Robust Perception for Formula Student Driverless Racing

Gustaf Broström

David Carpenfelt

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6152
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2021

Abstract

Building an autonomous system for race cars requires robust and highly accurate perception running in real time. This thesis proposes a novel ground removal strategy for 3D LiDAR perception, modelling the ground as several planes, and a novel clustering method for LiDARs that sweep the scene in a predefined pattern resulting in a 20-fold performance increase over clustering methods commonly used for this problem.

To weed out spurious information produced by relying only on LiDAR perception, two sensor fusion methods using a camera are introduced and evaluated, one using the immensely popular YOLO network. All algorithms were evaluated in real world scenarios using a fully functioning Formula Student driverless vehicle built by the Lund Formula Student 2021 team.

Keywords

3D LiDAR, autonomous racing, sensor fusion, Formula Student

Acknowledgements

First and foremost we would like to thank our supervisor Björn Olofsson for his great guidance and expertise throughout the project. We would also like to thank the Department of Automatic Control at LTH for providing us with crucial tools and equipment, and last but not least, a huge thank you to all the members that stayed in the 2021 Lund Formula Student team. Without their dedication, there would have been no vehicle to make autonomous.

Contents

1. Introduction **9**

2. Background **11**
2.1 The Formula Student Competition 11
2.2 Previous Work 12

3. Theory **14**
3.1 LiDAR 14
3.2 RANSAC 15
3.3 *k*-d Tree 16
3.4 Euclidean Clustering 16
3.5 Projection Matrix 16
3.6 Plane Geometry in 3D-Space 17
3.7 Cone Model in 3D-Space 18
3.8 Linear Regression Using Ordinary Least Squares 18
3.9 Deep Learning 19

4. Hardware and Software Description **20**
4.1 Hardware 20
4.2 Software 21

5. Simulation Environment and Real World Testing **23**
5.1 LFS Simulator 23
5.2 Slamdog 24
5.3 The Lund Formula Student 2021 Racecar — Tunnan 25

6. Method and Project Development **26**
6.1 Ground Removal 27
6.2 Clustering 29
6.3 Cluster Classification 30
6.4 Color Estimation using Calibrated Reflectivity Measurement from LiDAR 33
6.5 Cone Detection Using Camera 34

7. Testing and Evaluation **38**
7.1 Experiment Setup 38
7.2 Evaluation of LiDAR Cone Detection Algorithms 39
7.3 Execution Time Evaluation of Clustering Algorithms 40
7.4 Evaluation of LiDAR Color Classification Algorithm 40
7.5 Evaluation of Sensor Fusion Algorithms 41
8. Results **42**
8.1 Parameter Tuning 42
8.2 Evaluation of LiDAR Cone Detection Algorithms 44
8.3 Execution Time Comparison of Clustering Algorithms 49
8.4 LiDAR Color Estimation 50
8.5 Evaluation of Sensor Fusion Algorithms 51
9. Discussion **56**
9.1 Ground Removal 56
9.2 Clustering 56
9.3 Color Classification with LiDAR 56
9.4 Cone Classification with Camera 57
9.5 Color Classification with Camera 57
10. Conclusions **58**
10.1 Final Result 58
10.2 Ground Removal 59
10.3 Future Work 59
Bibliography **60**

1

Introduction

During 2021, Lund Formula Student has been designing its first ever autonomous racing vehicle, with the intention to compete in at least one of the driverless Formula Student competitions held around the world [*FSG: Concept* 2021]. This Master Thesis is focused on the perception aspect of designing and developing the autonomous racing system for this car.

Problem Formulation

The problem that this thesis aims to solve can be stated as follows. The objective is to get an accurate position and color estimation of the cones that make up the racetrack relative to the car, while the car is in motion. This requires the perception algorithm to run in real time, below 100 ms, as that is the time between scans of the LiDAR available. The problem has to be solved with the available hardware, consisting of a 16 layer 3D LiDAR, a 1080p camera and the NVIDIA Jetson AGX Xavier Developer Kit, an embedded computing board for autonomous machines. For further specifications see Section 4.1.

Other Formula Student teams at other universities have already tried solving the same perception problem and produced good result, which will be discussed further in Section 2.2. This thesis will build on some of their work and explore if it is possible to achieve as good results with a simpler hardware setup.

A video showing the data captured from the LiDAR and camera can be found here [*Sensors only — E-building Parking Lot* 2021].

Integration with Other Parts of the Formula Student Project

This thesis only covers the perception aspect of the autonomous system developed for the Lund Formula Student 2021 race car. The full autonomous system consists of four key components:

- Perception — Interpreting the input from the car's sensors to create a useful description of the surrounding environment.

- Simultaneous Localization and Mapping (SLAM) — Mapping the environment while simultaneously localizing the car within that map.
- Path Planning — Calculate a trajectory.
- Control — Calculate inputs for the car so that it will follow the trajectory.

A flowchart describing how the input information from the car's sensors travels through the system to finally result in a control input can be seen in Figure 1.1.

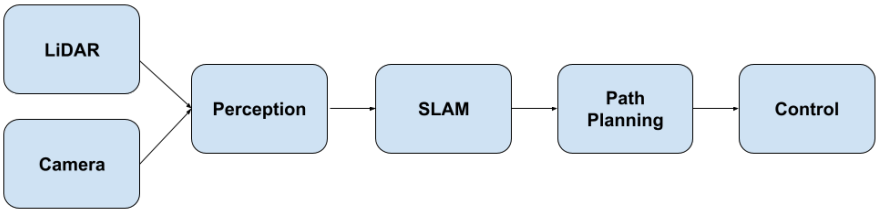


Figure 1.1: High Level Autonomous System Pipeline.

2

Background

2.1 The Formula Student Competition

Formula Student is an annually held engineering competition where teams from around the world build racing cars to compete in engineering design and racing [*FSG: Concept* 2021]. Recently, a driverless class has been added to this competition where teams design an autonomous system to compete on different race tracks defined by visually distinct cones (see Figure 2.1).

	small orange cone single white stripe	small yellow cone single black stripe	small blue cone single white stripe	big orange cone two white stripes
dimensions:	228 mm × 228 mm × 325 mm			285 mm × 285 mm × 505 mm
weight:	0.45 kg			1.05 kg

Figure 2.1: Cones in Formula Student competition, from [*FSG Competition Handbook* 2021].

The type of tracks relevant for this Master Thesis are closed loop tracks that are driven 10 consecutive laps and have a clear set of rules regarding their layout: [*Formula Student Rules* 2020]

1. Straights: No longer than 80 m.
2. Constant Turns: up to 50 m diameter.
3. Hairpin Turns: Minimum of 9 m outside diameter (of the turn).

4. Miscellaneous: Chicanes, multiple turns, decreasing radius turns, etc.

5. The length of one lap is approximately 200 m to 500 m.

There are also a few restrictions on how the cones have to relate to each other (see Figure 2.2). Consecutive yellow/blue cones have to be a maximum of 5 meters apart and the width of the track is not allowed to be less than 3 meters at any point. The start/finish line is marked by four big orange cones and is 6 meters wide.

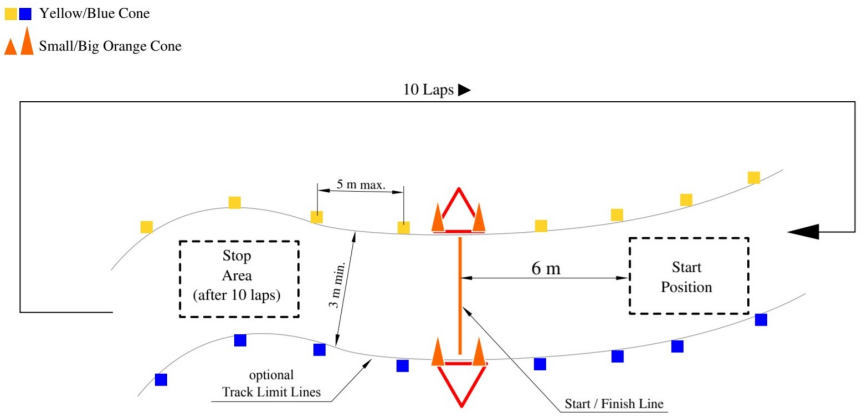


Figure 2.2: Definition of the racing track, from [*FSG Competition Handbook* 2021].

2.2 Previous Work

By Other Teams

Since Formula Student is an annual engineering competition, previous teams from other universities have tried solving the same perception problem, but with slightly different hardware. Two teams that have previously performed well are AMZ Racing and BCN eMotorsport. Their respective reports "AMZ Driverless: The Full Autonomous Racing System" [Kabzan et al., 2019] and "LiDAR cone detection as part of a perception system in a Formula Student Car" [Roche López, 2019] were studied and used as a reference for this work. The idea of dividing the self-driving task into the components shown in Figure 1.1 was inspired by these teams.

The key difference between their approach and the one by Lund Formula student is the hardware. They have higher resolution sensors and more computing power. As a comparison, AMZ's setup consists of three cameras and a LiDAR with twice the resolution. They also have more computing power, allowing the use of more complex algorithms in terms of computing time.

At Lund Formula Student

The autonomous system for the Lund Formula Student car has been in development since summer 2019. The project starting it all aimed at developing an autonomous system good enough to guide an RC car (see Figure 5.2) through the same dynamic events as in the Formula Student competition.

The perception system developed for the RC car uses a previously developed object detection algorithm trained to detect the cones from Figure 2.1 in images [Ahlqvist et al., 2019] and that work will be used in this thesis.

3

Theory

The purpose of this section is to give the reader some familiarity with the core technologies used in this project. The main focus will be on explaining the fundamentals of LiDAR technology, selected machine learning concepts and the basis of the mathematics used.

3.1 LiDAR

LiDAR, formally known as Light Detection and Ranging, is a technology used for determining distances to objects by shooting at them with a laser and measuring the time of flight for the beam to return to the receiver. Since the speed of the beam is known (the speed of light), one can then transform these time measurements into accurate distances (see Figure 3.1).

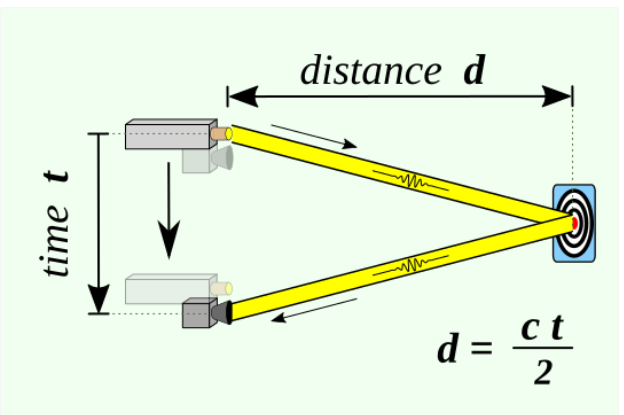


Figure 3.1: Visualisation of Time-of-flight to distance calculation used in LiDAR technology, from [Wikimedia Commons, 2021]. c denotes the speed of light.

In most modern applications, LiDAR is used to create a 3D representation of some environment. This is done by shooting a multitude of beams in a short time frame, where each beam has a known angular difference from the previous. This means that each measurement has a known distance, azimuth angle and polar angle, and together they can form a point cloud describing the surrounding environment as seen from the LiDAR sensor. An example of such point cloud can be seen in Figure 3.2.

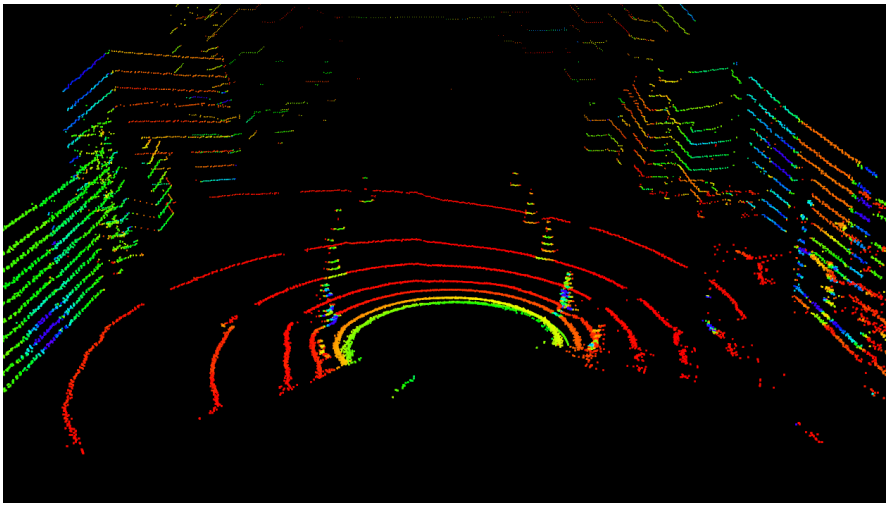


Figure 3.2: Example visualization of a point cloud.

The LiDAR used in this thesis uses monochromatic light with a wavelength of 903 nm. By measuring the returned intensity of the beam, certain information about targeted objects' color can be extracted. For example, targeting an object with a black surface will return a low intensity value, whereas a white surface will return a high intensity value. In Figure 3.2, this is visualized as points returning a low intensity being more red and points returning high intensity more green.

3.2 RANSAC

Random sample consesus (RANSAC) is an iterative method of fitting a mathematical model to a set of data points [Bolles, 1981]. By sampling the minimum required data points to unambiguously determine the parameters of the model and calculating the number of inliers within a given threshold of this model several times, an

estimate of the model's parameters can be made using the parameters that gave the most inliers.

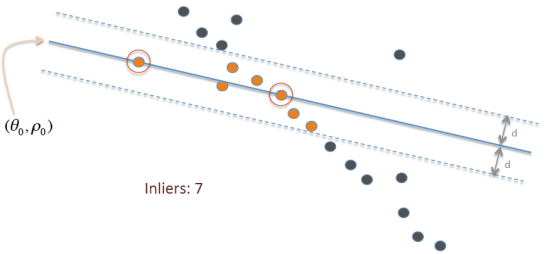


Figure 3.3: Example of RANSAC with a line as mathematical model. θ_0 and ρ_0 are the parameters of the line and d the given threshold, from [Wikipedia, the free encyclopedia, 2021].

RANSAC will later be used to fit the model of a plane to a point cloud containing the ground.

3.3 *k*-d Tree

A k-d tree is a data structure that orders k-dimensional data in k-dimensional space as a binary tree [Bentley, 1975]. Such a data structure results in a search time complexity of $O(\log n)$ where n is the number of data points.

3.4 Euclidean Clustering

Euclidean Clustering or hierarchical clustering using the Euclidean norm [Nielsen, 2016], is a clustering method using the Euclidean distance to determine which data points that should be clustered together. This method will be used later in Chapter 6.

3.5 Projection Matrix

Transforming points from world coordinates to image coordinates of a camera requires the intrinsic as well as the extrinsic parameters of the camera to be known [Zhang, 2014]. The intrinsic parameters of the camera depend on the type of image sensor and lens used, and must therefore be calculated for each individual camera. The extrinsic parameters are only the rotation and translation of the camera with respect to the coordinate system chosen.

The intrinsic parameters of the camera can be defined in an intrinsic camera matrix K. This together with the extrinsic rotation matrix R and translation t gives the projection matrix

$$P = K[R|t] \tag{3.1}$$

where $[R|t]$ means the 3x3 matrix R and 3x1 vector t are concatenated into a 3x4 matrix. The projection matrix P is used to project from world space to image coordinates with the relation

$$x = PX \tag{3.2}$$

where X is the world coordinates and x is the image coordinates.

3.6 Plane Geometry in 3D-Space

A plane in 3D-space can be described by some point $P_0 = (x_0, y_0, z_0)$ located on the plane and a normal vector $N = (a, b, c)$ describing its inclination. The plane then consist of all points with position vector $P = (x, y, z)$ such that the vector from P_0 to P is perpendicular to N. Since two vectors are perpendicular if and only if their dot product is 0, this leads to the set of points P having to fulfill the following equation:

$$N \cdot (P - P_0) = 0$$

Expanding this leads to:

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$$

This is known as the point-normal form of the equation of the plane and is more commonly written as:

$$ax + by + cz + d = 0$$

where

$$d = -(ax_0 + by_0 + cz_0)$$

A plane in 3D-space can also be determined by three points $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$, $P_3 = (x_3, y_3, z_3)$ as long as the points do not lie on the same line. One algorithm for finding the point-normal form of such a plane can be described as follows:

Algorithm 1 Compute the point-normal form given three points

1: $P_1 = (x_1, y_1, z_1)$
2: $P_2 = (x_2, y_2, z_2)$
3: $P_3 = (x_3, y_3, z_3)$
4: $V_1 = P_2 - P_1$ ▷ V_1 will lie on the plane
5: $V_2 = P_3 - P_1$ ▷ V_2 will lie on the plane
6: $N = (a, b, c) = V_1 \times V_2$ ▷ N will be perpendicular to the plane
7: $d = -(ax_1 + by_1 + cz_1)$

A benefit of representing a plane by its point-normal form is that one can easily calculate the distance $\mid D \mid$ from a point $P = (p_x, p_y, p_z)$ to the plane as

$$D = \frac{ap_x + bp_y + cp_z + d}{\sqrt{a^2 + b^2 + c^2}} \tag{3.3}$$

By looking at the sign of D, one can also determine if the point P lies above or below the plane (above corresponds to the direction of the normal vector for the plane). Note that by normalizing the normal vector N, the denominator in Equation (3.3) equals one. Doing this normalization once will speed up the process when consecutive distance calculations are made.

3.7 Cone Model in 3D-Space

A model of a cone in 3D-space can be described by the following coefficients (see Figure 3.4):

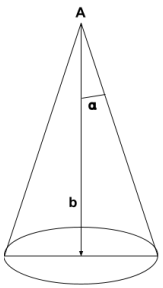


Figure 3.4: Mathematical model of a cone in 3D-space.

- A: the X, Y and Z coordinate of the cone's apex
- b : the X, Y and Z coordinate of the cone's axis direction. We let the length of the direction vector define the cones height
- α : The cone's opening angle

3.8 Linear Regression Using Ordinary Least Squares

Ordinary Least Squares (OLS) [Böiers, 2010] is a linear unbiased estimator used to solve the linear minimization problem

$$\min_x |Ax - b| \tag{3.4}$$

with the solution

$$x = (A^T A)^{-1} A^T b \tag{3.5}$$

where A is a matrix and x and b are vectors.

3.9 Deep Learning

Artificial Neural Network

An artificial neural network (ANN) [Goodfellow et al., 2016] is a computational model loosely based on the way the neural networks are structured in our brains. An artificial neural network uses nodes, weights and activation functions to best fit a non-linear function to the data the model is trained with.

Convolutional Neural Network

Convolutional neural networks (CNNs) [Goodfellow et al., 2016] are a type of artifical neural network architecture using convolutional layers to extract spatial information from input data. The convolutional layer applies a filter of predetermined size to the 2D data and the filter weights are learned by the network using training data.

You Only Look Once

You Only Look Once (YOLO) [Redmon and Farhadi, 2018] is a real-time object detection algorithm used for general object detection in images. It can be trained to detect and classify objects in a picture.

4

Hardware and Software Description

The purpose of this section is to give a description of the hardware components used in this thesis and their specifications.

4.1 Hardware

LiDAR

The LiDAR used is a VLP-16 Velodyne LiDAR Puck [*Velodyne LiDAR Puck Datasheet* 2021] with the relevant specifications listed in Table 4.1.

Table 4.1: Relevant specifications of the LiDAR

Measurement range	Up to 100 m
Distance accuracy	± 3 cm
Angular resolution (Vertical)	2.0°
Field of view (Vertical)	$+15.0^\circ$ to $-15.0^\circ (30^\circ)$
Angular resolution (Horizontal)	$0.1^\circ - 0.4^\circ$
Field of view (Horizontal)	360°
Rotation rate	5 Hz – 20 Hz
Wavelength	903 nm
Dimensions	103 mm Diameter x 72 mm Height
Weight	830 g

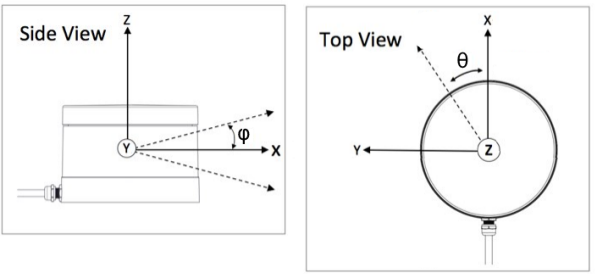


Figure 4.1: Coordinate system chosen for the 3D LiDAR.

Camera

The camera used is a Basler acA1920-40uc USB 3.0 camera with the Sony IMX249 CMOS sensor [*Basler ace acA1920-40uc* 2021]. The camera has the relevant specifications listed in Table 4.2.

Table 4.2: Relevant specifications of the camera.

Resolution	1920x1200 pixels
Frame Rate	41 fps
Shutter	Global Shutter
Color Depth	10 bits

Computer

The computer used on the actual car is an Nvidia Jetson AGX Xavier Developer Kit [*Jetson AGX Xavier Developer Kit* 2021] with the specifications listed in Table 4.3.

Table 4.3: Computer specifications.

CPU	8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
GPU	512-core Volta GPU with Tensor Cores
Memory	32GB 256-Bit LPDDR4x | 137GB/s

4.2 Software

ROS

The Robot Operating System (ROS) is a framework for developing robot software [*ROS* 2021]. The framework's main features is providing an easy and structured way of sharing information between applications that are running concurrently. A good

example regarding the cone detection information is how the LiDAR application relays its information about the possible cone clusters to the sensor fusion application.

The distribution of ROS used in this project was ROS Melodic Morenia.

PCL

Point Cloud Library (PCL) is an open project released under the BSD licence for image and point cloud processing [*Point Cloud Library (PCL)* 2021]. The library holds a collection of useful functions used in this Master Thesis, such as RANSAC and other model fitting functions.

TensorFlow

TensorFlow is an open-source software library for machine learning [*TensorFlow* 2021]. The library provides a framework and fully implemented functions for creating as well as training state of the art machine learning models.

The version of TensorFlow used in this project was 1.14.0.

5

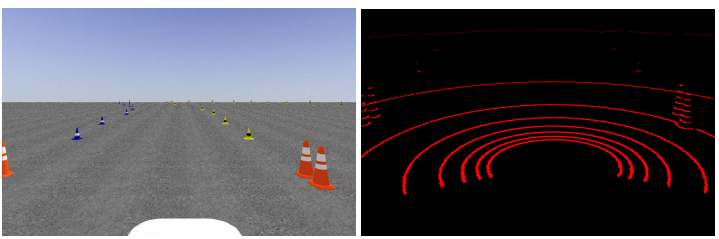
Simulation Environment and Real World Testing

An integral part of the development of the autonomous vehicle has been the ability to test the implementations both easily, quickly, as well as realistically. This has been done through what has been chosen to be called the LFS (Lund Formula Student) simulator and the RC car known as Slamdog, and finally the Lund Formula Student 2021 car Tunnan.

5.1 LFS Simulator

The developed simulator is based on a simulator made by the Edinburgh University Formula Student team [*EUFS Simulator* 2021] and was modified to fit the sensor setup and car dynamics of the Lund Formula Student team as well as the chosen system architecture.

The simulator simulates a VLP-16 LiDAR without the intensity return and a generic camera with the same resolution and frame rate as the Basler camera used on the real car (see Figure 5.1).



(a) Camera in simulator (b) Point cloud from simulator.

Figure 5.1: Visualization of the output produced by the LFS simulator.

5.2 Slamdog

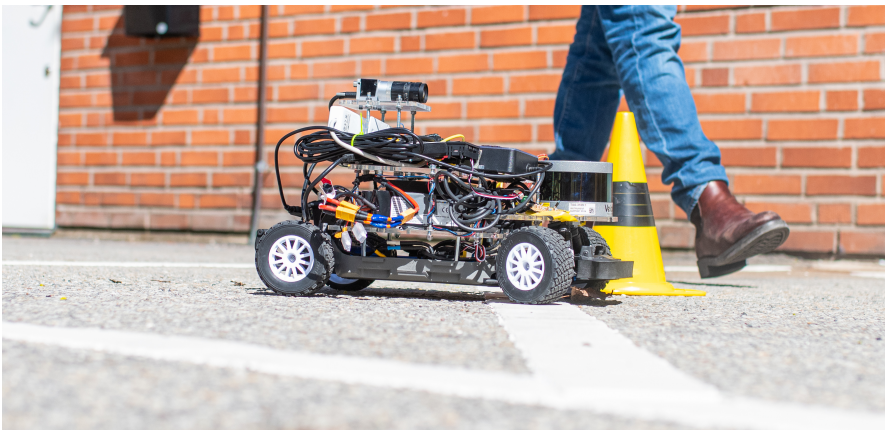


Figure 5.2: The RC testing vehicle known as Slamdog.

When implementations worked in the LFS simulator, they had to be tested in the real world. Since the real full-sized car was not finished until the end of this Master Thesis, a radio controlled vehicle built on the Traxxas TRX-4 foundation was used with the same sensor setup as the final vehicle. The biggest difference between the RC vehicle and the final vehicle was the height of the camera and translation relative to the LiDAR sensor. For more information regarding Slamdog, see [Ahlqvist et al., 2019].

5.3 The Lund Formula Student 2021 Racecar — Tunnan

Figure 5.3: LFS21 racing car.

The first autonomous and electric race car developed by Lund Formula Student was finished in the summer of 2021. Using the same VLP-16 LiDAR and Basler camera for perception, a Simplex Motion SH100A servo motor was used for steering actuation. An EMRAX 228 Axial flux motor together with an HV-500 Liquid-cooled inverter from Drivetrain Innovation were used as the drivetrain.

The car has a top speed of 83 km/h and a 0 to 50 km/h time of 3.7 seconds.

6

Method and Project Development

This section focuses on describing the methods used and developed to solve the problem stated in Chapter 1, as well as motivating some of the design choices made.

The architecture chosen for the perception system can be broken down into the subparts seen in Figure 6.1.

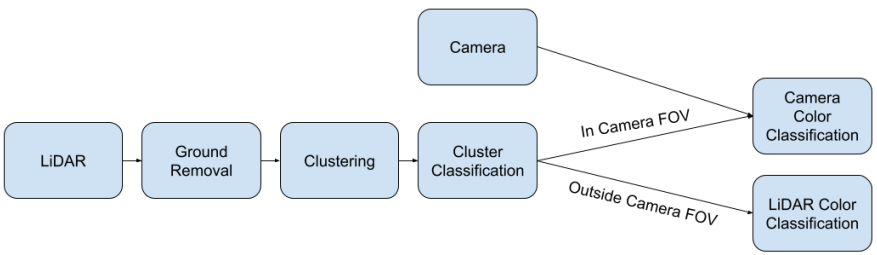


Figure 6.1: Overview of the perception system. FOV = *Field of view*.

The motivation for choosing this structure was to utilize the strengths of the two sensors. Detecting cones in a camera image and classifying their color can be done accurately as shown by [Ahlqvist et al., 2019]. It is, however, hard to get an accurate position estimate. The LiDAR, however, provides very good position measurements and the idea is to fuse these sensor inputs into a single output in the best possible way. As stated in Chapter 1, the aim was to design a perception system that could keep up with the LiDAR's output rate of 10 Hz.

The following sections will focus on how the input from the LiDAR and the camera travels through the system to compute an output in terms of the position and color of the cones in the local environment.

6.1 Ground Removal

The first step in the perception system is to remove the points hitting the ground in the point cloud produced by the LiDAR. Here, we experimented with two different approaches, which are described in the following sections.

Using RANSAC

Ground removal using the RANSAC algorithm (see Section 3.2) fits the mathematical model of a plane (see Section 3.6) to the entire point cloud received from the LiDAR. This is done with the assumption that the overwhelming majority of laser beams will hit the ground, and therefore a plane in the ground should have the most inliers.

Using the Point Cloud Library (PCL), the RANSAC algorithm was applied to the entire point cloud received and the parameters of the plane best fitting to the ground were estimated. All the points that were considered inliers were then removed from the point cloud as well as all points lying below the plane.

Novel Ground Removal

One of the drawbacks of using the RANSAC approach to find and remove points belonging to the ground is the assumption that the ground will be somewhat flat and can be described by the mathematical model of a plane. This led to the decision of developing a novel ground removal algorithm without these drawbacks.

The steps of the algorithm can be described as follows:

1. Divide the 3D-space of the point cloud into circular area segments (see Figure 6.2).

2. Find the lowest point in each segment. This is done by simply converting the polar coordinates into Euclidean coordinates and choosing the one with the lowest z-value.

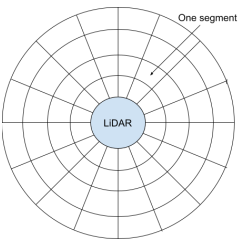


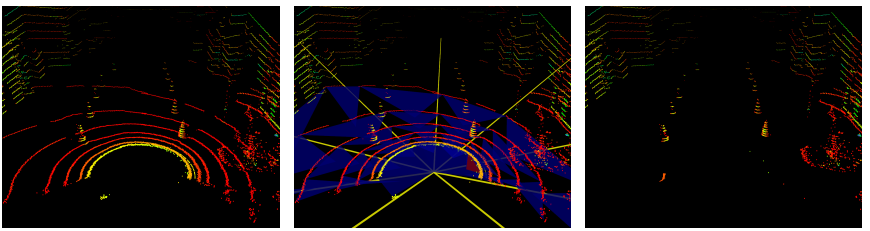
Figure 6.2: Segmentation of the point cloud space as seen from above.

3. Iteratively use 3 points and fit a plane to them. To fit a plane to 3 points in 3D-space we use Algorithm 1 described in Section 3.6. The first planes use the coordinate right below the center of the LiDAR and 2 points from the neighboring segments closest to the LiDAR. Next, use two points from the second closest ring of segments and the center-point of the previous plane to create the next ring of planes. Repeat for all rings of segments. When creating

the next ring of planes, a calculation is also done comparing the normal vectors from the previous plane to the current by a simple dot product. If the dot product is less than a certain threshold (this corresponds to the angle between the planes), then the plane is classified as no longer being part of the ground. This will, for example, be the case if the points start hitting a wall or some other large object. The result of all the planes can be seen as a triangular mesh that approximates the surrounding ground.

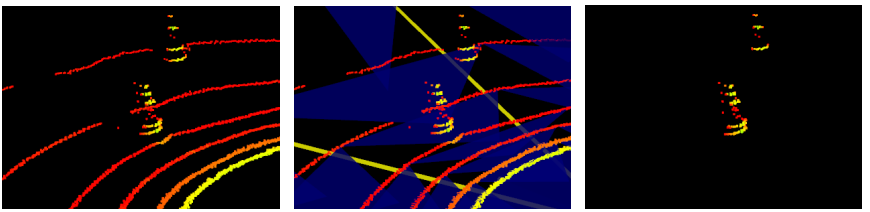
4. The next step is to iterate over all the points in the point cloud again and calculate the distance to the closest plane. The distance calculation is done according to Equation (3.3) described in Section 3.6. If the distance is larger than a threshold value, the point is classified as not being part of the ground, otherwise it is removed from the point cloud.

Figure 6.3 illustrates the novel ground removal algorithm in action. The yellow lines in Subfigure 6.3b show the segmentation and the blue triangles are the locally fitted planes.



(a) Raw input (b) Fitting the local planes (c) Result

Figure 6.3: Illustration of the Novel Ground Removal algorithm.



(a) Raw input (b) Fitting the local planes (c) Result

Figure 6.4: Zoomed version of Figure 6.3.

A video comparing RANSAC ground removal with the novel ground removal can be found here [*RANSAC Ground Removal vs Novel Ground Removal* 2021].

6.2 Clustering

The next step is to order the remaining points of the point cloud into clusters. The idea is that points that lie close together are likely to belong to the same object.

The first approach uses Euclidean Clustering, see Section 3.4. This has been used before by other Formula Student teams, for example AMZ Racing [Kabzan et al., 2019].

It was, however, noticed that this could become slow for inputs where the point clouds have a lot of points. This led to the development of a novel clustering algorithm called String Clustering, using the order of the points in the cloud returned from the LiDAR as an advantage to speed up the clustering process. The implementation of both of these algorithms is described in the following sections.

Euclidean Clustering

Using Point Cloud Library (PCL), the remaining point cloud is structured in a k-D tree to reduce search time complexity, then Euclidean Clustering is performed. The Euclidean distance threshold is set to 285 mm which is the maximum diameter of the defined cones.

String Clustering

Firstly, an empty set of clusters is initialized. The point cloud points are then looped through from the top to bottom in elevation angle and left to right in azimuth angle, which is the same ordering they have when obtained from the LiDAR. For every point, the radial distance between the point and every cluster's right-most point as well as the arc length distance are compared, see Figure 6.5. For computational efficiency, we then conclude that the final distance from the cluster to the point is the maximum of these distances. If the closest cluster is closer than the threshold distance of 285 mm, the point is added into the cluster, otherwise a new cluster is initialized with that point.

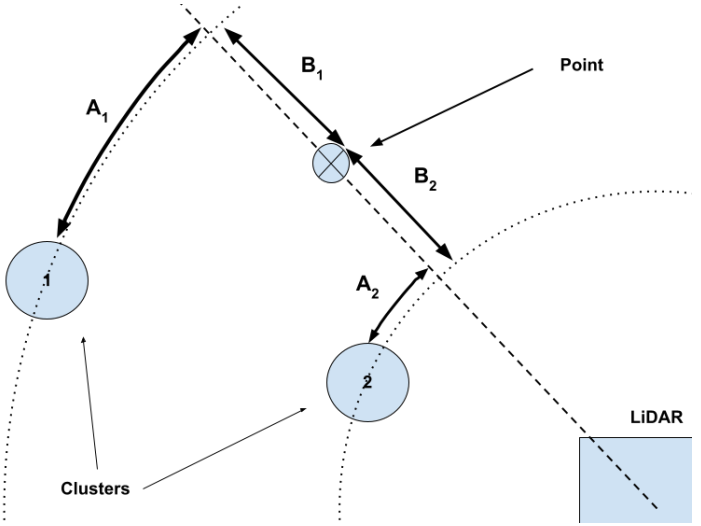


Figure 6.5: The distances marked by A_i correspond to the arc-length distances and B_i the radial distances to cluster i.

Every time a cluster is updated, a check is also made whether or not this cluster could possibly be a cone, by checking its height above the ground as well as its size. If the cluster can no longer be a valid cone, it is removed from the set of clusters.

Since the points are looped over from left to right in azimuth direction, clusters with an arc-length distance larger than the threshold distance will never have more points added to them. Therefore, they can be removed from the set of current clusters, which speeds up the process.

To emphasize on the key difference, String Clustering uses the known structure of the point cloud to its advantage to only loop over set of points once. For every point, all current clusters are evaluated. This set of current clusters will, however, always be relatively small, since it will only contain clusters close to the currently evaluated point (along the azimuth direction).

6.3 Cluster Classification

Once the remaining part of the point cloud has been divided into clusters, the next step is to classify these clusters as either being one of the cone types seen in Figure 2.1 or not a cone.

Two different approaches were tried. The first approach was to use RANSAC

(see Section 3.2) to fit a mathematical model of a cone (see Section 3.7) to the cluster.

Calculating Distance to a Cone Model

To be able to use the RANSAC algorithm, a method for calculating the distance to the model is needed. The method implemented can be described as follows:

We start out by looking back at Figure 3.4. Since a cone is symmetrical around its direction vector b, we can imagine a plane defined by the cone's apex A, the point $B = A + b$ and the point P that we wish to determine the distance from. We can now transform the problem into 2D, since we know that the shortest path will lie on this plane, see Figure 6.6.

This leaves us with three cases corresponding to the areas marked with blue, red and green in Figure 6.6. If the point lies in the blue area, the shortest path will be the length of v_1, in the red area the length of v_2, and in the green area the length of v_3. Note that we are calculating the distance to the surface area of the cone model, which in the 2D-case is the distance to the line AC

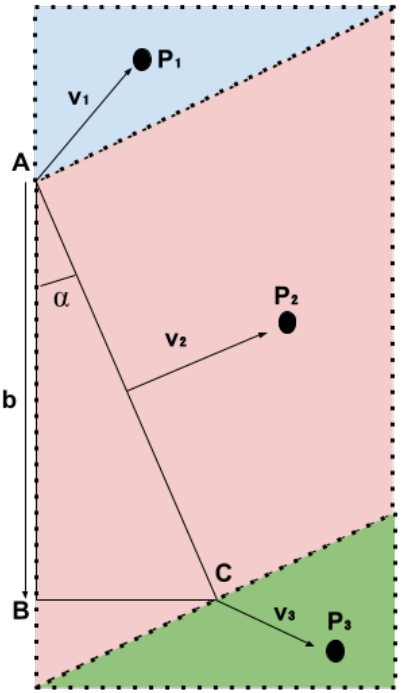


Figure 6.6: Three cases for computing the closest distance from a point to a cone model.

This leads us to the first step of the algorithm being to determine which case we are dealing with. We start by computing the coordinates of point C. To do this, we first need a direction vector d as:

$$d = (P - A) - \frac{(P - A) \cdot b}{b \cdot b} b \tag{6.1}$$

which will be perpendicular to b. We then compute C as:

$$C = A + b + \frac{d}{\|d\|} \tan\alpha \, \|b\| \tag{6.2}$$

Next we compute which case we are dealing with. If we are dealing with the blue case we know that $(P-A)\cdot(C-A) < 0$. Similarly, if we are dealing with the green case, $(P-C)\cdot(A-C) < 0$ should hold true. If neither of these cases are true, we know it is the red case and can compute the distance as:

$$\|v_2\| = \left\| (P-A) - \frac{(P-A)\cdot(C-A)}{(C-A)\cdot(C-A)}(C-A) \right\| \tag{6.3}$$

Novel Fitting Approach

One of the drawbacks of using RANSAC for the fitting task is that it can become quite computationally heavy, since it is an iterative algorithm and a lot of distance calculations have to be done. This led to the decision of trying a simpler approach, where we simply estimate the parameters of the cone model. Firstly, we make the assumption that the cone model's direction vector has to be aligned with the ground's normal vector, which we can get from the earlier ground removal step. Secondly, we assume that the apex point of the cone model has to be the same distance from the ground as the height of the cone. The last assumption we make is that the center point (defined as the average of all points in the cluster) is one fourth of the cone's diameter in front of the cone's direction vector in the direction as seen from the LiDAR.

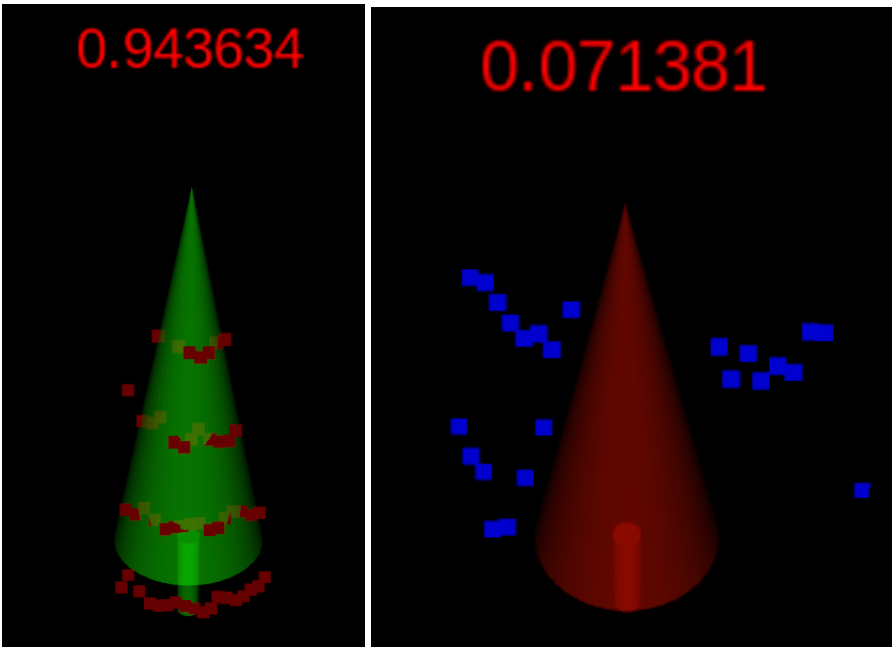
Model Fit Scoring

Once the parameters for the cone model are determined, the next step is to decide how good the cluster fits the cone model. For this purpose, the scoring function defined as

$$\text{score} = \frac{1}{n}\sum_{i=1}^{n} 1 - \min\left(\frac{{d_i}^2}{t^2}, 1\right) \tag{6.4}$$

is used, where d_i is the distance of point i in the cluster to the cone model, t is an inlier-threshold distance set to 3 cm (the distance accuracy of the LiDAR) and n is the number of points in the cluster. This results in a bounded scoring function between 0 and 1, where a perfect fit would have a score of 1. A cut-off threshold is then chosen where every cluster with a score above this threshold is classified as a cone.

Figure 6.7 illustrates the scoring function applied to two clusters, one being an actual cone and the other a cluster formed from the LiDAR hitting a bush at the side of the track.



(a) Cluster is an actual cone. (b) Cluster formed from hitting a bush.

Figure 6.7: The scoring function visualized. The red number indicates the actual score for the fit.

6.4 Color Estimation using Calibrated Reflectivity Measurement from LiDAR

Using the returned intensities from the LiDAR, the color of the cones can be estimated using their intensity patterns. As white is known to be highly reflective and black the opposite, it was reasonable to assume that the intensity pattern of the blue cone would be higher in the middle for the points reflecting on the white stripe than those on the top and bottom reflecting on the blue color. The opposite was assumed for yellow.

A quadratic equation is fitted to the average intensity of each layer that hits the cone. Given the layer indices $x_1, x_2, x_3, \ldots, x_n$ of the n layers and the intensities

$y_1, y_2, y_3, \ldots, y_n$ that hit the cone, the minimization problem

$$\underset{a,b,c}{\text{minimize}} \left| \begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ & \vdots & \\ x_n^2 & x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} \right| \tag{6.5}$$

is set up and solved using ordinary least squares. Since a represents the concavity of the equation, this estimate is used to estimate the color of the cone as defined in Table 6.1.

Table 6.1: Color estimation using linear regression.

$a > 0$	yellow
$a < 0$	blue

6.5 Cone Detection Using Camera

Using the camera for cone detection can be done in several ways. It can be done using object detection on the entire image or image classification of the parts of the image where cones are believed to be.

Calibration

For proper sensor fusion, the relative transformations between the sensors need to be known. This together with the intrinsic camera parameters makes it possible to transform measurements from the LiDAR's coordinate system to the camera space using the projection matrix P. The intrinsic camera parameters are calibrated for the specific camera setup using already existing software [*Camera Calibration - ROS Wiki* 2021], while the extrinsic parameters are measured manually.

Projection Sensor Fusion

The first sensor fusion approach chosen uses the LiDAR's estimates of where cones are present around the car, extracts those parts in image space and then classifies the extracted part.

The possible cones captured by the LiDAR are projected into image space first using the projection matrix P. This gives the image coordinates where the cones should appear in the camera. With the range measurement, the approximate size of the cone in the image can be calculated, and thereafter a patch of the image can be extracted to then be classified.
The patch can be classified as either one of the possible cones from Figure 2.1 or

not a cone at all using an image classifier.

The image classifier made for the task of classifying the image patches is a convolutional neural network ending in a 5 node softmax layer representing the probability of the 5 possible classes the image patch can contain as defined in Table 6.2.

Table 6.2: Possible cone classes.

yellow cone	blue cone	orange cone	big orange cone	not a cone

The full network architecture used is shown in Table 6.3 and the images are resized to 16x16x3 before being fed into the network.

Table 6.3: Image classification network structure.

layer type	output shape	filter shape	activation function
2D Convolution	14x14x16	3x3	ReLu
2D Max Pooling	7x7x16	-	-
2D convolution	5x5x8	3x3	ReLu
2D Max Pooling	2x2x8	-	-
Flatten	32	-	-
Dense	16	-	ReLu
Dense	5	-	Softmax

Figure 6.8 shows the resulting bounding boxes projected on the image, the color of the bounding boxes represent what that image patch has been classified as.

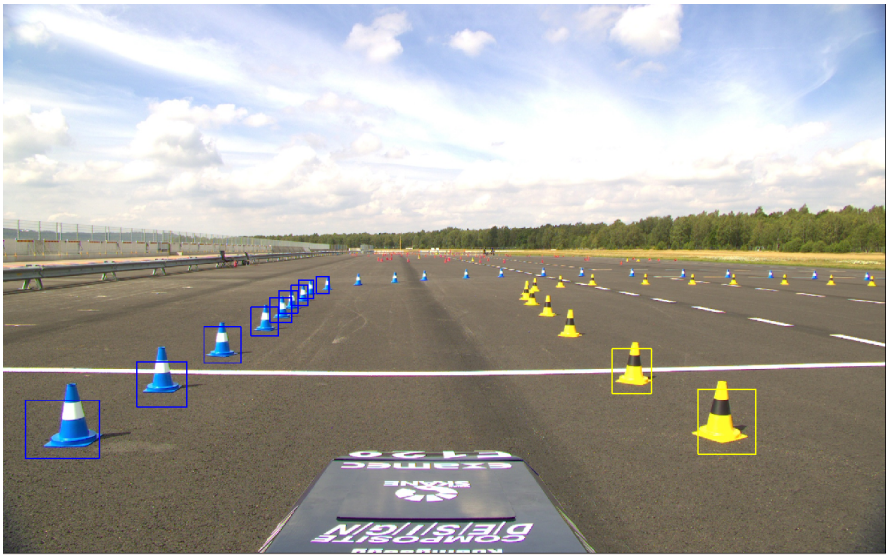


Figure 6.8: Bounding boxes made when projecting LiDAR cone estimates to the camera and classifying.

YOLO Sensor Fusion

The second sensor fusion approach uses the pretrained YOLO-network from [Ahlqvist et al., 2019] to detect the cones in the images. This approach was developed since the projection sensor fusion gave too many false positives, which is a greater problem than false negatives for the final performance of the car.

The approach transforms the possible cones captured by the LiDAR to image space in the same fashion as the first. Instead of extracting the image patches, bounding boxes are generated around the cone in the image. The bounding boxes are then paired in image space with the ones computed by the YOLO-network. The pairs are firstly made by distances between the midpoints of the bounding boxes and secondly the difference in bounding box height. If the midpoint of a possible cone from the LiDAR is too far away from any YOLO bounding box, the possible cone is discarded. Color of the cone is chosen by the color of the YOLO bounding box and the location is chosen by the LiDAR position estimate. Figure 6.9 shows the YOLO sensor fusion algorithm. The yellow and blue bounding boxes are the detected cones of the YOLO algorithm and the green bounding boxes are the estimated bounding boxes of the cones from the LiDAR algorithm. The pairs are shown with the green lines between the bounding boxes.

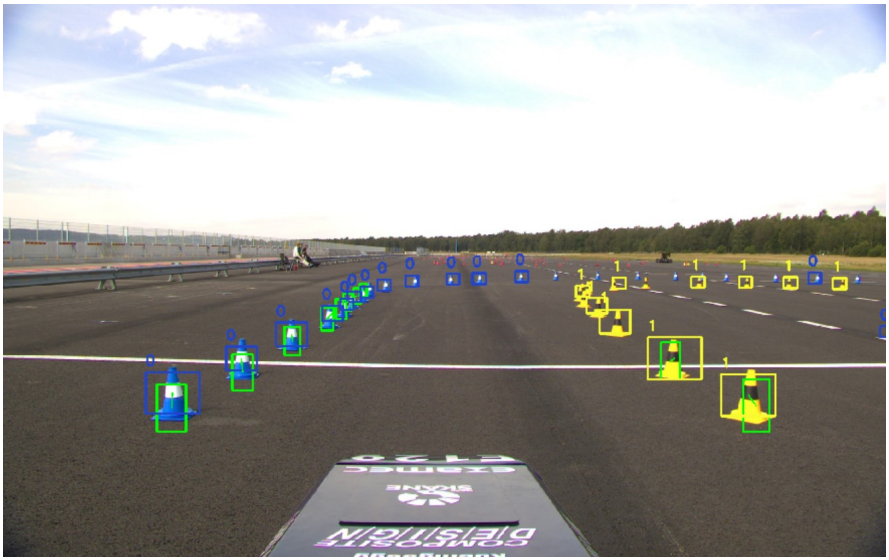


Figure 6.9: Bounding boxes from YOLO together with bounding boxes from the LiDAR algorithm (green).

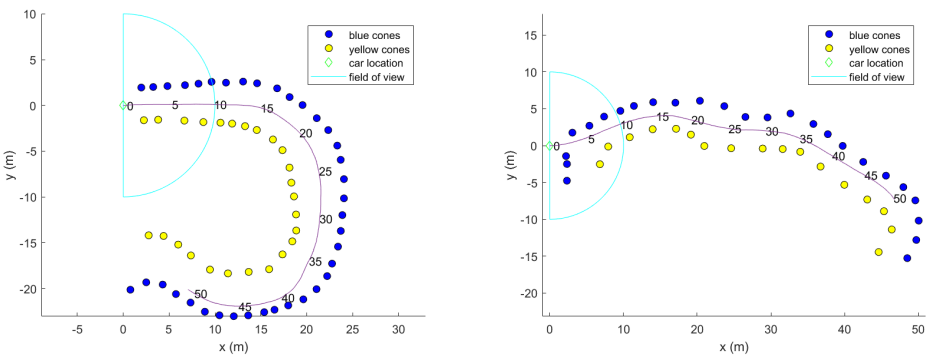
7

Testing and Evaluation

The purpose of this section is to describe the tests and experiments set up to evaluate the performance of the developed algorithms and methods used in this thesis.

7.1 Experiment Setup

For evaluating the performance of the proposed algorithms, sensor data from the LFS21 driverless vehicle driving 50 meters at about 2.5 m/s were recorded. One recording is from testing the car on the parking lot behind the E-building at the Faculty of Engineering at Lund University (LTH) and the other from testing at the Ljungbyhed Raceway. The sensor data were then used to create maps of how the car had moved and where the cones were present using the SLAM algorithm mentioned in Chapter 1. These maps were then manually annotated with the correct colors of the cones.



(a) Ljungbyhed Raceway.

(b) E-building Parking Lot.

Figure 7.1: Maps used as ground truth.

(a) Ljungbyhed Raceway. (b) E-building Parking Lot.

Figure 7.2: The maps from Figure 7.1 as seen from the camera.

7.2 Evaluation of LiDAR Cone Detection Algorithms

To evaluate the performance of the cone detection algorithms, the maps were used as reference. Based on qualitative evaluations of the point cloud measured by the LiDAR, we defined a field of view for the algorithm, where it is reasonable for the algorithm to detect cones. This field of view is 10 meters and 180° in front of the vehicle as shown in Figure 7.1.

Comparing the output of the algorithms with the cone locations inside the field of view at the time the sensor data were captured, measurements were made with the following criteria:

- How many of the cones we perceive are within 0.3 meters of their true location (true positives).
- How many of the cones we perceive are more than 0.3 meters away of any cone (false positives).
- How many of the cones that are in the field of view do we not perceive at all (false negatives).

With these measurements, two metrics were constructed to evaluate the algorithms' performances. These metrics were the hit rate (also known as sensitivity) defined as

$$\text{hit rate} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{7.1}$$

which measures how well the algorithm finds cones.

Further, the precision is used as a metric:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{7.2}$$

which measures how many of the cones found actually are cones. The combination of all ground removal and clustering algorithms were then tested, which equates to four separate algorithms:

1. RANSAC Ground Removal with Euclidean Clustering
2. RANSAC Ground Removal with String Clustering
3. Novel Ground Removal with Euclidean Clustering
4. Novel Ground Removal with String Clustering

7.3 Execution Time Evaluation of Clustering Algorithms

The reason for developing the String Clustering algorithm was to gain speed compared to using Euclidean Clustering. To evaluate the difference in execution time, the size of the input point cloud and the corresponding execution time of only the clustering algorithms were recorded. This was done by taking parts of the complete point cloud of different sizes and feeding those parts into the clustering algorithms. The point cloud data used were from the E-building parking lot, since these point clouds were denser because of the surrounding bushes and trees.

7.4 Evaluation of LiDAR Color Classification Algorithm

The evaluation of the LiDAR color classification algorithm was performed by taking the correctly associated cones from the novel ground removal with String Clustering and comparing the estimated color to the true color. The algorithm has three possible outputs per clusters as defined in Table 7.1

Table 7.1: Possible cone classes.

blue cone	yellow cone	no color

Two metrics were constructed to evaluate the algorithm's performance. Firstly, we measure at what range the algorithm can estimate the correct cone color, which we define as the color hit rate:

$$\text{color hit rate} = \frac{\text{correct color}}{\text{incorrect color} + \text{no color} + \text{correct color}} \tag{7.3}$$

Secondly, we calculate the misclassification rate at different ranges, which does not take clusters with no color estimate into account:

$$\text{misclassification rate} = \frac{\text{incorrect color}}{\text{incorrect color} + \text{correct color}} \tag{7.4}$$

7.5 Evaluation of Sensor Fusion Algorithms

The evaluation of the sensor fusion algorithms used the cone position estimates from the novel ground removal with String Clustering algorithm together with the sensor fusion approach being evaluated. The metrics used to evaluate the algorithm performances are the hit rate (7.1) and the precision (7.2) as well as the color hit rate (7.3).

The sensor fusion approaches were also compared with just the novel ground removal and String Clustering algorithm, but only including the cones that fall within the camera field of view of 10 meters of the car. The evaluation of the color accuracy takes every cone detected and determines if it was detected as the correct color. The color accuracy is then calculated as in

$$\text{color accuracy} = \frac{\text{correct color}}{\text{incorrect color} + \text{correct color}}. \tag{7.5}$$

8

Results

The following chapter will firstly describe how the parameters were tuned for the algorithms and then the results when performing the evaluation methods introduced in Chapter 7, are presented.

8.1 Parameter Tuning

Model Fit Scoring Threshold

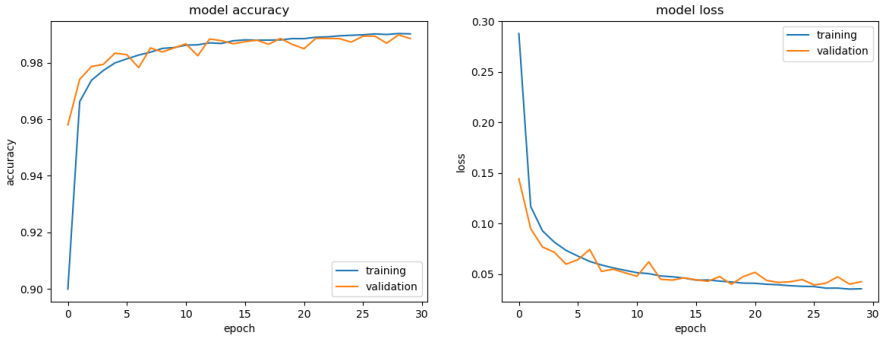
The cut-off threshold for the cone model fit scoring explained in Section 6.3 was set to 0.7. This gave good results in early testing and was thereafter kept the same through the development of the project. It is therefore possible that a different value could produce a better result, but since the improvement gained through finding the optimal value was deemed marginal, no further testing in this respect was done.

Projection Sensor Fusion

The model defined in Table 6.3 was trained with $126,784$ training images and $31,696$ validation images in a $80/20$ training/validation split for 30 epochs. The training data are split as evenly as possible between the classes, more specifically, as specified in Table 8.1.

Table 8.1: Cone class image distribution

yellow cone	blue cone	orange cone	big orange cone	not a cone
31.97%	24.92%	4.64 %	6.50%	31.97%



(a) Accuracy of the model per epoch.

(b) Loss of the model per epoch.

Figure 8.1: Accuarcy and loss of the training and validation data per epoch.

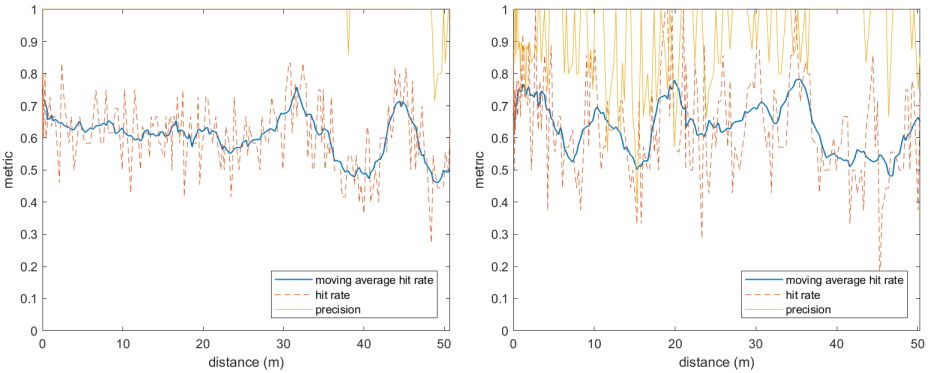
Table 8.2: Final accuracy and loss of the classifier.

	Traning set	Validiation set
Accuracy	0.9889	0.9875
Loss	0.0396	0.0481

Figure 8.1 shows the accuracy and loss calculated for both the training and validation data per epoch during training and Table 8.2 shows the final loss and accuracy at epoch 30.

8.2 Evaluation of LiDAR Cone Detection Algorithms

RANSAC Ground Removal with Euclidean Clustering



(a) Ljungbyhed Raceway.

(b) E-Building Parking Lot.

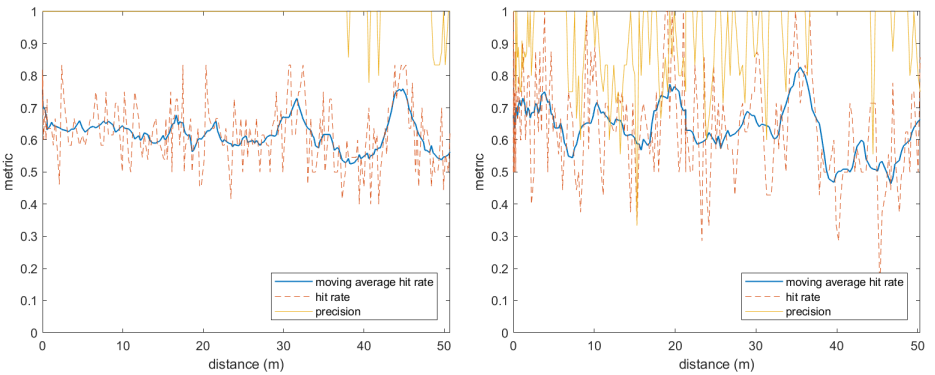
Figure 8.2: Hit rate and precision over distance.

Table 8.3: Mean hit rate and precision.

Location	Hit Rate	Precision
Ljungbyhed Raceway	0.618	0.994
E-Building Parking Lot	0.642	0.915

Figure 8.2 shows the hit rate and precision of the RANSAC ground removal with Euclidean Clustering at each distance on each of the tracks. Table 8.3 shows the average over the entire travelling distance. This will be the reference values when comparing LiDAR cone detection algorithms in this thesis because it is the most commonly used cone detection algorithm used by other Formula Student teams.

RANSAC Ground Removal with String Clustering



(a) Ljungbyhed Raceway.

(b) E-Building Parking Lot.

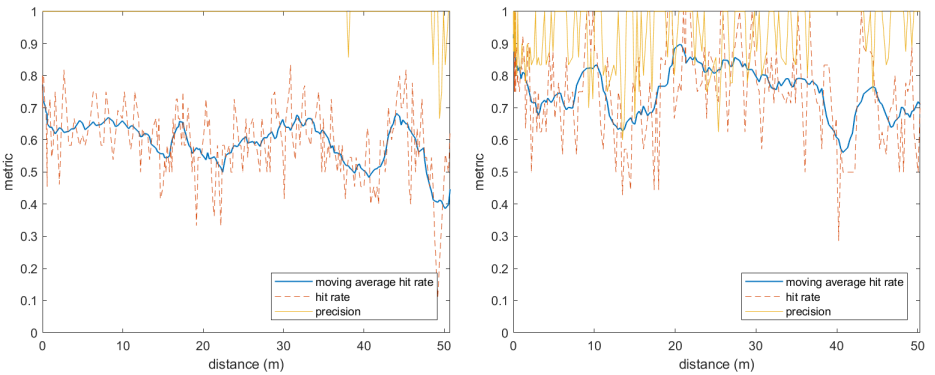
Figure 8.3: Hit rate and precision over distance.

Table 8.4: Mean hit rate and precision.

Location	Hit Rate	Precision
Ljungbyhed Raceway	0.63	0.993
E-Building Parking Lot	0.641	0.915

Figure 8.3 and Table 8.4 show the results of the RANSAC ground removal with the String Clustering algorithm. This is evaluated to show the difference in hit rate and precision between Euclidean Clustering and String Clustering.

Novel Ground Removal with Euclidean Clustering



(a) Ljungbyhed Raceway.

(b) E-Building Parking Lot.

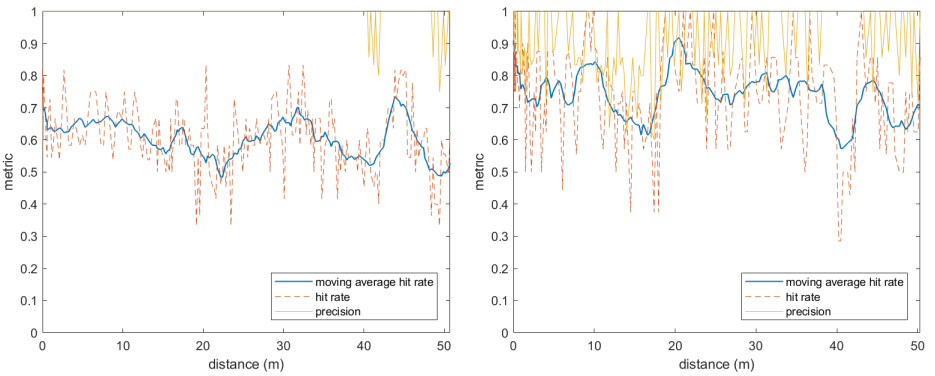
Figure 8.4: Hit rate and precision over distance.

Table 8.5: Mean hit rate and precision.

Location	Hit Rate	Precision
Ljungbyhed Raceway	0.606	0.995
E-Building Parking Lot	0.759	0.924

Figure 8.4 and Table 8.5 show the results of the novel ground removal with Euclidean Clustering, which was evaluated to investigate the effect of modelling the ground with several planes instead of just one.

Novel Ground Removal with String Clustering



(a) Ljungbyhed Raceway.

(b) E-Building Parking Lot.

Figure 8.5: Hit rate and precision over distance.

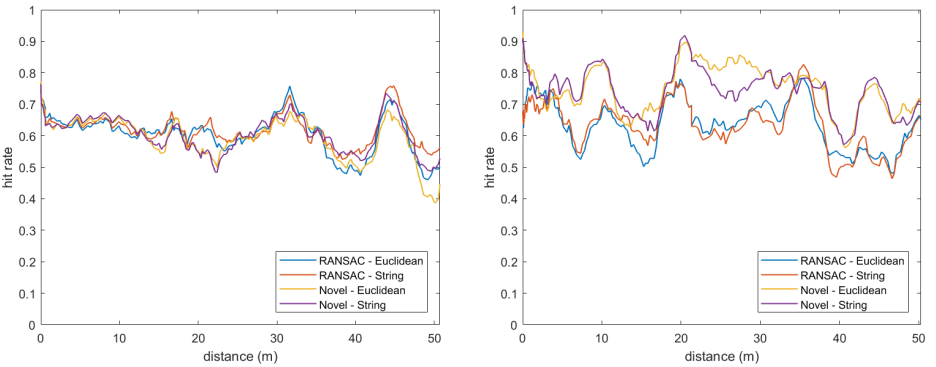
Table 8.6: Mean hit rate and precision.

Location	Hit Rate	Precision
Ljungbyhed Raceway	0.617	0.994
E-Building Parking Lot	0.752	0.921

Figure 8.5 and Table 8.6 show the results of the novel ground removal with String Clustering, which uses both algorithms developed for this Master Thesis.

Comparison

To easily compare the performance of the four cone detection algorithms, we plot their moving average hit rates together, as can be seen in Figure 8.6. Tables 8.7 and 8.8 then show the average hit rate and precision for all algorithms over the entire distance travelled at Ljungbyhed Raceway and the E-building parking lot, respectively.



(a) Ljungbyhed Raceway.

(b) E-building Parking Lot.

Figure 8.6: Moving average of the hit rate for all LiDAR algorithms.

Table 8.7: Mean hit rate and precision at Ljungbyhed Raceway.

Ground Removal	Clustering	Hit Rate	Precision
RANSAC	Euclidean	0.618	0.994
RANSAC	String	0.63	0.993
Novel	Euclidean	0.606	0.995
Novel	String	0.617	0.994

Table 8.8: Mean hit rate and precision at E-building Parking Lot.

Ground Removal	Clustering	Hit Rate	Precision
RANSAC	Euclidean	0.642	0.915
RANSAC	String	0.641	0.915
Novel	Euclidean	0.759	0.924
Novel	String	0.752	0.921

As can be seen in Figure 8.6b and Table 8.8, the most notable difference on hit rate and precision is when the ground removal algorithm is changed on the E-building

parking lot, which is because the ground there is not as even as at Ljungbyhed Raceway.

A video showing the perception using just novel ground removal and String Clustering can be found here [*LiDAR only peception — E-building Parking Lot* 2021].

8.3 Execution Time Comparison of Clustering Algorithms

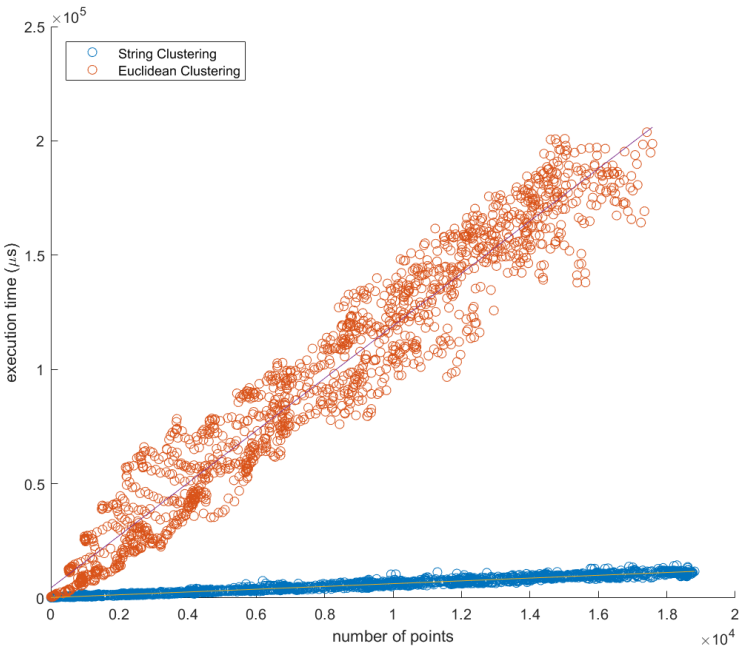


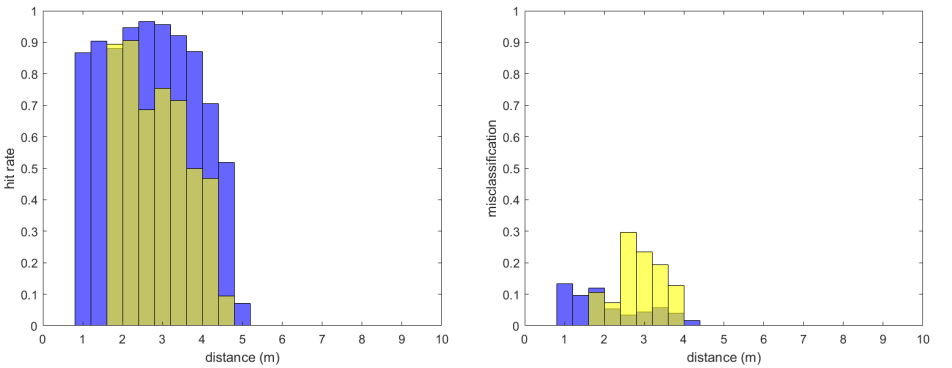
Figure 8.7: Execution time of the implemented clustering algorithms over the number of points.

In Figure 8.7 the execution times recorded for the two algorithms are visualized in correspondence to the number of points fed into the algorithms. The execution times were measured on the hardware specified in Section 4.1. With a simple linear regression, the increase in execution time per point added can be estimated for each algorithm and the results are shown in Table 8.9.

Table 8.9: Estimated time increase per point added in microseconds.

Algorithm	Time Increase per Point Added (μs)
Euclidean Clustering	11.469
String Clustering	0.608

8.4 LiDAR Color Estimation



(a) Hit rate of the color estimation.

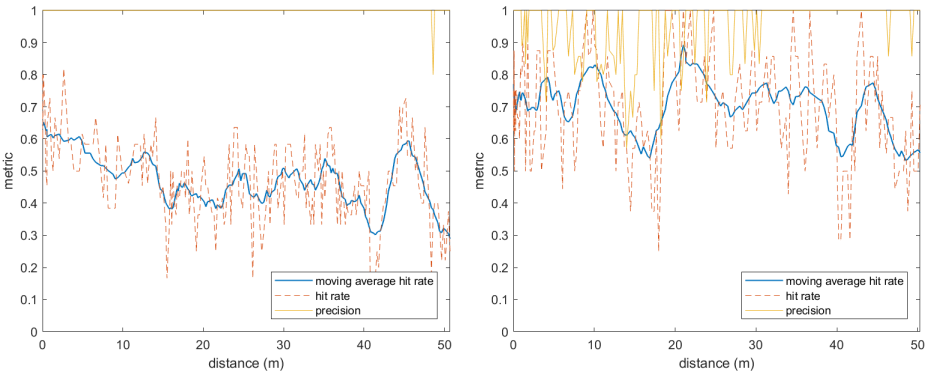
(b) Misclassification rate of the color estimation.

Figure 8.8: LiDAR color estimation accuracy and misclassification rates.

Figure 8.8 shows the accuracy and misclassification rate of the proposed LiDAR color classification algorithm at different distances from the LiDAR. The colored bars in the figures represent how often that color cone was accurately estimated and misclassified as something else. We see how the accuracy of estimates deteriorates faster for the yellow cones. This is because of the low intensity return from the black strips on the yellow cones causing the information to be sparse.

8.5 Evaluation of Sensor Fusion Algorithms

Projection Sensor Fusion



(a) Ljungbyhed Raceway.

(b) E-Building Parking Lot.

Figure 8.9: Hit rate and precision over distance.

Table 8.10: Mean hit rate and precision.

Location	Hit Rate	Precision
Ljungbyhed Raceway	0.467	0.999
E-Building Parking Lot	0.702	0.956

Figure 8.9 and Table 8.10 show the results when adding the projection sensor fusion algorithm to the cone estimates determined by the novel ground removal with String Clustering cone detection algorithm.

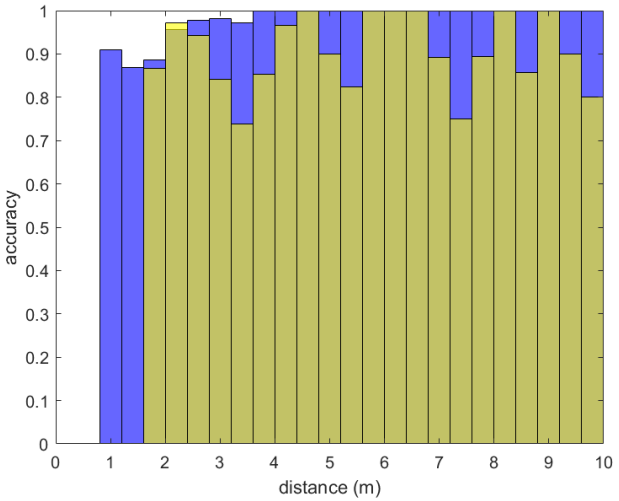
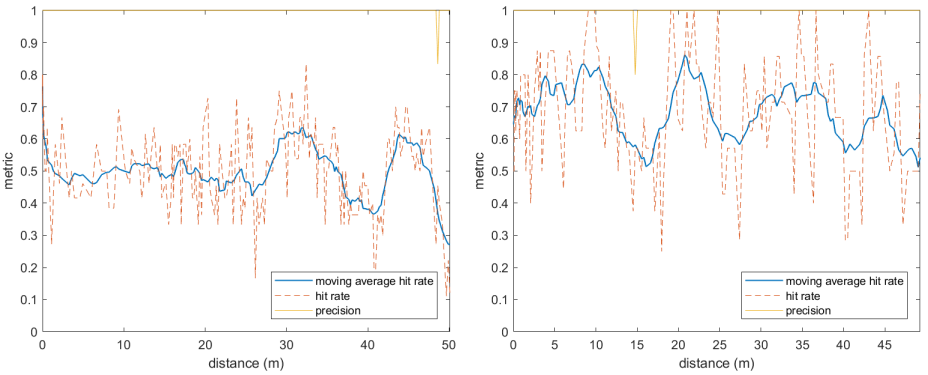


Figure 8.10: Color accuracy at different distances from the car.

Figure 8.10 shows the accuracy in the color estimate at different distances from the car center with the projection sensor fusion algorithm. The blue and yellow bars represent the accuracy of the color estimate when classifying that color of a cone. The yellow bars in Figure 8.10 are overlaying the blue bars which is why they are darker in some places.

YOLO Sensor Fusion



(a) Ljungbyhed Raceway.

(b) E-Building Parking Lot.

Figure 8.11: Hit rate and precision over distance.

Table 8.11: Mean hit rate and precision.

Location	Hit Rate	Precision
Ljungbyhed Raceway	0.517	0.999
E-Building Parking Lot	0.685	0.999

Figure 8.11 and Table 8.11 show the results when adding the YOLO sensor fusion algorithm to the cone estimates computed by the novel ground removal with String Clustering cone detection algorithm.

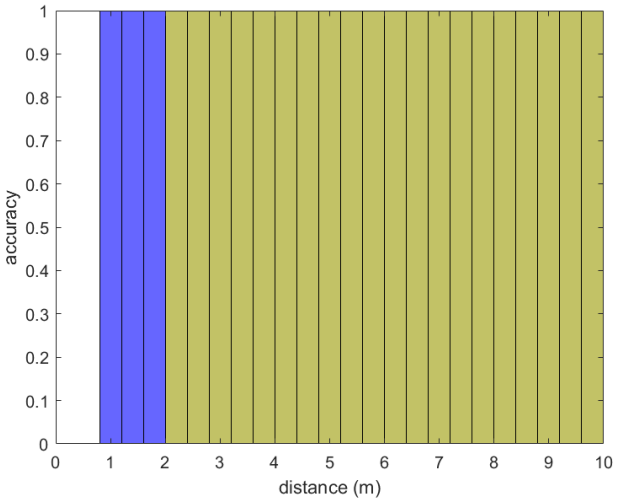
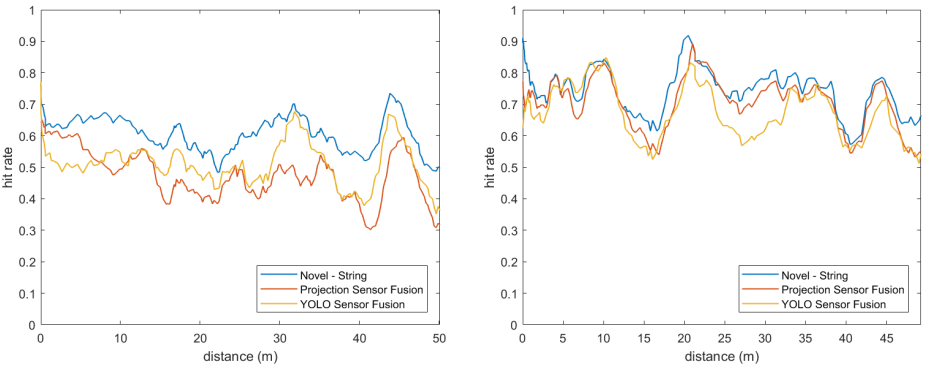


Figure 8.12: Color accuracy at different distances from the car.

Figure 8.12 shows the accuracy of the color estimate for the YOLO sensor fusion. We see here that the color estimation has a perfect accuracy for both colors. This is because we only look at cones that the algorithm managed to make an association in camera space.

Comparison

Comparison is not only done between the sensor fusion approaches, but also between the novel ground removal and String Clustering algorithm with only the cones that fall within the camera field of view and are also within 10 meters of the car. This is to show the improvement in precision when adding sensor fusion.



(a) Ljungbyhed Raceway.

(b) E-Building Parking Lot.

Figure 8.13: Comparison of moving average hit rate.

Table 8.12: Mean hit rate and precision at Ljungbyhed Raceway.

Ground Removal	Clustering	Sensor Fusion	Hit Rate	Precision
Novel	String	None	0.617	0.994
Novel	String	Projection	0.467	0.999
Novel	String	YOLO	0.517	0.999

Table 8.13: Mean hit rate and Precision at E-building Parking Lot.

Ground Removal	Clustering	Sensor Fusion	Hit Rate	Precision
Novel	String	None	0.752	0.921
Novel	String	Projection	0.702	0.956
Novel	String	YOLO	0.685	0.999

A video showing the perception using YOLO Sensor Fusion can be found here [*YOLO Sensor Fusion — E-Building Parking* 2021].

9

Discussion

9.1 Ground Removal

The results of the ground removal algorithm can be seen in Figure 8.6 and Tables 8.7 and 8.8. We see that the performance increases with the novel ground removal method is negligible at Ljungbyhed Raceway but substantial for the E-building Parking Lot data. This is because the ground at Ljungbyhed Raceway is very smooth and almost perfectly flat, while the parking lot outside of the E-Building on the other hand is uneven and bumpy. This is where the adaptive nature of the novel ground removal algorithm shines.

9.2 Clustering

Again looking at Figure 8.6 and Tables 8.7 and 8.8, we see the results of the clustering algorithms and can conclude that they have very similar hit rates and precision for both data sets. When considering the execution times in Figure 8.7 and slope estimate in Table 8.9, we see an almost 20-fold increase in performance with a small difference in clustering performance.

9.3 Color Classification with LiDAR

The results from the color classification are seen in Figure 8.8. We see that the accuracy for the blue cones is quite good up until 4 meters, and then quickly deteriorates to no accuracy over 5 meters. This is because color estimation can only be done when 3 or more LiDAR rays hit the cone.

The classification is generally worse for the yellow cones, which is because of the black strip not returning enough of the monochromatic light from the LiDAR.

9.4 Cone Classification with Camera

The results from Figure 8.13 and Tables 8.12 and 8.13 show the performance differences when adding sensor fusion into the mix. Both sensor fusion approaches lower the hit rate compared to just using the LiDAR cone detection, but also increases precision. The increase in precision is more noticeable at the E-Building parking lot, since the surrounding area provided way more clusters that were not part of the ground.

9.5 Color Classification with Camera

Results from Figures 8.10 and 8.12 show the color accuracy of the two proposed sensor fusion algorithms. We can instantly see how the YOLO sensor fusion algorithm outperforms the projection sensor fusion with a perfect color accuracy over the tests.

10

Conclusions

10.1 Final Result

The goal of this Master Thesis as defined in Chapter 1 was to create a robust perception system for the Lund Formula Student 2021 racing car, to provide a stable footing for the future team. The perception system used when the car was finished in August 2021 was the novel ground removal approach together with String Clustering and the YOLO sensor fusion. The LiDAR color estimation was not used, as a wrongful color estimation could have more devastating outcomes for the car's performance than not having a color estimation at all.

The final resulting perception system provides a precise perception on uneven terrain because of the novel ground removal approach modelling the ground as several interconnected planes, and a faster and scalable clustering method up to par with the ones used by other Formula Student teams. The sensor fusion approach does not require perfect calibration and is using all of the advantages the camera has over the LiDAR for cone detection.

The full LiDAR cone detection algorithm had an average execution time of 17 ms, from grabbing the LiDAR data to outputting cone locations, over both datasets on the hardware provided. The sensor fusion part of the algorithm, had an average execution time of 60 ms, making a total of 77 ms average for the entire pipeline. These measurements were done on the available hardware, and we therefore reached our goal of 100 ms and using every LiDAR scan for the perception.

A video showing the final perception and car driving can be found here [*Final Ljungbyhed Drive* 2021].

10.2 Ground Removal

As seen in the results, the novel ground removal only shows an improvement at the E-building parking lot, whereas at Ljungbyhed Raceway the difference of the two algorithms is negligible. Since most Formula Student competitions are held at raceways similar to the one at Ljungbyhed Raceway where the environment is "beneficial", one can argue that the novel ground removal approach is unnecessary. However, to counter this point, it has been of great use to be able to drive and test the car in environments where the ground is bumpy. Overall, it is a more robust algorithm, which is what this thesis aimed at achieving.

10.3 Future Work

The perception system proposed in this thesis can be improved in many ways. One major aspect is that it is entirely dependent on the LiDAR cone detection doing its job and does not take cones that only the camera sees into account. This would be much improved by choosing a LiDAR of higher resolution, which has been made possible because of the computationally inexpensive clustering method.

Another way the perception system can be improved is finding a more practical and accurate way of calibrating the extrinsic parameters of the projection matrix for the camera.

Bibliography

Ahlqvist, J., O. F. Hedbrant, and M. G. Nilsson (2019). "Slamdog Millionaire - An autonomous vehicle for cone demarcated tracks". *Rapport i Projeketkurs i Reglerteknik, LTH, Lunds Universitet*. URL: https://drive.google.com/file/d/1OZ59cRWaIJwGs1-E36P3FS3z8ZTz2DM3/view?usp=sharing.

Basler ace acA1920-40uc (2021). URL: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca1920-40uc/ (visited on 2021-10-20).

Bentley, J. L. (1975). "Multidimensional binary search trees used for associative searching". *Commun. ACM* **18**:9, pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007.

Böiers, L.-C. (2010). *Mathematical Methods of Optimization*. Eng. Studentlitteratur AB. ISBN: 978-91-44-07075-9.

Bolles, M. A. F. R. C. (1981). "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*. URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a460585.pdf.

Camera Calibration - ROS Wiki (2021). URL: http://wiki.ros.org/camera_calibration (visited on 2021-09-21).

EUFS Simulator (2021). URL: https://gitlab.com/eufs/eufs_sim (visited on 2021-10-11).

Final Ljungbyhed Drive (2021). URL: https://www.youtube.com/watch?v=PcsF_VZSDYQ.

Formula Student Rules (2020). URL: https://www.formulastudent.de/fileadmin/user_upload/all/2020/rules/FS-Rules_2020_V1.0.pdf (visited on 2021-10-21).

FSG Competition Handbook (2021). URL: https://www.formulastudent.de/fileadmin/user_upload/all/2021/rules/FSG21_Competition_Handbook_v1.0.pdf (visited on 2021-10-21).

FSG: Concept (2021). URL: https://www.formulastudent.de/about/concept/ (visited on 2021-10-11).

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Jetson AGX Xavier Developer Kit (2021). URL: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit (visited on 2021-10-20).

Kabzan, J., M. de la Iglesia Valls, V. Reijgwart, H. F. C. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, A. Dhall, E. Chisari, N. Karnchanachari, S. Brits, M. Dangel, I. Sa, R. Dubé, A. Gawel, M. Pfeiffer, A. Liniger, J. Lygeros, and R. Siegwart (2019). *AMZ driverless: The full autonomous racing system*. arXiv: 1905.05150 [cs.RO].

LiDAR only peception — E-building Parking Lot (2021). URL: https://www.youtube.com/watch?v=ruPoroedVaw.

Nielsen, F. (2016). *Hierarchical Clustering*. ISBN: 978-3-319-21902-8. DOI: 10.1007/978-3-319-21903-5_8.

Point Cloud Library (PCL) (2021). URL: https://pointclouds.org/ (visited on 2021-10-11).

RANSAC Ground Removal vs Novel Ground Removal (2021). URL: https://www.youtube.com/watch?v=OmYJ1PAgGTo.

Redmon, J. and A. Farhadi (2018). *Yolov3: An incremental improvement*. arXiv: 1804.02767 [cs.CV].

Roche López, A. (2019). *LiDAR cone detection as part of a perception system in a Formula student car*. PhD thesis. UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament de Teoria del Senyal i Comunicacions. URL: http://hdl.handle.net/2117/168711.

ROS (2021). URL: http://wiki.ros.org/ (visited on 2021-10-13).

Sensors only — E-building Parking Lot (2021). URL: https://www.youtube.com/watch?v=WPpPNfMNKQA.

TensorFlow (2021). URL: https://www.tensorflow.org/ (visited on 2021-10-05).

Velodyne LiDAR Puck Datasheet (2021). URL: https://velodynelidar.com/wp-content/uploads/2019/12/63-9229_Rev-K_Puck-_Datasheet_Web.pdf (visited on 2021-10-20).

Wikimedia Commons (2021). *Lidar*. URL: https://commons.wikimedia.org/wiki/File:20200501_Time_of_flight.svg?uselang=en (visited on 2021-10-21).

Wikipedia, the free encyclopedia (2021). *Ransac*. URL: https://en.wikipedia.org/wiki/Random_sample_consensus#/media/File:RANSAC_Inliers_and_Outliers.png (visited on 2021-10-21).

YOLO Sensor Fusion — E-Building Parking (2021). URL: https://www.youtube.com/watch?v=u3YgTxhC0KE.

Zhang, Z. (2014). *Camera Parameters (Intrinsic, Extrinsic)*. Ed. by K. Ikeuchi. Springer US, Boston, MA. ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_152.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
December 2021

Document Number
TFRT-6152

Author(s)
Gustaf Broström
David Carpenfelt

Supervisor
Björn Olofsson, Dept. of Automatic Control, Lund University, Sweden
Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Robust Perception for Formula Student Driverless Racing

Abstract

Building an autonomous system for race cars requires robust and highly accurate perception running in real time. This thesis proposes a novel ground removal strategy for 3D LiDAR perception, modelling the ground as several planes, and a novel clustering method for LiDARs that sweep the scene in a predefined pattern resulting in a 20-fold performance increase over clustering methods commonly used for this problem.

To weed out spurious information produced by relying only on LiDAR perception, two sensor fusion methods using a camera are introduced and evaluated, one using the immensely popular YOLO network. All algorithms were evaluated in real world scenarios using a fully functioning Formula Student driverless vehicle built by the Lund Formula Student 2021 team.

Keywords
3D LiDAR, autonomous racing, sensor fusion, Formula Student

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-62

Recipient's notes

Security classification