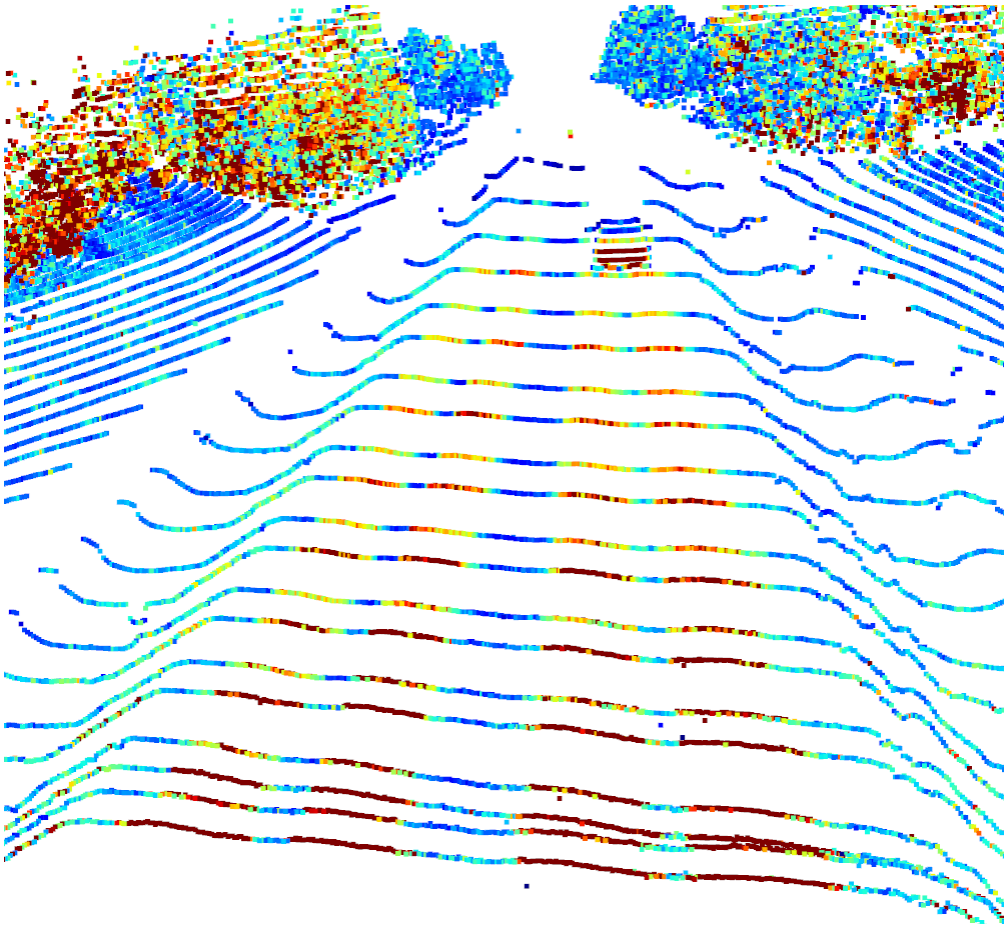


LiDAR-based Road State Estimation

Using LiDAR and Machine Learning to Classify the Road Friction Coefficient and Surface Condition

Master's thesis in Systems, Control and Mechatronics

PHILIP JOHANSSON, VIET LE

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

LiDAR-based Road State Estimation

Using LiDAR and Machine Learning to Classify the Road Friction Coefficient and Surface Condition

Philip Johansson
Viet Le

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2023

LiDAR-based Road State Estimation
Using LiDAR and Machine Learning to Classify the Road Friction Coefficient and Surface Condition
Philip Johansson
Viet Le

Supervisors:
Hasith Karunasekera, Department of Electrical Engineering,
Chalmers University of Technology
Arun Shenbaga Priyan Vijayan, Volvo Cars
Derong Yang, Volvo Cars

Examiner:
Fredrik Kahl, Department of Electrical Engineering,
Chalmers University of Technology

Master's Thesis 2023
Department of Electrical Engineering
Division of Signal processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Reflectance value of every point in a point cloud visualized. Blue points represent low reflectance values while red points represent high reflectance values.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

LiDAR-based Road State Estimation
Using LiDAR and Machine Learning to Classify the Road Friction Coefficient and Surface Condition
Philip Johansson, Viet Le
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This master thesis explores the potential of LiDAR sensors and machine learning techniques for classifying the road friction coefficient and surface condition of road surfaces. The primary focus is to investigate the significance of various features derived from LiDAR data, including the reflectance value. To enable the research, a dataset representing road state was collected, requiring the development of methods for extracting road points and accumulating point clouds over time. Furthermore, a multilayer perceptron and a transformer network were designed and evaluated on the created dataset. The results indicate promising performance of both the multilayer perceptron and the transformer models in distinguishing between road surfaces with high and low friction coefficients, as well as accurately identifying surface conditions associated with each respective class. In this study, dry and moist road surfaces are together associated with the high friction coefficient class. Similarly, snowy and icy surfaces are grouped together and associated the low friction coefficient class. The achieved balanced accuracy and F1-score reached 77.7% and 74.6% when discriminating between high or low friction coefficients, as well as 77.1% and 74.6% when estimating surface conditions within each friction coefficient class. However, when considering the surface condition classes individually, the models could not make accurate classifications. Moreover, the best result was obtained using the reflectance value of LiDAR points in combination with the distances to them. The findings of this research contribute to the understanding of the reflectance value of LiDAR points on various surface conditions. Additionally, the study revealed that utilizing single LiDAR points for estimations yielded adequate results, although a trend was seen where leveraging multiple points indicated an improved performance. The insights gained from this research can guide further advancements related to prediction of the friction coefficient and surface condition.

Keywords: LiDAR, Machine learning, Deep learning, Friction coefficient, Surface condition, Reflectance value, Supervised learning, Classification, Multilayer perceptron, Transformer

Acknowledgements

We would like to express our sincere gratitude to the individuals who have contributed to the completion of this master thesis project. First and foremost, we extend our heartfelt appreciation to our supervisors for their guidance, support, and invaluable expertise throughout the research process. We are immensely grateful to Hasith, Arun and Derong for their valuable insights and contributions. Furthermore, we would also like to acknowledge our examiner, Fredrik, for his involvement and assessment of the thesis. We would also like to extend our gratitude to our fellow classmates and colleagues for their stimulating discussions, suggestions, and encouragement, which have contributed to the development of our ideas. Without the collective efforts and support of all those mentioned above, this research would not have been possible. Thank you all for your contributions.

Philip Johansson and Viet Le, Gothenburg, May 2023

List of Abbreviations

Below is the list of abbreviations that have been used throughout this thesis listed in alphabetical order:

GPF	Ground plane fitting
ICP	Iterative closest point
LiDAR	Light detection and ranging
LPR	Lowest point representative
MLP	Multilayer perceptron
NIR	Near-infrared
RCM	Road condition monitor
ReLU	Rectified linear unit
SGD	Stochastic gradient descent

Contents

List of Figures

List of Tables

1

Introduction

Autonomous vehicles rely heavily on sensors such as radar, LiDAR, camera and ultrasonic sensors to perceive the surrounding environment. Apart from detecting vehicles, pedestrians and obstacles, it is also crucial to know the road state. With the road state information available, the driving can be adapted to improve safety, comfort and energy efficiency. In this thesis, the road state will be defined as the road friction coefficient and the surface condition. The aim is to investigate the potential of combining a LiDAR sensor with machine learning algorithms to estimate the road state.

1.1 Background

Future autonomous vehicles will be equipped with a variety of sensors. One such sensor is the light detection and ranging (LiDAR) sensor that measures the distance to an object and the reflected light [1]. In the field of autonomous driving, data obtained from LiDAR sensors has mainly been used for perception and localization [2]. However, the full potential of LiDAR is still being researched and there is an interest in investigating how LiDAR can be used to estimate the road state.

Methods for road friction coefficient estimation can be classified as either effect-based or cause-based. Effect-based methods make estimations based on the vehicle motion resulting from the contact between the tire and the road, whereas cause-based methods make estimations based on sensor readings, such as those of the road surface ahead of the vehicle [3]. Cause-based methods are important for autonomous vehicles since they allow for using the most optimal control signals based on upcoming road conditions, and are also relevant when estimating the surface condition. Previous cause-based methods for estimating the road state include the use of cameras and optical IR sensors [4]-[8]. However, since cameras require that the surroundings are properly illuminated, they perform worse in dark environments. Additionally, optical IR sensors only focus on a single spot of the road, which means that larger areas can not be observed. Since the LiDAR is not dependent on good lighting conditions, emits light pulses in many directions and has the capability to reach long distances [1], it could potentially be used to solve both weaknesses connected to the camera and optical IR sensor.

1.2 Purpose

The purpose of this study is to investigate what road state information can be extracted from LiDAR sensors. Specifically, the following research questions will be investigated.

- Using a LiDAR sensor, is it possible to estimate road state? How accurately can a LiDAR-based method make such estimations?
- What kind of information from the LiDAR is relevant for estimating the road state? Which properties are critical for accurate road state estimation, and are there any additional properties that can improve the performance?

1.3 Contributions

The objective of this study was to develop a machine learning algorithm for accurate estimation of the road state based on LiDAR data. The main contributions can be categorized into the following two parts, both of which had a significant influence on the outcome:

- **Creating a road state dataset**:
  To train and evaluate a machine learning algorithm for estimation of the road state, it was required to have data with labels that represent the actual road state. Thus, an algorithm for automatically creating a dataset containing LiDAR points labeled with the road state was developed in the first part of the thesis.

  In order to achieve this, two additional algorithms had to be implemented. The purpose of the first algorithm was to extract all LiDAR points belonging to the road from a point cloud. This was done to remove non-road points, which can be considered as noisy observations that do not provide information about the road state. The goal of the second algorithm was to accumulate LiDAR point clouds over multiple time instances into a single point cloud. By transforming this point cloud to a global reference system, the position of the data collection vehicle in the point cloud could be located. This way, measurements of the friction coefficient and surface conditions provided by the data collection vehicle could be matched with LiDAR points.

- **Estimation of the surface condition**:
  The second part of this thesis was to develop and adapt machine learning models capable of processing LiDAR data. Specifically, two models, namely a multilayer perceptron and a transformer network, were examined and tested for their suitability in predicting the friction coefficient and surface condition. Further, the study investigated the impact of various combinations of LiDAR features on the performance of road state estimation.

1.4 Delimitations

To restrict the scope of the thesis, the following limitations was introduced.

- The terms ”road state” and ”surface condition” are broad terms that could be used to describe many properties of the road. However, in this study, only the friction coefficient, as well as dry, moist, wet, slushy, snowy and icy surface conditions will be investigated. Thus, properties such as the roughness of the road, the inclination or deterioration will not be considered.
- When developing machine learning models for the road state estimation, the main focus will be to investigate the potential of such methods for the task at hand. Therefore, less emphasis will be put on finding the optimal method, architecture and parameters.

1.5 Thesis Outline

The thesis is organized into several chapters, each addressing specific aspects of the study.

Chapter 2, ”Sensor Setup”, gives an overview of the main sensors used in this thesis.

Chapter 3, ”Theory”, provides an overview of the relevant theories and concepts related to the methods used to create a road state dataset and the implementation of machine learning models. Furthermore, previous studies describing the use of near-infrared light to estimate the surface condition are presented.

Chapter 4, ”Extracting LiDAR Points on the Road Surface”, explains the methodology used to extract points that belong to the road surface from LiDAR data. The results of the road point extraction technique are also presented.

Chapter 5, ”Point Cloud Accumulation”, focuses on the approach used to accumulate point clouds over time to obtain a denser point cloud. Here, two methods of merging point clouds are introduced, after which their performances are compared.

Chapter 6, ”Road State Dataset”, details the process of creating a dataset containing information about the road state. An overview of the created dataset is then presented.

Chapter 7, ”Effects of Various Road Surfaces on LiDAR Features”, investigates the effects of different surface conditions on two features derived from the LiDAR data, namely the reflectance value and the return index. First, the methodology for examining these features is described, whereafter the results are displayed and analyzed.

Chapter 8, "Road State Estimation", explains the employed machine learning models for estimating the friction coefficient and surface condition, as well as the performed preprocessing methods. Finally, the results are presented.

Chapter 9, "Discussion", discusses the implications of the research findings that were obtained in the previous chapters. The limitations of the research are acknowledged, and potential future directions are suggested. Finally, ethical aspects related to the thesis are discussed.

Chapter 10, "Conclusion", summarizes the key findings of the research. The contributions of the study to the field of road state estimation using LiDAR sensors and machine learning techniques are highlighted. The chapter concludes with implications and recommendations for further advancements in the prediction of friction coefficient and surface condition.

2

Sensor Setup

The sensor setup used in this research is shown in Fig. 2.1. A Luminar Iris LiDAR, emitting light with a wavelength of 1550 nm, is placed at the top above the windshield and has a horizontal field of view of 120° in the driving direction. In the front of the car, a Teconer Road Condition Monitor RCM511 (RCM) [9] is mounted with a field of view directed down at the road directly ahead of the right tire. At the top of the car, an inertial navigation system (INS) is placed, which measures the velocities, accelerations, orientation, angular velocities and more. Specifically, the INS sensor used is part of the OxTS RT3000 series. A camera with view of the driving direction is also located at this position. The following two sections will provide more details about the LiDAR and the RCM.

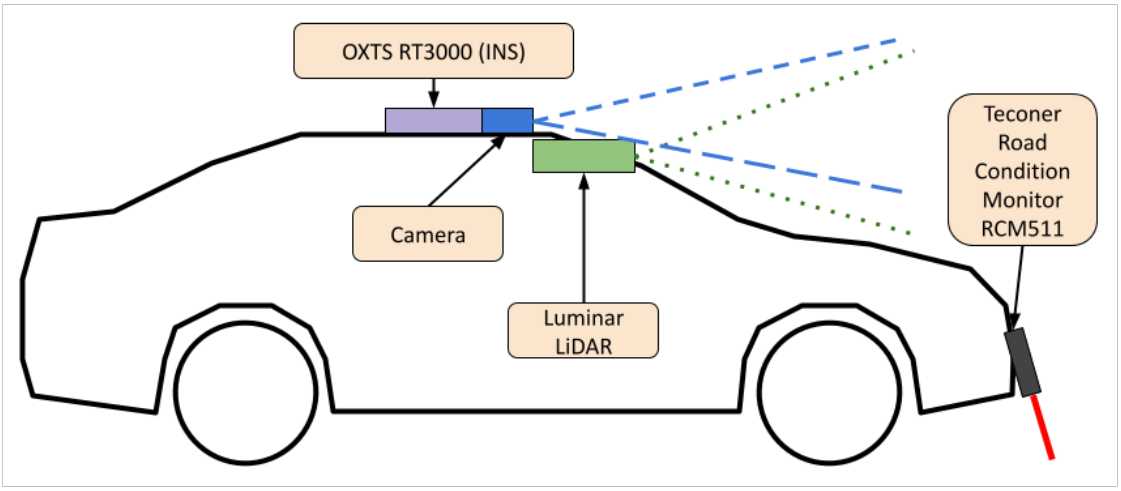


Figure 2.1: The sensor setup when gathering data.

2.1 LiDAR

The LiDAR sensor emits pulses of light that bounces off surfaces back to the LiDAR. This way, it can measure the distance to an object based on the travel time of the light pulses, resulting in a 3D description of the environment being obtained in the form of a point cloud [1]. The position of each point in the point cloud can be described with spherical coordinates, which is illustrated in Fig. 2.2. Here, d is the distance to the point, while θ and ψ represent the elevation and azimuth angles, respectively. Note that in this work, references to the elevation angle will explicitly contain the word ”angle”, while the use of ”elevation” without the ”angle” refers to the word synonymous with height.

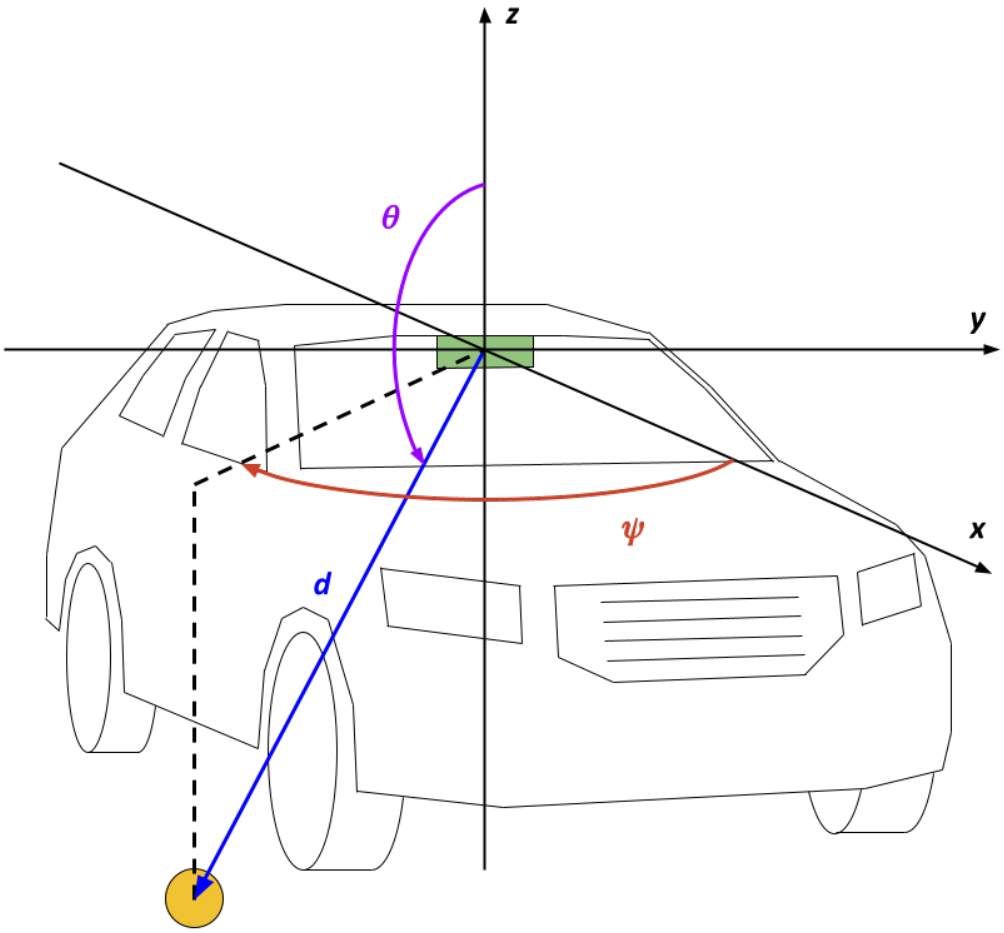


Figure 2.2: An illustration of the distance to a point, d, together with its elevation angle, θ, and azimuth angle, ψ. The point is depicted by a yellow circle, the distance by a blue arrow, and the elevation and azimuth angle by a purple and orange arrow, respectively. The LiDAR sensor is represented by a green box, whose coordinate frame is illustrated with black arrows.

Each light pulse that returns to the LiDAR is assigned a reflectance value and a return index. The reflectance value is a measurement of how much light of an emitted pulse that bounces back to the LiDAR. The return index is derived from the behaviour of a pulse of light when it hits a surface. A fraction of a pulse can reflect back to the LiDAR directly, while another fraction redirects and hits another surface before being reflected. This is illustrated in Fig. 2.3, where a single light pulse comes into contact with a puddle of water and causes three different light pulses to be returned to the LiDAR. The returning pulse resulting in the highest reflectance value is assigned return index 1, the returning pulse with the next highest reflectance value is assigned return index 2, and so on.

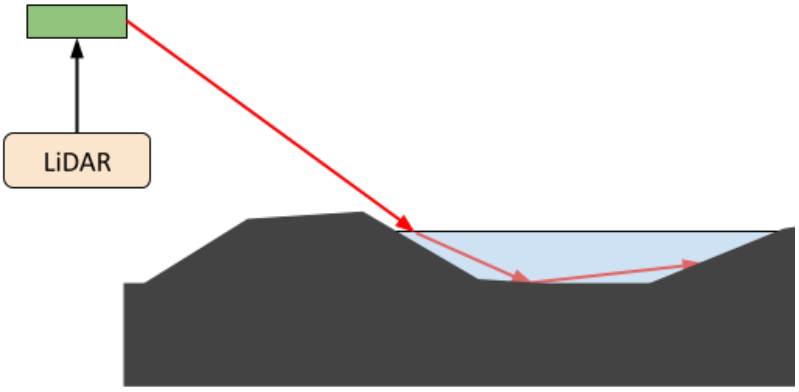


Figure 2.3: An example where the LiDAR receives three reflected light pulses. The returning pulse with the highest reflectance value is assigned return index 1, the returning pulse with the next highest reflectance value is assigned return index 2 and the returning pulse with the lowest reflectance value is assigned return index 3.

2.2 Road Condition Monitor

The RCM sends out lasers of different wavelengths that hit the road and the reflect back to the sensor. These reflectance values are then accumulated over a time period of 1 second. The information from the accumulated measurements are used to get accurate estimations of the surface condition, the friction coefficient, ice and water thickness and more. Here, the surface condition consists of dry, moist, wet, slushy, snowy and icy conditions. In this study, the measurements provided by the RCM will be regarded as the ground truth of the road state.

3

Theory

This chapter is structured into three sections, each focusing on a specific aspect of the research methodology. The first section explores theory related to extracting road points from a LiDAR point cloud, while the second section presents theory and techniques associated with accumulating point clouds over multiple time instances. The final sections of the chapter address theory related to the estimation of the road state and machine learning.

3.1 Detection of Ground Points

When estimating the road state using a LiDAR sensor, it is desirable to separate the points in the point cloud that belong to the road and all other points. By doing so, noisy observations that do not provide information about the road can be removed. Zermas et al. [10], presented a method that separates the segmentation process into two parts, one for extracting all of the point that belong to the ground and one that processes all of the remaining points.

Zermas et al. utilized two assumptions when developing their technique for extracting ground points. The first assumption is that the road has a planar structure, allowing points belonging to the road to be identified using a plane fitting technique. The second assumption is that the points in the point cloud with the lowest height are most likely to belong to the ground surface.

The second assumption enables initial ground points to be found by first extracting a predefined number of the lowest points, N_{LPR}, in the point cloud. A value that represents the lowest point can then be computed by taking the average of the extracted points. In the paper by Zermas et al., this point is referred to as the lowest point representative (LPR). By using the LPR as a reference point, initial seed points can be found by iterating through the point cloud and collecting all points with height values that are within a certain threshold, Th_{seeds}. Using the first assumption, a plane is then fitted to the collected points. When an initial plane has been found, all points within a predefined threshold, Th_{dist}, from the plane are collected and labeled as ground points. This process of plane fitting and collection of points is then repeated for a specified number of iterations, N_{iter}, using the labeled ground points as input to the plane fitting technique. The method for

extracting ground points from a point cloud, referred to as ground plane fitting (GPF), is summarized in pseudo-algorithm 1.

Algorithm 1 Extraction of ground points, Zermas et al. [10]

Arguments:

$\mathbf{P}$	:	initial point cloud
N_{iter}	:	number of iterations of plane fitting and ground point collection
N_{LPR}	:	number of points used to estimate the LPR
Th_{seeds}	:	threshold for points to be considered initial seeds
Th_{dist}	:	threshold distance from the plane

Output:

$\mathbf{P_g}$	:	points that belong to the ground
$\mathbf{P_{ng}}$	:	points that do not belong to the ground

$\mathbf{P_g} = \textbf{ExtractInitialSeeds}(\mathbf{P}, N_{\text{LPR}}, Th_{\text{seeds}})$
for $i = 1 : N_{\text{iter}}$
  $plane\_model = \textbf{EstimatePlane}(\mathbf{P_g})$
  clear($\mathbf{P_g}$,$\mathbf{P_{ng}}$)
  for $k = 1 : |\mathbf{P}|$
    if $plane\_model(p_k) < Th_{\text{dist}}$
      $\mathbf{P_g} \leftarrow p_k$
    else
      $\mathbf{P_{ng}} \leftarrow p_k$
    end
  end
end

ExtractInitialSeeds:
  $\mathbf{P_{sorted}} = \textbf{ExtractInitialSeeds}(\mathbf{P})$
  $LPR = \textbf{Average}(\mathbf{P_{sorted}}(1 : N_{\text{LPR}}))$
  for $k = 1 : |\mathbf{P}|$
    if $p_k.height < LPR.height + Th_{\text{seeds}}$
      $seeds \leftarrow p_k$
    end
  end
return *seeds*

A problem that might occur with this approach is that it is possible for parts of the road to have different inclinations, resulting in that points will not align with one single plane. To reduce the effect of this, Zermas et al. proposed to divide the point cloud into a number of segments, N_{seg}, along the traveling direction of the vehicle and apply the algorithm for extracting ground points in each segment.

3.2 Theoretical Background to Point Cloud Accumulation

This section provides relevant theory related to the accumulation of point clouds. First, theory behind homogeneous transformations is introduced, followed by how Kalman filters are used for state estimation. Finally, a short description of the iterative closest point algorithm is given.

3.2.1 Homogeneous Transformations

When it is possible to obtain information about the orientation and position of the ego vehicle, the aligning transformation of two or more point clouds can be computed through a rotation matrix, $\mathbf{R}$, and a translation vector, $\mathbf{u}$, using the following equation [11].

$$\mathbf{P}^t = \mathbf{u}^t + \mathbf{R}^t\mathbf{P}^{t-1} \tag{3.1}$$

Eq. (3.1) shows the transformation of a point cloud, $\mathbf{P}$, from time instance $t-1$ to t. Here, $\mathbf{u}$ represents the positional change of $\mathbf{P}$ between the time instances while $\mathbf{R}$ represents the change in orientation. The rotation matrix can be computed using the equation,

$$\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\varphi)\,, \tag{3.2}$$

where

$$\mathbf{R}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix}, \tag{3.3}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{3.4}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

and φ, θ and ψ denote the roll, pitch and yaw at time t, respectively [11].

3.2.2 Kalman Filter

To improve the accuracy of state estimates for a system, Kalman filters can be employed. State estimates that are solely based on a state space model that describes the system's dynamics can be inaccurate due to models being simplifications of reality. At the same time, relying on only measurements can also be suboptimal due to the noise in sensors. A Kalman filter takes advantage of both methods by fusing the information provided by models and measurements [12]. As an example, the position of a vehicle can be estimated by using a model for vehicle dynamics together with measurements of the velocity.

In a Kalman filter, the observed system is described with a state vector $\hat{\mathbf{x}}_k$ and the state covariance matrix $\mathbf{P}_k$, where k denotes the current time instance. The state vector contains the variables that describe the conditions of the system, such as the position and the velocity of the vehicle, while the state covariance matrix represent the uncertainty in the state estimate [12].

The filtering process occurring at each time instance can be divided into two steps, the prediction step and the update step. In the prediction step, the state at time $k-1$ is used to predict the state at time k. The prediction step is defined as

$$\begin{aligned} \hat{\mathbf{x}}_{k|k-1} &= \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1|k-1}\,, \\ \mathbf{P}_{k|k-1} &= \mathbf{A}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1}\,, \end{aligned} \tag{3.6}$$

where $\mathbf{A}$ is the process model describing the dynamics of the system and $\mathbf{Q}$ denotes the process noise matrix that models uncertainties in the model [12].

The prediction is then used in the update step, where it is combined with the measurements at time k. The equations for the update step are

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{v}_k\,, \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T\,, \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1}\,, \\ \mathbf{v}_k &= \mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}\,, \\ \mathbf{S}_k &= \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\,. \end{aligned} \tag{3.7}$$

Here, $\mathbf{H}$ is the measurement model, which models how the sensors measure the states, while $\mathbf{R}$ is the noise covariance matrix containing information about the covariance of the sensor noise [12]. By performing the update step, the state information produced by the process model is refined, resulting in the state estimate $\hat{\mathbf{x}}_{k|k}$.

3.2.3 Iterative Closest Point

When accumulating point clouds from different time instances, it is important that they are aligned correctly. One common algorithm for finding the transformation between two point clouds is the iterative closest point (ICP) algorithm. The ICP algorithm estimates a transformation between a source and a target point cloud in an iterative manner. First, an initial guess of the transformation is applied to the source point cloud. Then, every point in the transformed source point cloud is matched with the closest point in the target point cloud. The matched points are then filtered, such that only point correspondences within a distance smaller than a predefined threshold are kept. Finally, the point correspondences are used to compute the transformation matrix that best aligns the source and the target point clouds. Using the newly found transformation matrix, the procedure is repeated until convergence [13].

3.3 Near-infrared Spectroscopy

Previous research has established that near-infrared (NIR) spectroscopy can be used when estimating the surface condition. A study made by Andersson et al. [5], described one approach of identifying road surface conditions, including dry, wet, icy and snowy conditions, by measuring the reflectance of NIR light at different wavelengths. From the identified road surface conditions, a friction coefficient was obtained from a table containing predefined friction coefficient values for each specific condition. The study used a Road Eye sensor that emitted two infrared lasers with wavelengths of 1310 and 1550 nm, which was able to detect if the road was dry or covered with either snow or ice. It was however mentioned that the proposed method had some trouble to distinguish between water and ice. The report was followed up by second study [6], that added a third laser with a wavelength that had a low absorption coefficient for water, snow and ice. This was done to increase the reflective intensity, which was assumed to cause the reflectance of ice and water to differ more.

Casselgren et al. [7] also analyzed spectroscopy readings of road surfaces including dry, wet, moist, frosty, snowy, icy, and a combination of snowy and icy conditions, when NIR lasers with wavelength of 980, 1310 and 1550 nm were used. The difference in reflected intensity partly comes from that the absorption coefficient of different media changes with the wavelength of the laser, as well as the scattering of the laser when it hits a surface, which is depicted in Fig. 3.1. The result of the study showed that it was possible to differentiate between dry, wet, icy and snowy asphalt and that the proposed method had a low probability of making false classifications. It was however noted that most difficult condition to classify was when there existed ice underneath snow.

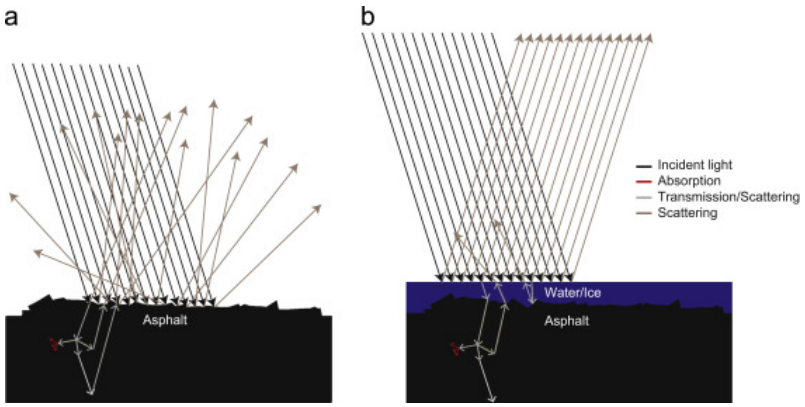


Figure 3.1: The scattering of light on dry asphalt compared to asphalt covered in water. From [7]. Reproduced with permission.

The studies by Andersson et al. and Casselgren et al. were both focusing on only a small area, around 10 mm, of the road. Jonsson et al. [8] presented an imaging system that could distinguish between surfaces of whole road sections that were dry, wet or covered in snow, using an infrared camera equipped with optical wavelength filters. Similarly to the previously mentioned work, this study utilized the reflected light when estimating the road conditions. It was shown that the reflectance differed based on the media absorbing the light, as well as wavelength that was used, as seen in Fig. 3.2. Furthermore, it is mentioned that the roughness of the surface affects how much the light will scatter. Dry asphalt is considered as a rough surface and gives a scattering of the light in all directions. Similarly, snow is also known to be a diffuse light scatterer, due to its composition of small ice crystals and air pockets. However, asphalt will become more smooth when covered with a layer of water or ice, which instead results in that the scattering of the light will be more forward-directed. This behaviour is depicted in Fig. 3.1.

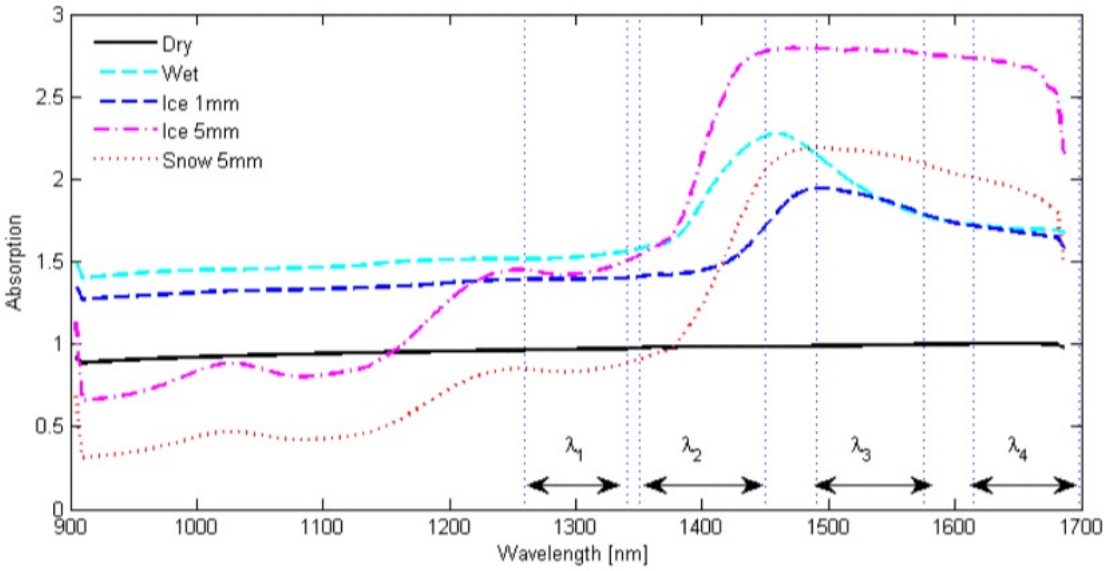


Figure 3.2: The absorption coefficient for different surface condition and wavelengths. From [8]. Reproduced with permission. © 2015 IEEE

The study tested a few machine learning algorithms for classification, such as K-Nearest Neighbour (KNN), Support Vector Machines (SVMs) and neural networks. From experiments performed with data obtained in a laboratory environment, it was found that almost all models had a high accuracy when classifying the road condition. However, the accuracy dropped significantly for all models, except the ones based on KNN and SVMs, when using data gathered from real-world conditions.

3.4 Machine Learning

This section introduces relevant theory related to deep learning, which is an area within the field of machine learning. Firstly, the training process of deep learning models is explained, followed by a description of common performance measures used for evaluating such models. Finally, the concepts of multilayer perceptrons and transformer networks are presented.

3.4.1 Neural Network Training

The work concerning surface state estimation in this project relates to the application of neural networks within supervised learning, a subfield of machine learning. More specifically, a classification task is to be solved. In supervised learning, each data point in the dataset consists of two parts, the input data and the target data. The input data, also known as input features, provides the information available to the algorithm for making predictions. Conversely, the labels, also referred to as targets, represent the ground truth that the model aims to predict. When performing classification, the label represents one of the two or more classes that the input can be classified into.

The overall objective in supervised learning is to train a machine learning model that uses the input features to predict the corresponding target. In this case, training refers to the process of optimizing the parameters of the model, $\boldsymbol{\theta}$, such that the predictions become as accurate as possible. The parameters are a set of values that the model uses to calculate its outputs [14]. Following this, the different parts related to the training process of a deep learning model are explained further.

Dataset partitioning
When training a machine learning model on a dataset, it is common to divide the dataset into a training, a validation and a test set. The training set is used to train the model, that is, to fit the model to the data in the training set. The validation and test sets are then used to evaluate the prediction error of the model. While the validation set is used to tune hyperparameters and find the best model during model selection, the test set is utilized to evaluate the performance of the final model. When dividing data into the training, validation and test sets, the majority of the data is placed in the training set, while smaller portions are allocated to the validation and test sets [15].

Epochs & batch size
Epochs is defined as the number of iterations over the whole training set and is often used as a measure of training duration [16]. To increase computational efficiency, an epoch is generally divided into smaller batches, with each batch containing a subset of points from the dataset that are to be processed in parallel. The number of points processed simultaneously is determined by a hyperparameter called batch size, and its upper limit is constrained by the amount of memory available. More details about the effects of the batch size can be found in [17].

Activation functions
For the deep learning model to be able to capture nonlinear relationships between the input features and the target variable, activation functions are used. Activation functions are nonlinear operations that are usually placed after different components in a neural network. A common choice of activation function is the rectified linear unit (ReLU), shown in the equation below, but there are many other options available [18] [19].

$$\mathrm{ReLU}(x) = \max(0, x) \tag{3.8}$$

Apart from introducing nonlinearities to the network's calculations, activation functions are also used to constrain the outputs of a network. For example, the softmax activation function, defined as

$$\mathrm{softmax}(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}\,, \tag{3.9}$$

where C is the number of classes, is commonly applied at the end of the network in classification tasks. It produces a probability distribution represented by a vector, where each element corresponds to a probability, while also ensuring that the elements in the vector sum up to one. This is particularly useful in classification, where each element of the output vector corresponds to the predicted probability of a specific class. Typically, the log softmax function,

$$\log \mathrm{softmax}(\boldsymbol{x})_i = e^{x_i} - \log \sum_{j=1}^{C} e^{x_j}\,, \tag{3.10}$$

is used instead of softmax since it provides better numerical stability and training efficiency [18].

Loss functions
In supervised learning, loss functions are used to quantify the deviation of the algorithm's prediction from the ground truth. When training the algorithm, the aim is to improve the accuracy of the predictions by minimizing the loss [14]. The choice of loss function depends on the nature of the problem to be solved. In classification tasks, it is common to employ the cross entropy loss, which is described by

$$-\sum_{i=1}^{C} t_i \ln p_i\,, \tag{3.11}$$

where C denotes the number of classes, t_i is an element in a one-hot encoding vector representing the true class and p_i is the predicted probability of class i [20].

Parameter optimization
As mentioned before, the objective is to improve the model's performance by updating the parameters $\boldsymbol{\theta}$ that are used in its computations. When the loss $J(\boldsymbol{\theta})$ has been calculated, the gradient of the loss function with respect to the parameters, $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, can be determined through an algorithm called backpropagation [18] [21]. Since the gradient points in the direction the loss function increases the most, the parameters are updated by taking a step along the negative gradient [14].

Usually, the loss is calculated as the average loss over multiple data points from the dataset. While it can be preferable to compute the loss of the whole dataset at once, it is often not feasible due to limited computational resources. As such, only a number of randomly selected data points equal to the batch size are considered at once. In that case, the gradient becomes

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m_{\mathrm{b}}} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m_{\mathrm{b}}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}) \,. \tag{3.12}$$

Here, m_{b} is the batch size, $L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$ is the loss for a single data point i, while $\boldsymbol{x}^{(i)}$ and $y^{(i)}$ are the input and target, respectively. The updated parameters can then be expressed as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \,, \tag{3.13}$$

where α is a hyperparameter called the learning rate. The previously presented optimizing method is commonly known as stochastic gradient descent (SGD) [14]. There are numerous variations of SGD that include additional techniques, such as momentum and adaptive learning rates. Some examples of methods include Adam and RMSprop [17].

Performance measures
A common method of evaluating machine learning models is to use precision, recall and the F1-score. As an example, consider a binary classification problem where the labels are either "Positive" or "Negative". Here, there are four possible outcomes; that the model classifies a "Positive" data point as "Positive", a "Negative" data point as "Positive", a "Positive" data point as "Negative" or a "Negative" data point as "Negative". This can be visualized as a confusion matrix, which is seen in Tab. 3.1.

Table 3.1: Example of a confusion matrix where there is a "Positive" and "Negative" class.

	Predicted Positive	**Predicted Negative**
Actual Positive	True positive	False negative
Actual Negative	False positive	True negative

The precision and recall can be computed using the following equations

$$\begin{aligned} \text{precision} &= \frac{\text{True positives}}{\text{True positives} + \text{False positives}}, \\ \text{recall} &= \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}, \end{aligned} \tag{3.14}$$

that is, the precision value is a ratio of how many of the predictions for one class that were correct, while the recall is a ratio of how many of the total number of instances for one class that were correctly classified [22]. The F1-score, also referred to as the balanced F measure, is a measurement of the harmonic mean of the precision and recall [22], and can be calculated according to the equation below.

$$\mathrm{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.15}$$

For a case when there are multiple classes, the precision, recall and F1-score is computed for each class. To asses the overall performance of the model, it is common to compute the average over each performance measure, which is referred to as macro averaging [23].

Another common approach is to compute the balanced accuracy when the dataset used is imbalanced. The balanced accuracy is computed by summarizing the class accuracies for each class and divide by the total number of classes [24]. This could be seen as the average of the recall value for each class. For the example above, where there are two classes, the balanced accuracy would be computed using

$$\text{Balanced accuracy} = \frac{1}{2}\left(\text{recall(Positive)} + \text{recall(Negative)}\right). \tag{3.16}$$

The benefit of computing the balanced accuracy is that it considers potential class imbalances in the dataset. If instead the accuracy of all combined classes would be computed from predictions made by a model that is biased towards the overrepresented class, the result would be more optimistic. Thus, the balanced accuracy provides a more accurate indication of the general performance of the model [24].

3.4.2 Multilayer Perceptrons

Multilayer perceptrons (MLPs), or feedforward neural networks, are composed of fully connected layers where every node of each adjacent layer is connected [25]. The number of nodes present in a layer is referred to as the size of the layer. MLPs typically consist of three types of layers; the input layer that processes the input information, hidden layers that learn hidden features and representations in the input data, and an output layer that translates the learned information to predictions [26]. Each node in the hidden and output layers performs the following computation for each input,

$$y = \sigma\left(\sum_{i=1}^{N} w_i x_i + b\right), \tag{3.17}$$

where x_i is the ith input to the node, w_i and b are its associated weights and bias, while N is the total number of inputs to the node [27] and σ is the activation function. Here, w_i and b are parameters that are to be learned by the network [28]. Fig. 3.3 below illustrates a simple MLP with one hidden layer.

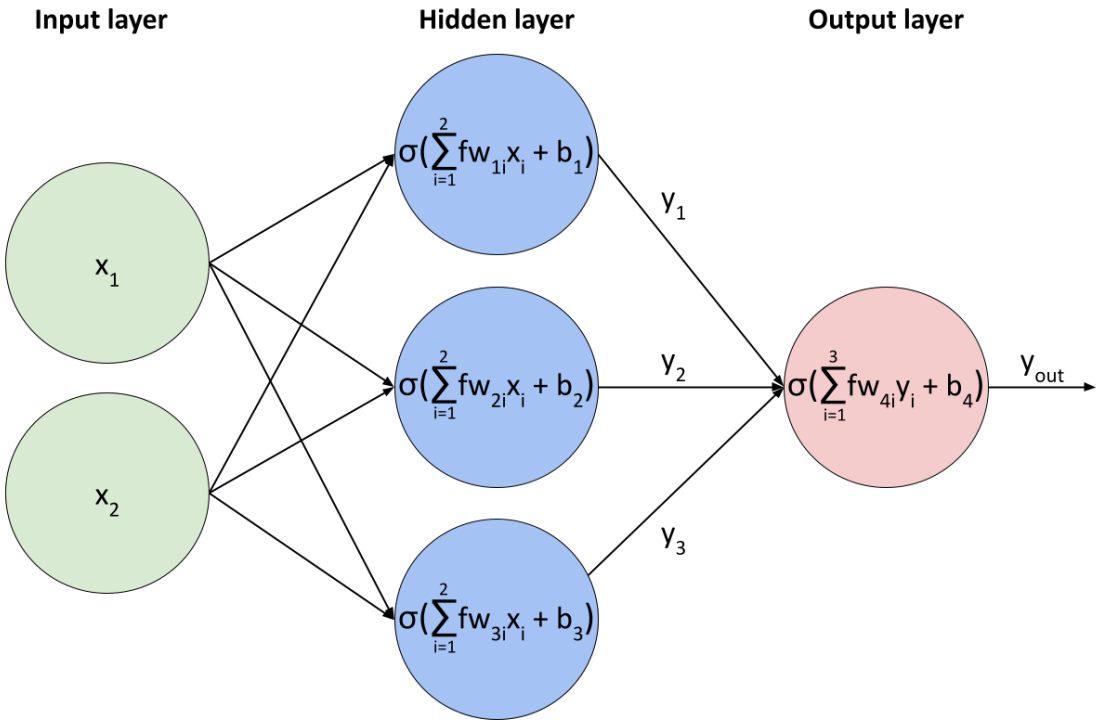


Figure 3.3: A simple multilayer perceptron with two input features, one hidden layer consisting of 3 nodes and an output layer. Each node in the hidden layer is associated with its own weights and bias. Here, y_{out} is the output from the network.

3.4.3 Transformer Networks

The transformer is a neural network architecture that stacks encoder and/or decoder layers and is intended to take sequences of data as inputs [29]. In this thesis, the decoder is not employed, and will therefore not be covered. Each encoder layer consists of two sublayers; the first is for multi-headed self-attention and second is a feedforward network. Both of the sublayers have residual connections, and their outputs are normalized [29].

An illustration of an encoder layer is found in Fig. 3.4.

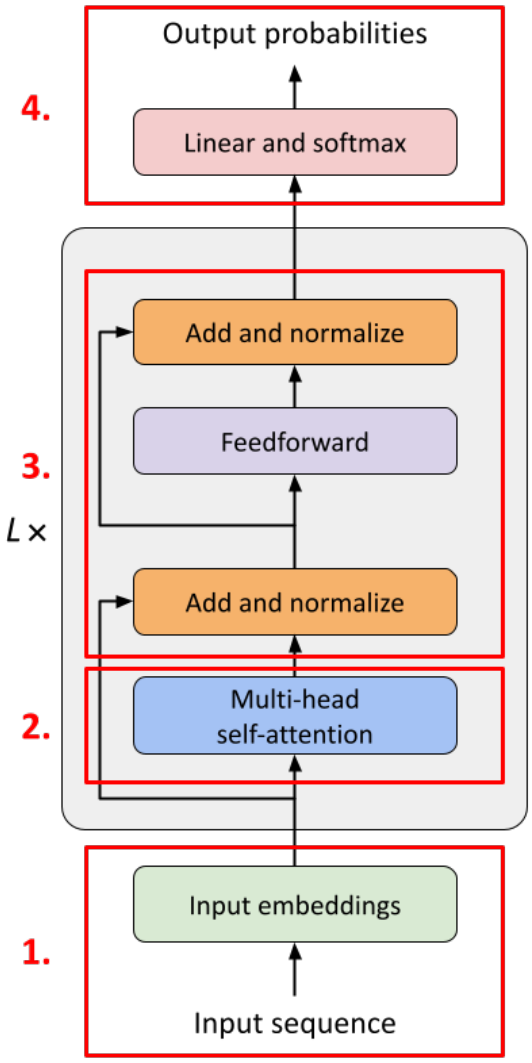


Figure 3.4: An illustration of a transformer network with L number of encoder layer. The different components of the transformer have been marked with red rectangles that have been numbered based their order in the network.

The input sequence to the transformer is first divided into tokens [29]. As an example, if the input sequence is a sentence, it can be divided such that each word within the sentence is a token. Often, a [CLS] token is added to the sequence of tokens. Its purpose is to capture information about the entire sequence [30]. Next, each token in the sequence is encoded into an embedding vector that is to be fed into the transformer. These embedding vectors are then learned during training of the transformer, and the goal is that a vector should capture the meaning of the corresponding token [29]. This process occurs within the first red rectangle in Fig. 3.4.

The second red rectangle in Fig. 3.4 shows the multi-head self-attention sublayer. The goal of self-attention is to update each embedding by capturing the significance of all the other embeddings in relation to the embedding being updated. In other words, it could be said that the purpose of self attention is to determine how much each input embedding should contribute to the output, based on the similarity between the embeddings. To do this, key, query and value vectors are computed from each embedding vector. These vectors are gathered in the matrices $\mathbf{K}$, $\mathbf{Q}$ and $\mathbf{V}$, respectively. These matrices are calculated with the following equations,

$$\begin{aligned} \mathbf{Q} &= \mathbf{X}\mathbf{W}_q \,, \\ \mathbf{K} &= \mathbf{X}\mathbf{W}_k \,, \\ \mathbf{V} &= \mathbf{X}\mathbf{W}_v \,, \end{aligned} \tag{3.18}$$

where $\mathbf{X}$ is a matrix containing embedding vectors, while $\mathbf{W}_q$, $\mathbf{W}_k$ and $\mathbf{W}_v$ are weight matrices containing parameters that are learned during training [31]. First, the dot product of $\mathbf{Q}$ and $\mathbf{K}$ is computed, resulting in that each embedding vector is compared with all other embedding vectors. This yields an output that indicates which embedding vectors that might contain useful information related to a specific input embedding being updated. To get more stable gradients, the output is divided by the square root of the dimension of the key vectors, d_k. Next, the output is passed through a softmax operation, which will determine how much each embedding will be influenced by other embeddings by assigning weights to them. Finally, each embedding vector is updated by computing the dot product between the weights from the softmax operation and $\mathbf{V}$. In doing so, embeddings that hold greater relevance to a specific embedding will have a stronger influence on it during the updating process [29]. The equations to the steps performed in self-attention is summarized as

$$\begin{aligned} \mathbf{Z} &= \frac{\mathbf{Q}\mathbf{K}}{\sqrt{d_k}} \,, \\ \mathbf{W} &= \mathrm{softmax}(\mathbf{Z}) \,, \\ \mathbf{Y} &= \mathbf{W}\mathbf{V} \,, \end{aligned} \tag{3.19}$$

where $\mathbf{Z}$ is the scaled output from the dot product between $\mathbf{Q}$ and $\mathbf{K}$, while $\mathbf{Y}$ is the updated embeddings [29].

Self-attention can be performed in multiple heads, where each head learns different parts of the embedding vectors. To this end, each head is assigned its own weight matrices with trainable parameters for the queries, keys and values. The output from each head is then concatenated into a single matrix, whereafter the dot product between this matrix and an additional weight matrix with trainable parameters, $\mathbf{W}^O$, is computed. This final step is done to transform the size of the output to the same size as the input [29]. An illustration of multi-head self-attention where 3 heads are used, is shown in Fig. 3.5.

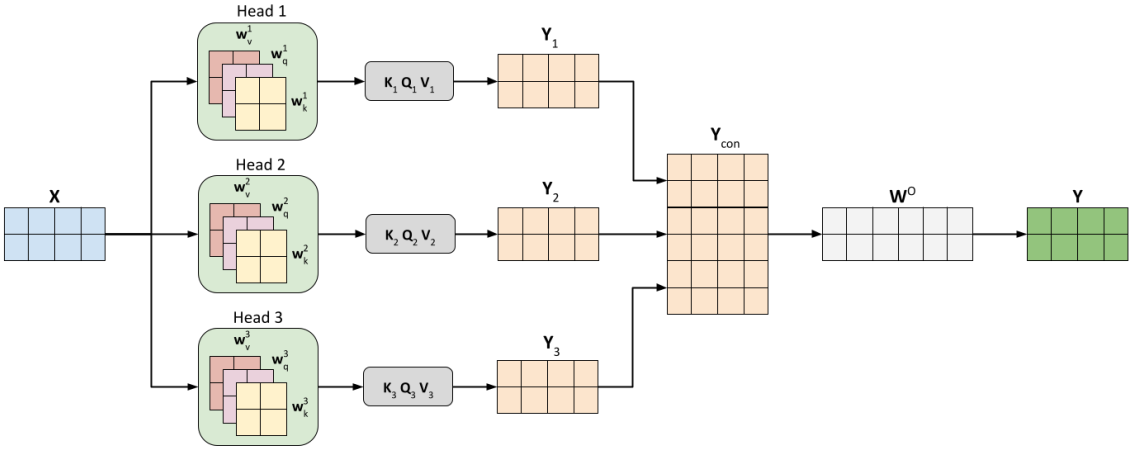


Figure 3.5: Multi-head self-attention using 3 heads. Here, $\mathbf{X}$ is the input to the multi-head self-attention sublayer and $\mathbf{Y}$ is its final output. Each head has its own set of weight matrices containing trainable parameters.

The third red rectangle in Fig. 3.4 shows that the encoder makes use of residual connections, which facilitates the training of deeper networks [32]. The encoder layer also utilizes layer normalization to improve the learning stability during training, as well as the generalization of the model. Additionally, layer normalization reduces the training time [33]. This third red rectangle also shows that the encoder layer contains a feedforward network [29].

The fourth red rectangle in Fig. 3.4 shows that the output from the encoder is passed through a final MLP. If a [CLS] token has been used, only this token will be input to the MLP [30]. Next, softmax is used to compute the probability distribution over all possible classes [29]. This means that the number of nodes in the output layer of the MLP needs to be equal to the number of classes.

4

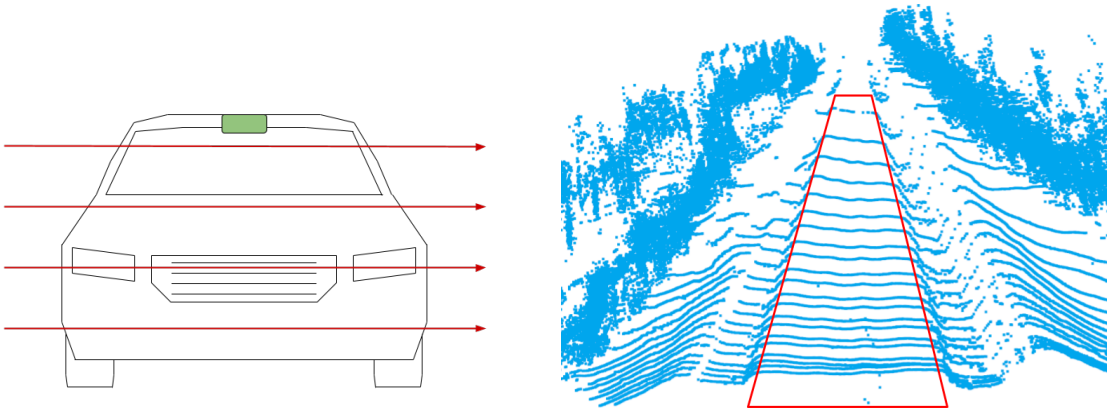
Extracting LiDAR Points on the Road Surface

In this chapter, the topic of extracting points that belong to a road from a point cloud is investigated. First, an existing method for finding all of the points on the ground plane is detailed, followed by the methodology used in this thesis. Lastly, the performance of the proposed method for extracting ground points is presented.

4.1 Methodology and Implementation of the Road Point Extraction Algorithm

The extraction of road points is divided into two stages. The idea of the first stage is to utilize the scan pattern of the LiDAR to find points in the point cloud that are more likely to belong to drivable road. The LiDAR sensor used scans its environment in a unidirectional manner, which is illustrated in Fig. 4.1a. At the same time, points of the point cloud that belong to the road surface have a tendency of spreading out in line-like formations, as seen in Fig. 4.1b. These two factors mean that road points that are placed side by side also will be indexed next to each other in the point cloud.

To find the road points that are more likely to belong to the road, three different starting points were defined. The first starting point was located at the center of the ego vehicle along its latitudinal axis, while the remaining starting points were placed with a ± 1.5 meters offset from the center. This offset ensured that nearby vehicles positioned closely in front of the ego vehicle, which could obstruct the view of the LiDAR from the initial starting point, were prevented from causing any potential interference. By assuming that the width of a typical vehicle is approximately 2 meters, using an offset of 1.5 meters at each side of the ego vehicle would assure that at least one of the starting points would have a clear view of the road ahead. The next task was to find points that potentially belong to the line formations seen in Fig. 4.1b. This was done by collecting all points that lined up with the starting points in the longitudinal direction, which from now on will be referred to as the middle points. This is illustrated in Fig. 4.2a.



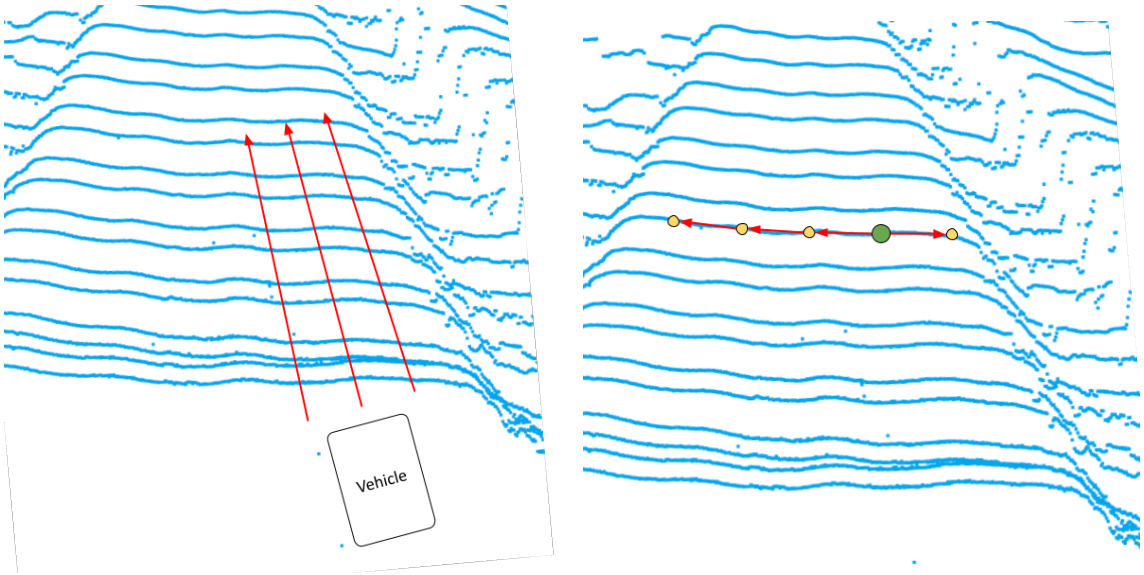
(a) A sketch of the scan pattern of the LiDAR sensor where the LiDAR is depicted as a green box. The LiDAR scans its environment in a unidirectional manner.

(b) A LiDAR scan of the road and its surroundings. The points that belong to the road are located within the red bounding box.

Figure 4.1: An illustration of the scan pattern and the resulting line formations located on the road.

For each gathered middle point, another point indexed at a predefined number of steps, S_{step}, before the middle point was examined to see if it belonged to the line. If the heights of the points and the distances between the points in longitudinal-latitudinal plane did not differ more than their respective threshold, Th_{height} and $Th_{\text{lon,lat}}$, both points and all intermediate points were considered to belong to the line formation. This search continued until one of the thresholds were exceeded, whereafter the search continued in the other direction, that is, by examining the points that were indexed after the middle point. When all the points of the line had been found, the line and its points were stored for further processing. The method of finding all points belonging to a line is illustrated in Fig. 4.2b and was repeated for all gathered middle points.

When all middle points had been examined, the process was repeated using the line formation that contained the most number of points as reference. This time, ten starting points were uniformly spread out along the horizontal direction of the line. This was done to capture as many lines as possible, since the occurrence of obstacles or other vehicles could form gaps in the line formations belonging to the road. The reason for choosing the line containing the most number of points as a reference was that it most likely was a longer line, thus increasing the chance of capturing missed line segments. Another method would be to compute the actual longest line based on the coordinates of the lines end points, however, the formerly mentioned method was deemed to perform sufficiently well in regards to capturing the missed line segments.



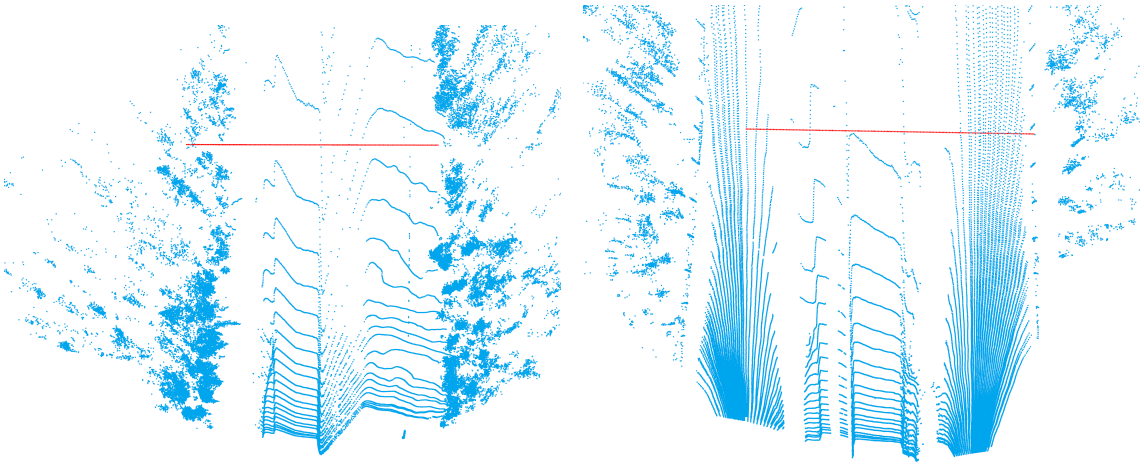
(a) An illustration of how the middle points are collected. All points that are lined up with the starting points in the direction of the red arrows are considered as middle points for the horizontal scan lines of the LiDAR.

(b) An illustration of the search for lines. The green circle symbolizes the middle point, the yellow circles symbolize the examined points and the red arrows show the search direction.

Figure 4.2: An illustration of the process for finding line formations.

After the actions of the first stage had been performed, line formations that were elevated above the ground plane might be amongst the gathered lines. The reason for this is that the proposed method does not consider at which height the lines are located at. As an example, if the ego vehicle is driving towards a tunnel that is captured by the LiDAR, lines that are placed above the entrance of the tunnel will be considered as valid line points. The purpose of the second stage is therefore to filter the found lines, such that only lines at ground level are kept. To this end, the GPF technique by Zermas et al. [10], presented in section 3.1, was used to filter out all points that did not belong to the ground surface. When the distance in the longitudinal direction increased, the number of LiDAR scan lines on the road decreased. This means that the segments used in the GPF algorithm that were further away from the ego vehicle could end up containing no road points. Additionally, higher elevated objects, such as an overpass, could potentially still be visible within these segments. If the scan lines of such an object were to be caught by the algorithm for finding line formations, the estimated ground plane in this segment would be placed well above the true ground plane.

By examining several point clouds, it was found that the scan line formations of the LiDAR had stopped being located on the road, or occurred with a much lower frequency, at distances higher than 55 m for flat roads. Two example cases for this are shown in Fig. 4.3, where the blue points represent the point cloud obtained by the LiDAR and a red line is placed at 55 meters from the ego vehicle. With this observation in mind, the algorithm for extracting road points was only applied to points within 55 meters.



(a) Example case 1 **(b)** Example case 2

Figure 4.3: Visualisation of the sparsity of LiDAR scan lines on flat roads as the distance from the ego vehicle increases. The blue points represent the LiDAR point cloud while the red line marks the distance of 55 meters from the ego vehicle.

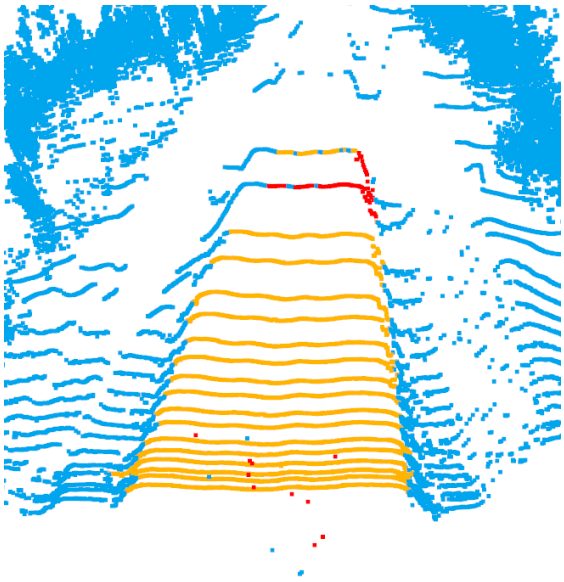
4.2 Evaluation and Analysis of the Road Point Extraction Algorithm

To evaluate the performance of the road point extraction algorithm, the points of the point cloud were coloured in different colours based on if they had been classified as road points, filtered away during the search for line formations or removed by the GPF algorithm. Using the coloured points, the result was only examined by looking at the coloured sections, while no numerical analysis was conducted. The reason for this was that no ground truth was available to identify which points belonged to the road or not. The evaluation process was carried out by investigating the following questions.

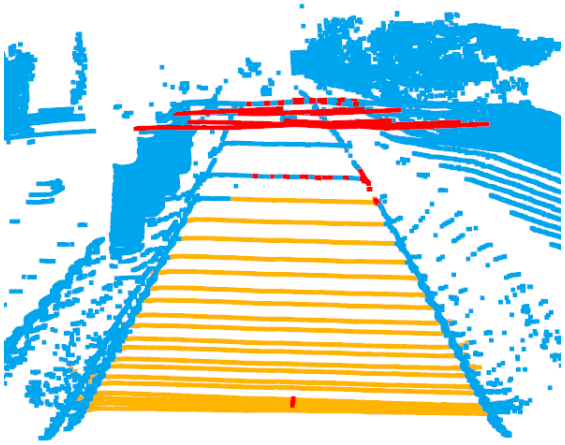
- How many road points were correctly classified as road points?
- How many points were incorrectly classified as road points?
- How many road points were not found by the first stage of the algorithm, that is, at the search for line formations that belonged to the road?

- How many road points were removed by the GPF algorithm?
- How many of the points located above the road were removed by GPF?
- How did the method perform when other vehicles were present?
- How did the method perform on curved roads?

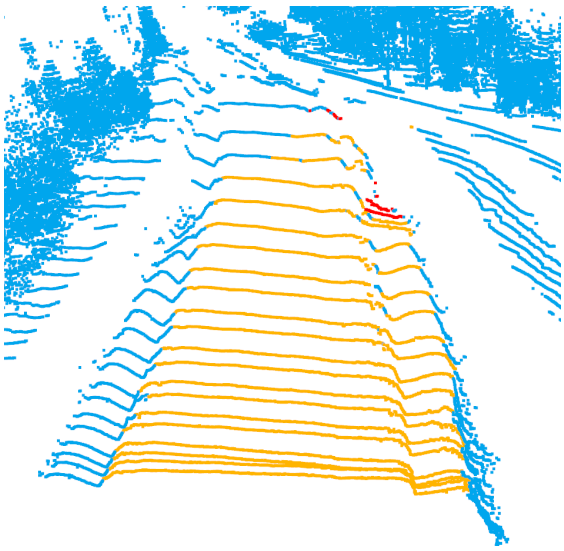
The result of the road point extraction algorithm is shown in Fig. 4.4 and 4.5, where the yellow points have been classified as road points, blue points represent points that have been filtered away during the initial search for lines and the red points are points that was removed by the GPF algorithm.



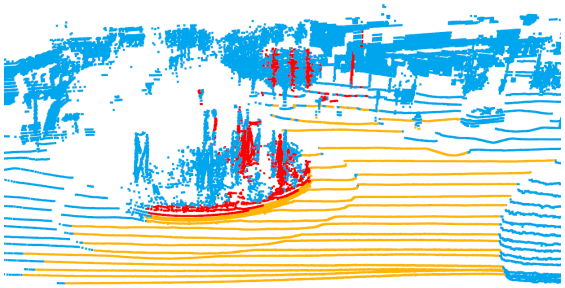
(a) Extraction of road points on a straight empty road.



(b) Extraction of road points when driving under an overpass.

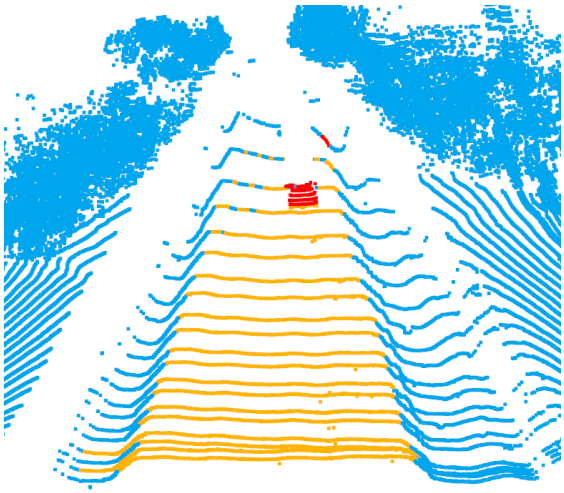


(c) Extraction of road points when driving on a curved road.

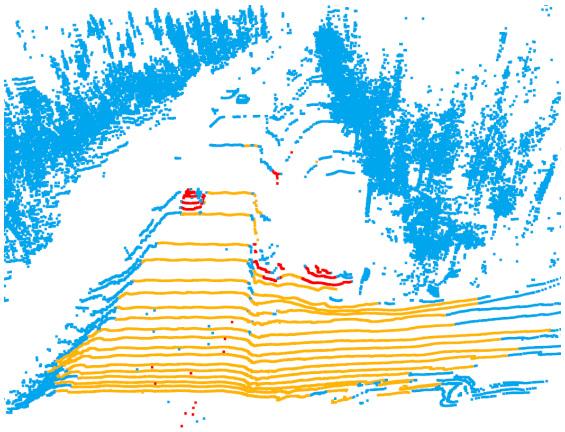


(d) Extraction of road points when driving into a roundabout.

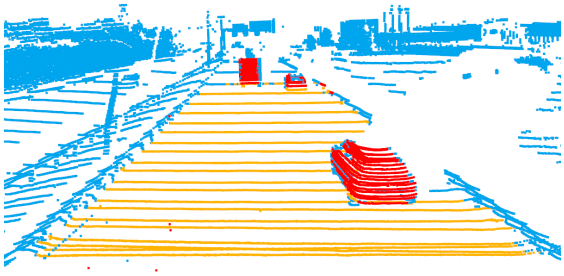
Figure 4.4: Performance of the road point extraction algorithm in different settings. Yellow points have been classified as road points, blue points represent points that have been filtered away during the initial search for lines and the red points are points that was removed by the GPF algorithm.



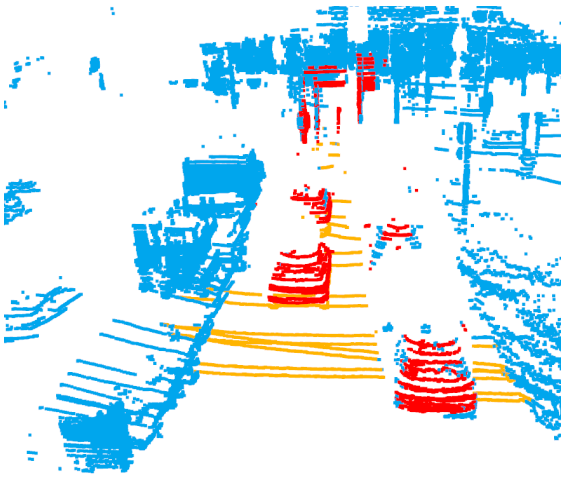
(a) A scene where a vehicle is driving further away from the ego vehicle on a straight road.



(b) Extracting road points from a scene where two vehicles are driving towards the ego vehicle. An entrance to a side road present on the right.



(c) Extraction of road points when driving next to other vehicles.



(d) Extraction of road points at a close proximity to other vehicles

Figure 4.5: Performance of the road point extraction algorithm in different settings where vehicles are present. Yellow points have been classified as road points, blue points represent points that have been filtered away during the initial search for lines and the red points are points that was removed by the GPF algorithm.

In all of the figures, most of the points on the road are correctly classified. It was however noticed that some of the road points were excluded in the process of finding line formations. This can especially be seen when the distance to the line formation is increased, as in Figs. 4.4a, 4.4b, 4.5a. In Fig. 4.4c, it is also seen that the found line formations sometimes extend to areas outside of the road. A similar behaviour can be observed in Fig. 4.5b, where the algorithm manages to find lines at the entrance of a side road.

When examining the effect of the GPF algorithm, it is seen that it successfully manages to remove points that have a much higher elevation than the ground plane, as in Fig. 4.4a where the ego vehicle drives under an overpass. The GPF algorithm also removes the outlier points that occurs close to the ego vehicle in some of the figures. However, Figs. 4.4d and 4.5d show that points closer to the ground plane, such as those on sidewalks or on the lower parts of vehicles, are sometimes considered as being part of the road. Additionally, it is found that the GPF algorithm also excludes some of the points that are further away from the ego vehicle, as seen in Figs. 4.4a and 4.5a.

Figs. 4.4d and 4.5 show the performance when other vehicles are included in the point cloud at various distances. When points belonging to these vehicles are captured in the found line formations, the GPF algorithm often does not remove the points that are close to the ground. It can also be noticed that the presence of other vehicles easily obstruct the LiDARs view of road patches, as seen in Figs. 4.5c and 4.5d.

Fig. 4.4c shows the extraction of road points on a curved road. It is seen that the algorithm manages to correctly classify almost all of the road point, and only fails at longer distances from the vehicle. A more challenging scene is presented in Fig. 4.4d, where the ego vehicle is driving into a roundabout. Here, it can be observed that all of the close road points are classified successfully. It can however noticed that the points that are placed further away from the vehicle in the latitudinal direction are not found when searching for line formations. Furthermore, the sidewalk at the base of the roundabout is falsely classified as belonging to the road.

5

Point Cloud Accumulation

This chapter is dedicated to the process of accumulating point clouds. First, the approaches implemented in this thesis are explained, after which the performance of the methods is presented. Accumulating point clouds allows for the creation of denser and more detailed point clouds by merging data from different time instances. Obtaining denser point clouds is desirable since it could provide more information about the road state.

5.1 Proposed Approach for Point Cloud Accumulation

When accumulating points from different time instances, two different methods were compared; one that transformed the points from the previous time instance using the sensor data of the velocity and orientation of the ego vehicle, and another that used ICP with the transformation of the first method as initial guess for the rotation matrix.

5.1.1 Using Transformations Based on Vehicle Motion

A global reference system was defined with its origin at the center of the INS sensor at the time instance of the first INS measurement. The reason why the INS sensor was chosen as a reference point was because it is the location from which the velocity and orientation of the vehicle is measured. From this global reference frame, the change of the vehicle's position and orientation were tracked using the Kalman filter estimations and INS measurements, respectively. To accumulate point clouds from multiple time instances, Eq. (3.1) was used. Here, the translation vector $\mathbf{u}$ represents the change in position since the first measurement of the INS sensor, and was therefore set to be equal to the current position of the vehicle. Similarly, the rotation matrix $\mathbf{R}$, was set to the represent current orientation of the vehicle in the global reference system.

By denoting x as the coordinate in the longitudinal direction, y as the coordinate in the latitudinal direction, z as the elevation and φ, θ and ψ as the rotations around the respective axes, the global position and orientation of the vehicle were described as

$$\mathbf{V}_{\text{pos}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad \mathbf{V}_{\text{ori}} = \begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}. \tag{5.1}$$

To compute the orientation in the global reference system, the change in orientation was first computed and added to the current global orientation of the vehicle,

$$\mathbf{V}^t_{\text{ori}} = \mathbf{V}^{t-1}_{\text{ori}} + (\mathbf{V}^t_{\text{ori,INS}} - \mathbf{V}^{t-1}_{\text{ori,INS}}), \tag{5.2}$$

where $\mathbf{V}_{\text{ori,INS}}$ is the orientation of the vehicle measured by the INS. Afterwards, the result was used to compute $\mathbf{R}$, as described in Eq. (3.2). Since the velocity measurements provided by the INS were measured in its local reference system, they had to be rotated according to the equation

$$\mathbf{v}_{\text{global}} = \mathbf{R}\mathbf{v}_{\text{INS}}, \tag{5.3}$$

where $\mathbf{v}_{\text{global}}$ is the velocity in the global reference system and $\mathbf{v}_{\text{INS}}$ is the velocity in the reference frame of the INS. The global velocity is then used as measurement input to the Kalman filter when tracking the position, $\mathbf{V}_{\text{pos}}$.

When computing the change in position between two point clouds, the position of the LiDAR in the reference system of the INS had to be computed for both time instances. Fig. 5.1 illustrates the LiDAR position in terms of spherical coordinates, where d_l is the distance to the center of the LiDAR from the origin of the INS reference frame. The position of the LiDAR was then converted from spherical to rectangular coordinates using the following equations,

$$\begin{aligned} l_x &= d_l \cos\theta \cos\psi, \\ l_y &= d_l \cos\theta \sin\psi, \\ l_z &= d_l \sin\theta, \end{aligned} \tag{5.4}$$

which made it possible to express the position in vector form,

$$\mathbf{L}_{\text{pos}} = \begin{bmatrix} l_x \\ l_y \\ l_z \end{bmatrix}. \tag{5.5}$$

The change in position was then computed with the following equation,

$$\mathbf{u} = (\mathbf{L}^t_{\text{pos}}(\theta^t, \psi^t) + \mathbf{V}^t_{\text{pos}}). \tag{5.6}$$

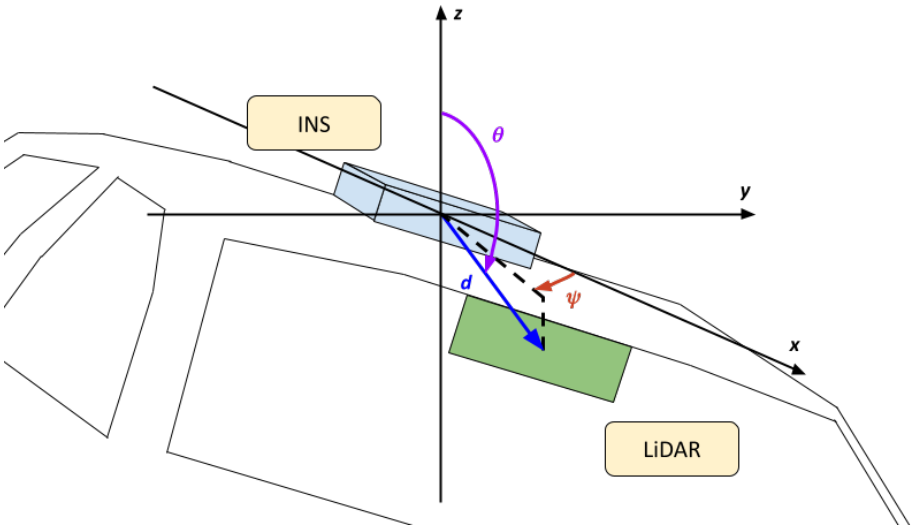


Figure 5.1: An illustration of the LiDAR's position in the reference frame of the INS sensor in terms of spherical coordinates, where d_l is the distance to the origin of the LiDAR, θ is the pitch angle and ψ is the yaw angle. The LiDAR is represented by the green box, while the blue box symbolizes the INS sensor.

5.1.2 Using Transformations Based on Vehicle Motion and ICP

In section 3.2.3, it was explained that an initial guess for the transformation matrix is required when using ICP. To find this initial guess, the methods for computing the translation vector and rotation matrix introduced in subsection 5.1.1, was used. Since the ICP algorithm matches point correspondences, it works best with the presence of rigid objects, such as buildings, stationary vehicles and road signs, as well as objects with distinct features, such as corners or edges. However, some objects make it harder for ICP to find suitable point correspondences. For example, a moving vehicle could make the algorithm try to match point correspondences on the vehicle, even though it has traveled a distance in between the point cloud instances. A consequence of this could be that the point clouds become misaligned. Another example of objects that could reduce the accuracy of the ICP algorithm are bushes, which lack distinct features. To reduce the effect of less distinct objects, the ICP algorithm was only applied to limited area around the ego vehicle.

5.2 Qualitative Evaluation of Proposed Point Cloud Accumulation Methods

In this section, the evaluation process of the two accumulation methods is detailed. Subsequently, the resulting performance of both methods is presented.

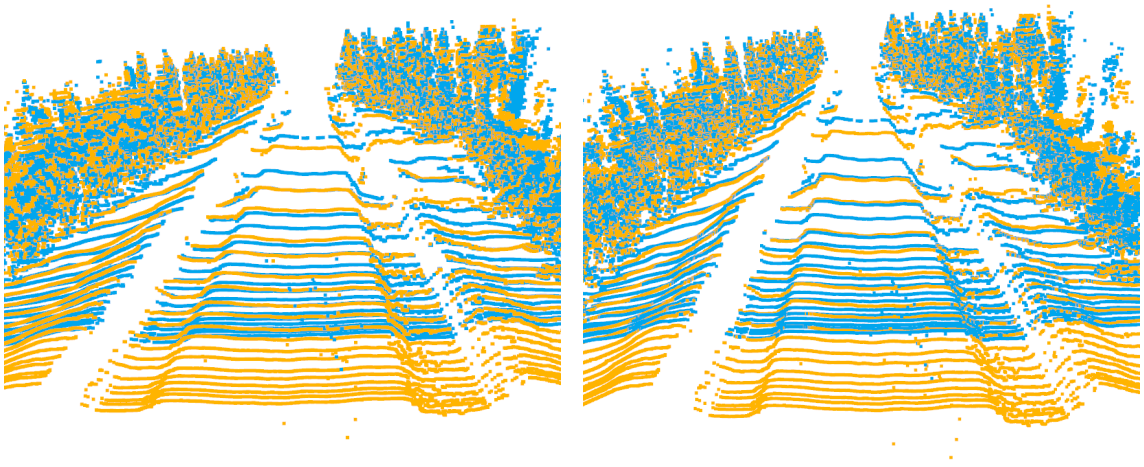
5.2.1 Evaluation of Point Accumulation Methods

The evaluation of both point cloud accumulation methods was performed by visualizing the alignment of point clouds from two adjacent LiDAR scans and observing how much they deviated from each other. When comparing the different methods to each other, 5 minutes of driving time was analyzed. Of these 5 minutes, 4 were on and around a highway, while the remaining minute took place in a suburban area. The performance of the different methods were compared in terms of whether the alignment was considered successful or not. For the method using ICP, the area around the ego vehicle was adapted from case to case, but used a default range of 0 to 100 m in the longitudinal direction and -20 m to 20 m in the latitudinal direction.

Finally, both methods were integrated with the road point extraction algorithm and tested by accumulating a series of LiDAR scans. For the method using ICP, this meant that the full point cloud from the previous time instance had to be stored, such that it could be used to find point correspondences with the newly obtained point cloud. The methods were then compared to see if they performed differently.

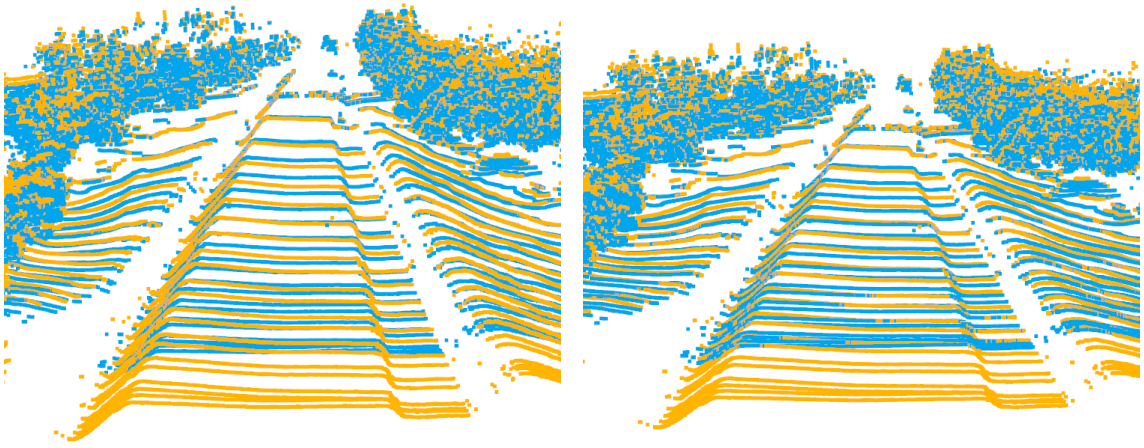
5.2.2 Point Cloud Accumulation Result

Figs. 5.2 and 5.3 show the result when aligning two subsequent point clouds using homogeneous transformations alone and homogeneous transformations together with ICP. Here, the result from using only a homogeneous transformation is displayed in the left column, while the combination of a homogeneous transformation and ICP is presented in the right column. Additionally, yellow points represent a point cloud obtained at time t, while blue points belong to a point cloud obtained at time $t+1$.



(a) Only homogeneous transformation.

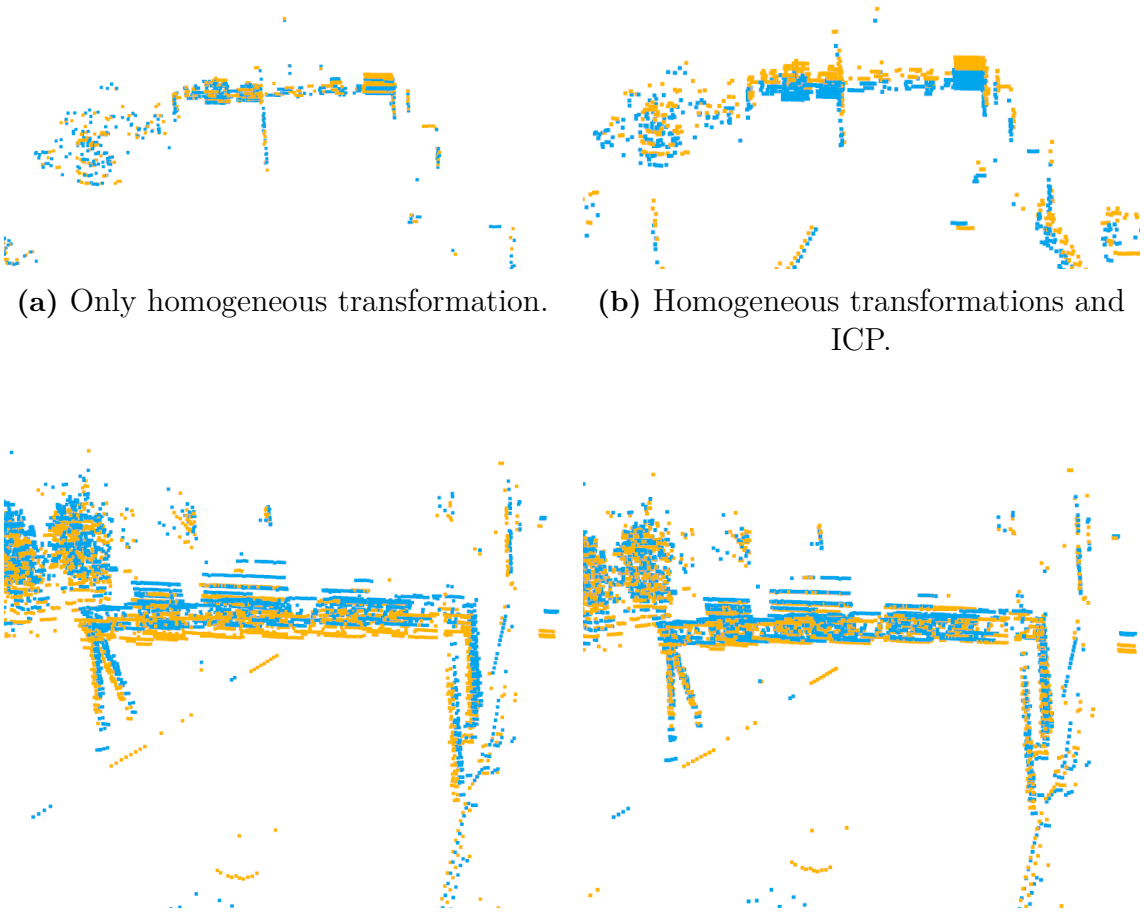
(b) Homogeneous transformations and ICP.



(c) Only homogeneous transformation.

(d) Homogeneous transformations and ICP.

Figure 5.2: Comparison of the two aligning methods on roads that are surrounded by trees. Yellow points represent a point cloud obtained at time t, while blue points belong to a point cloud obtained at time $t+1$.



(a) Only homogeneous transformation.

(b) Homogeneous transformations and ICP.

(c) Only homogeneous transformation.

(d) Homogeneous transformations and ICP.

Figure 5.3: Comparison of the two aligning methods on a highway. Yellow points represent a point cloud obtained at time t, while blue points belong to a point cloud obtained at time $t + 1$.

Fig. 5.2 shows the typical environment that was present in the available data. In these cases, none of the methods manage to correctly align the point clouds, due to the blue points being more visible than the yellow points. This behaviour is most prominent in Figs. 5.2c and 5.2d. Additionally, Figs. 5.3a and 5.3b present an example where the method using ICP misaligns the point clouds, even though the alignments from the homogeneous transformation was seemingly good. The reverse case is visible in Figs. 5.3c and 5.3d, where the method using ICP corrects the misalignment made by the homogeneous transformation.

Figs. 5.4 and 5.5 show the performance of both methods when accumulating road points over multiple time instances.

(a) Only homogeneous transformation.

(b) Homogeneous transformations and ICP.

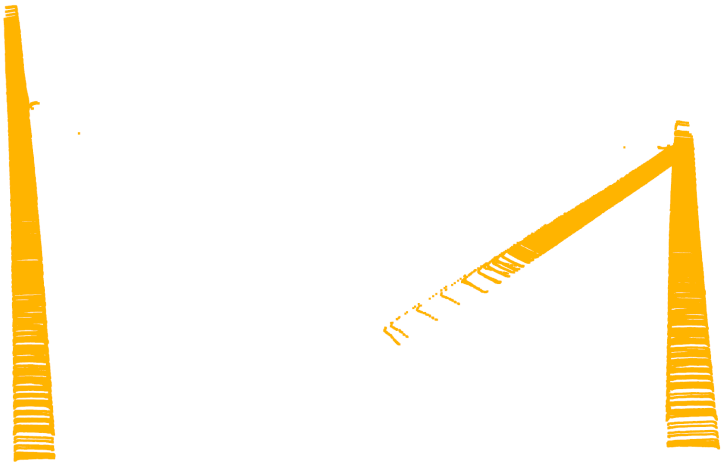
(c) Only homogeneous transformation.

(d) Homogeneous transformations and ICP.

Figure 5.4: Accumulation of road points over multiple time instances. Viewed from the side.

(a) Only homogeneous transformation.

(b) Homogeneous transformations and ICP.



(c) Only homogeneous transformation.

(d) Homogeneous transformations and ICP.

Figure 5.5: Accumulation of road points over multiple time instances. Viewed from above.

When comparing the methods, it is observed that the performance was similar, although it was noticed that the method using ICP more often aligned the elevation of the road points better, as is seen when comparing Fig. 5.4a with Fig. 5.4b. Still, the performance of both methods often faltered in this regard, resulting in that the road points float above or below the ground plane. When looking at the road points from above, see Fig. 5.5, it is observed that the performance is mostly the same for both methods. However, Fig. 5.5d shows that it is possible for the ICP-based method to completely misalign point clouds and start accumulating the road points wrongly. While the method using ICP could be unpredictable, the method using

only homogeneous transformations always resulted in adequate alignments in the longitudinal-lateral plane, as seen in Fig. 5.5c.

To combat the worst of the misalignments, a set of thresholds were defined based on the measurements of the accelerations and angular rates. These thresholds were chosen because it was observed that the performance of the accumulation method decreased when the vehicle accelerated or made a turn. When one or more of the thresholds were not met, the algorithm did not align the related point cloud, but kept the previous accumulated points such that the next point cloud could be added. The thresholds are presented in Tab. 5.1, where a_x and a_y symbolize the acceleration in the direction of respective axes and ω_φ and ω_θ denote the angular rate around the x- and z-axis.

Table 5.1: Thresholds used in the point accumulation algorithm.

Thresholds
$-0.3 < a_x < 1.5$
$-0.8 < a_y < 0.8$
$-2 < \omega_\varphi < 2$
$-5 < \omega_\psi < 5$

6

Road State Dataset

This chapter describes the process of creating a dataset containing LiDAR points labeled with both friction coefficients and surface conditions. The first section explains how LiDAR data was matched with measurements from the RCM, and what data was collected. In the second section, an overview of the road state dataset is presented.

6.1 Creating a Road State Dataset using Automatic Annotation

The creation of the road state dataset was divided into two parts, one for gathering features of every point in the point cloud and the position of the vehicle, and the other for collecting the points containing road state information. An overview of both parts and their components is illustrated in Fig. 6.1.

In the first part, a file containing one minute of LiDAR data, with a scan frequency of 5 Hz, was replayed. While replaying the file, the position of the vehicle, as well as all of the road points in the LiDAR scans, were mapped. For every point, a set of characteristic features were collected, namely the original coordinate of the point before being transformed; the reflectance value; the return index; the distance, as well as the elevation and azimuth angle of the point with respect to the LiDAR; the elevation of the point; and finally an identifying number of the LiDAR scan. The coordinates of the points in the point clouds were given in rectangular coordinates. Thus, the distance d to the point from the origin of the lidar sensor was calculated with the formula for the Euclidean distance,

$$d = \sqrt{x^2 + y^2 + z^2}\,, \tag{6.1}$$

while the elevation angle θ and azimuth angle ψ were computed using equations for conversion of rectangular coordinates to spherical coordinates,

$$\begin{aligned} \theta &= \arcsin \frac{z}{d}\,, \\ \psi &= \arctan \frac{y}{x}\,. \end{aligned} \tag{6.2}$$

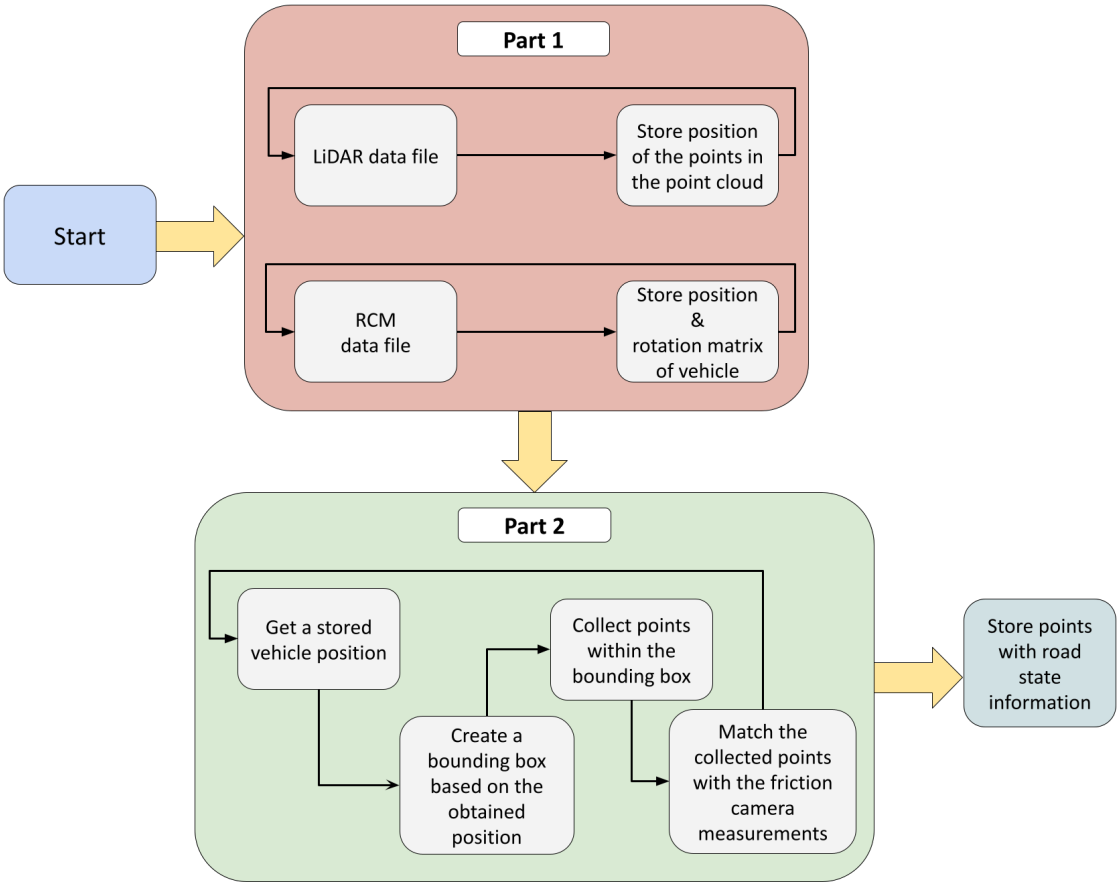


Figure 6.1: An overview of the process of creating a road state dataset.

Additionally, the elevation was defined as the height of the point in relation the ground at the vehicle's current position. To compute this feature, the distance from the ground to the lidar was added to the z coordinate of every point.

When replaying the file containing the LiDAR point cloud, another file containing measurements from the RCM was replayed simultaneously. Every time a new measurement was received, it was stored together with the global position of the vehicle and the related rotation matrix based on the current global orientation.

After replaying the data files, the second part of the dataset creation was initiated, where the goal is to match points together with their corresponding RCM measurement. To this end, a bounding box was placed with its center in the location of the RCM measurement, which was computed using the stored position and the dimensions of the ego vehicle. An illustration of this is showed in Fig. 6.2. All points that ended up inside of the bounding box were then stored, together with their characteristic features, RCM measurement and an identifying number of the bounding box that was used to extract the points. That is, all points within the bounding box was assigned the same surface condition and friction coefficient.

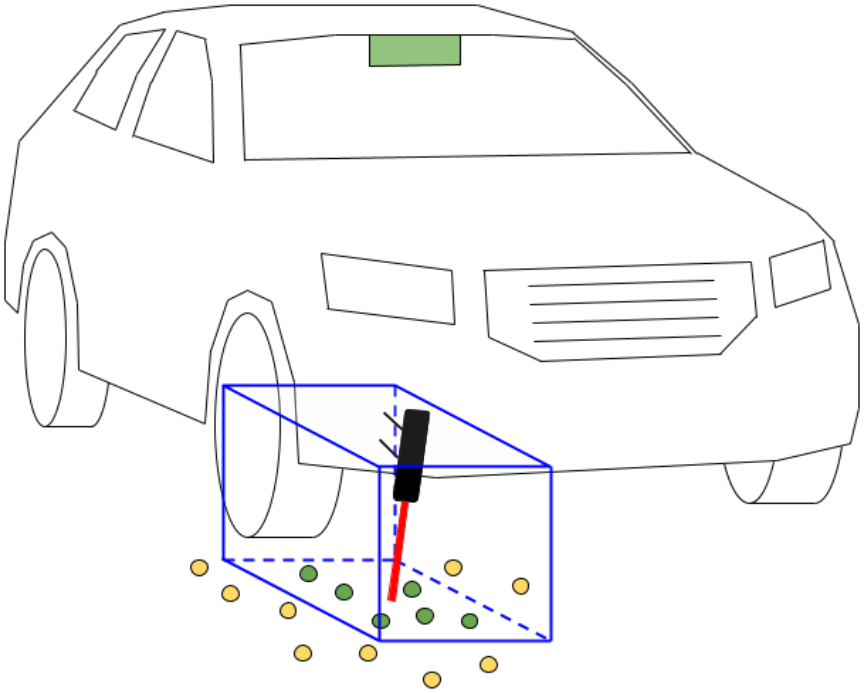


Figure 6.2: An illustration of the bounding box used to match points with RCM measurements. The black box with a red line coming out of it symbolises the RCM. The green circles represent points that will be labeled with RCM measurements, while the yellow circles are points that will not be used.

The method for creating a road state dataset relied on that the road points could be accumulated over multiple time instances. However, section 5.2.2 showed that the proposed accumulation methods performed rather poorly. Since the method using ICP was deemed unpredictable, the accumulation of point clouds when creating the road state dataset was performed using only homogeneous transformations.

The RCM collected measurements in a small area directly in front of the right tire of the ego vehicle, at an interval of 1 Hz. Due to the low frequency of these measurements, the bounding box used to gather road points was chosen to be larger in the longitudinal direction, and was set to 1.5 meters. The height of the bounding box was set to 3 meters, because of the previously mentioned behaviour of floating points inherited from the accumulation method. Finally, the width of the bounding box was set to 0.5 meters. The process of collecting road points for the road state dataset is visualized in Fig. 6.3, where the green box symbolizes the origin of the INS sensor and the red bounding box is used to match the points with RCM measurements.

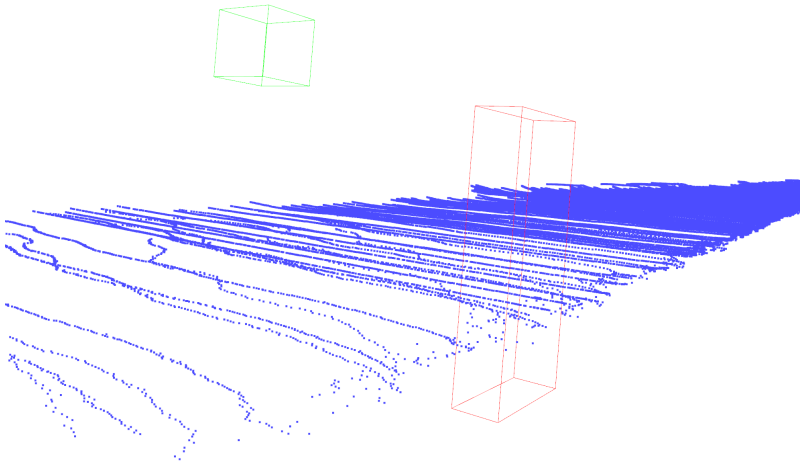


Figure 6.3: The process of collecting labeling road points with RCM measurements. The green box symbolizes the origin of the INS sensor and the red bounding box is used to collect the LiDAR points.

The following list summarises the content of the created dataset.

- Original point coordinates.
- The reflectance value of the point.
- The return index of the point
- The distance to the point from the LiDAR.
- The elevation and azimuth angle to the point with respect to the LiDAR.
- The elevation of the point.
- The index of the LiDAR scan the point belonged to.
- The extraction ID, meaning the ID of the bounding box used to match the point with the RCM data.
- The surface condition of the point.
- The friction coefficient of the point.

Most of the data was gathered in northern Sweden during January, February and March. Some data was also collected in Gothenburg during the month of April. When gathering the data, road sections containing sharp turns or sequences where other vehicles were driving close in front of the ego vehicle were mostly excluded. Furthermore, all data was gathered in conditions where the environment could be examined using the related camera feed. Therefore, no data was collected during nighttime.

6.2 Overview of the Dataset

After the road state dataset had been created, its content was examined from various aspects. The variety of points from different surface conditions was investigated to see if there were any classes that was over- or underrepresented. It was also of interest to observe if the distribution of different features differed based on the surface condition or friction coefficient. Since the reflectance value was suspected to be the key component when estimating the road state with the LiDAR, its distribution was examined in relation to other features to determine if any distinguishing patterns between could be found.

Fig. 6.4 shows the amount of labeled points for each surface condition, where the exact number of points for a surface condition is displayed above its corresponding bar. It is seen that the dataset is imbalanced, where the surface conditions dry and icy have most data, while wet is the surface condition with the least number of labeled points.

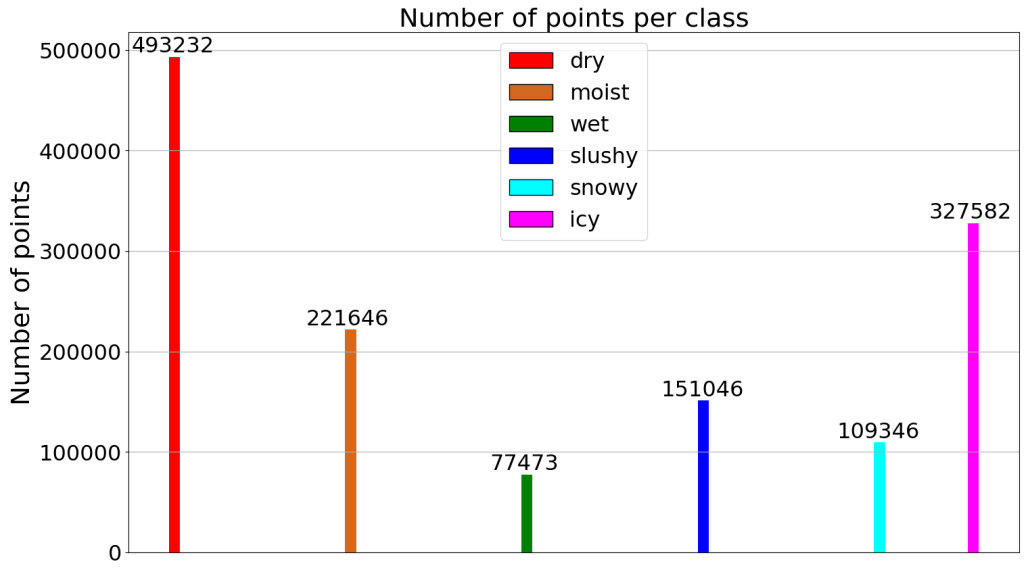


Figure 6.4: The number of points for each class in the dataset, where the exact number of points for a surface condition is displayed above its related bar.

The distribution of the distance to the points from the LiDAR in the dataset is found in Fig. 6.5. Most of the gathered points were located at a distance of between 6-12 meters from the LiDAR. It is also observed that the number of points decreases as the distance increases, and that there are few points within the interval of 30 to 50 meters.

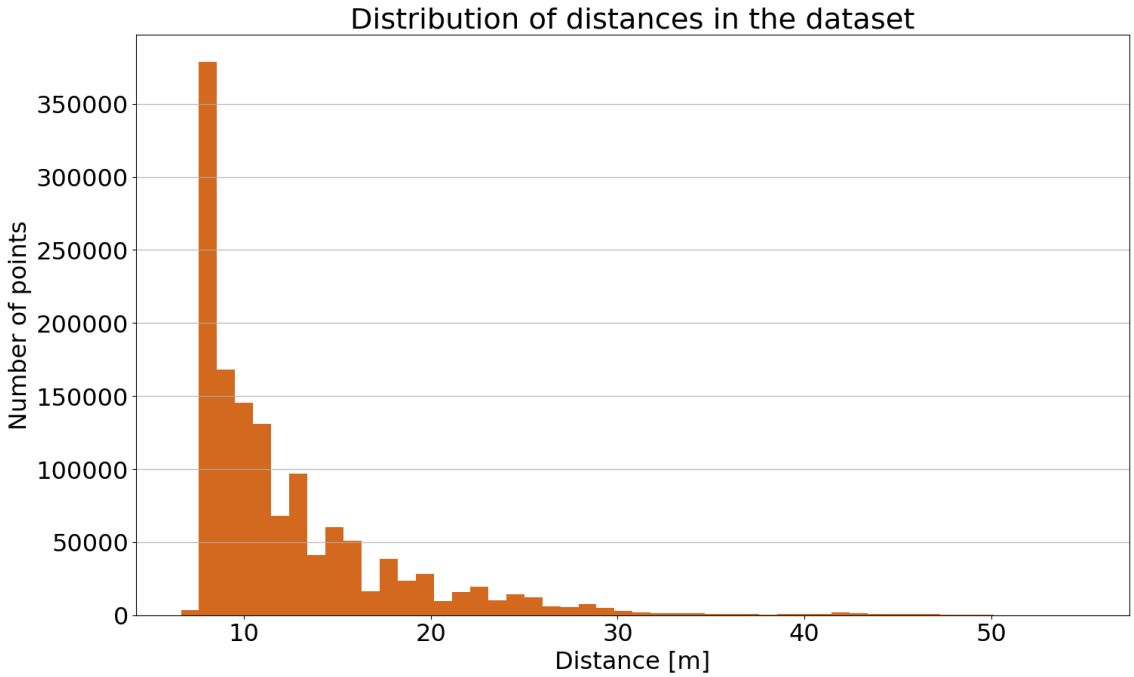


Figure 6.5: The distribution of the distance between the LiDAR and the points in the dataset.

The reflectance of the points for each surface condition is visualized in Fig. 6.6. Here, it is evident that all surface conditions have peaks in the lower range of the reflectance values. Further, the dry surface condition is more evenly spread out compared to the other surface conditions. The remaining surface conditions show a comparable behavior, but with fewer data points between 0.05 and 0.2. Additionally, Moist, wet, slushy, and snowy surface conditions have their reflectance values concentrated below 0.01. Finally, the distribution of icy points exhibits a peak at 0.025, which represents a somewhat higher reflectance value than that of other surface conditions.

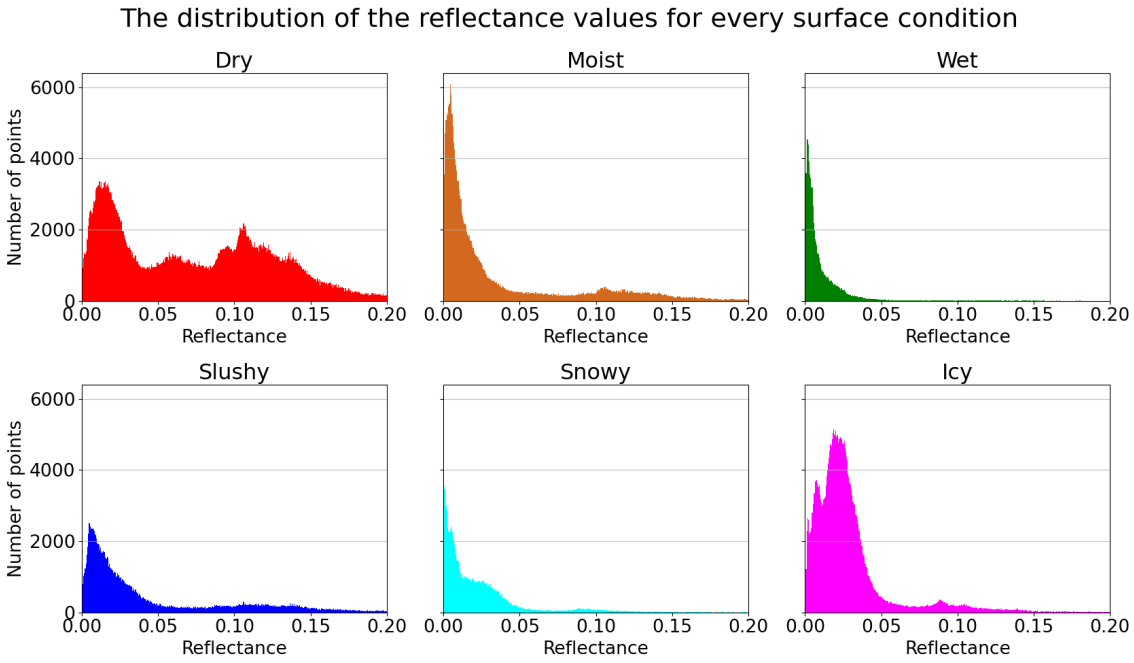


Figure 6.6: The distribution of the reflectance values for each surface condition.

Fig. 6.7 shows the reflectance values for each surface condition and the distance from the LiDAR at which the points where located. It is seen that the figures for moist, wet and snowy surface conditions look similar, but that the moist and snowy points both have more points and higher reflectance values than wet points when the distance is between 10 and 20 meters. Below 10 meters, the snowy surface condition has some points with higher reflectance values, while the wet points have few reflectance values that exceed 0.025. As for the dry surface condition, it can be observed that when the distance increases, points with relatively high reflectance values can still be found. In the interval between 10 and 20 meters, the reflectance values of points are mostly found around 0.01, 0.06 and 0.130. The number of points after 20 meters are much fewer, though reflectance values are still found at 0.01 and 0.06 in the interval between between 20 to 30 meters. The figures for slushy and icy surface conditions look similar to each other, in the sense that most of their points are located at around 7 meters and that the reflectance values are quite low as the distance increases. However, icy points generally have higher reflectance values, and more of these points are found at distances up to 30 meters.

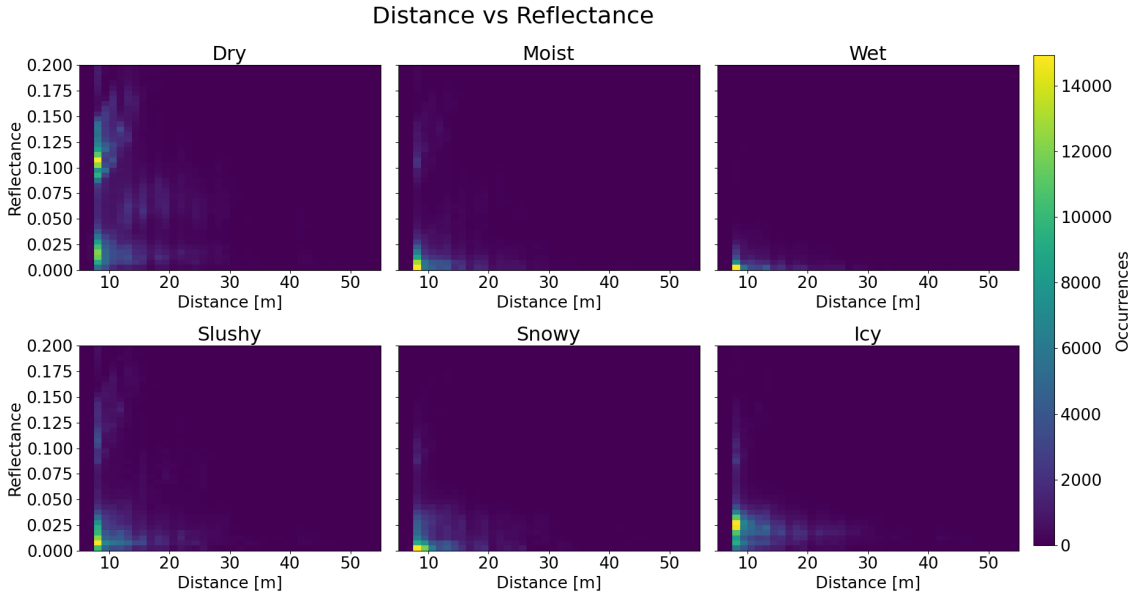


Figure 6.7: The reflectance value of the points for each surface condition, as well as the distance from the LiDAR the points were located at.

The distribution of the friction coefficients for every surface condition is displayed in Fig. 6.8. Most of the points belonging to the dry, moist, wet and slushy surface conditions have friction coefficients around 0.8. However, the distribution for slushy points has more points in the interval between 0.4 and 0.8. The points for snowy and icy surface conditions are on the other hand distributed in the interval between 0.2 and 0.6. It is observed that both have peaks around 0.4, though points for snow seem to be more evenly distributed.

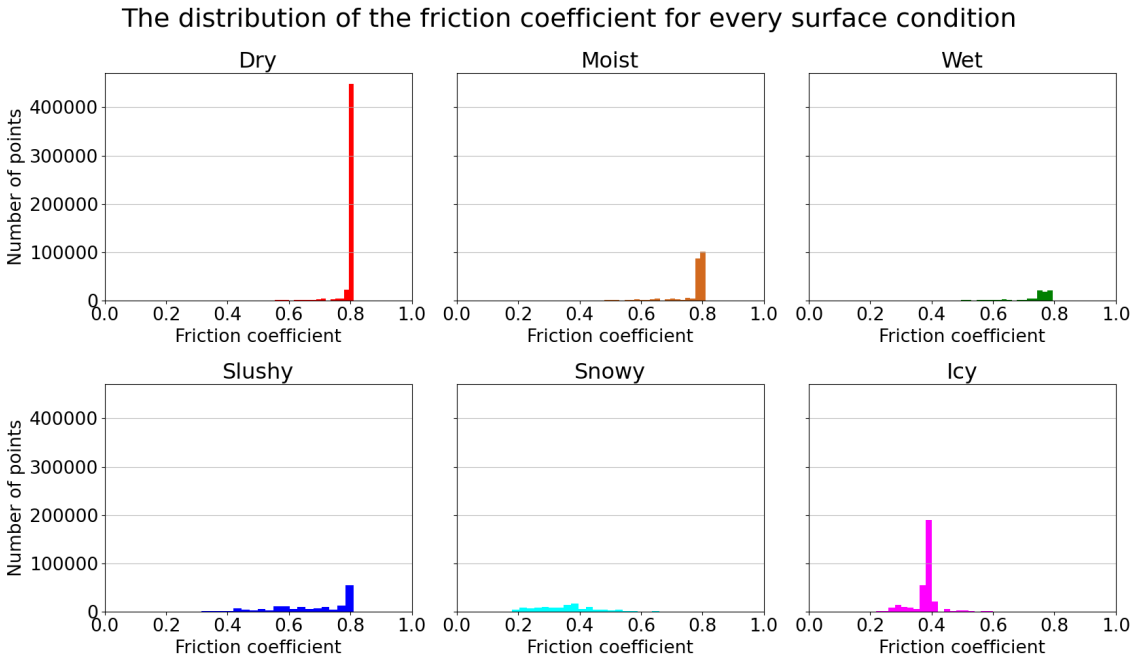


Figure 6.8: The distribution of the friction coefficients for every surface condition.

Fig. 6.9 shows the friction coefficient for different surface conditions and the corresponding reflectance value of each point. When looking at the figures, it is seen that reflectance values greater than 0.1 tend to be associated with higher friction coefficient measurements, and that these higher reflectance values occur for dry, moist and slushy surface conditions. Further, when examining the figures in the interval of 0 to 0.1 of the reflectance values, it is noticed that the friction coefficients range from roughly 0.2 to 0.8 for all surface conditions.

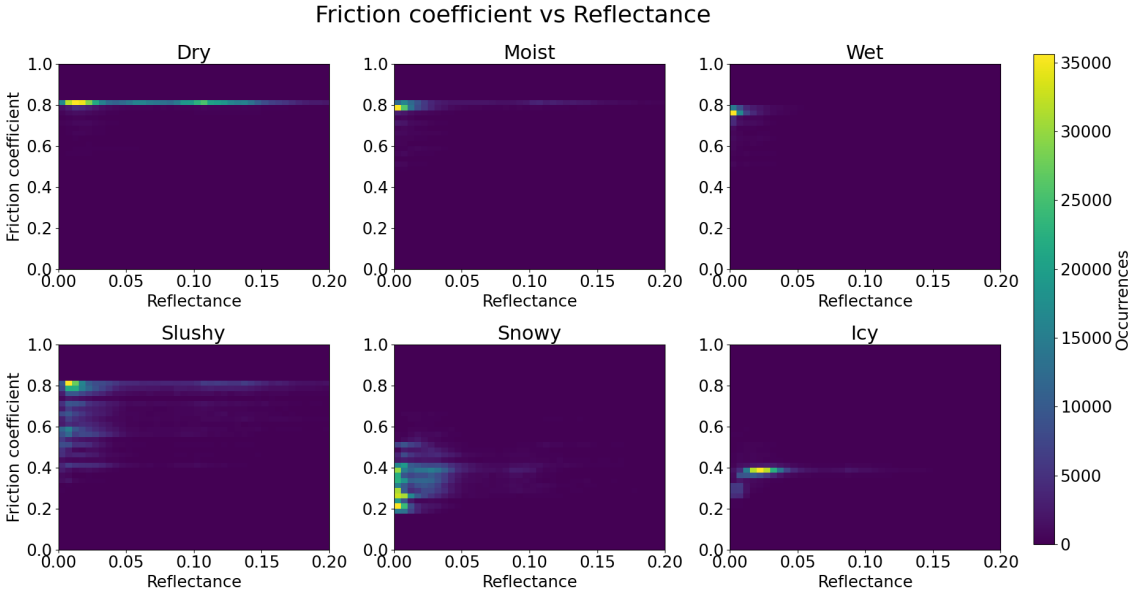


Figure 6.9: The friction coefficient of each point for the different surface conditions, as well as the corresponding reflectance value.

In Fig. 6.10, the distribution of the return index for every surface condition is presented as a probability density function. Most of the points for all classes have return index 1, while slushy, snowy and icy surface conditions have a few more points with higher return indices than the rest of the surface conditions.

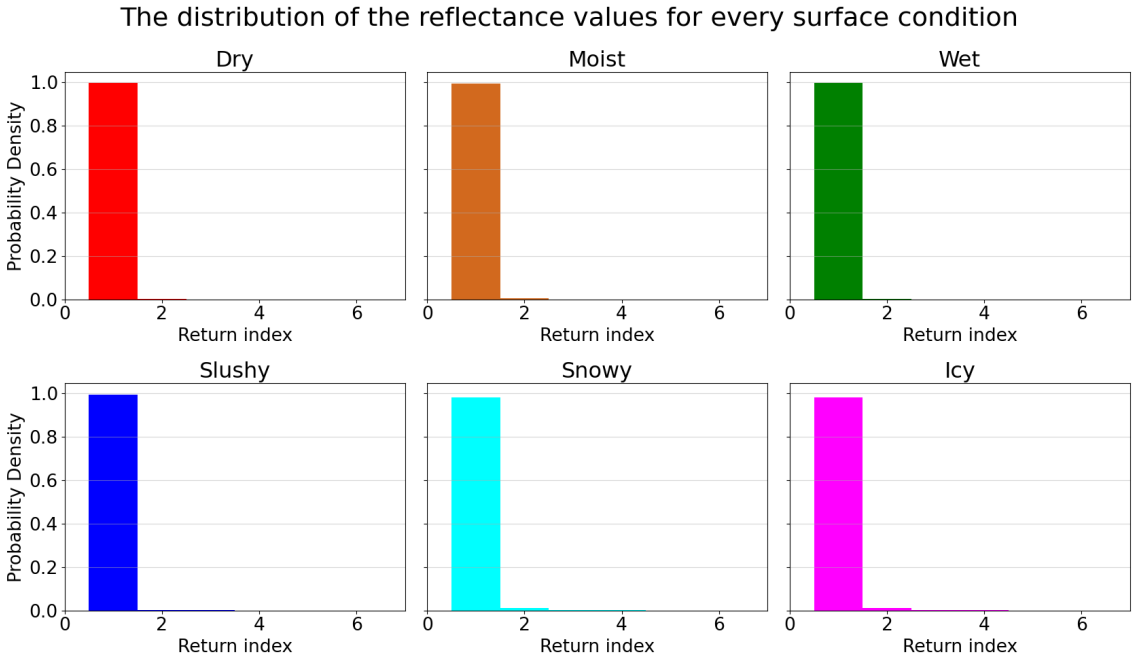


Figure 6.10: The distribution of the return index for every surface condition as probability density functions.

Fig. 6.11 shows the distribution of the friction coefficients for all return indices together, as well as for all return indices separately. It is difficult to find any differences between the figure containing all return indices and the figure containing only return index 1. At the same time, the number of points with a return index greater than 1 is significantly lower, and decreases as the return index increases. Also, the friction coefficient for return index 1 has more high than low values. Conversely, the lower friction coefficient values makes up the majority when the return index is greater then 1.

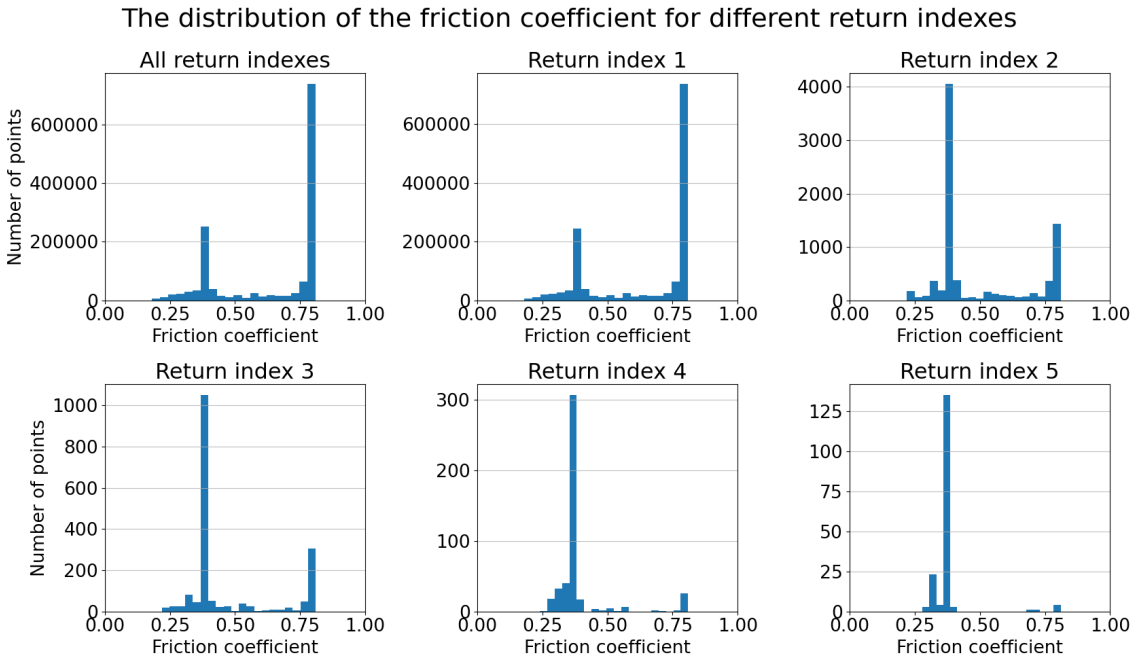


Figure 6.11: The distribution of friction coefficients for all return indices together, as well as for all return indices separately.

7

Effects of Various Road Surfaces on LiDAR Features

This chapter provides an explanation of the analysis process for the reflectance value and the return index. Furthermore, it presents the outcomes of this analysis, and highlights the significance of these features in characterizing the different surface conditions.

7.1 Reflectance Value and Return Index

To better understand how different surface conditions affect the reflectance value and return index, two different types of plots were created; one for examining the reflectance value and another for examining the return index. The LiDAR used in this thesis receives reflectance values between 0 and 2. Thus, a jet colormap was created that ranged between these values. When visualizing a point cloud, the reflectance value of each individual point was used to assign it a color. This way, points with a lower reflectance value were represented by a blue color, while points with a higher reflectance value were represented by a red color. Since all of the points on the road tended to have a low reflectance values, they were multiplied with a scaling factor of 12 to be able to distinguish between smaller differences in the reflectance values. To investigate if the return index varied for different road surface conditions, each point in the point cloud was assigned a colour based on its return index. Here, only the 5 first return indices were examined.

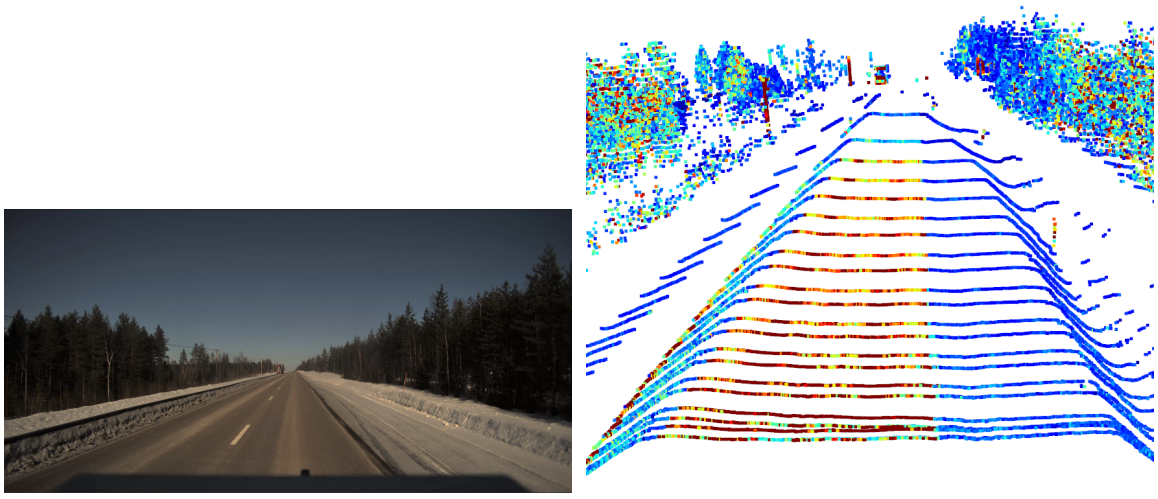
7.2 Analysis of the Reflectance Value and Return Index on Different Road Surfaces

The result from the analysis of the reflectance value and the return index is divided into two parts. The first part shows the findings from visualizing the reflectance value, after which the result from the analysis of the return index is presented.

7.2.1 Reflectance Value

The next series of figures show the reflectance value of the points in point clouds from different scenes, together with the corresponding camera image. As mentioned in section 7.1, the colours of the points range according to a jet colour map, where blue points represent lower reflectance values and red points represent higher reflectance values.

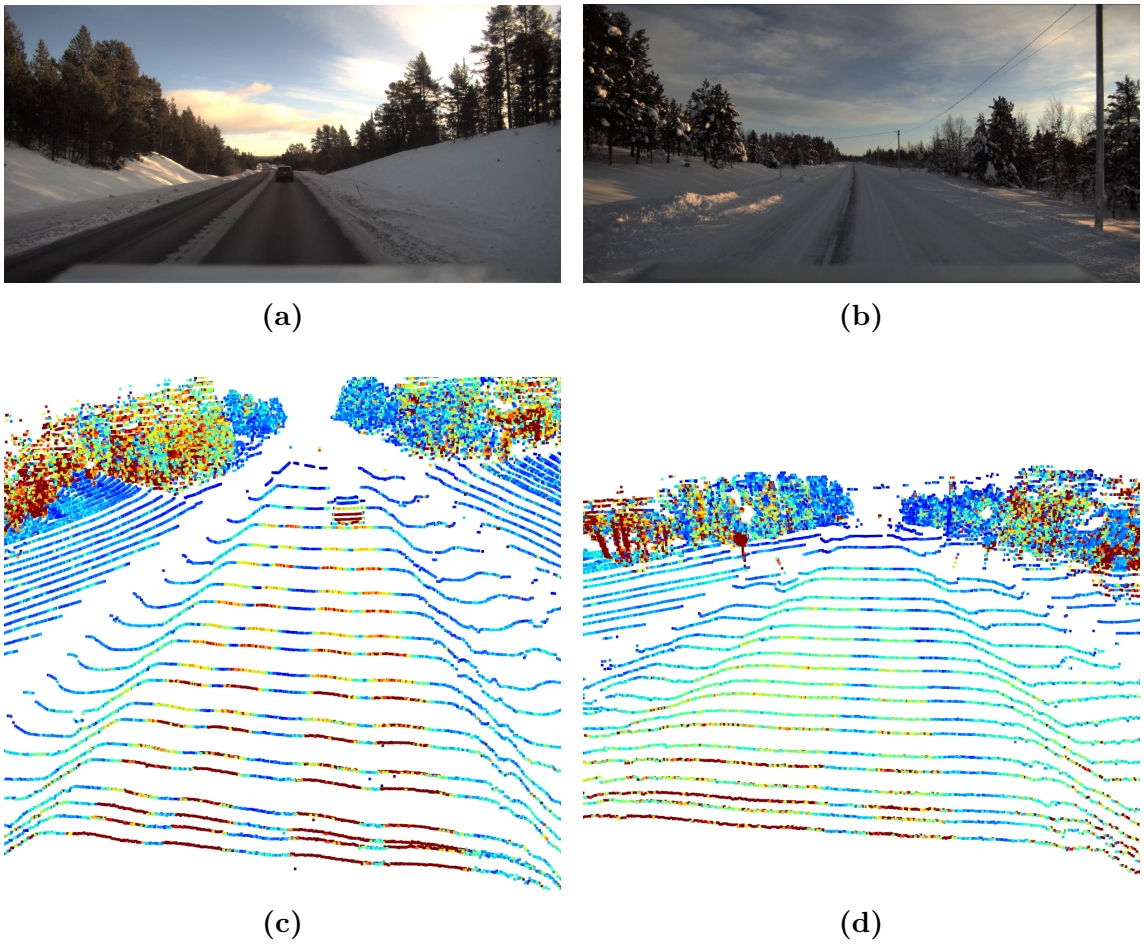
Fig. 7.1a shows a dry road which goes by a parking space that is covered with a layer of snow, while Fig. 7.1b shows the related point cloud of the scene. There is a clear distinction between the dry asphalt and the snowy part of the road, where the dry asphalt is seen to have higher reflectance than the snowy parking space.



(a) **(b)**

Figure 7.1: Reflectance values in a scene where a snow covered parking space is placed next to a dry road. The points on the snowy part have a lower reflectance value than those on the dry road.

In Fig. 7.2 the tire tracks of previous vehicle often had an impact on the reflectance value, where the camera image of a examined scene is found above the visualization of the point cloud. Fig. 7.2c represents the most commonly found scenario, where the tire tracks have a higher reflectance value than the surrounding road. However, sometimes the tire tracks instead have a lower reflectance value, which is visible in Fig. 7.2d. Comparing Figs. 7.2a and 7.2b, it is seen that the tire tracks in Fig. 7.2a seem to be mostly dry, while they are covered or partly covered in snow in Fig. 7.2b. By examining Figs. 7.2b and 7.2d, it can also be observed that the lane marking in the left most tire track does not seem to affect the reflectance value.

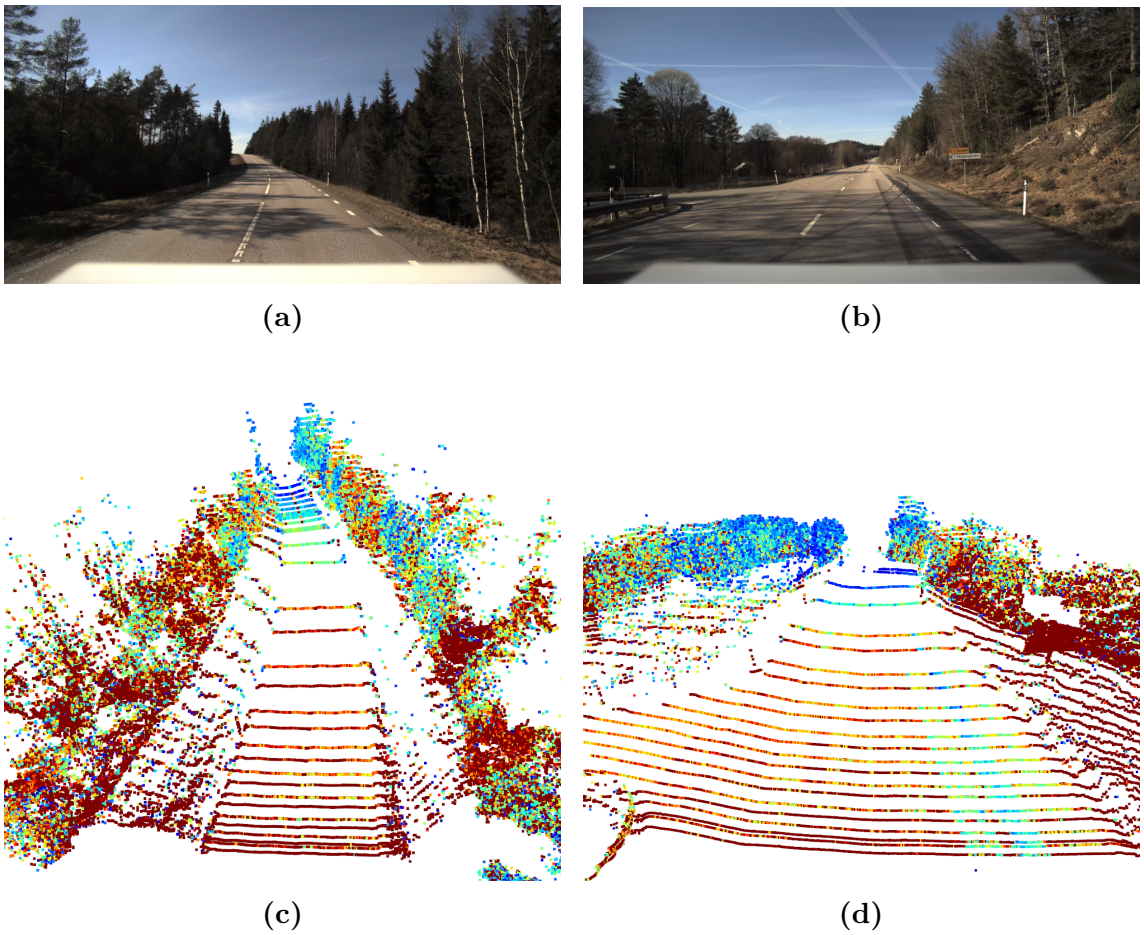


(a) **(b)**

(c) **(d)**

Figure 7.2: Two scenes where the tire tracks of previous vehicles affect the reflectance value. The tire tracks in figure 7.2a and 7.2c have a higher reflectance value than the surrounding road, while the tracks in figure 7.2b and 7.2d have a lower reflectance value.

In Fig. 7.3 two scenes with dry roads are displayed. Figs. 7.3a and 7.3c show a slope with an upward curvature, where the reflectance values decrease as the distance increases. Moreover, it is also noticed that the reflectance from the lane markings that separate the lanes still have a higher reflectance at a longer distance. Figs. 7.3b and 7.3d show brake tracks that result in lower reflectance values.

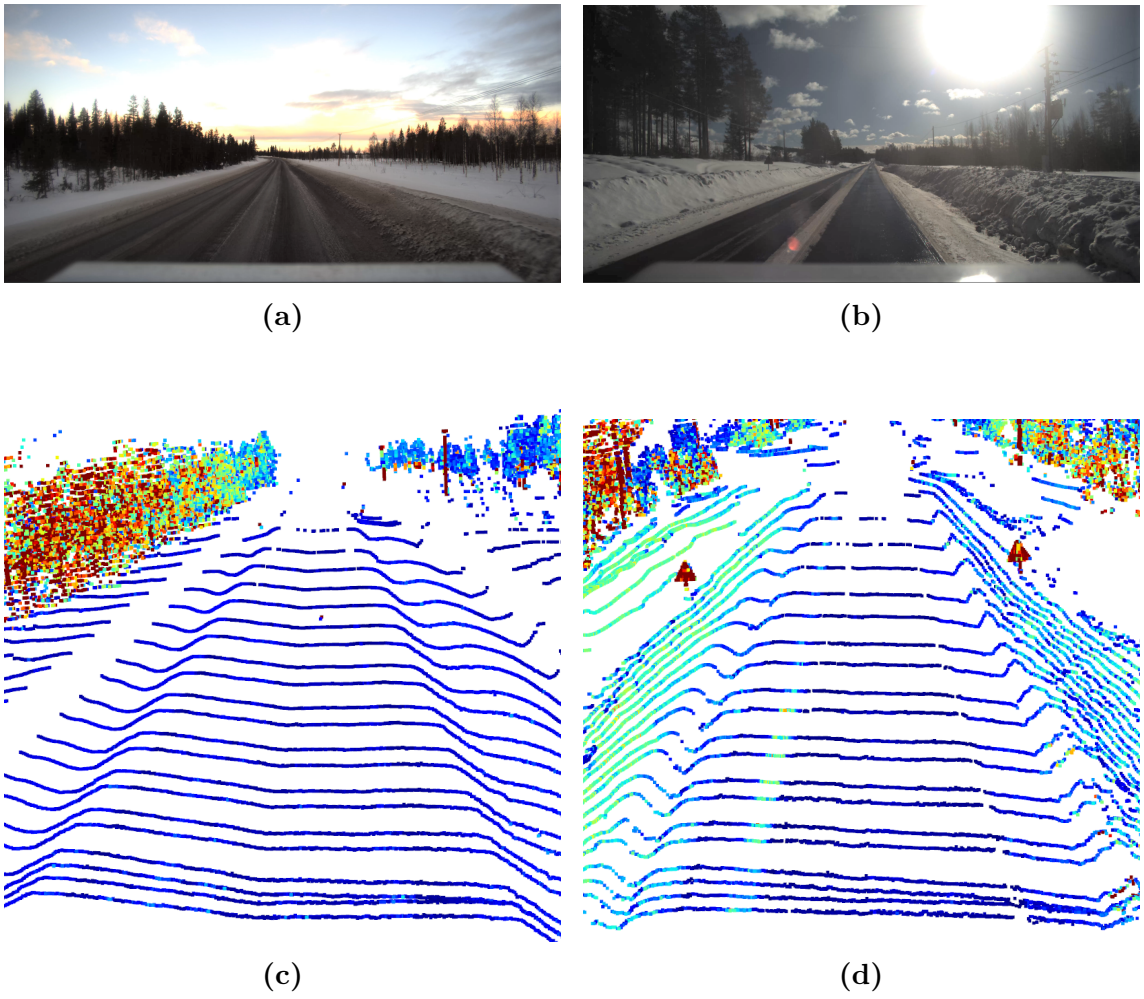


(a) **(b)**

(c) **(d)**

Figure 7.3: Two scenes where the road is dry. Figs. 7.3a and 7.3c show a slope with an upward curvature, where it is clear that the reflectance decreases with the distance. Figs. 7.3b and 7.3d show brake tracks that result in lower reflectance values.

Fig. 7.4 shows two new scenes, one where the road is covered with slush, seen in Figs. 7.4a and 7.4c, and another where the road is wet, seen in Figs. 7.4b and 7.4d. In both scenes, the road surfaces have quite low reflectance values. In Fig. 7.4d, a line of higher reflectance values is found on the left hand side of the road, where there appears to be some remnants of snow when comparing with Fig. 7.4b. However, the line of snow that is clearly seen in Fig. 7.4b is not visible in Fig. 7.4d.

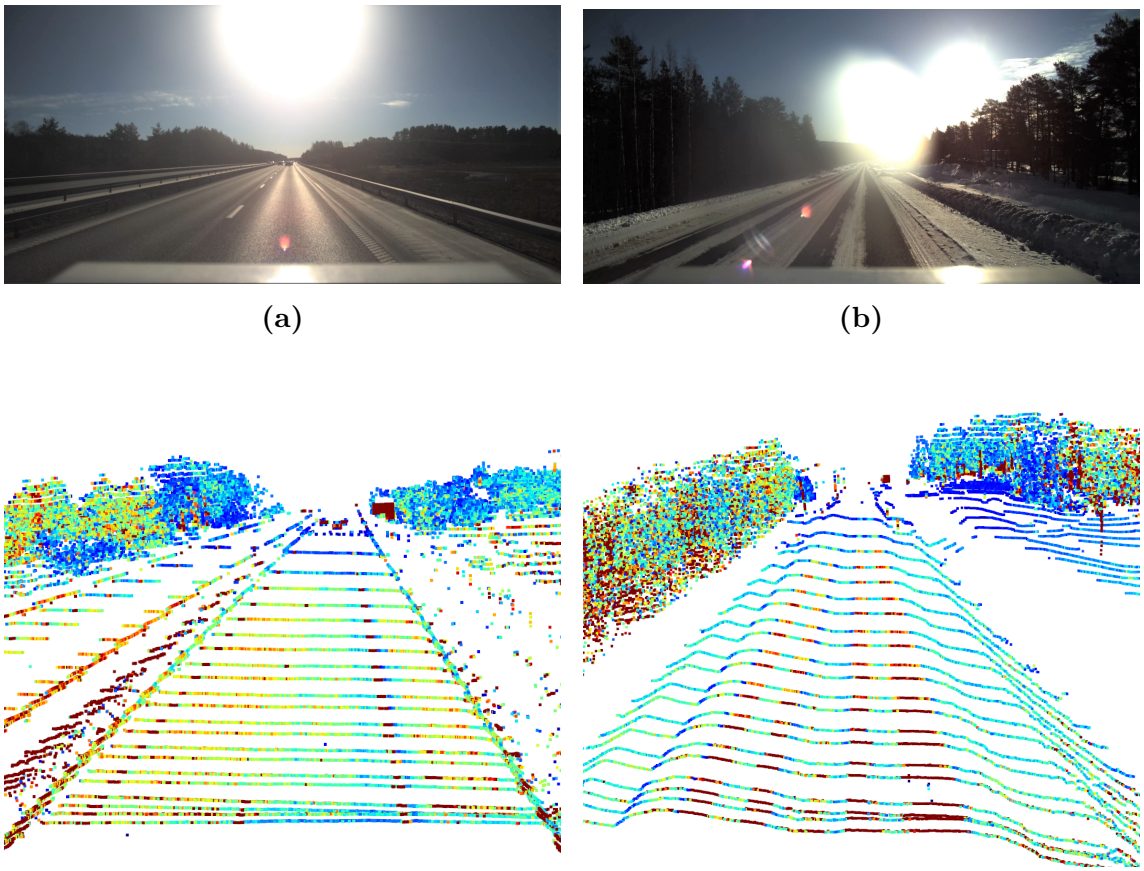


(a) **(b)**

(c) **(d)**

Figure 7.4: Figs. 7.4a and 7.4c show a scene where the road is covered in slush, while Figs. 7.4b and 7.4d show a wet road. In both cases, the points on the road have low reflectance values.

Fig. 7.5 presents two scenes where the sun is directly in front of the ego vehicle. In Figs. 7.5b and 7.5d, where the tire tracks seem to be dry while the rest of the road is covered in snow, the tire tracks have a higher reflectance than the surrounding road. This behaviour is consistent over the whole road surface, except for the points that are further away, which all have lower reflectance values. Figs. 7.5a and 7.5c show a scene where the whole road segment appears to be dry. It can however be seen that the road that is directly in front of the vehicle has a lower reflectance value than the road segments placed at the sides. The lane markings at the side of the road are clearly visible in the visualization of the point cloud, although the lane markings that separate the lanes are not as prominent.



(a) **(b)**

(c) **(d)**

Figure 7.5: Two scenes where the sun is placed directly in front of the ego vehicle. Figs. 7.5a and 7.5c show that the road directly in front of the vehicle has a lower reflectance value than the road segments on the sides, while Figs. 7.5b and 7.5d show a difference in reflectance values between the dry tire tracks and the snowy surrounding road.

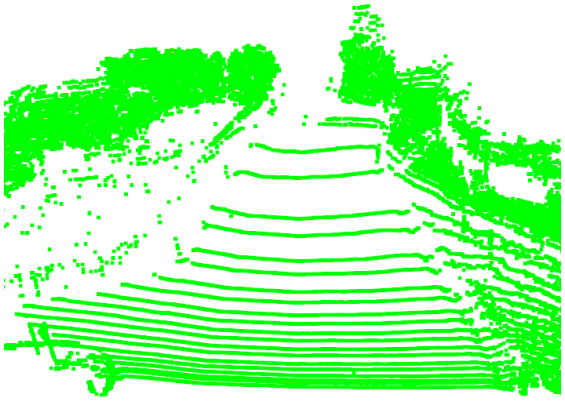
7.2.2 Return Index

The following figures show the points of a LiDAR scan with return index 1 to 5 together with the corresponding camera image. Since there are few points with return index 3, 4 and 5, these points are visualized together in the same figure. As explained in section 7.1, points are coloured based on their return index. The colour of each return index is listed below:

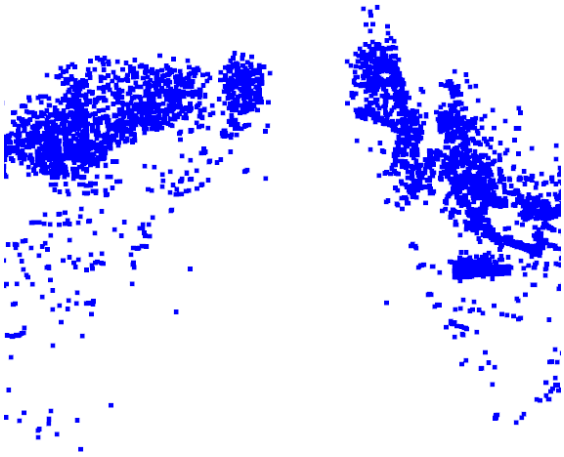
- **Return index 1:** Green
- **Return index 2:** Blue
- **Return index 3:** Red
- **Return index 4:** Yellow
- **Return index 5:** Cyan

Fig. 7.6 shows the return indices on a dry road. It is seen that all of the points that belong to the road section have return index 1.

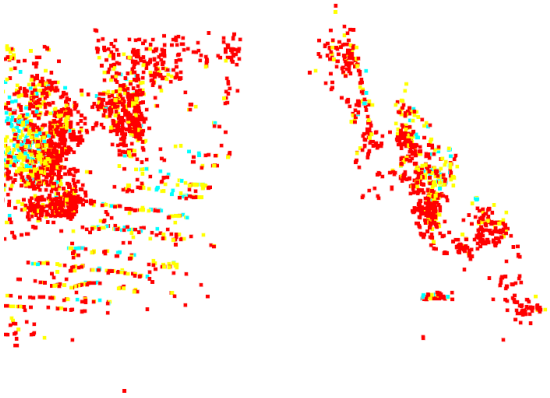
(a) Camera feed



(b) Return index 1



(c) Return index 2

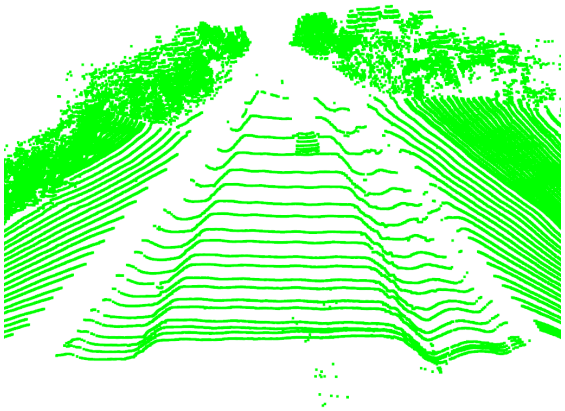


(d) Return index 3, 4 and 5

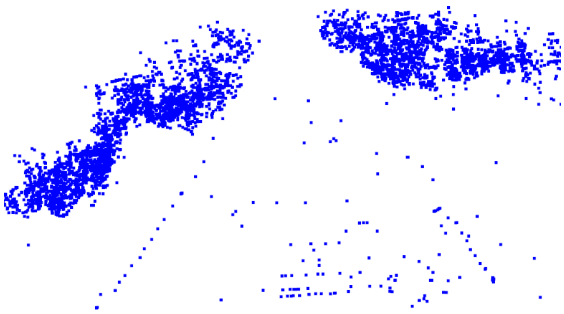
Figure 7.6: Return indices on a dry road.

Fig. 7.7 shows the return indices on a road that appears to be mostly dry, but with a patch of snow between the lanes. In Figs. 7.7b to 7.7d, the majority of the points that are on the road come from the first return index. Furthermore, there are few points that have return index 2, while there are no points with return index 3, 4 or 5 are on the road.

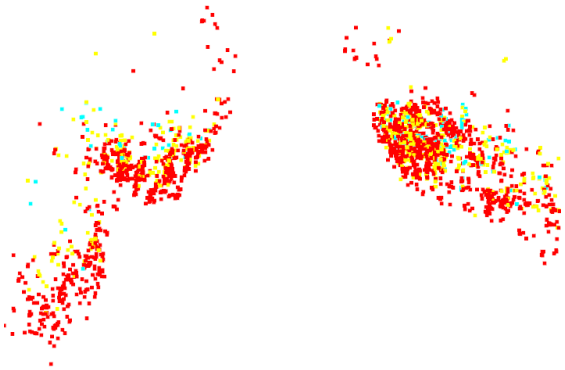
(a) Camera feed



(b) Return index 1



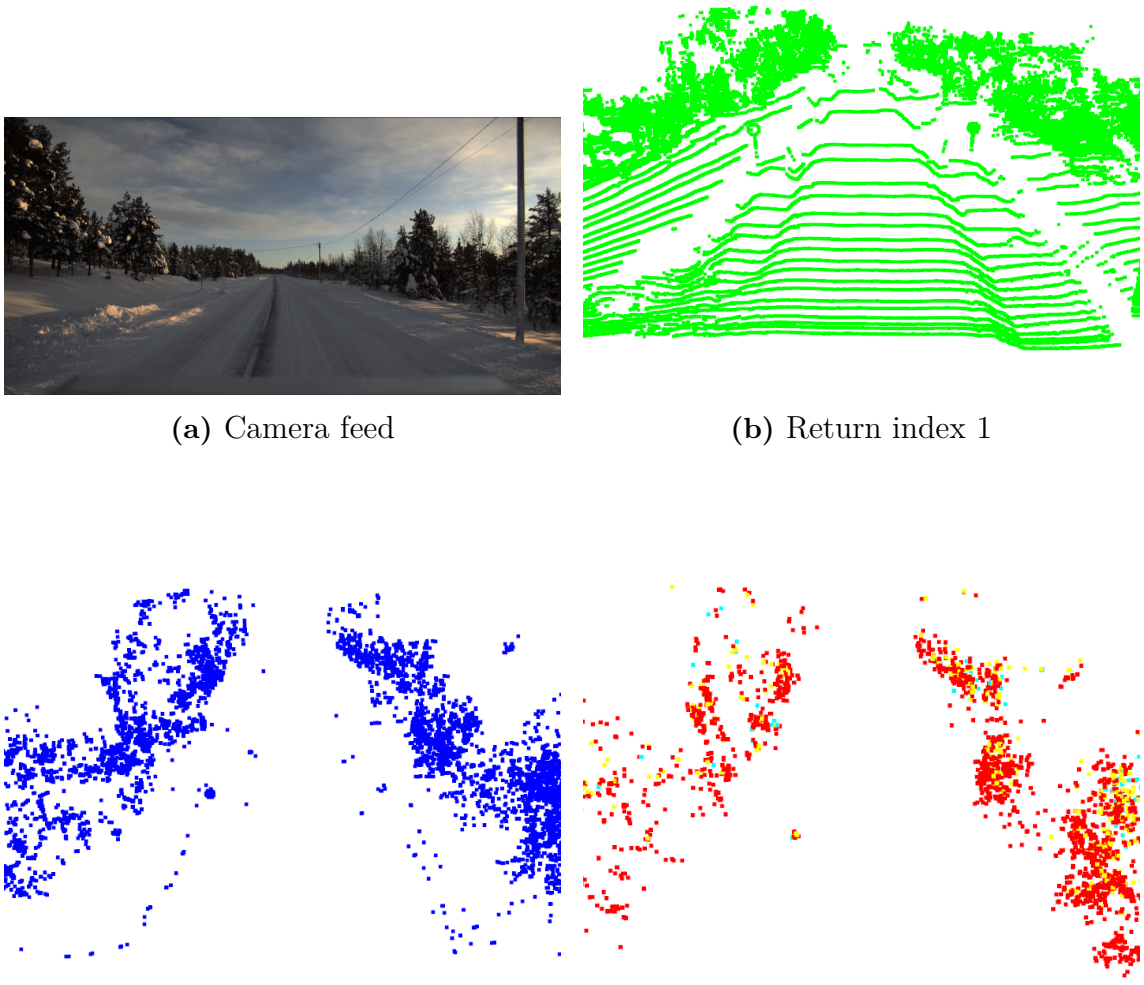
(c) Return index 2



(d) Return index 3, 4 and 5

Figure 7.7: Return indices on a road that has dry tire tracks, but also contain snow.

Fig. 7.8 shows the return indices on a road that is covered in ice and snow. Similarly to Fig. 7.6, most of the points on the road have return index 1. However, nearly no points with return index 2 end up on the road.



(a) Camera feed

(b) Return index 1

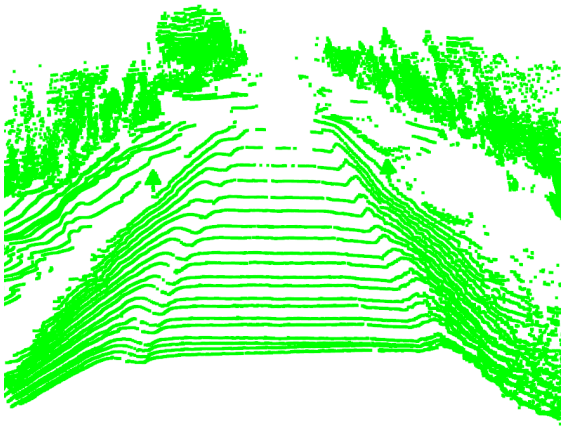
(c) Return index 2

(d) Return index 3, 4 and 5

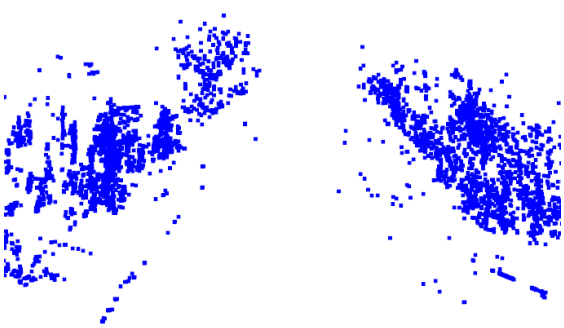
Figure 7.8: Return indices on a road that is covered in ice and snow.

Fig. 7.9 show a road segment that is mostly wet. As for the first two figures, almost all of the points on the road have return index 1.

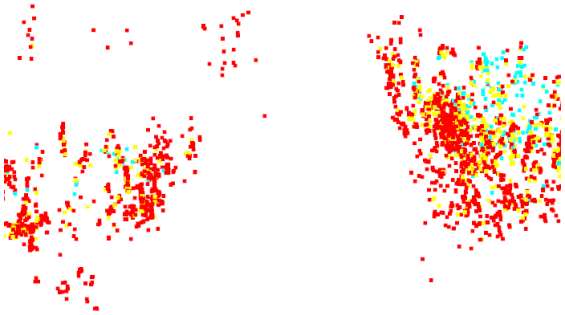
(a) Camera feed



(b) Return index 1



(c) Return index 2



(d) Return index 3, 4 and 5

Figure 7.9: Return indices on a road that is wet.

8

Road State Estimation

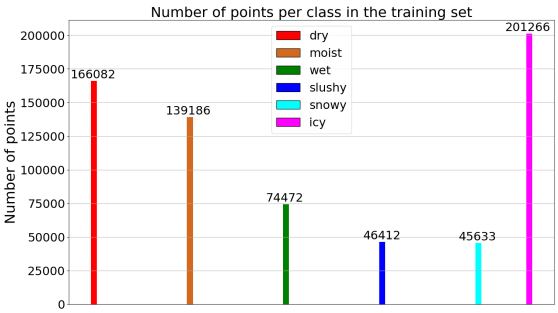
This chapter explores the potential of using the LiDAR sensor to estimate the road state. First, the architectures of the tested machine learning models are introduced, after which the evaluation of the models is described. Finally, the findings and results regarding the performance of the machine learning models are presented.

8.1 Proposed Method for Road State Estimation

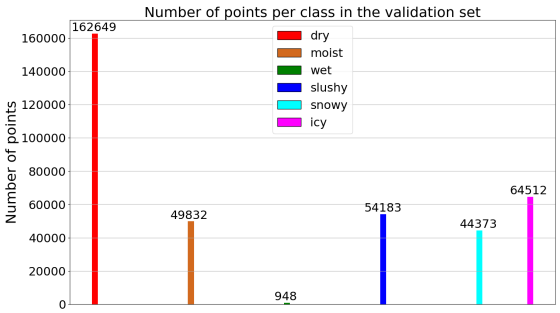
In the first section of this chapter, the methodology used for estimating the road state is introduced. It starts by detailing how data was divided into training, validation, and test sets, along with the preprocessing steps that were conducted. Furthermore, the network architectures of the multilayer perceptron and the transformer models are explained.

8.1.1 Partitioning and Preprocessing of Data

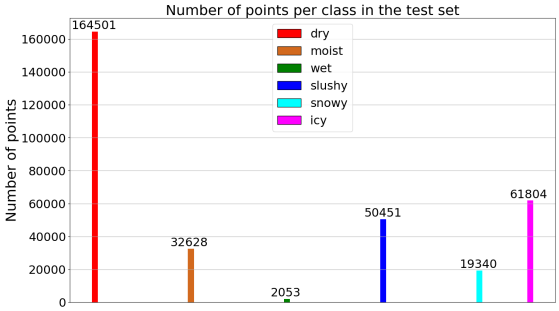
After creating the road state dataset, it was divided into a training, validation and test set. To ensure that the machine learning models would not be evaluated on the same data as they were trained on, it was desired to put data acquired from a single day into only one of the three datasets. However, due to the availability of data, this was not possible. Instead, only data sequences from a day that was separated with at least one hour were allowed to be put in different datasets. Fig. 8.1 shows the number of points per class in the training, validation and test sets. It is seen that all datasets are imbalanced, and that there are few data points for wet surfaces in the validation and test sets. The reason for this is that almost all data for wet surface conditions were present during a short time period on one single day. Therefore, it was not possible to split the wet data without allowing the points from the same sequence to be spread out into the training, validation and test set. A similar reason also explains why there is less snowy data in the test set. Finally, there are many points classified as dry present in all of the datasets, while there is considerably more moist, wet and icy data in the the training set compared to the validation and test sets.



(a) Points per class in the training set.



(b) Points per class in the validation set.



(c) Points per class in the test set.

Figure 8.1: The number of points per class in each dataset, where the exact number of points for a surface condition is displayed above its related bar.

Fig. 8.1 shows that there were not enough wet data points in the validation and test sets. Furthermore, including the slushy surface condition could create complications. As slush is a mixture of the other surface conditions, the classification problem might become ill-defined. This could potentially affect estimations of the remaining surface conditions. As a result, tests were primarily conducted without the wet and slushy classes. However, for the sake of completeness, additional tests were performed with these classes included.

To be able to estimate the road state, one idea was to use information from a group of points located next to each other on the road surface. When creating such groups, points were grouped based on LiDAR scan index and extraction ID. That is, all the points within a group had to belong to a single LiDAR scan, as well as a single RCM measurement. As a result, all points belonging to a specific group of points were labeled with the same surface condition and friction coefficient. However, the number of points belonging to a certain group could vary, depending on how many points that were located inside of the aforementioned bounding box, as seen in Fig. 6.2. The distribution of the number of points located within each group is depicted in Fig. 8.2. Here, it is seen that the most common number of points within a group is 10, and that the occurrences are much fewer when the number of points is 20 or higher.

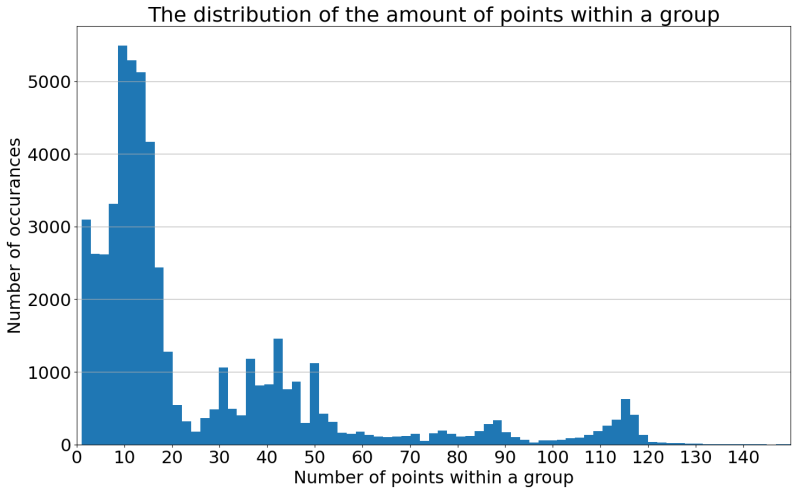


Figure 8.2: The distribution of the number points that were located within each group of points.

When estimating the surface state, it was desired to find out how the performance was affected when using different group sizes as input. Therefore, a fixed number of points that had to be present within each group, N_{group}, were introduced. After determining an appropriate N_{group}, all groups containing less points than N_{group} were removed, while an N_{group} number of points were randomly chosen from the groups that contained too many points. Tab. 8.1 shows the number of groups when setting $N_{\text{group}} = 1, 2, 3, 5, 10, 20, 40$ for the training, validation and test sets respectively, where it can be observed that the amount of data reduces as N_{group} grows. To minimize data loss, as well as the information loss in each individual group, $N_{\text{group}} = 5$ was chosen as default.

Table 8.1: The number of groups of points in the training, validation and test sets using different values of N_{group}.

N_{group}	**Training**	**Validation**	**Test**
1	21227	12232	10790
2	20545	11841	10487
3	19951	11529	10191
5	18890	10937	9628
10	15427	9099	7914
20	7204	4253	3620
40	4686	2795	2385

To reduce the effect of the aforementioned class imbalance, two methods were used. The first method was to perform undersampling on the training dataset, such that all of the classes had as many groups of points as the class with the least number of groups. While this removed the class imbalance in the training set, it also reduced the amount of data. The number of groups for the undersampled training set is

found in Tab. 8.2, together with the number of groups without resampling. The second measure for addressing the class imbalance was to compute class weights for the validation set. The reason for why weights were used instead of undersampling was to keep as much data as possible. Conversely, undersampling was used instead of class weights for the training set, as this gave better results in the initial tests with the machine learning models.

Table 8.2: Comparison of the number of groups in the training set with and without undersampling for different values of N_{group}.

N_{group}	**Training set without undersampling**	**Training set with undersampling**
1	21227	19256
2	20545	18586
3	19951	18024
5	18890	17046
10	15427	13844
20	7204	6434
40	4686	4162

In addition to dividing the points into groups, undersampling and assigning weights, the x and y point coordinates, the elevation, the distance and the angles for elevation and azimuth were normalized, since they sometimes assumed larger values. This was done to make sure that all input features would contribute equally to the training process, as well as make the trained model more robust and faster to train [34].

To explore the potential of using LiDAR when estimating the surface state, various combinations of the classes were tested. Therefore, five class groupings were created, as is shown in Tab. 8.3. The first class grouping shows that the classes were either considered to have a high or low friction coefficient, where all points with friction coefficient values above 0.5 were assigned to the high friction coefficient class, while the remaining points were sorted to the low friction coefficient class. In the second class grouping, the dry and moist surface conditions have been merged into one class, while the same has been done for snowy and icy surface conditions. In class grouping 3, dry, moist, snowy and icy surface conditions represent separate classes. In the two final classes, the wet and the slush data have been included, and were therefore not examined extensively. However, they were added to explore the capabilities of the machine learning models. In addition to the class groupings, different feature combinations were also investigated. All tested feature combinations are presented in Tab. 8.4.

Table 8.3: The different class groupings.

Class grouping	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Grouping 1	High friction coefficient	Low friction coefficient	-	-	-	-
Grouping 2	Dry Moist	Snowy Icy	-	-	-	-
Grouping 3	Dry	Moist	Snowy	Icy	-	-
Grouping 4	Dry Moist	Wet Slushy	Snowy Icy	-	-	-
Grouping 5	Dry	Moist	Wet	Slushy	Snowy	Icy

Table 8.4: The different feature combinations.

Feature combination	Feature 1	Feature 2	Feature 3	Feature 4
R	Reflectance	-	-	-
R, D	Reflectance	Distance	-	-
R, D, E	Reflectance	Distance	Elevation	-
R, P	Reflectance	x-coordinate	y-coordinate	z-coordinate
R, D, EA, AA	Reflectance	Distance	Elevation angle	Azimuth angle

8.1.2 Multilayer Perceptron

To explore the capabilities of neural networks for road state estimation, a simple multilayer perceptron was implemented to serve as a baseline model. The architecture of the MLP was chosen such that a collection of N_{group} points, where each point had N_{ft} number of features, could be considered at once when predicting the surface condition. To be able to process the features of N_{group} points at the same time, the features of all points were concatenated into a single vector of size $N_{\text{group}} \cdot N_{\text{ft}}$ before being sent to the input layer. Only one hidden layer was used in the MLP, and the size of this layer varied between 32, 64 and 128. The size of the output layer was set to match the number of classes to be predicted. As for activation functions, a ReLU function was applied in all layers except for the output layer, where a log softmax layer was used to predict the log probability of each class. The loss function used when training the MLP was the cross-entropy loss, while Adam was chosen as the optimizer.

8.1.3 Transformer

In addition to the MLP, a more advanced architecture was implemented to compare the performance of the two. The network chosen was a transformer due to its capability to process multiple points simultaneously. Given that the transformer network introduced by A. Vaswani et al. in [29] was originally designed to process text, it was necessary to modify the architecture before using it to process the information obtained from the LiDAR. As opposed to text, the LiDAR data does not strictly require tokenization, since each data point by itself can be regarded as a token. However, the data points still needed to be converted into embeddings. The embedding method chosen uses a single fully-connected layer, which takes the features of a data point and maps them into a higher-dimensional space.

The final architecture is illustrated in Fig. 8.3. The input points are first prepended with a [CLS] token, after which the set of points are embedded with a fully-connected layer. Then, the embeddings are processed in the encoder layers. Here, the feedforward layer consists of a fully connected layer with a size equal to four times the size of the embedding dimension, followed by a ReLU activation function and another fully connected layer with a size equal to the embedding dimension. At the end of the network, the embedding corresponding to the [CLS] token is extracted and propagated through an MLP, consisting of a single fully connected layer with a size equal to the number of classes, and a log softmax function to produce a prediction. As in section 8.1.2, cross-entropy loss and Adam were used when training the transformer.

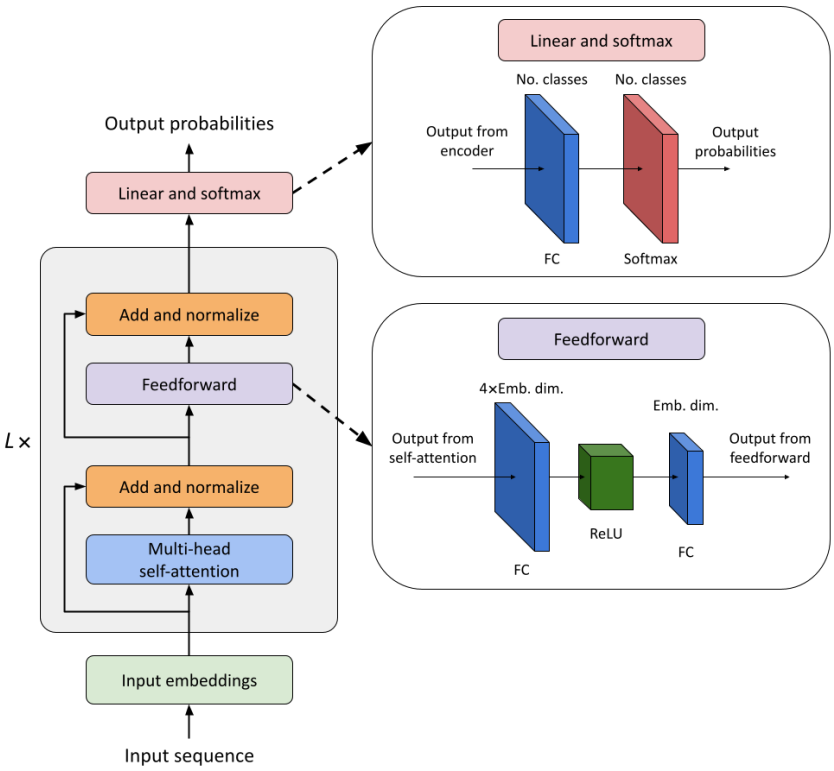


Figure 8.3: The chosen architecture of the transformer network.

8.2 Evaluation of the Road State Estimation Methods

This section introduces the experiments performed to evaluate the performance of the machine learning models introduced in section 8.1. It then proceeds to present the outcomes obtained from these experiments.

8.2.1 Experiments and Evaluation

To evaluate the potential of using machine learning algorithms for estimation of the surface condition or friction coefficient using LiDAR data, both models were tested for different class groupings and feature combinations. Due to the result presented in section 7.2.2, only points with return index 1 were considered. For class grouping 1, 2 and 3, a base model was found by tuning hyperparameters, such as the number of epochs and learning rate. Additionally, the size of the hidden layer was tuned for the MLP, while the embedding dimension, depth and number of heads was tuned for the transformer model. All base models were tuned using feature combination [R, D]. After finding a base model, it was tested on all feature combinations in Tab. 8.4. When testing the different feature combinations, some additional tuning was performed in some cases where it was necessary. Both models were evaluated by examining the macro average of the F1-score, precision and recall, as well as the balanced accuracies, class accuracies and confusion matrix.

The performance of both models was also examined when varying N_{group} according to the values described in Tab. 8.1. This experiment was performed using feature combination [R, D] and class grouping 1 and 2. When testing different values for N_{group}, it was however suspected that points chosen from groups that contained 5 points or less had a higher chance of having lower quality than the points that were chosen from groups that contained more than 5 points. To test this theory, an experiment was conducted for $N_{\text{group}} = 1, 2, 3, 5$, where the points instead were chosen from groups that contained 10 points or more. This experiment was only conducted with the transformer model.

Finally, the transformer was tested when wet and slushy data was included in the dataset. First, class grouping 4 and 5 were examined, to examine the performance when presented with more challenging cases. Then, class grouping 1 was also evaluated when friction coefficient values from wet and slushy points were present in the data, to see what impact these values had on the performance.

The models used were sensitive to how random operations were performed. As an example, when using the exact same data and parameters, the result could vary much between trials. Therefore, random operations in the implementation were set to be performed the exact same way every time the code was executed.

8.2.2 High and Low Friction Coefficient Classification: Group 1

The following tables display the results from when the MLP and the transformer were trained to estimate class grouping 1 with different feature combinations, that is, when the points used as input had either high or low friction coefficients. The headings of the tables are defined as follows:

- Feature comb. - The feature combinations that were used
- P - Precision
- R - Recall
- F1 - F1-score
- BA - Balanced accuracy
- High - The accuracy when estimating high friction coefficients
- Low - The accuracy when estimating low friction coefficients
- D - The accuracy when estimating the dry surface condition
- M - The accuracy when estimating the moist surface condition
- W - The accuracy when estimating the wet surface condition
- Sl - The accuracy when estimating the slushy surface condition
- Sn - The accuracy when estimating the snowy surface condition
- I - The accuracy when estimating the icy surface condition

The result when estimating class grouping 1 with the MLP is found in Tab. 8.5, where it is seen that feature combination [R, D] achieved both the best balanced accuracy and F1-score. The result was however similar for all feature combinations.

Table 8.5: The performance of the MLP when estimating class grouping 1, using different feature combinations.

Feature comb.	P	R	F1	BA	High	Low
R	0.722	0.769	0.726	0.769	0.812	0.726
R, D	0.729	0.775	0.736	0.775	0.808	0.743
R, D, E	0.727	0.775	0.733	0.775	0.813	0.736
R, P	0.725	0.772	0.730	0.772	0.811	0.733
R, D, EA, AA	0.726	0.770	0.732	0.770	0.798	0.743

Tab. 8.6 shows the results obtained when estimations were made with the transformer. The result was similar to that of the MLP, however the MLP generally had a marginally better balanced accuracy and F1-score.

Table 8.6: The performance of the transformer when estimating class grouping 1, using different feature combinations.

Feature comb.	P	R	F1	BA	High	Low
R	0.690	0.767	0.729	0.767	0.794	0.740
R, D	0.727	0.771	0.735	0.771	0.794	0.748
R, D, E	0.724	0.772	0.729	0.772	0.815	0.728
R, P	0.718	0.760	0.725	0.760	0.775	0.744
R, D, EA, AA	0.723	0.770	0.727	0.770	0.812	0.728

8.2.3 Dry/moist and Snowy/icy Classification: Group 2

The following tables present results from when the MLP and the transformer were trained to estimate class grouping 2 with different feature combinations, that is, when the points used as input either belonged to dry/moist or snowy/icy surface conditions. In the tables showing the results, "D/M" denotes the accuracy for estimating the dry and moist class, while "Sn/I" represents the snowy and icy class.

Tab. 8.7 shows the result when estimating class grouping 2 with the MLP. Here, feature combination [R, D, E], [R, D] and [R] gave the best performances, although as in section 8.2.2, the result was similar for all feature combinations.

Table 8.7: The performance of the MLP when estimating class grouping 2, using different feature combinations.

Feature comb.	P	R	F1	BA	D/M	Sn/I
R	0.725	0.764	0.730	0.764	0.735	0.794
R, D	0.728	0.766	0.734	0.766	0.744	0.789
R, D, E	0.728	0.765	0.735	0.765	0.750	0.780
R, P	0.721	0.757	0.728	0.757	0.748	0.765
R, D, EA, AA	0.726	0.765	0.732	0.765	0.742	0.788

Tab. 8.8 shows the results for the transformer. Compared to the MLP, the result was again similar.

Table 8.8: The performance of the transformer when estimating class grouping 2, using different feature combinations.

Feature comb.	P	R	F1	BA	D/M	Sn/I
R	0.724	0.759	0.731	0.759	0.756	0.761
R, D	0.727	0.766	0.733	0.766	0.742	0.789
R, D, E	0.724	0.759	0.731	0.759	0.754	0.765
R, P	0.719	0.757	0.725	0.757	0.738	0.775
R, D, EA, AA	0.726	0.762	0.733	0.762	0.752	0.773

8.2.4 Classification of Dry, Moist, Snowy and Icy Surfaces: Group 3

When examining class grouping 3, it was found that both of the models had little success in predicting any classes other than dry or icy, as seen in Fig. 8.4. Instead, moist was mostly misclassified as dry, while snow was mistaken for ice. Since both models and all feature combinations produced equally poor results, only one representative case is displayed below. Here, the transformer was used together with feature combination [R, D]. Tab. 8.9 shows the final result.

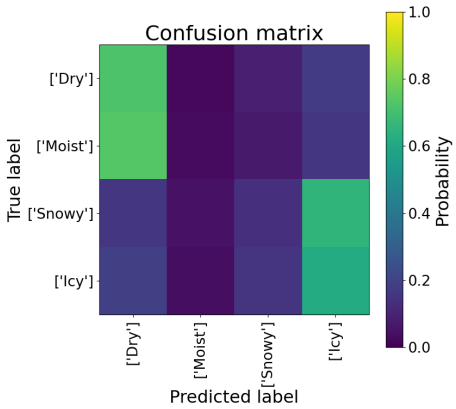


Figure 8.4: Confusion matrix when estimating class grouping 3 using the transformer model and feature combination [R, D].

Table 8.9: The performance of the transformer when estimating class grouping 3, using feature combination [R, D].

Feature comb.	P	R	F1	BA	D	M	Sn	I
R, D, E	0.355	0.376	0.354	0.376	0.719	0.03	0.14	0.616

8.2.5 Evaluation of the Input Size

This subsection introduces the result from the road state estimations using different values of N_{group} and feature combination [R, D].

Tab. 8.10 presents the result when using the MLP to predict class grouping 1. It is seen that the best performance is found when $N_{\text{group}} = 10$, and that using 5 or more points within a group gave a better result than using 3 or less.

Table 8.10: The performance of the MLP when estimating class grouping 1 for different values of N_{group}, using feature combination [R, D]

N_{group}	**P**	**R**	**F1**	**BA**	**High**	**Low**
1	0.712	0.759	0.712	0.759	0.816	0.702
2	0.718	0.766	0.719	0.766	0.822	0.709
3	0.721	0.768	0.725	0.768	0.814	0.723
5	0.729	0.775	0.736	0.775	0.808	0.743
10	0.733	0.782	0.741	0.782	0.816	0.748
20	0.724	0.774	0.731	0.774	0.811	0.737
40	0.724	0.779	0.728	0.779	0.833	0.725

Tab. 8.11 shows the result when using the transformer to estimate class grouping 1. Here, $N_{\text{group}} = 10$ had the best recall and balanced accuracy, while $N_{\text{group}} = 40$ had the best precision and F1-score. As for the MLP, it is observed that using $N_{\text{group}} \geq 5$ leads to better results. However, the precision and F1-score for $N_{\text{group}} = 5$ is a bit lower than for $N_{\text{group}} > 5$, but more similar to the result from $N_{\text{group}} = 3$.

Table 8.11: The performance of the transformer when estimating class grouping 1 for different values of N_{group}, using feature combination [R, D]

N_{group}	**P**	**R**	**F1**	**BA**	**High**	**Low**
1	0.710	0.758	0.705	0.758	0.838	0.679
2	0.713	0.755	0.719	0.755	0.776	0.734
3	0.718	0.762	0.723	0.762	0.793	0.732
5	0.724	0.772	0.729	0.772	0.815	0.728
10	0.736	0.777	0.746	0.777	0.776	0.778
20	0.730	0.775	0.739	0.775	0.788	0.762
40	0.739	0.771	0.750	0.771	0.729	0.812

The next table shows the performance when using the MLP to estimate class grouping 2. Similarly to the result for class grouping 1, the best result is found for $N_{\text{group}} = 10$. Again, it is seen that $N_{\text{group}} \geq 5$ gave better results than $N_{\text{group}} < 5$.

Table 8.12: The performance of the MLP when estimating class grouping 2 for different values of N_{group}, using feature combination [R, D]

N_{group}	**P**	**R**	**F1**	**BA**	**D/M**	**Sn/I**
1	0.711	0.748	0.715	0.748	0.723	0.772
2	0.718	0.757	0.722	0.757	0.726	0.788
3	0.721	0.760	0.725	0.760	0.73	0.79
5	0.728	0.766	0.734	0.766	0.747	0.785
10	0.733	0.772	0.741	0.772	0.757	0.788
20	0.724	0.765	0.732	0.765	0.751	0.778
40	0.728	0.770	0.736	0.770	0.753	0.788

Tab. 8.13 shows the result when using the transformer to estimate class grouping 2. The result is similar to the aforementioned cases, where $N_{\text{group}} = 10$ gave the best performance and $N_{\text{group}} \geq 5$ was more favorable than $N_{\text{group}} < 5$.

Table 8.13: The performance of the transformer when estimating class grouping 2 for different values of N_{group}, using feature combination [R, D]

N_{group}	**P**	**R**	**F1**	**BA**	**D/M**	**Sn/I**
1	0.714	0.753	0.717	0.753	0.716	0.79
2	0.718	0.756	0.723	0.756	0.732	0.779
3	0.721	0.759	0.726	0.759	0.734	0.784
5	0.727	0.766	0.733	0.766	0.742	0.789
10	0.736	0.771	0.746	0.771	0.776	0.766
20	0.728	0.763	0.737	0.763	0.775	0.75
40	0.734	0.763	0.744	0.763	0.798	0.728

The final tables show the results for $N_{\text{group}} = 1, 2, 3, 5$, but where the points have only been chosen from groups that contain 10 or more points. Tab. 8.14 shows the result when predicting class grouping 1 with the transformer. It is seen that the values of the different performance measures increases with N_{group}, and that the outcome is better compared to Tab. 8.11.

Table 8.14: The performance of the transformer when estimating class grouping 1 for different values of N_{group}, using feature combination [R, D] and only choosing points from groups with 10 or more points.

N_{group}	**P**	**R**	**F1**	**BA**	**High**	**Low**
1	0.725	0.772	0.733	0.772	0.799	0.745
2	0.728	0.772	0.737	0.772	0.787	0.758
3	0.729	0.774	0.737	0.774	0.794	0.755
5	0.734	0.778	0.743	0.778	0.794	0.762

Tab. 8.15 shows the result from predicting class grouping 2 using the transformer. As for the result when predicting class grouping 1, the performance improves as N_{group} increases. Additionally, the result is better compared to the result in Tab. 8.13

Table 8.15: The performance of the transformer when estimating class grouping 2 for different values of N_{group}, using feature combination [R, D] and only choosing points from groups with 10 or more points.

N_{group}	**P**	**R**	**F1**	**BA**	**D/M**	**Sn/I**
1	0.729	0.764	0.737	0.764	0.764	0.764
2	0.732	0.767	0.740	0.767	0.769	0.765
3	0.733	0.768	0.742	0.768	0.773	0.762
5	0.737	0.773	0.746	0.773	0.773	0.774

8.2.6 Wet and Slushy Surface Conditions in the Dataset

Due to the similar performance of the tested models, only the transformer was employed when examining cases containing wet and slushy surface conditions. Additionally, feature combination [R, D] was used together with $N_{\text{group}} = 10$. In the following tables, "D/M", "W/Sl" and "Sn/I" represent the dry and moist class, wet and slushy class and snowy and icy class, respectively.

First, class grouping 4 was tested, resulting in the performance presented in Tab. 8.16. Furthermore, the associated confusion matrix is found in Fig. 8.5. Both Tab. 8.16 and Fig. 8.5 show that the transformer has problems with estimating the wet and slushy class, which instead mostly gets confused with the remaining classes.

Table 8.16: The performance of the transformer when estimating class grouping 4, using feature combination [R, D] and $N_{\text{group}} = 10$.

Feature comb.	**P**	**R**	**F1**	**BA**	**D/M**	**W/Sl**	**Sn/I**
R, D, E	0.486	0.505	0.480	0.505	0.768	0.081	0.666

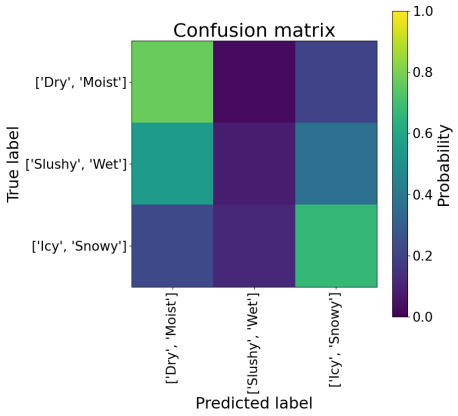


Figure 8.5: Confusion matrix when estimating class grouping 4 using the transformer model, feature combination [R, D] and $N_{\text{group}} = 10$.

Next, the transformer was tested on class grouping 5. The result is displayed in Tab. 8.17, while the confusion matrix is shown in Fig. 8.6. There is an indication that the model has a bias towards the dry surface condition. Furthermore, slush is classified with a low accuracy, and the model does not predict any input as belonging to a wet surface.

Table 8.17: The performance of the transformer when estimating class grouping 5, using feature combination [R, D] and $N_{\text{group}} = 10$.

Feature comb.	P	R	F1	BA	D	M	W	Sl	Sn	I
R, D, E	0.225	0.217	0.208	0.217	0.719	0.093	0	0.028	0.101	0.363

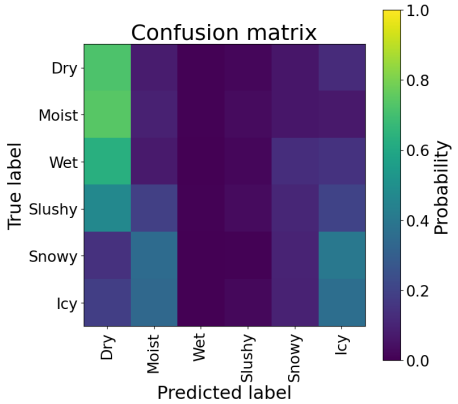


Figure 8.6: Confusion matrix when estimating class grouping 5 using the transformer model, feature combination [R, D] and $N_{\text{group}} = 10$.

Finally, class grouping 1 was once again examined, but with wet and slushy data included in the dataset. Tab. 8.18 presents the result and Fig. 8.5 shows the related confusion matrix. Even though the result is worse compared to when the wet and slush surface conditions were excluded, as seen in Tab. 8.11, the difference is only about 3-4 percentage points when examining the F1-score and the balanced accuracy.

Table 8.18: The performance of the transformer when estimating class grouping 4 with slush and wet data included in the dataset, using feature combination [R, D] and $N_{\text{group}} = 10$.

Feature comb.	**P**	**R**	**F1**	**BA**	**High**	**Low**
R, D, E	0.701	0.754	0.711	0.754	0.754	0.755

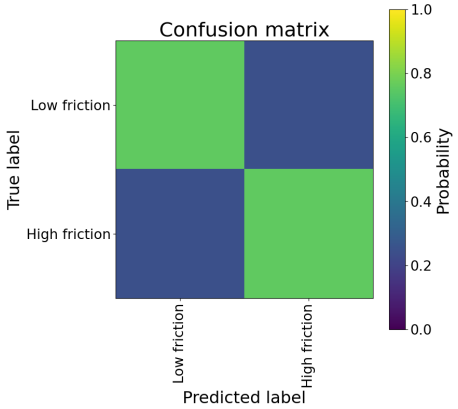


Figure 8.7: Confusion matrix when estimating class grouping 1 with wet and slush included in the dataset, using the transformer model, feature combination [R, D] and $N_{\text{group}} = 10$.

9

Discussion

This chapter discusses the methods and results from previous chapters. It is divided into multiple sections, where each section is assigned to a previous chapter. Thus, the order of appearance is road point extraction, point cloud accumulation, creation of the road state dataset, evaluation of LiDAR features on various surface conditions, and the estimation of the road state. Finally, ethical aspects related to the thesis are discussed.

9.1 Road Point Extraction

Section 4.2 presented the performance of the road point extraction algorithm. Even though most of the road points were correctly classified, road points that were located further away were more likely to not be found by the algorithm that searched for line formations. A reason for this is that the line formations lose their shapes when the distance increases, especially when the roads are covered or partially covered in snow, see Fig. 4.4b. This in turn causes the defined thresholds that are used when searching for line formations to stop the search prematurely, thus resulting in that points belonging to the road at greater distances are misclassified as non-road points. A possible solution to this could be to make the thresholds less strict when the distance increases. Though, this would increase the risk that points not belonging to the road, for example the point on another vehicle, will be considered as road points. Furthermore, the proposed method for extracting road points often misclassifies non-road points that are placed next to the road as road points. This behaviour can be seen in figure 4.4c.

Figs. 4.5c and 4.5d show another problem with the road point extraction, namely that some points near the ground plane are wrongfully classified as road points. These points could lead to a faulty estimation of the road state, which might affect the control signals of the vehicle. When considering the creation of the road state dataset, described in section 6.1, it is likely that some of the false road points were matched with RCM measurements and stored in the dataset. This means that the method for estimating the road surface state might have been based on data that to some extent is incorrect.

The problem of wrongfully classifying non-road points as road points, both when the points are located above and on the sides of the road, could be improved by setting stricter thresholds on the height of the points. For the case where line formations extend out from the sides of the road, a harsher threshold for the maximum difference in height between points, Th_{height}, would make the behaviour seen in Fig. 4.4c less prominent. As for when points above the ground plane are classified as road points, seen in Fig. 4.5c, a stricter threshold on the points' allowed distance to the ground plane, Th_{dist}, could be solve the problem. However, stricter conditions could result in the behaviour exhibited in Fig. 4.4a, where the search for line formation was stopped too early and the GPF algorithm removed road points, becoming both more prevalent and significant. It is therefore important to consider the tradeoff between wrongfully classified road points and the loss of road points.

A potential improvement to the proposed road point extraction algorithm is to fuse it with a camera based method that detects the boundaries of the road [35]. By using the detected boundaries, the 3D points of the point cloud could be projected onto the image, whereafter the points located within the boundaries could be extracted. Conversely, the extracted road points could be used to complement the camera based method in cases where the LiDAR is suited better, such as in dark environments.

The main takeaways from the extraction of road points are summarized as follows:

- Most road points are correctly classified, although many non-road points next to the road are also classified as road points.
- The proposed method has difficulties in identifying road points at longer distances.
- Non-road points that are placed close to the ground are often misclassified as road points.
- When tuning the algorithm, the tradeoff between missing road points or falsely classifying non-road points needs to be considered.
- Potential improvements could be made by fusing camera information to detect boundaries of the road.

9.2 Point Cloud Accumulation

In section 5.2.2, it was observed that the proposed methods for accumulating points over multiple time instances performed rather poorly. One of the reasons for this could be that the measurements of the velocity and orientation are not accurate enough. This could cause the points to be translated and rotated wrongly. Another reason could be that the Kalman filter uses a constant velocity model, which assumes that the vehicle is driving with constant speed in a straight line, when estimating the position of the vehicle. As a result, the positional estimations might be inaccurate when the vehicle is accelerating or turning.

When introducing ICP, the performance of the alignments varied. Some cases were improved by ICP, as seen in Figs. 5.3c and 5.3d, while others were worsened, see Figs. 5.3a and 5.3b. Other times, there was no difference in performance between the both methods. When examining the accumulated road points, it was noticed that the method using ICP sometimes performed a little better than the method that only used homogeneous transformations, see Figs. 5.4a and 5.4b. However, Figs. 5.5c and 5.5d, showed that the ICP method could result in completely misaligned road sections. A reason for this is that some of the environments have few objects with distinguishable features, thus making it harder to find good point correspondences. This can be seen in Figs. 5.2b and 5.2d, where much of the environment consists of bushes. Another difficult obstacle for the ICP algorithm is the presence of moving objects. A possible solution to this is to use the method developed by Chebrolu et al. [36], that used adaptive robust kernels together with ICP.

The key findings derived from the point cloud accumulation can be summarized as follows:

- Both methods for accumulating point clouds performed rather poorly.
- Noisy measurements, the use of a too simple process model, or a combination of both, may have contributed to faulty homogeneous transformations.
- The method using ICP was most likely disturbed by moving vehicles or lack of distinct features in the point clouds.

9.3 Road State Dataset

When creating the road state dataset, the algorithm for extracting road points, explained in section 4.1, and the method for accumulating point clouds, described in section 5.1.1, were used. Both these methods have flaws that may have resulted in faulty labels for the gathered LiDAR points. As discussed in section 9.1, the fact that the road point extraction algorithm does not always manage to remove all points located on other vehicles, means that there is a risk that the dataset contains some points that belong to vehicles instead of the road. At the same time, the method for point cloud accumulation has problems with aligning the ground plane in the z-direction, as seen in Fig. 5.4c. Additionally, even though the accumulation thresholds introduced in section 5.2.2 were added, it is still possible that some point clouds could be misaligned in the longitudinal-latitudinal plane as well. Such a misalignment would result in the wrong points being collected by the bounding box seen in Fig. 6.2.

The method of using a bounding box to extract the points in the area around the RCM and assigning all these points its measurement, might also be problematic. If the road inside the bounding box contains more than one surface condition, some of the extracted points might be given the wrong label. Conversely, this will not be a problem when the surface condition is the same over the whole road surface. An idea for future work is to gather data from roads where only one surface condition

is present when annotating points. That way, the bounding box could be excluded and all of the points on the road could be collected, resulting in substantially more labeled data. However, it might prove difficult to find scenarios where only one surface condition is present for all surface conditions.

Fig. 6.4 shows that the created dataset is imbalanced. It is seen that most data was collected for dry and icy roads, while least data was gathered on wet roads. The reason for this is that most of the data was collected in northern Sweden during January, February and March, when snow is more common than rain. Additionally, most of the data for wet surface conditions was also collected during the same occasion. This means that the dataset does not contain wet data points from different scenarios, and could as a consequence be biased. In future research, it could therefore be desirable to gather more data on wet surface conditions from different occasions. A similar argument could be made for the whole dataset, since it mostly contains data from winter time in northern Sweden. To obtain a more diverse dataset, data could also be gathered during the other seasons, as well as in other locations in the world.

Fig. 6.6 shows that all of the surface conditions have many points with low reflectance values, which could make it harder to distinguish between different surface conditions. It is however seen that the dry surface condition has higher reflectance values. Some of the surface conditions behave differently from each other at longer distances, as seen in 6.7. One finding is that mostly only the dry and icy surface conditions show reflectance values at 25 to 30 meters, where the reflectance from dry points often is higher. Furthermore, while moist and snowy surface conditions appear to behave fairly similarly, points on icy surfaces often have reflectance values that are a bit higher. These observations show that there are some differences between the surface conditions, especially when comparing dry and icy points with all other surface conditions. Moreover, they also show that the behaviour of the reflectance value is similar for many classes, which could potentially make it difficult to differentiate between them.

In Figs. 6.8 and 6.9, dry, moist and wet surface conditions all have many points with a higher measured friction coefficient, while points on icy and snowy surfaces have lower friction coefficients. Additionally, points on slushy surface conditions have both high and low friction coefficients. These friction coefficient measurements do however conflict with what is stated in results from previous research. The friction coefficient for dry, wet, snowy and icy roads when the driving speed is 40 km/h, according to [37], is listed below.

- Dry asphalt: 0.7-0.8
- Wet asphalt: 0.4-0.5
- Snowy road: 0.1-0.2
- Icy road: 0.1-0.2

The friction coefficients measured by the RCM is therefore higher than expected for many of the surface conditions. This might mean that measurements from the RCM are not suitable to be used as ground truth when the goal is to estimate an exact friction coefficient. However, they could still be used when classifying if the road has high or low friction coefficient values, since the high and low friction coefficient values seem to be sufficiently separated. For future research projects that intend to use the same RCM, it is recommended to investigate its accuracy.

Fig. 6.10 shows that most points have a return index of 1, while slushy, snowy and icy surface conditions have more points with higher return indices. However, the amount of points with a return index greater than one is relatively low compared to the points with a return index of 1. The fact that fewer points have higher return indices is seen more clearly in Fig. 6.11. A conclusion from this observation is that the return index might not be that helpful when trying to estimate the surface condition.

The main comments about the creation and analysis of the road state dataset are summarized below:

- The road state dataset likely contains faulty data to some extent due to inaccuracies in the employed methods.
- The current dataset is imbalanced and more data needs to be gathered from different seasons and locations.
- As a consequence of the two aforementioned flaws, the dataset might not reflect the real-world distributions of the road states.
- The reflectance value of the surface condition is similar for the different classes. Most difference is seen in the dry and icy surface conditions.
- The RCM sensor appears to provide friction coefficient measurements that are higher than they should be in reality.
- The return index does not seem to be a valuable feature for road state estimation.

9.4 Effects of Various Road Surfaces on LiDAR Features

This section discusses the reflectance value and the return index on various road surfaces. First, the effect of the reflectance value is analysed, after which the importance of the return index is discussed.

9.4.1 Reflectance

The figures of the reflectance values introduced in section 7.2.1 show that dry roads tend to have a higher reflectance value than roads that are covered in water, slush, snow or ice, as seen in Fig. 7.1. It can also be noticed that tire tracks usually have higher reflectance values when the surrounding road is covered in ice or snow, which is observed in Figs. 7.5b and 7.5d. A reason for this could be that bare or partially bare asphalt can often be seen in the tire tracks, thus, more light is reflected back in these areas. However, some cases were also found where the reflectance value of the tire tracks were lower than the surrounding snow and ice covered area, as in Figs. 7.2b and 7.2d.

When examining different scenarios where snow or ice was present on the road, it could be observed that the colour of the snow and ice points varies from blue to green, and in some scenes even red, as in Fig. 7.2d. A potential reason for the varying behaviour could be that the absorption coefficient is higher when the ice is thicker [8], as seen in Fig. 3.2. This behaviour could make it harder to distinguish between the different types of road conditions, although it might be useful for estimation of ice thickness. As for wet and slushy roads, the reflectance tends have consistently low values, as in Fig. 7.4. Common for all road conditions is that the magnitude of the reflectance value decreases as the distance to a point increases.

Another observation was that the sun might influence the reflectance value. When examining Figs. 7.5a and 7.5c, is seen that the reflectance values are lower in the direction of the sun compared to the road sections on its sides. This behaviour might be caused by how the LiDAR compensates for light that comes from external light sources. Although the method that is used by the LiDAR in this thesis is unknown, one way of approaching this problem is to first only measure the intensity of the background illumination, whereafter this background intensity is subtracted from the intensity of the reflected light that comes from the LiDAR [7]. If the background intensity were to be measured for different parts of the road, the sides of the road in Fig. 7.5a would have lower background intensity values than the middle section of the road, due to the placement of the sun. Since the intensity of the light emitted from the LiDAR is the same regardless of which direction the light pulses are directed in, the difference between the background intensity and the intensity measured by the LiDAR will be smaller in the middle part of the road than on its sides. This will result in the behaviour seen in Fig. 7.5c, where the reflectance values are lower in the direction of the sun. It is however noticed that this behaviour is not found in Figs. 7.5b and 7.5d, even though the situation is

similar to the one in Fig. 7.5a, where the sun is located at a lower position on the sky. These observations suggest that the placement of the sun could be a factor that affects the road state estimations.

Another potential disturbance that could affect the reflectance value is the presence of brake marks. This is seen in Figs. 7.3b and 7.3d, where the brake marks have lower reflectance value than the surrounding dry road. By comparing the brake mark that is to the right of the leftmost lane marking with the brake mark to the left of the same lane marking, it is seen that the right brake mark is both darker and has a lower reflectance value. That the darker parts of the road gives lower reflectance values is however reasonable, since it is commonly known that darker colors absorb more light than lighter colors. As a result, new asphalt and brake marks could potentially be disturbance elements when estimating the surface state.

The primary takeaways from the analysis of the reflectance value can be summarized as follows:

- Dry surface conditions generally have higher reflectance than other surface conditions, while wet surface conditions have the lowest reflectance. Icy and snowy surfaces alternate between low and medium high reflectance.
- Tire tracks generally have a higher reflectance value than the surrounding road. However, some exceptions of this have been observed.
- The reflectance value might be affected by the position of the sun.
- Darker road segments, such as patched up roads or roads containing brake marks, have lower reflectance. This might be a disturbance factor when estimating the road state.

9.4.2 Return Index

Figs. 7.6, 7.7, 7.8 and 7.9 show how the return indices for dry, icy/snowy and wet surface conditions respectively. For all surface conditions, it is seen that almost all of the points have return index 1. This behaviour could be expected when looking at Fig. 6.11, since it shows that the road state dataset contains a relatively small amount of points with return indices above 1. The fact that only points with return index 1 seem to belong to the road means that this feature will not be helpful when making estimations about the road surface condition.

9.5 Road State Estimation

In section 9.3, it was discussed that the RCM might not be suitable when estimating the exact friction coefficient. This is because each of its measurements is an average over 1 second. During this time, the friction coefficient could fluctuate significantly if the surface condition is not uniform. This makes the sensor less suitable for gathering ground truth data. However, since this is a relatively unexplored subject, it was deemed sufficient for exploring the potential of using a LiDAR to estimate the friction coefficient. Due to the inaccuracies of the sensor and the novelty of the use case, the estimation of the friction coefficient was treated as a binary classification problem. The problem of estimating the friction coefficient exactly with a regression model is left as a subject for future research. Before attempting the task, it is recommended to gather data with a sensor that is better suited for the purpose.

When examining the performance of the MLP and the transformer, it is seen that both models give similar results. This can be observed when comparing Tab. 8.5 to Tab. 8.6, where the models predicted whether the points belonged to the high or the low friction coefficient class, and Tab. 8.7 to Tab. 8.8, where the models estimated whether points were part of a dry/moist or snowy/icy road. Both class groupings achieved an F1-score that ranged between 72.5% to 73.6%, and a balanced accuracy between 75.7% to 77.5%. However, when trying to predict the dry, moist, snowy and icy surface conditions separately, both models performed rather poorly. As seen in the confusion matrix in Fig. 8.4, the moist surface condition was mostly confused with the dry surface condition, while the snowy surface condition was often confused with the icy surface condition. Even though this result shows that there are some difficulties with distinguishing between certain surface conditions, it also shows promise when discriminating between high and low friction coefficients, as well as the surface conditions associated with respective friction coefficient class.

The results could potentially be improved by fusing the LiDAR data with information from other sensors. As an example, since the location of the sun seemed to have an impact of the reflectance values, as discussed in section 9.4.1, a camera might be used to find its position. The position of the sun could then be included as an input feature to the machine learning models. Furthermore, as mentioned in section 8.1.3, the transformer used embeddings of the different features of the LiDAR that had been created using one fully connected layer. It could however be of interest to examine if there are any other type of embeddings that would result in a higher performance. An example of such an embedding could be to use separate fully connected layers for each feature and then concatenate the output from each layer.

In addition to the similar results of the models, it was also found that the feature combinations did not affect the performance as much as expected. As an example, since the intensity of light decreases from the time it leaves the light source, it was believed that adding the travel distance of the light pulse as an input together with the reflectance value would give a significantly better result than when only using the reflectance value. However, the improvement seemed to only be marginal. This

might mean that the reflectance is relatively unaffected or minimally reduced at the distances found in the road state dataset, which are mostly less than 20 meters according to Fig. 6.5. As for the remaining feature combinations, none of them appear to consistently yield better results than the use of reflectance and distance as the input features to the models.

As for the capabilities of the transformer when wet and slushy data were included in the dataset, it was seen that it was not possible to accurately estimate the surface conditions. The confusion matrices in Figs. 8.5 and 8.6 both show that the transformer misclassifies the classes related to wet and slushy data. Fig. 8.5 also shows that when trying to predict all six classes, almost none of the classes are classified correctly, and that the model was biased towards dry surface conditions. One reason for this might be that the dataset does not contain that much data for many of the classes. Adding more data to all classes might improve the model's ability to generalize across the surface conditions. It was however noticed that the transformer still managed to achieve an F1-score of 70.1% and a balanced accuracy of 75.4% when classifying the friction coefficient. This result is only a little worse than the case where the wet and slushy surface conditions were excluded from the dataset. One reason for the lower performance could come from that the friction coefficient of slushy surface conditions is more spread out in the interval between 0.4 to 0.6, seen in Fig. 6.9, when 0.5 is the friction coefficient value that separates high from low friction coefficients. Since the reflectance value appears to mostly assume similar values in this interval, a reasonable assumption is that it will be harder to predict the friction coefficient for this surface condition.

When examining how different values of N_{group} affected the performance, that is, how many LiDAR points were used as input to the network at a time, it was observed that $N_{\text{group}} = 10$ gave the best result for both class grouping 1 and 2. This can be seen in Tabs. 8.10, to 8.13. It was also noticed that using up to $N_{\text{group}} = 40$ points might have some potential for an increase in performance, as it often gave a result that was similar to $N_{\text{group}} = 10$, though the number of groups was drastically lower for $N_{\text{group}} = 40$, as presented in Tab. 8.1. Conversely, Tabs. 8.10 to 8.13 show that using $N_{\text{group}} < 5$ resulted in a worse performance for both the MLP and the transformer, where the points within each group was chosen from groups of any size. However, when repeating the test for $N_{\text{group}} \leq 5$ when the points were only chosen from groups that contained at least 10 points, the performance was increased to match the performance for $N_{\text{group}} = 10$, as seen in Tabs. 8.14 and 8.15. This suggests that it might be possible to get accurate predictions with few points as input. As a result, the friction coefficient and surface condition could potentially be estimated for smaller segments on the road, thus allowing a higher resolution mapping of the road state to be made. Furthermore, the performance is improved when gradually increasing N_{group} from 1 to 10, and it is recommended for future work to gather more data and investigate how using larger groups of points as input would affect the performance of the models. Another future research topic could be to investigate why points extracted from groups with less than 10 points reduced the performance of the models. Until a reason for this behavior is identified, it is recommended to exclude points not belonging to a group of at least 10 points.

In section 8.2.1, it was mentioned that the models were sensitive to how random operations were performed. One method that uses random operations is the undersampling process. As a consequence, the amount of noisy data in the training set might vary depending on the points that are chosen. Thus, if a model starts training on a training set containing much noisy data, it could learn wrong connections between the input features and the road state. For future research, it is therefore recommended to improve the quality of the road state dataset by either solving the problems related to the road point extraction and the automatic annotation, or using an alternative method of gathering the data. An additional note related to the sensitivity of the models is that it was much easier to find acceptable results using the transformer than the MLP. An observation is therefore that the transformer might be better than the MLP at handling cases where the data is noisy.

As mentioned in chapter 2, the LiDAR in this project only emits light with a wavelength of 1550 nm. Previous research, however, has noted that using lasers with different wavelengths for other sensors can improve the result when estimating different surface conditions [5]-[7]. If it is possible to include light with other wavelengths in the LiDAR, the result from estimating the surface state could potentially be improved.

While the two deep learning models showed some promising results, it should be noted that they are among the simpler alternatives available. The multilayer perceptron is one of the most straightforward architectures, while the transformer employed is a simple adaptation of the original transformer by Aswani et al.. It is possible that more elaborate architectures might yield better results. As a potential improvement, machine learning models intended for tabular data could be utilized, since the dataset in this work can be regarded as a table where each row is a data point and each column is a feature. Examples of such models include the Self-Attention and Intersample Attention Transformer (SAINT) [38] and the Non-Parametric Transformer (NPT) [39]. Both of these models consider multiple points at once when predicting, and can perform attention between either the data points or the features.

Considering the adequate results obtained for $N_{\text{group}} = 1$, alternative machine learning algorithms such as gradient boosted decision trees (GBDTs) could be a viable option to explore because of their competitive performance [40]. These kind of models only take one data point at a time as input however, and therefore lack the ability to take multiple points into account when classifying a single target. As previously discussed, better data in regard to both quality and quantity is needed in order to properly assess and compare the performance of the models.

In addition to the models designed for tabular data, it could be worth to explore networks that are tailored for point clouds. Networks like PointNet++ [41] have the capability of capturing structural patterns in point clouds, and could be adapted to investigate their applicability for surface condition estimation. For example, this can be relevant since dry roads are typically rougher than roads covered in ice [8].

Since the usage of a single LiDAR point as input was sufficient to achieve adequate results, and utilizing reflectance as the only input feature produced comparable results to the remaining feature combinations, it appears that minimal information was needed to perform the binary classifications in this study. Therefore, simpler machine learning models, or even non-machine learning methods could be explored, since the complexity of neural networks might not be necessary. An incentive for doing so is the potential reduction in computational load of the estimation algorithm, which can be advantageous in automotive applications, where the processing power may be more limited.

The main conclusions drawn from the road state estimation can be summarized as follows:

- The RCM can be used to gather data when the friction coefficient is divided into high and low friction coefficient classes, but it is not suited for regression problems.
- The results obtained by the MLP and the transformer were similar, although the transformer was less sensitive to noisy data.
- Both models showed promise when estimating the two friction coefficient classes, as well as when identifying surface conditions associated with each respective friction coefficient class.
- Both models had difficulties to predict each surface condition class separately.
- Both the MLP and the transformer achieved adequate results when only using one LiDAR point as input. However, a higher number of points improved the performance as long as there was enough data present.
- Adding lights with multiple wavelengths to the LiDAR could increase the performance.
- Using models made for tabular data or for processing LiDAR point clouds might improve the result.
- It is worth considering the possibility of estimating the road state with simpler methods than those presented in this work.

9.6 Ethical Aspects

This thesis examines the use of LiDAR sensors to estimate the surface condition and friction coefficient. As this technology might be of use for self driving vehicles, it is important to consider and analyze the implications that emerge in the context of autonomous driving. To this end, the 17 Sustainable Development Goals (SDGs) [42], provided by the United Nations (UN) could be used as ethical guidelines.

A major topic when discussing autonomous driving is the safety aspect, that can be related to the 3rd SDG, *Good Health and Well-being*, which is to ensure healthy lives and promote well-being for all at all ages. One argument that contributes to this goal is that self-driving vehicles have the potential to enhance the road safety, by reducing accidents caused by human error. However, there is also a risk that autonomous vehicles could be introduced prematurely, resulting in that they instead cause accidents that easily could have been avoided by human drivers. It is therefore critical that the technology behind these vehicles is extensively researched and tested before any commercial release.

Despite the potential reduction in traffic accidents through the use of autonomous vehicles, it is highly probable that accidents will continue to occur. This raises multiple questions regarding liability. As the human driver will be removed from the driving process, much more of the liability will be shifted towards the vehicle manufacturer. This could be related to the 9th SDG, *Industry, Innovation, and Infrastructure*, in regards to that the automotive industry has an obligation to ensure safe and reliable infrastructure and innovations. Similarly, SDG 11, *Sustainable Cities and Communities*, could be used to argue that governments should have laws and regulations in place that addresses the challenges related to this technology and prioritizes the safety and well-being of residents in cities and communities. An additional question is related to how autonomous vehicles should behave during an accident, as this will be preprogrammed by the manufacturer. Therefore, it is important that vehicle manufacturers and governments collaborate to set reasonable rules and guidelines regarding the accident scenarios, which is related to the 17th goal of the SDGs, *Partnerships for the Goals*.

An autonomous vehicle will rely on sensors that record data about itself and its surroundings. As privacy and data protection are fundamental rights, it is important for automotive manufacturers to establish robust data protection mechanisms that protect individuals' privacy rights. This relates both to SDG 16, *Peace, Justice, and Strong Institutions*, as well as the previously mentioned SDG 9.

10

Conclusion

In this study, the objective was to investigate the possibility to estimate the road state using machine learning and data from a LiDAR. The results showed that discernment between dry/moist and snowy/icy roads was achievable with a balanced accuracy of 77.1% and an F1-score of 74.6%. The corresponding results for the classification of the road friction coefficient into either high or low values reached 77.7% and 74.6%, respectively. As for the models employed, the multilayer perceptron and the transformer network were observed to perform similarly to each other. The best results for the MLP and the transformer differed by at most 0.5 percentage points in both F1-score and balanced accuracy. Although the best results were obtained when using ten LiDAR points as input to the network at once, it was found that adequate estimations could be made with even a single point. At most, the difference between using one and ten points was 3.4 percentage points for the balanced accuracy and 4.1 percentage points for the F1-score.

The analysis of the reflectance plots indicated that the reflectance is a crucial component in the road state estimation, though it was difficult to distinguish between different surface conditions in some cases. This is reflected in the findings from the estimation results, where it was also discovered that the usage of reflectance by itself produced results comparable to those of the remaining feature combinations. Furthermore, the distance was shown to provide small enhancements to the predictions, but overall, the different feature combinations only gave marginal differences.

The data in this study was gathered under suboptimal conditions, where the road point extraction, point accumulation method and the road condition monitor are likely to have contributed to the inclusion of erroneously labeled points. As such, the dataset might not be representative of the road state distribution in the real world.

To conclude, the investigation showed promising results. However, there were limitations in the data, both in terms of both quantity and diversity. These limitations may have had an adverse effect on the performance of the classification methods. Further research will be needed in order to fully assess the potential of the LiDAR for the purpose of road state estimation.

Future work

This thesis demonstrates the potential of utilizing LiDAR for estimating the friction coefficient and surface condition. In future work, it is recommended to enhance the performance by fusing LiDAR data with information from other sensors. This option can be explored to determine if the integration of multiple sensor inputs improves the accuracy and reliability of the estimation. Moreover, there is an opportunity to combine LiDAR data with existing methods, such as camera-based approaches [4], to leverage the strengths of both technologies and potentially achieve better results in the road state estimations.

In future studies aiming to estimate the friction coefficient and surface condition using LiDAR data, it is advisable to explore alternative machine learning networks. Specifically, it is recommended to investigate the potential of utilizing machine learning models designed for tabular data, such as SAINT [38], as well as models specifically intended for processing LiDAR point clouds, such as PointNet++ [41]. It is also recommended to examine simpler methods, both within and outside the field of machine learning. By considering these alternative methods, future research could potentially enhance the accuracy of the road state estimates.

Finally, it is also recommended to investigate the potential of utilizing LiDAR sensors in regression tasks. Instead of categorizing surfaces into high or low friction coefficient classes, future research could focus on developing regression machine learning models that directly estimate the exact friction coefficient. Such an approach would provide more detailed information about the road surfaces.

Bibliography

[1] T. Vanessa, "Lidar (Remote Sensing Technology)," in *Salem Press Encyclopedia of Science*, 2023. [Online]. Available: `https://search.ebscohost.com/login.aspx?direct=true&db=ers&AN=87323881&site=eds-live&scope=site`, Accessed on: 2023-05-16.

[2] Y. Li, J. Ibanez-Guzman, "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50-61, Jul. 2020, doi: 10.1109/MSP.2020.2973615.

[3] L. Gao et al. "Multi-sensor Fusion Road Friction Coefficient Estimation During Steering with Lyapunov Method," *Sensors*, vol. 19, no. 18. Jul. 2019, doi: 10.3390/s19183816.

[4] L. Cheng, X. Zhang, J. Shen, "Road surface condition classification using deep learning," *Journal of Visual Communication and Image Representation*, vol. 64, Oct. 2019, doi: 10.1016/j.jvcir.2019.102638.

[5] M. Andersson et al. "Road Friction Estimation," Intelligent Vehicle Safety Systems, Trollhättan, Sweden, 2004:17750, 2007. [Online]. Available: `https://scholar.google.com/citations?view_op=view_citation&hl=en&user=uQskUJ8AAAAJ&citation_for_view=uQskUJ8AAAAJ:d1gkVwhDpl0C`, Accessed on: 2023-05-15.

[6] M. Andersson et al. "Road Friction Estimation, part II," Intelligent Vehicle Safety Systems, Sweden, 2004:17750, 2010. [Online]. Available: `https://fudinfo.trafikverket.se/fudinfoexternwebb/Publikationer/Publikationer_001101_001200/Publikation_001109/IVSS_RFEII_Slutrapport.pdf`, Accessed on: 2023-05-15.

[7] J. Casselgren, S. Rosendahl, M. Sjödahl, P. Jonsson, "Road condition analysis using NIR illumination and compensating for surrounding light," *Optics and Lasers in Engineering*, vol 77, pp. 175-182, Feb. 2016, doi: 10.1016/j.optlaseng.2015.08.002.

[8] P. Jonsson, J. Casselgren, B. Thörnberg, "Road Surface Status Classification Using Spectral Analysis of NIR Camera Images," *IEEE Sensors Journal*, vol. 15, no. 3, pp. 1641-1656, Oct. 2015, doi: 10.1109/JSEN.2014.2364854.

[9] *Road Condition Monitor RCM511*, Helsinki, Finland, Teconer. [Online]. Available: https://www.teconer.fi/wp-content/uploads/Road-Condition-Monitor-RCM511-data-sheet_en-1.pdf, Accessed on: 2023-05-22.

[10] D. Zermas, I. Izzat, N. Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 2017, pp. 5067-5073. [Online]. Available: https://ieeexplore.ieee.org/document/7989591, Accessed on: 2023-05-17.

[11] B. Siciliano, L. Sciavicco, L. Villani G. Oriolo, "Kinematics," in *Robotics Modelling, Planning and Control*, B. Siciliano, L. Sciavicco, L. Villani G. Oriolo, Ed. London, England: Springer-Verlag London Limited 2010, 2010, ch. 2, pp. 41-42, 51-52, 56. [Online]. Available: https://doi.org/10.1007/978-1-84628-642-1_2, Accessed on: 2023-03-16.

[12] S. Särkkä, "Bayesian filtering equations and exact solutions," in *Bayesian Filtering and Smoothing*, Ed. New York, USA: Cambridge University Press, 2013, ch. 4, pp. 56-57. [Online]. Available: https://doi.org/10.1017/CBO9781139344203.005, Accessed on: 2023-05-22.

[13] Z. Zhang, "Iterative Closest Point (ICP)," in *Computer Vision*, K. Ikeuchi, Ed. Boston, USA: Springer, 2014, pp. 433. [Online]. Available: https://doi.org/10.1007/978-0-387-31439-6_179, Accessed on: 2023-03-15.

[14] I. Goodfellow, Y. Bengio, A Courville, "Machine Learning Basics," in *Deep Learning*, Ed. Cambridge, Massachusetts, USA: MIT Press, 2016, ch. 5, pp. 96-106, 149-150. [Online]. Available: https://www.deeplearningbook.org/, Accessed on: 2023-05-26.

[15] T. Hastie, R. Tibshirani, J. Friedman, "Model Assessment and Selection," in *The Elements of Statistical Learning*, 2nd ed., New York, USA: Springer, 2008, ch. 7. pp. 219–259. [Online]. Available: https://doi-org.proxy.lib.chalmers.se/10.1007/978-0-387-84858-7_7, Accessed on: 2023-05-16.

[16] I. Goodfellow, Y. Bengio, A Courville, "Regularization for Deep Learning," in *Deep Learning*, Ed. Cambridge, Massachusetts, USA: MIT Press, 2016, ch. 7, pp. 243. [Online]. Available: https://www.deeplearningbook.org/, Accessed on: 2023-05-26.

[17] I. Goodfellow, Y. Bengio, A Courville, "Optimization for Training Deep Models," in *Deep Learning*, Ed. Cambridge, Massachusetts, USA: MIT Press, 2016, ch. 8, pp. 274-279, 292-296, 302-306. [Online]. Available: `https://www.deeplearningbook.org/`, Accessed on: 2023-05-26.

[18] I. Goodfellow, Y. Bengio, A Courville, "Deep Feedforward Networks," in *Deep Learning*, Ed. Cambridge, Massachusetts, USA: MIT Press, 2016, ch. 6, pp. 168-171, 177-193, 200-217. [Online]. Available: `https://www.deeplearningbook.org/`, Accessed on: 2023-05-26.

[19] S. R. Dubey, S. K. Singh, B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92-108, Sep. 2022, doi: 10.1016/j.neucom.2022.06.111.

[20] C. M. Bishop, "Linear Models for Classification," in *Pattern Recognition and Machine Learning*, Ed. New York, USA: Springer, 2006, ch. 4, pp. 209-210.

[21] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533-536, Oct. 1986, doi: 10.1038/323533a0.

[22] N. Japkowicz, M. Shah, "Performance Measures I," in *Evaluating Learning Algorithms: A Classification Perspective*, Cambridge University Press, 2011, ch. 3, pp. 74-110. [Online]. Available: `https://doi.org/10.1017/CBO9780511921803`, Accessed on: 2023-05-16.

[23] M. Sokolova, G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427-437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.

[24] K. H. Brodersen, C. S. Ong, K. E. Stephan, J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution," in *2010 20th International Conference on Pattern Recognition*, Istanbul, Turkey, 2010, pp. 3121-3124. [Online]. Available: `https://ieeexplore.ieee.org/document/5597285`, Accessed on: 2023-05-16.

[25] J. Noh et al., "Inverse design meets nanophotonics: From computational optimization to artificial neural network," in *Intelligent Nanotechnology*, Elsevier, 2023, ch. 1, pp. 15. [Online]. Available: `https://doi.org/10.1016/B978-0-323-85796-3.00001-9`, Accessed on: 2023-05-06.

[26] S. Abinaya, M. K. K. Devi, "Enhancing crop productivity through autoencoder-based disease detection and context-aware remedy recommendation system," in *Application of Machine Learning in Agriculture*, M. A. Khan, R. Khan, M. A. Ansari, Academic Press, 2022, ch. 12, pp. 241. [Online]. Available: `https://doi.org/10.1016/B978-0-323-90550-3.00014-X`, Accessed on: 2023-05-06.

[27] M. D. Mohanty, M. N. Mohanty, "Verbal sentiment analysis and detection using recurrent neural network," in *Advanced Data Mining Tools and Methods for Social Computing*, Academic Press, 2022, ch. 5, pp. 93-94. [Online]. Available: https://doi.org/10.1016/B978-0-32-385708-6.00012-6, Accessed on: 2023-05-08.

[28] P. Gupta, N. K. Sinha, "Neural Networks for Identification of Nonlinear Systems: An Overview," in *Soft Computing and Intelligent Systems*, N. K. Sinha, M. M. Gupta, Academic Press, 2000, ch. 14, pp. 339-340. [Online]. Available: https://doi.org/10.1016/B978-012646490-0/50017-2, Accessed on: 2023-05-06.

[29] A. Vaswani et al., "Attention Is All You Need," in *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 5998-6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf, Accessed on: 2023-05-08.

[30] D. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota, USA, 2019, pp. 4171-4186. [Online]. Available: https://aclanthology.org/N19-1423/, Accessed on: 2023-05-14.

[31] Y. Tai, M. Dehghani, D. Bahri, D. Metzler, "Efficient Transformers: A Survey," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–28, Dec. 2022, doi: 10.1145/3530811.

[32] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016, pp. 770-778. [Online]. Available: https://ieeexplore.ieee.org/document/7780459, Accessed on: 2023-05-10.

[33] J. L. Ba, J. R. Kiros, G. E. Hinton, "Layer Normalization," Jul. 2016, doi: 10.48550/arXiv.1607.06450, unpublished.

[34] L. Huang et al., "Normalization Techniques in Training DNNs: Methodology, Analysis and Application," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Early Access, 2023, pp. 1-20. [Online]. Available: https://ieeexplore.ieee.org/document/10056354, Accessed on: 2023-05-26.

[35] Y.-W. Seo; R. R. Rajkumar, "Detection and tracking of boundary of unmarked roads," in *17th International Conference on Information Fusion (FUSION)*, Salamanca, Spain, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6916256, Accessed on: 2023-05-27.

[36] N. Chebrolu, T. Läbe, O. Vysotska, J. Behley, C. Stachniss, "Adaptive Robust Kernels for Non-Linear Least Squares Problems," *IEEE Robotics and Automation Letters,* vol. 6, no. 2, pp. 2240-2247, Feb. 2021, doi: 10.1109/LRA.2021.3061331.

[37] A. Novikov, I. Novikov, A. Shevtsova, "Study of the impact of type and condition of the road surface on parameters of signalized intersection," *Transportation Research Procedia*, vol. 36, pp. 548-555, 2018, doi: 10.1016/j.trpro.2018.12.154

[38] G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, T. Goldstein, "SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training," Jun. 2021, doi: 10.48550/arXiv.2106.01342, unpublished.

[39] J. Kossen et al., "Self-Attention Between Datapoints: Going Beyond Individual Input-Output Pairs in Deep Learning," Jun. 2021, doi: 10.48550/arXiv.2106.02584, unpublished.

[40] V. Borisov et al., "Deep Neural Networks and Tabular Data: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, Dec. 2022, doi: 10.1109/TNNLS.2022.3229161.

[41] C. R. Qi, L. Yi, H. Su, L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, California, USA, 2017, pp. 5105–5114. [Online]. Available: `https://dl.acm.org/doi/10.5555/3295222.3295263`, Accessed on: 2023-05-31.

[42] United Nations, "THE 17 GOALS". [Online]. Available: `https://sdgs.un.org/goals` (accessed on: 2023-05-25).

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY