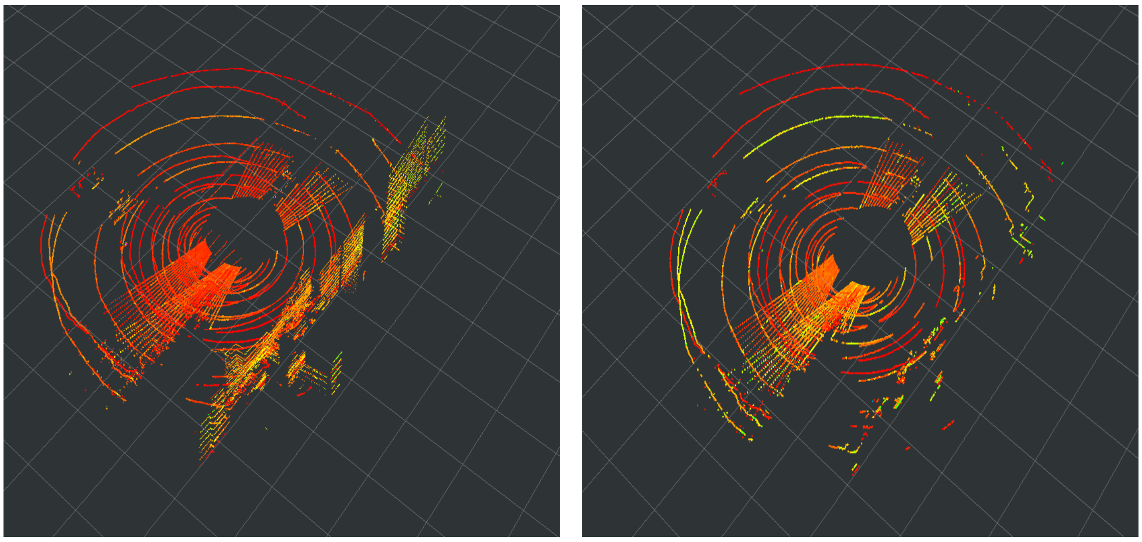


Figure 4.3: (a) Filtered LIDAR Scan (b) Extracted Road Surface from Filtered LIDAR Scan

intensity value. This was accomplished by creating an intensity histogram for each LIDAR scan. The intensity values in this histogram were normalised prior to determining an optimal thresholding value that differentiates road marking features from non-road marking features. Figure 1.1a shows some relevant road marking features we are interested in extracting as compared to non-road marking features shown in Figure 1.1b.

Intensity Thresholding

The optimal threshold value is determined to separate the road marking points into two different classes based on intensity. The intensity data was processed to determine what values were road surface points and what values could be classified as road marking points. It was observed that the histogram of the intensity value is bimodal as shown in Figure 4.4 with the lower intensity values representing road surface points and the higher values representing road marking points. A certain optimal threshold value that correctly classifies the points was recognised from the histogram of the intensity. To maximise the chances of obtaining this threshold value for every LIDAR scan, OTSU Thresholding algorithm was used [43]. A slight modification of the OTSU algorithm by introducing a separability condition used to determine whether the threshold value is ideal or not, gave an excellent threshold value that correctly separates the road markings points from the road surface points. The implementation of the Otsu algorithm is based on the work of Otsu .N [43] and it was computed using the steps outlined:

- The normalised histogram of the intensity was computed by dividing each intensity, i, value by the total number of points, N, on the floor using equation (4.1).

$$p_i = \frac{i}{N} \tag{4.1}$$

- We separated the points in the histogram into 2 classes (road marking points and floor points). We aim to find the optimal discriminant intensity value k between these 2 classes. We computed the probability of points being road marking points and non-road markings

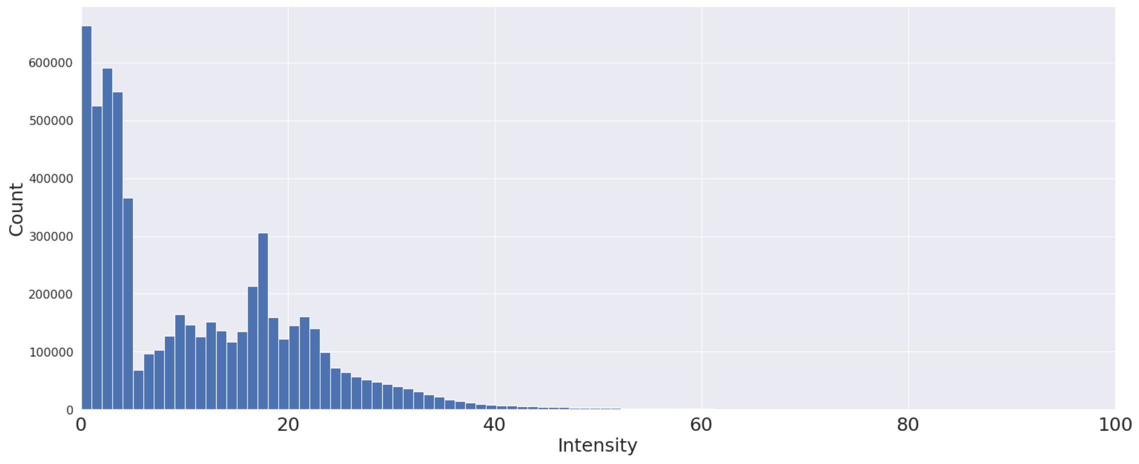


Figure 4.4: Intensity Histogram of Road Surface Points

using the equations (4.2) and (4.3).

$$P(F) = \sum_{i=0}^{k} p_i \tag{4.2}$$

$$P(M) = \sum_{i=k+1}^{L-1} p_i \tag{4.3}$$

where,
$P(M)$ is the probability of an intensity value from a LIDAR channel being a road marking point
$P(F)$ is the probability of an intensity value from a LIDAR channel being a floor point
i is the i^{th} histogram value
p_i is the normalised intensity value of the i^{th} histogram value
k is the upper-intensity value of floor point class
L is the upper-intensity value of road markings point class

The total probability of any point being in the histogram is calculated using equation (4.4).

$$P(T) = \sum_{i=0}^{L-1} p_i = 1 \tag{4.4}$$

From equation (4.2), we can infer that.

$$P(F) = 1 - P(M) \tag{4.5}$$

We computed the cumulative mean values of each class using equations (4.6) and (4.7).

$$\mu(F) = \sum_{i=0}^{k} i p_i \tag{4.6}$$

$$\mu(M) = \sum_{i=k+1}^{L-1} ip_i \tag{4.7}$$

where,
$\mu(F)$ is the cumulative mean of the floor points
and $\mu(M)$ is the cumulative mean of the road markings points

The cumulative mean for the whole histogram is calculated by using equation (4.8).

$$\mu(T) = \sum_{i=0}^{L-1} ip_i \tag{4.8}$$

The class variance for the road markings point class and the floor point class was computed using equations (4.9) and (4.10).

$$\sigma_F^2 = \sum_{i=1}^{k} (i - \mu(F))^2 \, \mathrm{P}(i \mid M) = \sum_{i=1}^{k} (i - \mu(F))^2 \, p_i/P(F) \tag{4.9}$$

$$\sigma_M^2 = \sum_{i=k+1}^{L-1} (i - \mu(M))^2 \, \mathrm{P}(i \mid F) = \sum_{i=1}^{k} (i - \mu(M))^2 \, p_i/P(M) \tag{4.10}$$

where, σF^2 is the variance of the points belonging to the floor point
σM^2 is the variance of the points belonging to the road markings point class
$\mathrm{P}(i \mid F)$ is the probability of the i^{th} histogram intensity value belonging to the floor point class
$\mathrm{P}(i \mid M)$ is the probability of the i^{th} histogram intensity value belonging to the road marking point class

Using 4.11, the total variance over all points was computed.

$$\sigma_R^2 = \sum_{i=0}^{L-1} (i - \mu(T))^2 \, p_i \tag{4.11}$$

The optimal threshold value T_{opt} that separates the histogram into road markings point and floor points is obtained using equation (4.12).

$$T_{opt} = \underset{0 \le k \le L-1}{arg\, max} \sigma_M^2(k) \tag{4.12}$$

T_{opt} occurs at a value of k that maximizes the variance σ_M of a point being a road marking point.

T_{opt} is evaluated using the separability criterion proposed by K. Fukunage [44]. This separability criterion is given by equation (4.13).

$$\eta(k) = \sigma_M^2(k)/\sigma_R^2 \tag{4.13}$$

where

$$0 \leq \eta(k) \leq 1$$

Our optimisation objective was to find k^* that maximises this separability condition. To do this, we ensured that T_{opt} gives a value of η_k that is close to 1. We computed $\eta(T_{opt})$ using equation (4.14) and checked whether $\eta(T_{opt})$ is greater than 0.85.

$$\eta(T_{opt}) = \sigma_M^2(T_{opt})/\sigma_R^2 \tag{4.14}$$

Once this condition is satisfied, T_{opt} is chosen as the optimal value for separating the two classes. If this condition is not satisfied, T_{opt} is multiplied by a constant value of 1.5, and the condition is rechecked. It must be noted that in the case where the condition is not satisfied, there is a possibility of slight misclassification, however, this situation is not common and as such, the T_{opt} in most cases correctly classified the points into the right classes. The steps are constantly repeated for every new LIDAR data obtained. Figure 4.5b shows the output of the road marking extraction using these for a single LIDAR Scan. The points detected corresponds to points in that scan that were classified as road marking points based on the intensity thresholding condition.

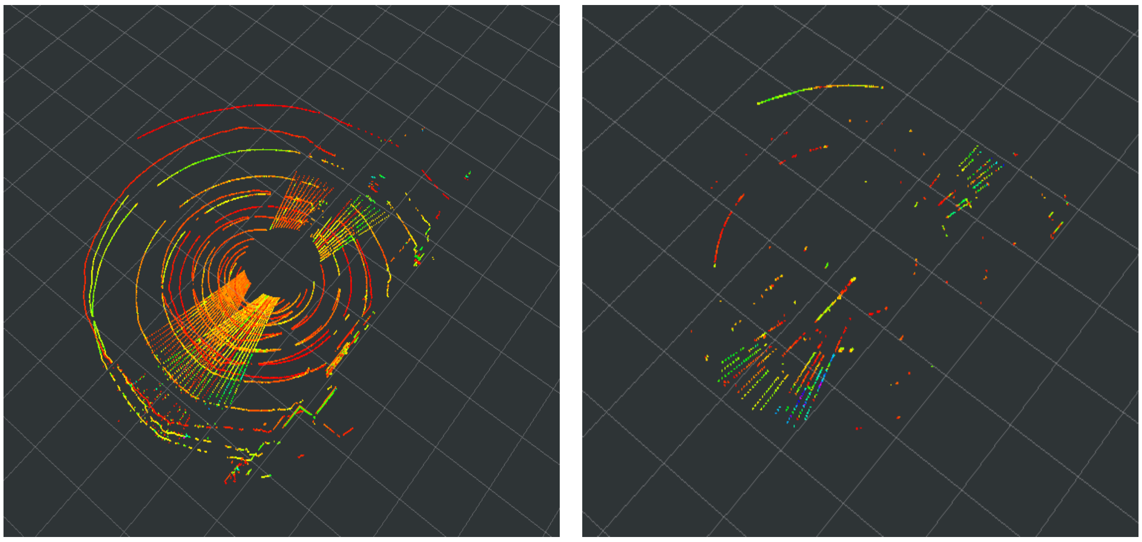


Figure 4.5: (a) Road Surface Points (b) Road Marking Points Extracted based on OTSU Intensity Threshold

4.4 Floor Replacement with Road Marking Points

Following the road marking extraction from the road surface points, the replace floor is used to replace the the points on the road surface with the road markings points obtained. This takes the filtered points obtained from the pre-filtering nodelet and replaces with the road marking points obtained from the road markings extraction nodelet. This is passed to the HDL graph SLAM nodelet which uses the road marking as inputs to construct pose graph detect loop and outputs the corresponding map after optimisation. The overall flowchart showing the interlink between all these steps is shown in Fig 4.8. Figure 4.6 and 4.7 shows the output of the HDL graph SLAM after applying all these filters and threshold. The map has walls and other feature as in the normal HDL graph SLAM processing. However, road surface points have been replaced with the road marking extraction. To evaluate this extraction, the map will be segmented to have only road surface points and tested against a ground truth road markings map.

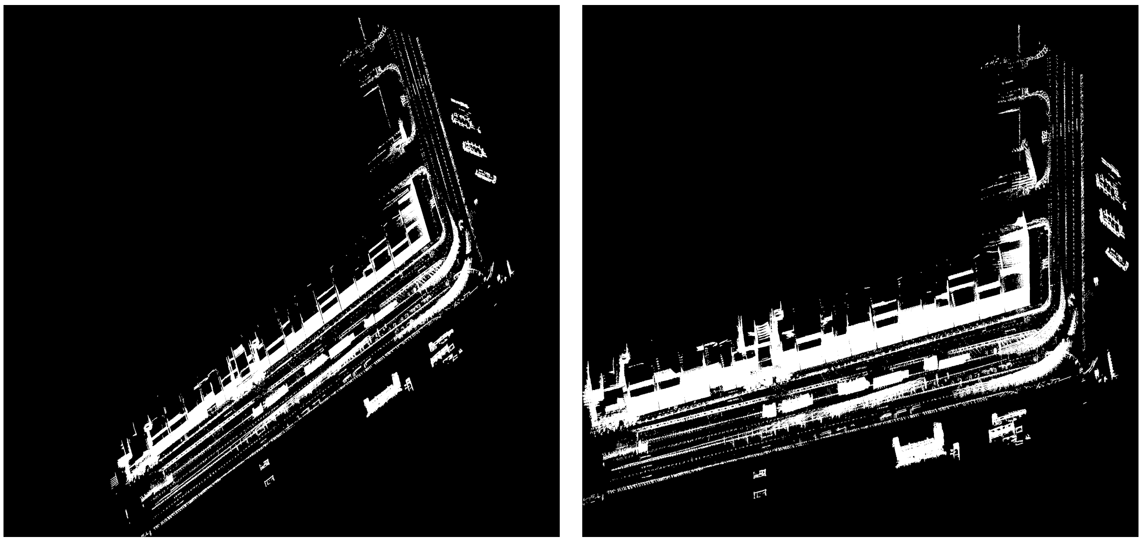


Figure 4.6: Mapping showing the output of HDL graph SLAM with road markings

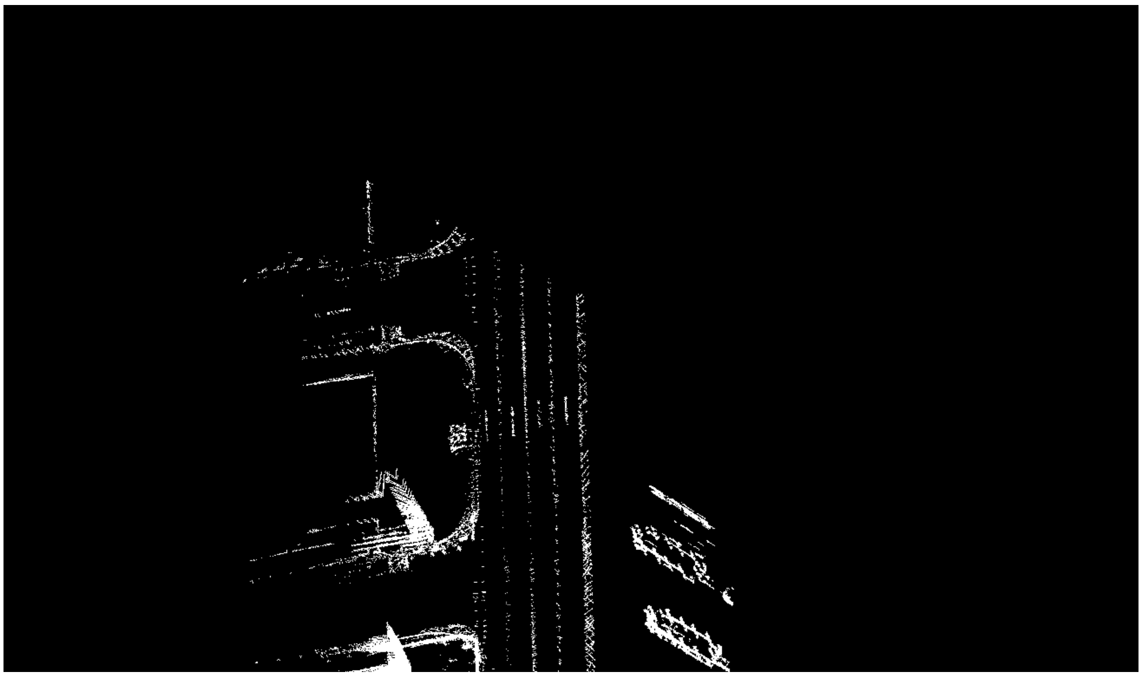
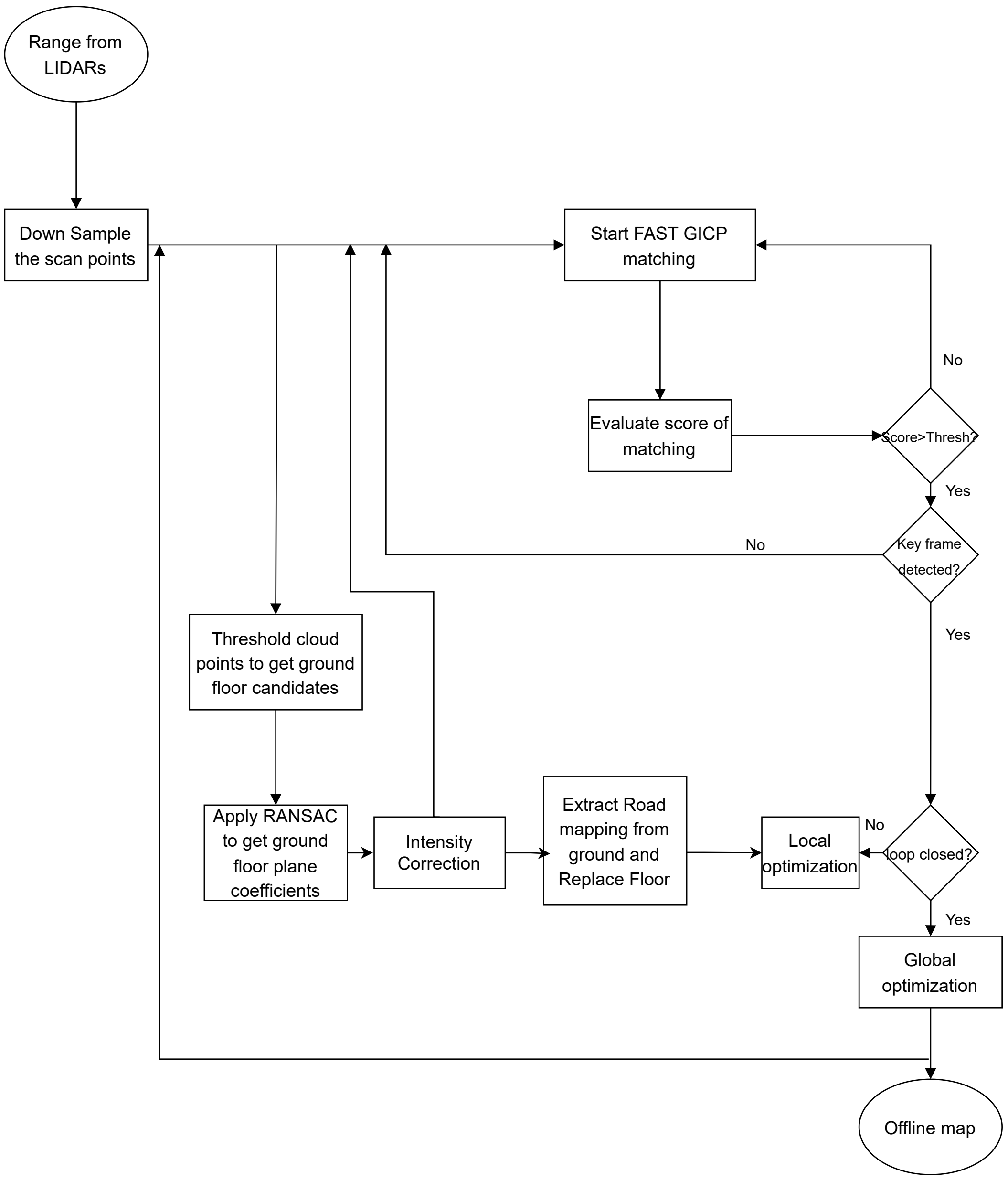


Figure 4.7: Mapping showing the output of HDL graph SLAM with road markings



Range from LIDARs
Down Sample the scan points
Start FAST GICP matching
No
Evaluate score of matching
Score>Thresh?
Yes
No
Key frame detected?
Yes
Threshold cloud points to get ground floor candidates
Apply RANSAC to get ground floor plane coefficients
Intensity Correction
Extract Road mapping from ground and Replace Floor
Local optimization
No
Loop closed?
Yes
Global optimization
Offline map

Figure 4.8: The Flow Chart of the Road Marking Extraction Algorithm integrated into the HDL graph SLAM system

Chapter 5

Experimental Results

In this chapter, the system designed was tested to evaluate its performance. Firstly, the system and dataset used for testing are described. The preprocessing of the ground truth map used for evaluation is discussed. The following sections evaluate the various stages of the applied methodology. The map obtained using the road marking extraction algorithm is preprocessed and aligned to match the ground truth map. In the last section, the map built using the road marking extraction algorithm is evaluated against the ground truth map based on completeness, correctness, and quality metrics, as discussed in section 2.5.

5.1 System Overview

The dataset analysed for this investigation was compiled by AIdrivers at the Singapore Port. Five Robosense RS LIDAR 16 were positioned strategically on top of a moving vehicle to collect data. Three were put on top (in a configuration that was top centre, top left, and top right), and two were put in the bottom left and right configurations, respectively. Regarding locating road markings, the two LIDARs situated at the bottom of the vehicle are more effective than those located higher up, while the 3 LIDARs at the top were more useful for environmental map building. The Robosense RS LIDAR 16 is a 16-channel LIDAR that has a range of 150 metres and is capable of capturing high-resolution images of its surroundings at a frame rate of 20 hertz and a rotational speed of 1200 revolutions per minute [4]. It features a vertical field of vision of 30 degrees and a horizontal field of view that spans 360 degrees [4].

The dataset used in this investigation was gathered on 6th July 2022 and captured as bag files to be processed further; the total number of points included inside these bag files is 8,137,215. This dataset included all the road aspects we are interested in extracting, such as lane markings(broken and continuous), pedestrian crossings, hatch, arrows, words, zigzags, and other pertinent road features.

5.2 Preprocessing the Ground Truth Map

Gpslands Singapore [45], created a 3D ground truth map of the port area used for this project using high-resolution 3D LIDAR. This map contained the point cloud data of the whole port area with its intensity values. There were 28,819,195 points in this dataset. Since the dataset worked with contained only an area of the port, we preprocessed the ground truth map such that only the area of the port the experiment was carried out on was compared with the experimental results. Using the cloudcompare segmentation tool [46], the ground truth map was segmented and thresholded to include only road marking points of the area being compared, resulting in a map with 252,694 points. Figure 5.1 shows the segmented area of the ground truth map that we utilised for evaluation.

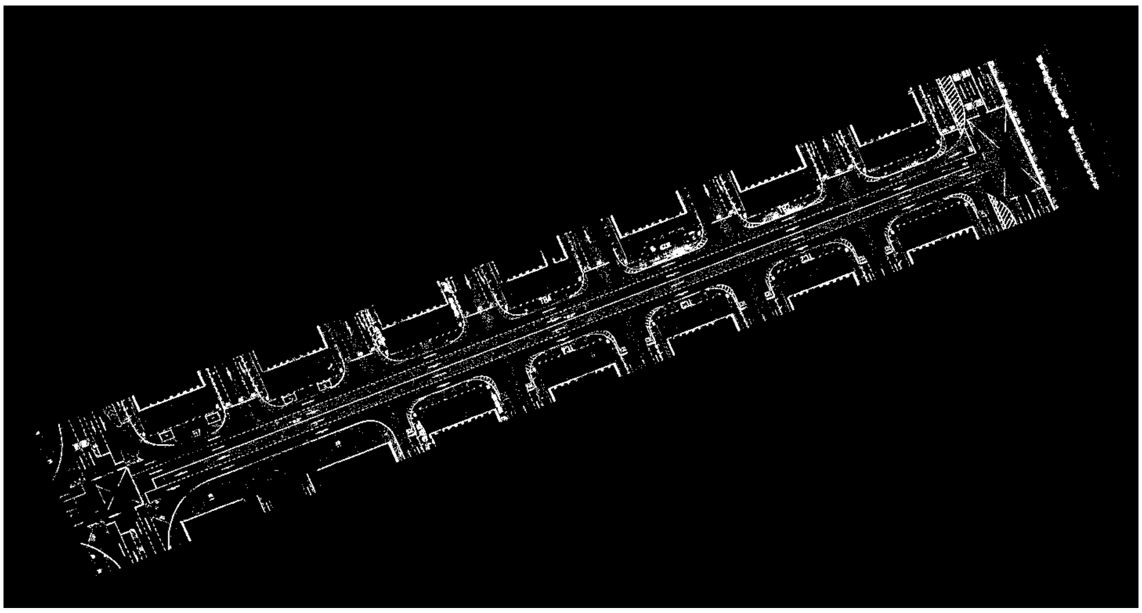


Figure 5.1: Ground Truth Road Marking map of Port.

5.3 Preprocessing the map obtained from Road Marking Extraction Algorithm

The map obtained using our road marking extraction is also preprocessed. Due to the busy nature of the port, during the data collection process, there are dynamic objects within the port area, such as other trucks and containers. These objects sometimes occlude the LIDAR sensor and impact the data collection. Hence, there is a need to eliminate the effects of these objects before evaluating the road marking extraction algorithm used for building this map.

To eliminate these dynamic objects and process only points detected as ground points, a cloth simulation filter (CSF) developed by Zhang, W. et al. is applied using cloudcompare [47]. The CSF separates point cloud data into ground and non-ground points. The CSF plugin in cloudcompare has three parameters: Cloth Resolution, Max Iterations, and Classification thresholds. The cloth resolution parameter is the grid size of the cloth used for filtering the ground and non-ground points; the max iterations refer to the number of times the simulation takes place, while the classification threshold determines the height threshold, which is used to separate the points into ground and non-ground points. The default values for the cloth resolution and Max Iteration were left unchanged at 2 m and 500 iterations, respectively. The height threshold was, however, changed to 0.35m to ensure only points relatively close to the ground are classified as ground points and points above the height are classified as non-ground points. This substantially eliminates the effects of other trucks within the scene and container. The map built using the developed algorithm was subsequently compared with the ground truth map. Figure 5.2 shows the extracted road marking data obtained using this step.

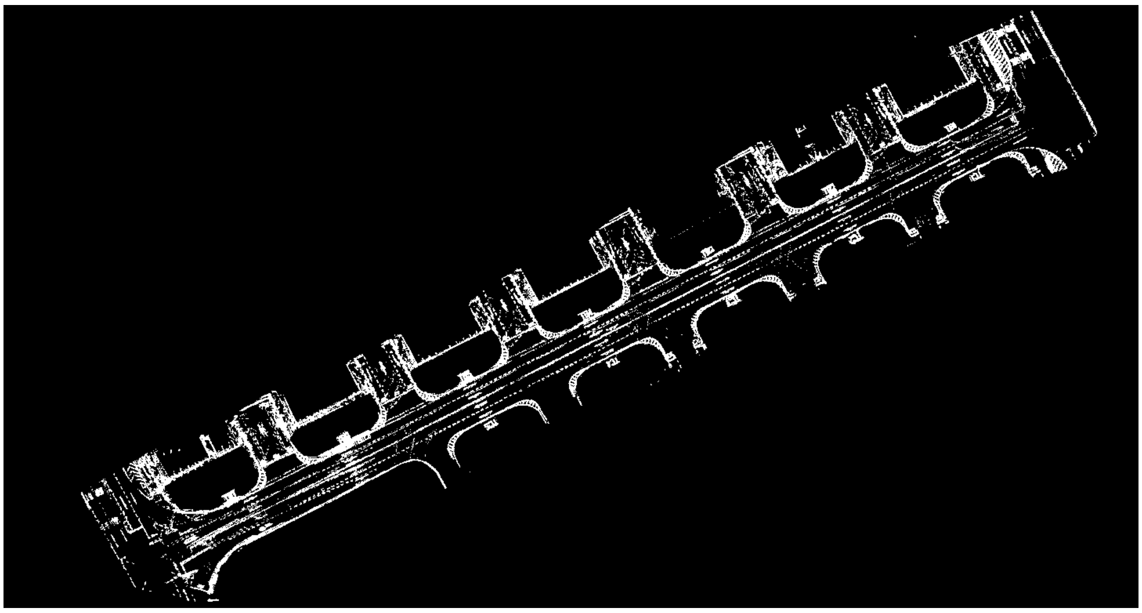


Figure 5.2: Map showing only the extracted road marking points

5.4 Alignment of Map

The last step in preparing the datasets before evaluation involves aligning the ground truth map with the map built with the road marking algorithm. Figure 5.4 shows the ground truth map and the map obtained using the road marking extraction algorithm before alignment. Although the two maps represent the same area, they are in different coordinate frames and must be transformed in order to be compared. ICP, a tool on cloudcompare does a fine registration of two cloud entities and it was used to align these two maps [48]. The ICP works by locating points corresponding to the two maps and calculating the transformation matrix to decrease the distance between them. This is done several times until the two maps overlap. The distance between them is computed by equation (2.3) and (2.4)

The ICP computes the transformation matrix such that the distance between two points on the different maps is minimal. The distance computation optimisation function is given by equation (2.5) and (2.6).

The transformation matrix, **T**, of this alignment is given by equation (5.1)

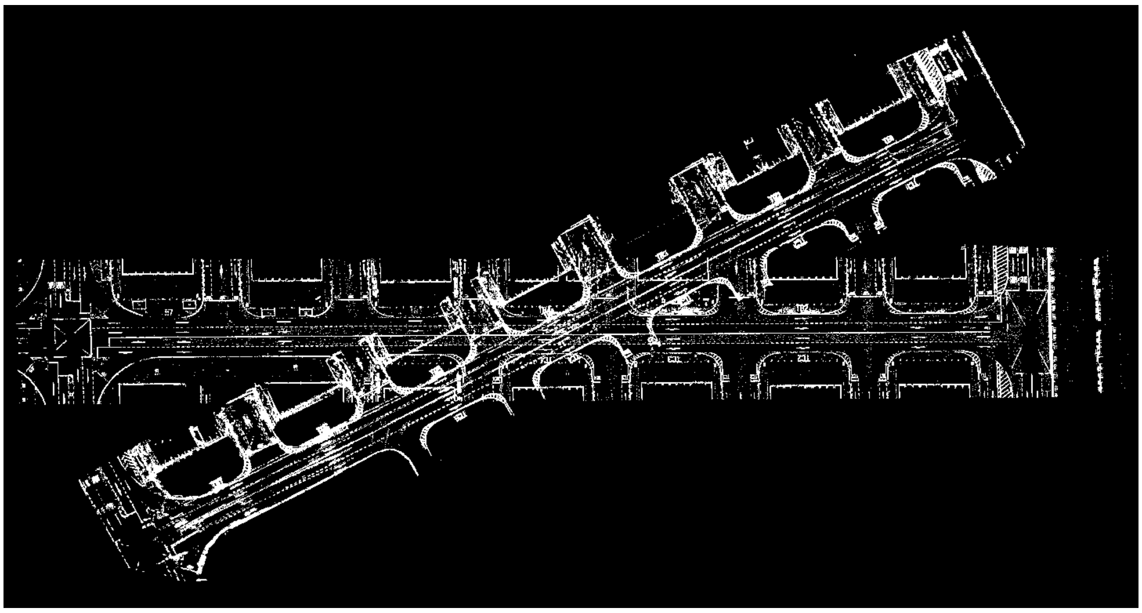


Figure 5.3: Ground Truth and Extracted Road Markings maps before alignment

$$\mathbf{T} = \begin{bmatrix} 1.000 & 0.406 & 0.010 & 103.190 \\ -0.406 & 1.000 & 0.001 & -91.947 \\ -0.009 & -0.005 & 1.000 & -2.973 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{5.1}$$

Figure **??** shows the alignment of the two point clouds The theoretical overlap between the two maps is 100%, which implies that the points are perfectly aligned and the final Root Mean Square (RMS) error difference is 1.16356×10^{-6}.

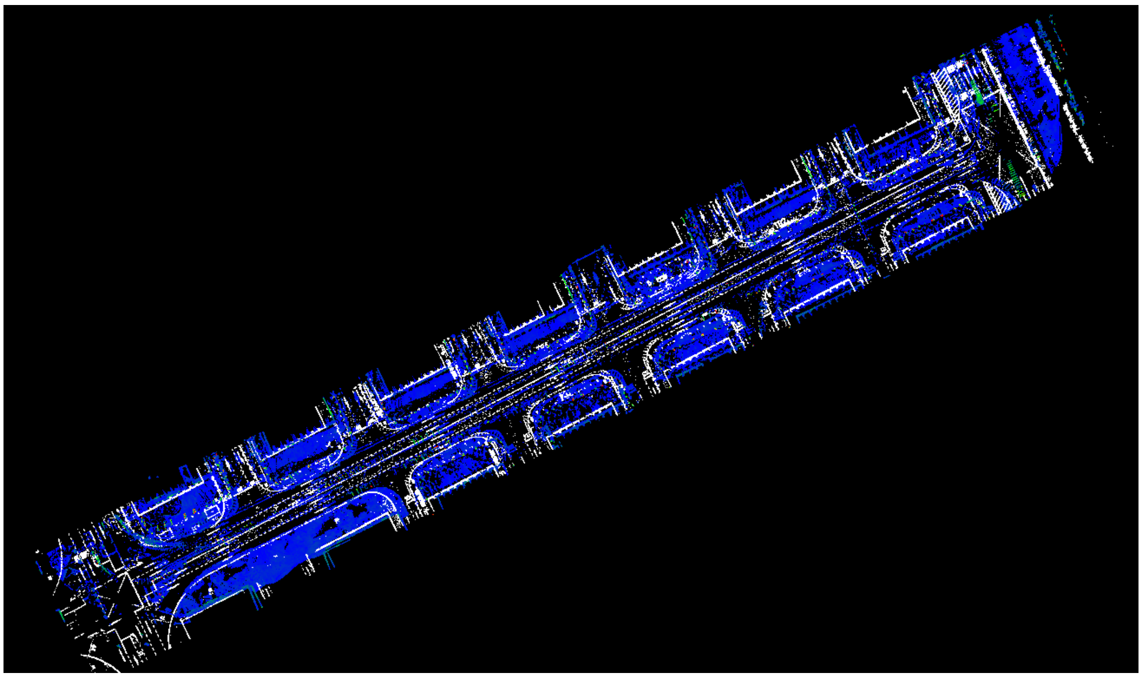


Figure 5.4: Ground Truth and Extracted Road Markings maps after ICP alignment

5.5 Evaluation - Road Markings Extraction

After aligning the two maps, the ground truth map and the map obtained using the road marking extraction algorithm were converted to images and pixel on pixel comparison was done and used to compute the precision, recall and F1 score. The precision, recall and F1- score for the dataset used for evaluation is listed in Table 5.1. The road marking extraction algorithm proposed achieved a precision of 91.62%. This indicates that 91.62% of the point cloud classified as road marking are by the algorithm are indeed road markings point. The road marking extraction algorithm proposed also achieved a recall of 94.03%. This indicates that 94.03% of the point detected in the ground truth data are also detected in the extracted road marking data. Figure 5.5 and 5.6 shows the raw point before applying the road marking algorithm and after the road marking extraction algorithm was applied.

Table 5.1: Quantitative Evaluation of Road Markings Extraction Algorithm

Dataset	TB Area
Raw Point Cloud	8,173,215
Extracted Point Cloud	237,654
Precision (%)	91.62
Recall (%)	94.03
F1-Score(%)	92.81

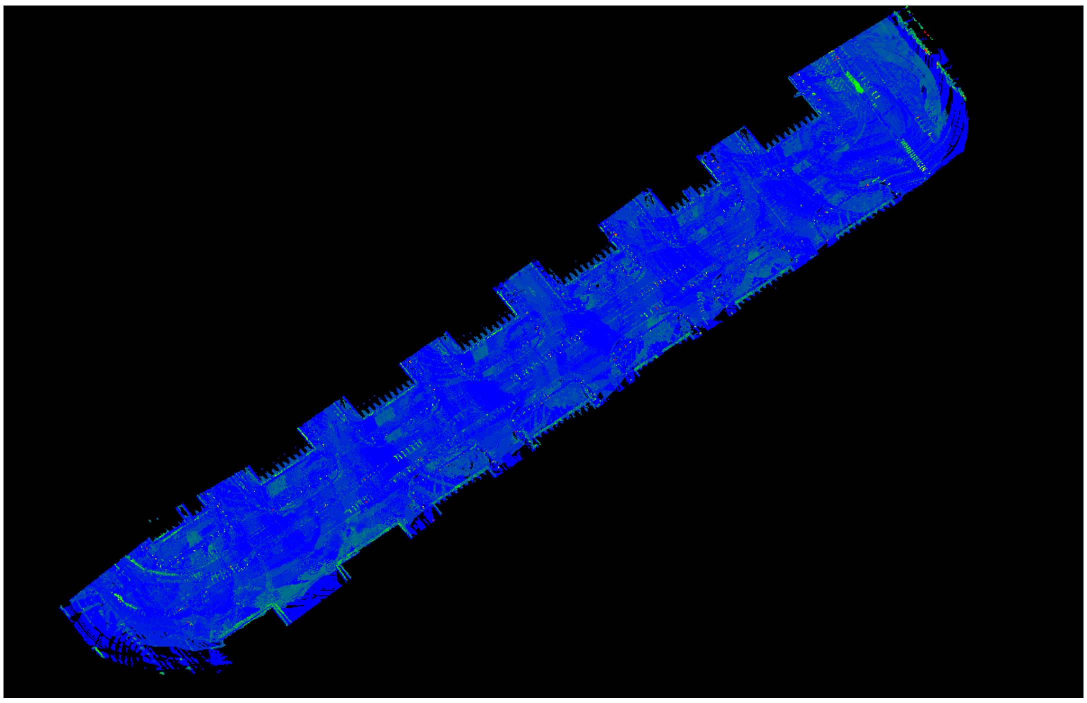


Figure 5.5: Intensity Point Cloud Data Before Applying the Extraction Algorithm value

A fixed threshold over all intensity value was also used to check if the threshold value selected using the OTSU corresponds to the best possible threshold value. Figure 5.6 and 5.7 show the results obtained.

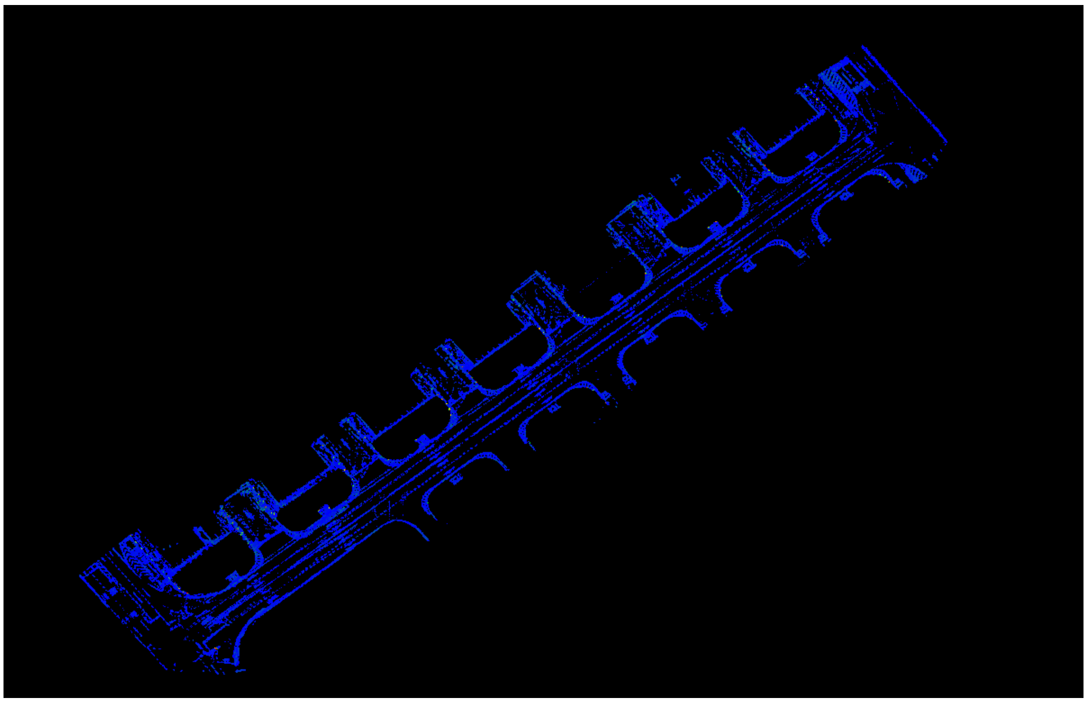


Figure 5.6: Intensity Point Cloud Data After Applying the Extraction Algorithm value

The threshold-precision curve in Figure 5.7 demonstrates that the precision level is relatively high until very high threshold values are reached. This is because the intensity value of road markings is high, and precision measures the accuracy of the classification; therefore, even at the extremely high threshold, the precision score is still high. It can be observed that the precision level starts reducing at a threshold of about 144 in Figure 5.7; this is due to the fact that road marking intensity values rarely exceed this intensity value; thresholding the intensity with this value and testing against a ground truth significantly reduces the number of points correctly classified and consequently reduces the precision score.

For the threshold-recall curve depicted in Figure 5.5, the recall value begins to decrease at a threshold of 30. Beyond this threshold, the completeness of the classification starts to reduce. This threshold value indicates that points beyond the intensity value of 30 are typically not road marking points. Despite the high precision at this level, the number of points classified as road marking points is lower than the actual number of road marking points in the ground truth map. This continues until the recall value is almost 0 at a high-intensity threshold value.

The two figures, however, show that the highest levels of precision and recall occur between intensity values of 0 and 30. Using an intensity threshold between 0 and 20 means that most of the point cloud data is still processed and that nearly all points are utilised in building the map. With the OTSU, however, the threshold is selected such that enough points are eliminated without affecting the road marking detection and extraction process, leading to a high precision and recall rate.

This approach of road marking extraction is able to eliminate the majority of the point cloud while retaining all available scene information like non road surface points like walls and buildings along with the road marking points. As this approach does not involve high level scene information, it is expected to contribute well to the accuracy of the SLAM system while reducing the computational demands of the system by substantial amounts.

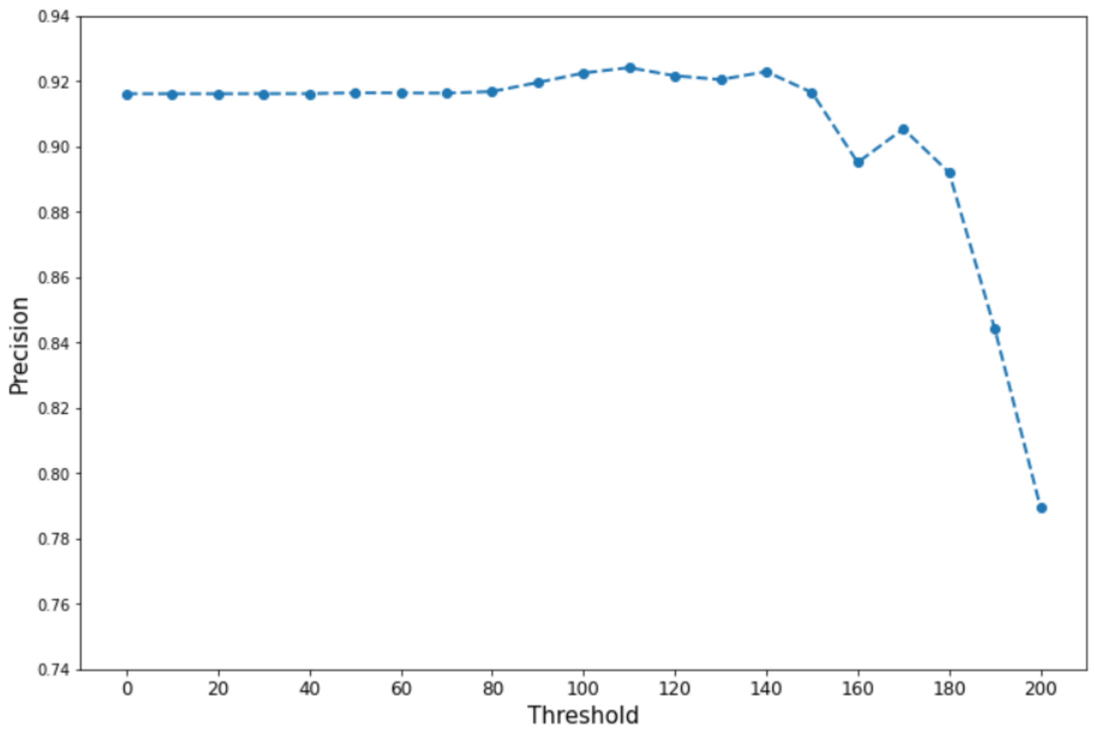


Figure 5.7: Precision-Threshold Curve for constant Intensity Threshold value

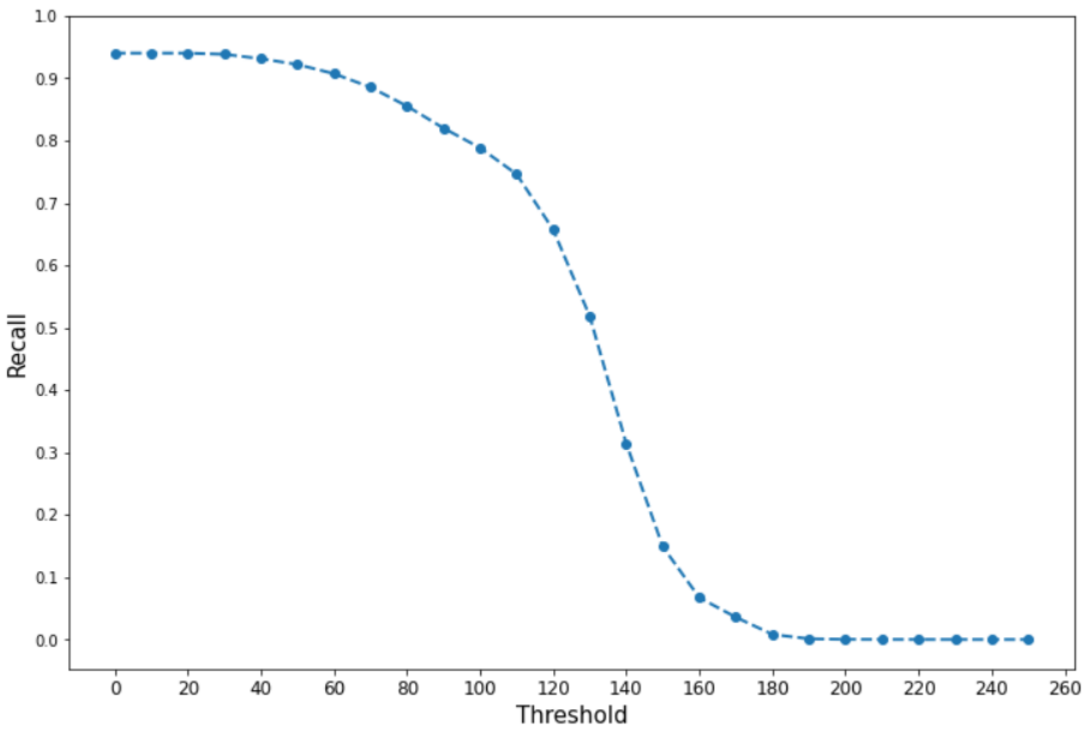


Figure 5.8: Recall-Threshold Curve for constant Intensity Threshold value

Chapter 6

Limitations, Future Works and Conclusion

In this chapter, I discuss this study's contribution and its conclusion. The limitations of this work and suggestions for improvement and future work is also discussed.

6.1 Limitations and Future Works

The road marking extraction algorithm implemented had some limitations. Due to the presence of other vehicles, some road marking points were obstructed. In addition, because detection is based solely on intensity, there is a high likelihood of being unable to detect road marking points if objects with closely similar intensity values are present on the road when the road marking extraction process is being performed. To mitigate these limitations, attempts were made to develop a classification model capable of identifying and classifying these road marking points following intensity thresholding. This attempt, however, had its challenges. After intensity thresholding, the map has significantly fewer points, resulting in the classification model's inability to recognise these road marking points. Future research may investigate how to augment the road marking points after applying an intensity threshold such that all relevant road marking points are accurately represented. This will aid the classification model's recognition and detection of road markings. Although this work relied solely on LIDAR intensity data for detecting road marking features, combining data from other sensors will result in a more effective road marking extraction system.

6.2 Conclusion

A road marking extraction algorithm is presented in this report. This thesis focused on how road marking points can be extracted from raw point cloud data. A systematic approach to extracting road markings in real time was designed and incorporated into the HDL graph SLAM framework. The HDL graph SLAM was used to preprocess the raw LIDAR data and filter out noisy point clouds. The road surface detection was also done with the HDL graph SLAM. Road markings are extracted from the road surface by correcting the intensity values of the road surface and then using a modified version of the OTSU thresholding algorithm to compute the optimal intensity value that separates road marking points from other elements on the road surface. These were passed to the HDL graph SLAM for map building. The extracted road marking was tested against a ground truth dataset of the area used for data collection. The standard evaluation metrics of precision, recall and F1-Score were used to compare the map built using extracted road marking points against a ground truth map. The result produced a precision of 91.62%, a recall of 94.03% and an F1 score of 92.81%. This approach produced a small map size while ensuring that essential road features of interest are captured and reducing the computational complexity of processing all points.

Bibliography

[1] P. Kumar, C. P. McElhinney, P. Lewis, and T. McCarthy, "Automated road markings extraction from mobile laser scanning data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 32, pp. 125–137, 2014.

[2] "Services – paragon," https://paragonroadmarkings.co.uk/our-offer, (Accessed on 09/23/2022).

[3] "Runway surfacing — lancashire — uk - asphalt paving services," https://asphaltpaving.uk/runway-surfacing/, (Accessed on 09/26/2022).

[4] "Robosense lidar rs lidar 16," Available on: https://www.robosense.ai/en/rslidar/RS-LiDAR-16, 2021, (Accessed on 08/31/2022).

[5] A. Bar Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[6] A. S. Huang, D. Moore, M. Antone, E. Olson, and S. Teller, "Finding multiple lanes in urban road networks with vision and lidar," *Autonomous Robots*, vol. 26, no. 2, pp. 103–122, 2009.

[7] S. Kammel and B. Pitzer, "Lidar-based lane marker detection and mapping," in *2008 ieee intelligent vehicles symposium*. IEEE, 2008, pp. 1137–1142.

[8] "3d vision technology principles," https://www.zivid.com/3d-vision-technology-principles, (Accessed on 08/31/2022).

[9] A. Saxena, M. Sun, and A. Y. Ng, "3-d reconstruction from sparse views using monocular vision," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[10] A. Hata and D. Wolf, "Road marking detection using lidar reflective intensity data and its application to vehicle localization," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 584–589.

[11] S. Thrun, W. Burgard, and D. Fox, "Probalistic robotics," *Kybernetes*, 2006.

[12] J. Sanchez, F. Denis, P. Checchin, F. Dupont, and L. Trassoudaine, "Global registration of 3d lidar point clouds based on scene features: Application to structured environments," *Remote Sensing*, vol. 9, no. 10, p. 1014, 2017.

[13] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[14] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[15] D. C. Hernández, D. Seo, and K.-H. Jo, "Robust lane marking detection based on multi-feature fusion," in *2016 9th International Conference on Human System Interactions (HSI)*. IEEE, 2016, pp. 423–428.

[16] X. Chen, B. Kohlmeyer, M. Stroila, N. Alwar, R. Wang, and J. Bach, "Next generation map making: Geo-referenced ground-level lidar point clouds for automatic retro-reflective road feature extraction," in *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2009, pp. 488–491.

[17] H. Guan, J. Li, Y. Yu, C. Wang, M. Chapman, and B. Yang, "Using mobile laser scanning data for automated extraction of road markings," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 93–107, 2014.

[18] M. Soilán, B. Riveiro, J. Martínez-Sánchez, and P. Arias, "Segmentation and classification of road markings using mls data," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 123, pp. 94–103, 2017.

[19] M. Cheng, H. Zhang, C. Wang, and J. Li, "Extraction and classification of road markings using mobile laser scanning point clouds," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 3, pp. 1182–1196, 2016.

[20] Y. Yu, J. Li, H. Guan, F. Jia, and C. Wang, "Learning hierarchical features for automated extraction of road markings from 3-d mobile lidar point clouds," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 2, pp. 709–726, 2014.

[21] J. Jeong and A. Kim, "Lidar intensity calibration for road marking extraction," in *2018 15th International Conference on Ubiquitous Robots (UR)*. IEEE, 2018, pp. 455–460.

[22] B. He, R. Ai, Y. Yan, and X. Lang, "Lane marking detection based on convolution neural network from point clouds," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 2475–2480.

[23] C. Wen, X. Sun, J. Li, C. Wang, Y. Guo, and A. Habib, "A deep learning framework for road marking extraction, classification and completion from mobile laser scanning point clouds," *ISPRS journal of photogrammetry and remote sensing*, vol. 147, pp. 178–192, 2019.

[24] C. Heipke, H. Mayer, C. Wiedemann, and O. Jamet, "Evaluation of automatic road extraction," *International Archives of Photogrammetry and Remote Sensing*, vol. 32, no. 3 SECT 4W2, pp. 151–160, 1997.

[25] D. Fernández-Arango, F.-A. Varela-García, D. González-Aguilera, and S. Lagüela-López, "Automatic generation of urban road 3d models for pedestrian studies from lidar data," *Remote Sensing*, vol. 14, no. 5, p. 1102, 2022.

[26] Y. Pan, B. Yang, S. Li, H. Yang, Z. Dong, and X. Yang, "Automatic road markings extraction, classification and vectorization from mobile laser scanning data." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2019.

[27] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419841532, 2019.

[28] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of Field Robotics*, vol. 24, no. 10, pp. 803–827, 2007.

[29] E. Nelson, "Blam-berkeley localization and mapping," Available on: https://github.com/erik-nelson/blam, 2016, (Accessed on 09/01/2022).

[30] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[31] K. Koide, "Hdl graph slam implementation: A 3d lidar-based graph slam package," Available on: https://github.com/koide3/hdl_graph_slam, 2018, (Accessed on 09/01/2022).

[32] "Ros: Robot operating system," Available on: https://www.ros.org/, 2015, (Accessed on 09/01/2022).

[33] GeoSLAM, "3d scanning glossary: Defining laser scanning jargon," Available on: https://geoslam.com/blog/2016/05/01/laser-scanning-jargon/, May 2016, (Accessed on 09/02/2022).

[34] A. G. Kashani, M. J. Olsen, C. E. Parrish, and N. Wilson, "A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration," *Sensors*, vol. 15, no. 11, pp. 28 099–28 128, 2015.

[35] D. D. Lichti, S. J. Gordon, and T. Tipdecho, "Error models and propagation in directly georeferenced terrestrial laser scanner networks," *Journal of surveying engineering*, vol. 131, no. 4, pp. 135–142, 2005.

[36] PCL, "Point cloud library(pcl):voxel grid function class," Available on: https://tinyurl.com/ar74jj6f, 2022, (Accessed on 08/31/2022).

[37] "Removing outliers using a statistical outlier removal filter," Available on: https://pcl.readthedocs.io/projects/tutorials/en/latest/statistical_outlier.html, (Accessed on 08/31/2022).

[38] "Pcl: Radius outlier removal class reference," Available on: https://tinyurl.com/2p93ujh3, (Accessed on 09/04/2022).

[39] "Point cloud library (pcl): pcl::planeclipper3d class template reference," Available on: https://pointclouds.org/documentation/classpcl_1_1_plane_clipper3_d.html#details, (Accessed on 08/31/2022).

[40] "pcl normal estimation," Available on: https://tinyurl.com/3f8axb72, (Accessed on 08/31/2022).

[41] PCL, "How to use random sample consensus model — pcl documentation," Available on : https://pcl.readthedocs.io/projects/tutorials/en/latest/random_sample_consensus.html, (Accessed on 08/31/2022).

[42] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[43] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[44] K. Fukunaga, *Introduction to statistical pattern recognition.* Elsevier, 2013.

[45] "gpslands projects," Available on: https://www.gpslands.com/projects, (Accessed on 08/29/2022).

[46] "Interactive segmentation tool," Available on: https://www.cloudcompare.org/doc/wiki/index.php/, (Accessed on 08/29/2022).

[47] W. Zhang, J. Qi, P. Wan, H. Wang, D. Xie, X. Wang, and G. Yan, "An easy-to-use airborne lidar data filtering method based on cloth simulation," *Remote sensing*, vol. 8, no. 6, p. 501, 2016.

[48] "Iterative closest point," Available on: https://www.cloudcompare.org/doc/wiki/index.php/ICP, (Accessed on 08/30/2022).

[B11] Dr Mohammed Abdellatif: Personal communication with supervisor. 2022.

Appendix A

Supplementary Material

The code repository is available at *https://github.com/abdulbaasitt/road_markings*.

The dataset used for this study was collected by AIDrivers team and access to it needs authorization from the company.