OXFORD BROOKES UNIVERSITY

Faculty of Technology, Design and Environment

MASTER OF SCIENCE DISSERTATION

Dissertation Title:	“3D Object Detection for Autonomous Vehicles”
Surname:	Do
First Name:	Mai
Student Number:	19132761
Supervisor:	Peter Ball
Dissertation Module:	DALT 7017
Course Title:	MSc Data Analytics
Date Submitted:	

Statement of Originality

Except for those parts in which it is explicitly stated to the contrary, this project is my own work. It has not been submitted for any degree at this or any other academic or professional institution.

Signature of Author		Date	

Regulations Governing the Deposit and Use of Master of Science Dissertations in the School of Engineering, Computing and Mathematics, Oxford Brookes University.

1. A copy of dissertation final reports submitted in fulfilment of Master of Science course requirements shall normally be kept by the School.

2. The author shall sign a declaration agreeing that, at the supervisor's discretion, the dissertation will be submitted in electronic form to any plagiarism checking service or tool.

3. The author shall sign a declaration agreeing that the dissertation be available for reading and copying in any form at the discretion of either the dissertation supervisor or in their absence the Programme Lead of Postgraduate Programmes, in accordance with 5 below.

4. The project supervisor shall safeguard the interests of the author by requiring persons who consult the dissertation to sign a declaration acknowledging the author's copyright.

5. Permission for anyone other than the author to reproduce in any form or photocopy any part of the dissertation must be obtained from the project supervisor, or in their absence the Programme Lead of Postgraduate Programmes, who will give his/her permission for such reproduction only to the extent to which he/she considers to be fair and reasonable.

I agree that this dissertation may be submitted in electronic form to any plagiarism checking service or tool at the discretion of my project supervisor in accordance with regulation 2 above.

I agree that this dissertation may be available for reading and photocopying at the discretion of my project supervisor or the Programme Lead of Postgraduate Programmes in accordance with regulation 5 above.

Signature of Author		Date	

Table of Contents

1. Abstract

Safe navigation along a road in the presence of other vehicles, cyclists, pedestrians, and other objects is an essential requirement for autonomous vehicles. An autonomous vehicle must be able to recognise and locate relevant objects around it. Developers need to make appropriate design choices on the vision sensors to be used, and the object detection and localisation algorithms to achieve an effective and safe solution.

In Formula Student Artificial Intelligence competition 2021, Oxford Brookes Racing Autonomous team decided to use only Cameras for object detection. However, this work provides a review of such choices and develops a Camera - LiDAR fusion solution for the perception system. I also use a camera for 2D object detection, but I combine LiDAR as a tool for measuring distance to those objects.

By doing measurement experiment with Camera and LiDAR compared to actual distances which is measured manually by tape measure, I found that the average distance error for Camera is 63cm, and for LiDAR is 8cm. Fusing the sensor data together could improve the object localisation accuracy from 63cm error (only Cameras) to 8cm error for the objects in the range of 10 meters.

However, to do the fusion method, we need an accurate camera LiDAR calibration matrix. Another camera LiDAR calibration experiment is carried out showing that OBR RS-16 LiDAR does not have enough channels to get an accurate calibration matrix for fusion method. In this way, the fusion method is tested on KITTI dataset, which is a benchmark dataset for 3D object detection for autonomous vehicles.

2. Acknowledgements

I would like to thank my supervisors Dr. Peter Ball and Dr. Andrew Bradley for their guidance. I am also grateful to Oxford Brookes Racing Autonomous for giving this project a chance to be applied on a real car and providing some insight, data, and analytics. I would like to extend my grateful thanks to all the Oxford Brookes Racing Autonomous team members, especially Ivan Fursa, who supported me a lot to complete this dissertation.

3. Introduction

3.1. Autonomous Vehicles

The rapid advancement of technology has significantly increased the level of automation in road vehicles in recent years. Compared to conventional vehicles, autonomous vehicles promise considerably improved universal access, safety, efficiency, convenience, and cost savings. In a short time, car manufacturers have moved from advertising Advanced Driver Assistance Systems or automated parking assistance systems to commercial vehicles that offer a high level of automation on roads. This advancement would not have been possible without the improvements in artificial intelligence algorithms, which allow the control system to extract knowledge from large amounts of real-world driving data and develop a more vigorous control and perception system. An autonomous driving system must detect essential objects around it, like drivable areas, boundaries, other vehicles, pedestrians, etc. and estimate the location relative to the autonomous system. Such cars should be able to move amongst other road users without human intervention at least for a limited time and in limited spaces [1].

3.2. 3D Object Detection with Vision Sensors

To perform object detection and localisation, the system can use vision sensors e.g., video cameras and distance measurement sensors like radars and LiDARs. Some systems rely on a single type of sensor, and some combine different types [2]. The problem of 3D object detection consists of several subproblems: recognising an object in the unstructured data (i.e., images and point clouds), classifying the object, measuring the distance to it (relative to the car's origin).

3.3. Formula Student Artificial Intelligence - IMechE

An autonomous system is being developed by Oxford Brookes Racing (OBR) Autonomous project. They are developing the software and hardware for a racing car with the aim of completing a racing track individually as fast as possible. Formula Student Artificial Intelligence (FS-AI) has been introduced to challenge student teams to develop an Artificial Intelligence driver capable of controlling a purpose-designed

Formula Student car through a series of racing challenges [3]. The perception system requires it to detect and locate traffic cones that define the racing track. By correctly locating them relative to itself, the car can plan a path between the track boundaries and move along the track and complete the whole circuit.

The primary race consists of completing ten laps faster, around a track which is defined by small cones (228 × 335 mm). The left and right boundaries are marked with blue and yellow cones respectively. The racetrack is a closed-circuit up to 500 meters in length, minimum track width is 3 meters, and cones in the exact boundary line are spaced up to 5 meters apart. According to the FS-AI Ruleset, the racetrack might contain straight lines, chicanes, hairpin, multiple turns, and decreasing radius. This work aims to enable a race car to autonomously complete multiple laps of an unknown racetrack without any human intervention and in a single attempt.

3.4. Problem Statement

This task specification sets particular requirements on the performance of the vehicle. Since the cones are placed 5 meters apart on a 3 meters' wide track, it is important to be able to accurately locate the cones that are at least 5m away. The demand for high racing speed might require an even larger lookahead distance [4]. Correct detection and localisation of cones is necessary for a working SLAM (simultaneous localisation and mapping) solution that requires a reconstruction of the map of the track to plan a racing line across the whole track [5].

Currently, OBR Autonomous uses a stereo camera for both detection and localisation of objects [6]. This approach might limit the efficiency of the system due to low object localisation accuracy *(figures 3.41, 3.4.2 and 3.4.3).*

Figure 3.4.1. 2D cone bounding boxes by YOLO method. Source: OBR Autonomous

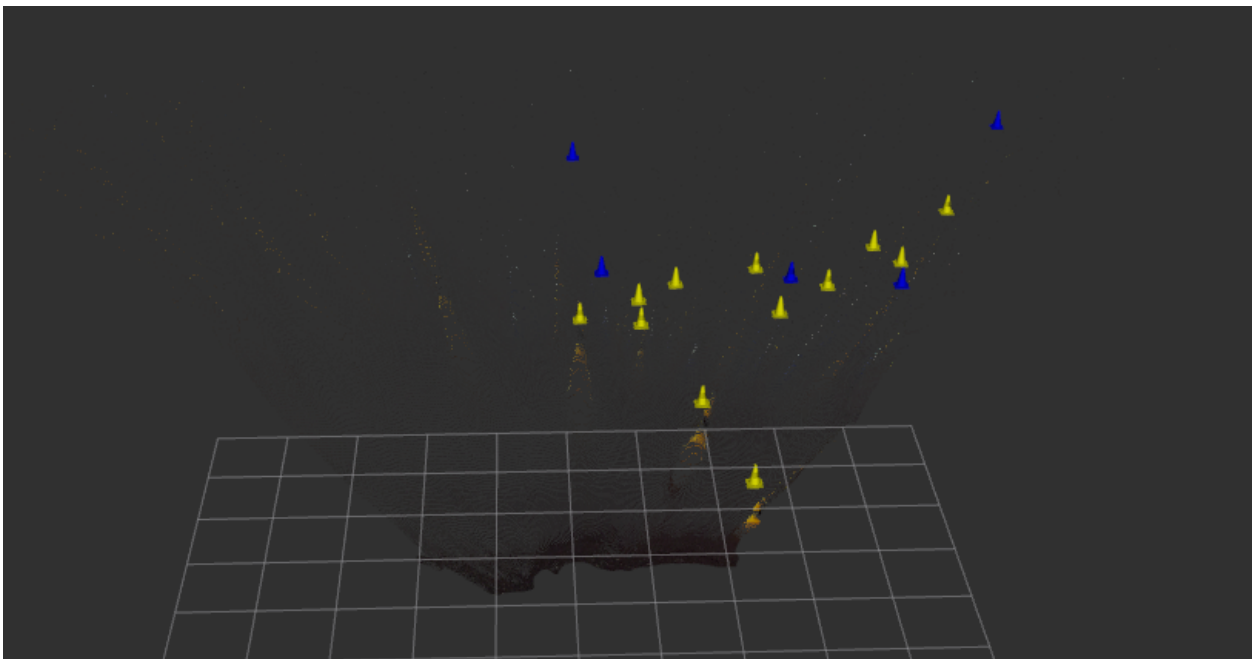


Figure 3.4.2. Visualisation of 3D cones on ROS2. Source: OBR Autonomous.

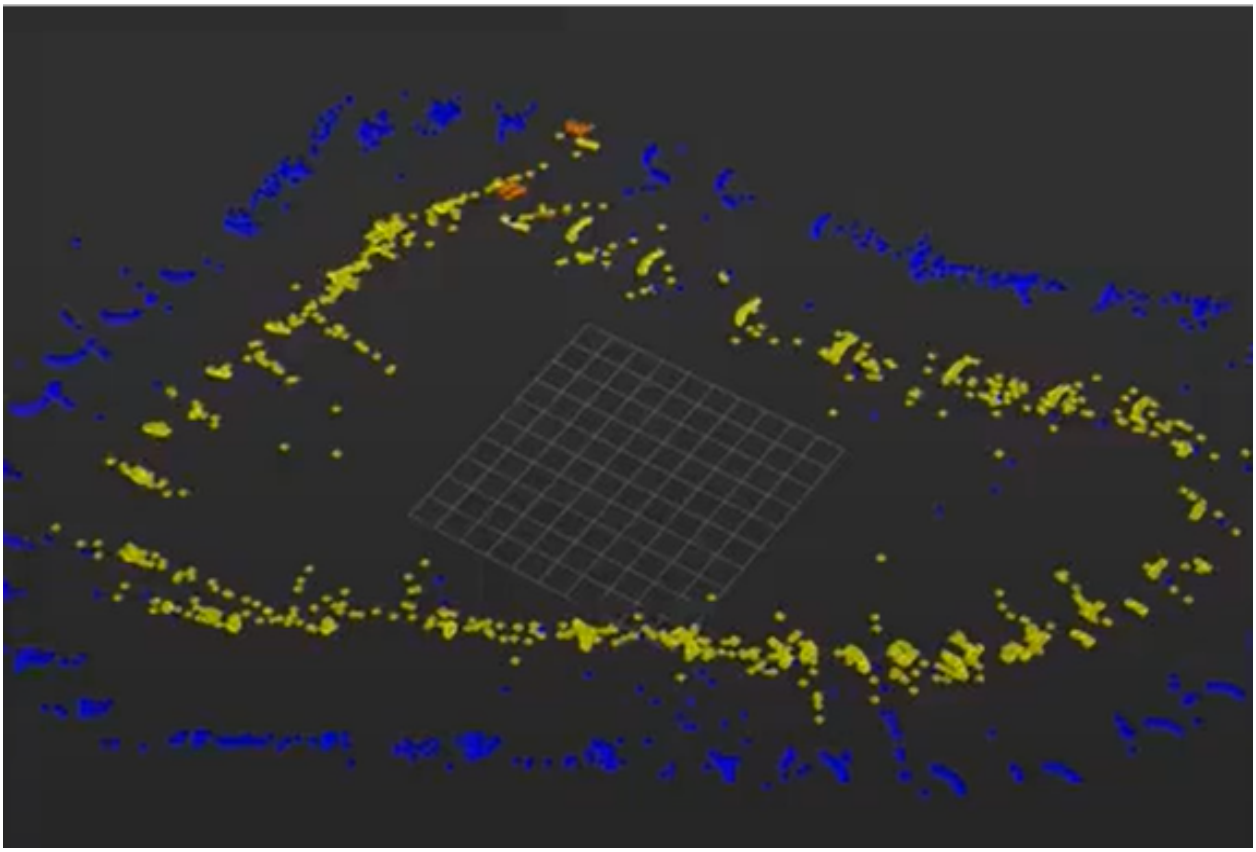


Figure 3.4.3. The cones in Gazebo simulation environment. Source: OBR Autonomous

Figure 3.4.1 shows output of 2D bounding boxes of cones detected during a Formula Student competition using YOLO. In *figure 3.4.2*, those cones are visualised with 3D positions based on the stereo camera point cloud by Rviz2 software in ROS2. The position estimations do not resemble the real layout of the track producing a large error. In *figure 3.4.3*, blue, yellow, and orange cones are localised after a run of an OBR Autonomous system in a Gazebo simulation environment. The camera behaviour is simulated.

Judging visually, the accuracy of localisations is low. The team has access to a 16-channel LiDAR sensor and two radar sensors that might potentially improve the perception subsystem. However, to make an informed decision on what method is most effective for localisation, a comparative study of camera-based and LiDAR-based localisation will be conducted in this report.

The main aim of this work is to study and develop an effective 3D object detection and localization solution using the available sensors.

4. Literature Review

The most commonly used sensors for autonomous cars are Cameras, LiDARs and RADARs [6,7]. In this section, I will review the sensors and compare their advantages and disadvantages. Then, I will look at the available 3D object detection methods for each sensor and make a decision on which method should be used for the OBR Autonomous car.

4.1. Camera Sensor Overview

Video cameras return sequences of image data that can be effectively used for object recognition. The most common way of annotating a data is with bounding boxes (where an object of interest is enclosed into a rectangle that labels all included pixels) and segmentation map (where each pixel relating to the object is labelled). The detected objects, however, are only localized in a 2D space of the image. Extra effort is required to infer the 3D location of the object relative to the camera.

Using the output from an object detector (ex. bounding box) and knowing the extrinsic and intrinsic parameters of a camera (i.e., focal length, sensor dimension, and lens distortion coefficients) it is possible to calculate the object's 3D position. There are two problems with this approach. Firstly, if the object detector returns a bounding box with a slightly smaller or bigger bounding box, the distance measurement suffers. Object detectors might detect objects correctly but drawing a perfect bounding box is a challenge. Secondly, it can be difficult to do cameras calibration and calculate its precise parameters.

These sensors combine 2 cameras. Knowing the offset between the cameras it is possible to process both images at the same time, calculate the disparity between them and calculate the distance to every pixel of an image. This approach has several challenges. It is computationally intensive to calculate the distance this way and it leads to a latency. Considering the latency from the object detector (e.g., 0.045 seconds on the

OBR Autonomous CQ60G In-CAR PC. Source: OBR Autonomous), the additional latency of this type of localisation can add a lot of time to the processing. Because short latency is essential for safety critical calculations for real-time systems as autonomous driving, this is problematic. The second issue is that the performance may degrade in certain situations: e.g., different lighting in both images, different occlusion in both images, etc. In general, this approach is subject to error, and therefore might not be adequate for real time systems that move at high speed [8].

The camera that is used on the Oxford Brookes Racing Autonomous is a ZED stereo camera by Stereolabs [9]. It provides software that calculates distance information. Following experimental research on the evaluation of this sensor *(figure 4.1.1)* [10], the RMS error increases fast with increasing in distance. Objects that are 10m away will be measured with RMS error of about 0.2m for full HD images, this will help for planning the movement from long distance. However, for the more distant objects, the quality will decrease much more.

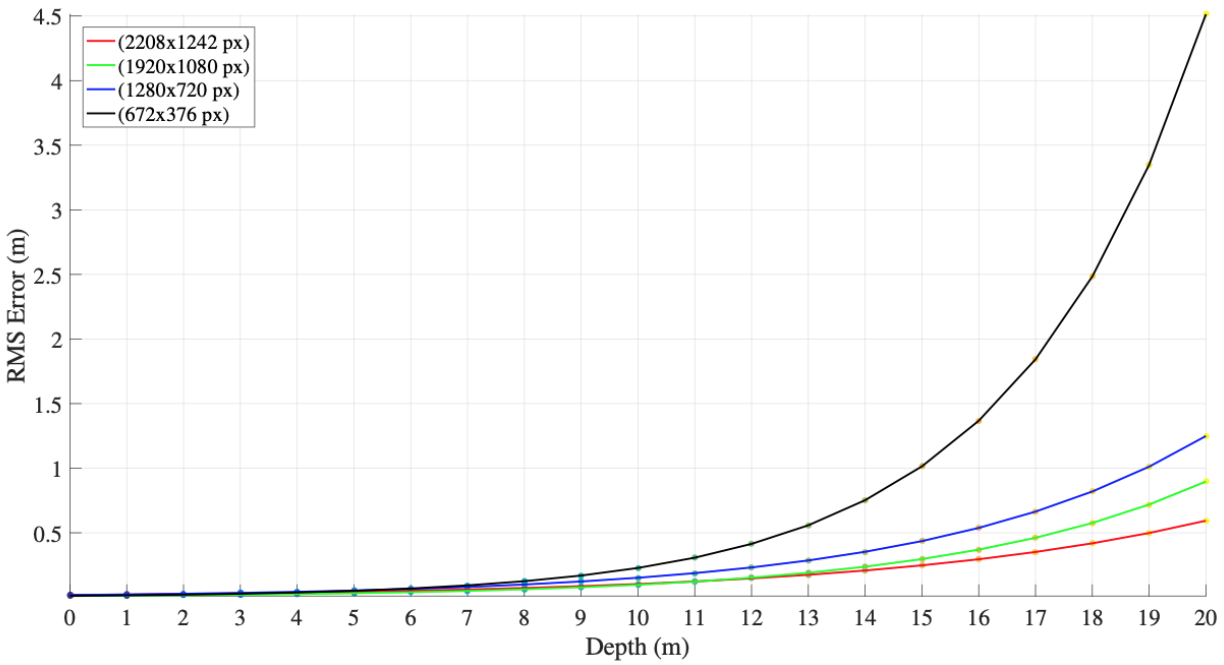


Figure 4.1.1. Expected RMS error for 1-20 ZED resolutions (px). Source: Depth Data Error Modeling of the ZED 3D Vision Sensor from Stereolabs [10]

Despite the figures from this research, however, the experimental data collected by OBR Autonomous students suggests a much lower quality in a real-world scenario *(figure 4.1.2).* The error in this case is drastically larger. This might be because of the different experimental settings and the objects (cones are small objects) to which the distance was measured.

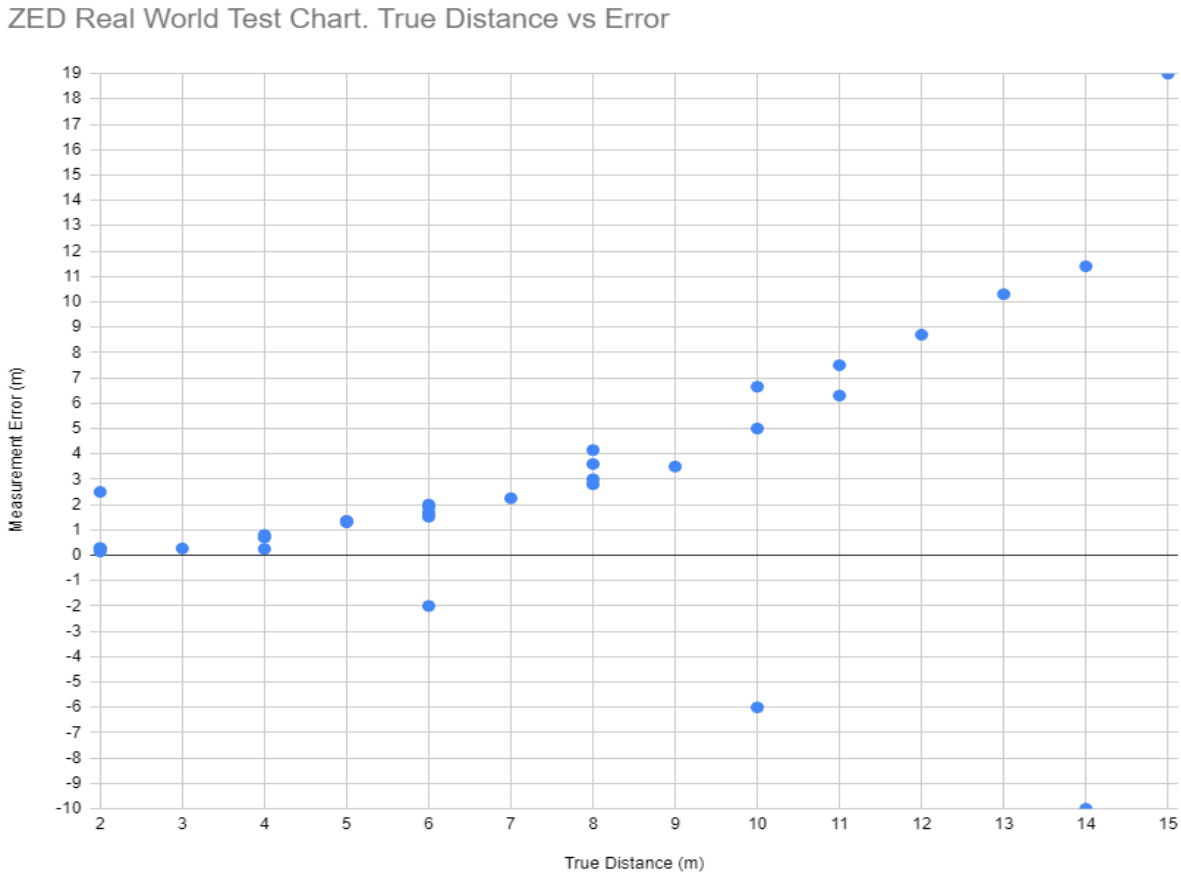


Figure 4.1.2. Experimental data for zed camera accuracy from OBR Autonomous. Error is calculated as error = measured distance - true distance. (Appendix B)

Such a big difference in the experimental results requires further analysis of the accuracy.

4.2. LiDAR Overview

LiDAR refers to a light detection and ranging device, which scans laser beams across the field of view and transmits millions of light pulses per second which reflect of

objects. The Lidar calculates the round-trip time of these pulses and builds a picture of the objects in the field of view. After calculating the latency between the start and reception of the beam, it's possible to calculate how far away the object that was hit by the beam is located. With its rotational axis, it can create a dynamic, three-dimensional map of the environment. LiDAR is the heart of object distances for most existing autonomous vehicles [11]. The points returned by the LIDAR in a natural setting, however, are never perfect. Scanning point sparsity, missing points, and disorganized patterns are problems when dealing with LiDAR points. The surrounding environment contributes to the difficulty of perception by presenting random and irregular surfaces. Raw output from the RS-16 LiDAR is a stream of 3D point positions (i.e., a point cloud). The manufacturer specified the sensor with up to 2cm accuracy of measurements [12].

Because of its ability to identify small plastic cones, a LiDAR was chosen over a Radar, as this is not effective at identifying smaller objects [63]. The type of LiDAR sensor is chosen based on physical parameters such as horizontal, vertical resolution, and field-of-view. The sensor has a horizontal field-of-view of 360 degrees. It means that the vehicle is able to see most of the racing track and produce a detailed map for it. However, the active field-of-view on the OBR Autonomous car is 270 degrees because the sensor plate and the vehicle obscure the remainder. The vertical resolution is the essential parameter in this application since it restricts the number of returns per cone, directly proportional to the distance at which cones can be observed *(figure 4.2.2)*. The RS-16 LiDAR that OBR Autonomous uses has 16 lasers, and the vertical field of view (FOV) is 30 degrees *(figure 4.2.1) [54]*. This is smaller compared to the camera's 60 degrees FOV and restricts the sensor from detecting closer cones [12].

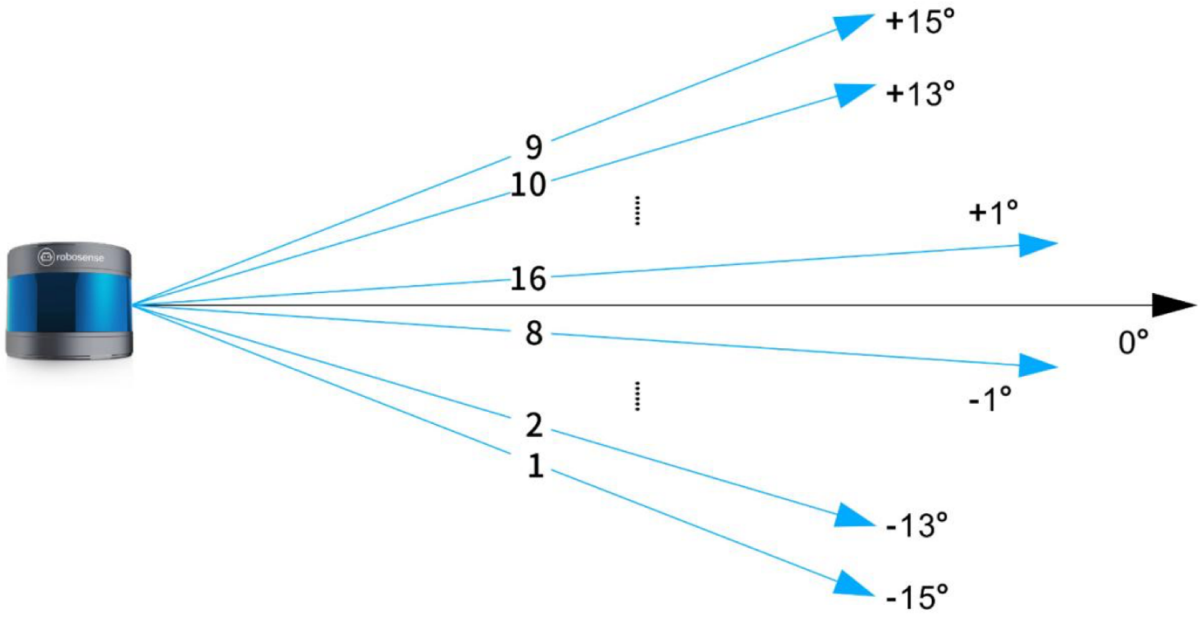


Figure 4.2.1: RS-LiDAR-16 Laser Channels and Vertical Angles. Source: RS-LiDAR-16 User Manual [54]

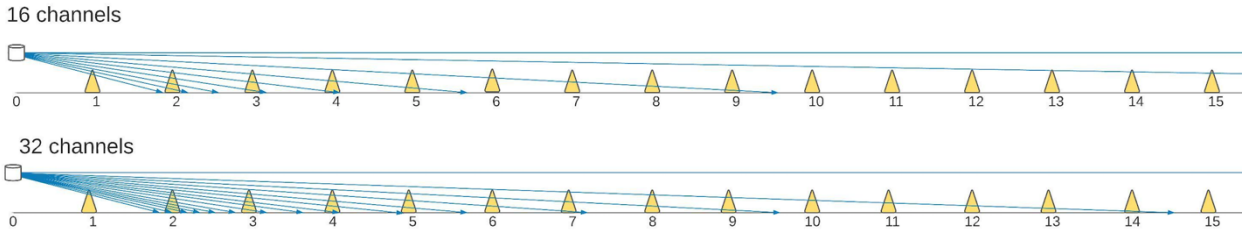


Figure 4.2.2. Theoretical coverage of the 16 channel LiDAR and 32 channel LiDAR.

I drew a graph *(figure 4.2.2)* showing theoretical coverage of the 16 channel LiDAR and 32 channel LiDAR. Lower axis denotes cones located in 1-meters' intervals. Notice that the closest cone is missed due to a lower vertical FOV of the LiDAR. Also, the cone that is 10 meters away is not perceived by a 16-channel sensor. This, however, can be fixed if a LiDAR with higher resolution is used.

Figure 4.2.3 shows performance analysis of 10 models LiDAR, there is no RS-16 LiDAR but from performance of RS-32 LiDAR, which is missing points on car object, we can see RS-32 LiDAR seems to be unacceptable for cone detection.

LiDAR	Sec. Reflections	Int. Aberrations	Blooming	Missing Points	Line Visibility
VLS-128	No	Yes, <25m	No	No	Very Good
Pandar64	No	No	Yes, small	No	Acceptable, distorted
HDL-64	No	Yes	No	Yes	Poor
OS1-64	No	Yes	Yes, large	Yes, many	Poor
Pandar40p	No	No	No	Yes, few	Acceptable, distorted
VLP-32c	Yes, <25m	No	No	Yes, few	Acceptable
HDL-32	No	No	Yes, small	Yes, few	Poor
RS-LiDAR-32	No	No	No	Yes, car	Very Poor
OS1-16	Yes, <15m	Yes	Yes, large	Yes, many	Very Poor
VLP-16	No	No	Yes, small	Yes, car	Acceptable

Figure 4.2.3. Qualitative observations of different detrimental effects each LiDAR. Source: Performance Analysis of 10 Models of 3D LiDARs for Automated Driving [55]

4.3. LiDAR Data Representation

Point clouds, features, and grids are the three common representations of the points [13].

The point cloud-based method uses the raw sensor data and further processing. This approach gives a better environmental representation but increases the processing time and reduces the efficiency of memory. To mitigate this, a voxel-based filtering mechanism is usually used to reduce the amount of raw point cloud, e.g., [14,15]. Voxel-based methods select a sub-scene to process and split it into a 3D grid of voxels (volumetric elements) [16] to apply 3D convolutions, or a 2D grid of pillars [17,18,19] to apply 2D CNNs [20].

Feature-based approaches start by extracting parametric aspects from the point cloud and then using those features to represent the environment. Lines [21] and surfaces [22] are two features that are often used. This technique uses the least amount of memory; however, it is frequently overly abstract, and the accuracy depends on the point cloud since not all environmental features can be accurately reproduced by the collection as mentioned above.

Grid-based method divides the area to small grids which are filled with information from the point cloud, forming a point neighbourhood [23]. This technique is memory-

efficient and does not rely on predetermined features, as stated in [24]. However, it is not straightforward to determine the size of the discretization. An adaptive octree a tree data structure was constructed in [25] to guide division from coarse to fine grids.

5. Object Detection for Camera and LiDAR (3D and 2D)

Cameras and LiDAR return data that is different in structure, nature, and quality, both data types can be used to detect objects. There are several LiDAR-based object detectors, the localization of objects is high quality. However, their performance is lower than the performance of camera-based approaches in object detection. Both sensors have their advantages and disadvantages. Therefore, several researchers have proposed combining the outputs from the sensors to benefit from the advantages of both.

5.1. LiDAR-only Object Detection

VoxNet, a 3D convolutional neural network that classifies point clouds (in occupancy grid or volumetric representation), was proposed in [26]. In [27], the volumetric based 3D CNN was improved by introducing auxiliary learning tasks on part of an object and combining data augmentation with multi-orientation pooling. In [28], from raw CAD data, a 3D Constitutional Deep Beliefs Network was suggested to acquire the distribution of complex 3D shapes through different object categories and arbitrary poses. An additional issue of applying LiDAR-based object detection for racing is classification between identically shaped objects based on colour. LiDAR does not work with colour data but with intensity. Differentiating between yellow and blue is difficult, and therefore there might be a high chance of misclassification.

5.2. Camera-only Object Detection

For deep learning systems, the usual workflow is to produce a collection of proposal bounding boxes around the image then send the proposal boxes through the CNN network to classify and fine-tune bounding box positions. There are two frameworks of bounding box proposal, the region and the regression/classification:

The region proposal based framework predicts bounding boxes directly from locations of the topmost feature map after obtaining the confidences of underlying object categories

[56]. The first deep learning that combined the bounding box and detection into a single network and complete a training process was Faster-RCNN [29]. MS-CNN is a unified multi-scale deep learning network that the authors in [30] proposed. In [31], the authors trained the deep learning network to recover 2D and 3D information from the 2D image (Sub-CNN). However, this method has a high computational cost.

One-step frameworks based on global regression/classification, mapping straightly from image pixels to bounding box coordinates and class probabilities, can reduce time expense [56]. One of the faster alternatives for image-based detection is YOLO ("You Only Look Once"). YOLO splits an image into cells and predicts and classifies an object for each of those cells. Then, the most likely candidates are chosen to produce a final prediction. This architecture produces competitive results and is capable of doing it in real time [32,33,34].

5.3. Sensor Fusion

According to [35], the fusion of the 2 sensors improves 3D object detection ability. Detections are particularly better for small far-away objects compared to single sensor approaches. This work identified 3 strategies for fusing the sensors together:

2D to 3D [36,37,38,39]: requires performing the 2D object detection on an RGB image [40,29] and then converting the resulting 2D bounding boxes into 3D bounding boxes within the LiDAR data.

Proposal fusion [41,42]: consists of, first, proposing potential 3D bounding boxes for objects based on LiDAR data, and, second, pooling features from each sensor to decide on the most likely proposals.

Dense fusion [43]: requires projecting features from the 2D camera data into a common 3D space with the LiDAR features. Then, the 3D detections are performed on this new 3D data.

Despite the first (2D to 3D) method not using LiDAR data to identify the objects but just measuring distance, this method is potentially faster because it avoids frequent feature pooling and voxel-based computations which are required by the other 2

methods. Speed is essential for the autonomous racing task. Considering the sparsity of the 3D data that the available RS LiDAR-16 sensor is producing, this data might not benefit the overall detection quality a lot if the proposal and dense fusion are used. The availability of 3D annotated data is another challenge that is faced when using either proposal or dense fusion. With the "2D to 3D" method, it's possible to use the existing annotated camera image data. This means that OBR Autonomous could equip the sensor fusion without spending long hours on 3D data annotation.

5.4. Datasets

To train the neural network that detects and localizes objects, a large high-quality dataset is required. One of these datasets is KITTI [44]. It is a benchmark dataset for the visual odometry tasks, SLAM and 3D object detection. The dataset comprises more than 200,000 objects annotated in 3D and contains synchronised Velodyne LiDAR data and stereo camera data. This dataset will provide sufficient data for tests.

6. Methodology

OBR autonomous is using ZED stereo camera working in 1920x1080 resolution, RS-16 LiDAR with 16 channels, set up to spin at the speed of 20Hz.

6.1. Sensor Experimental Analysis

The problem with the available LiDAR sensor is its small resolution and significant amount of noise. The annotations and graphical user interface were provided by Supervise.ly [7], a free data annotation web-service for images and point clouds. I used Supervise.ly to visualise a LiDAR point cloud data which was collected from an OBR Autonomous test car on a racing track *(figures 6.1.1).*

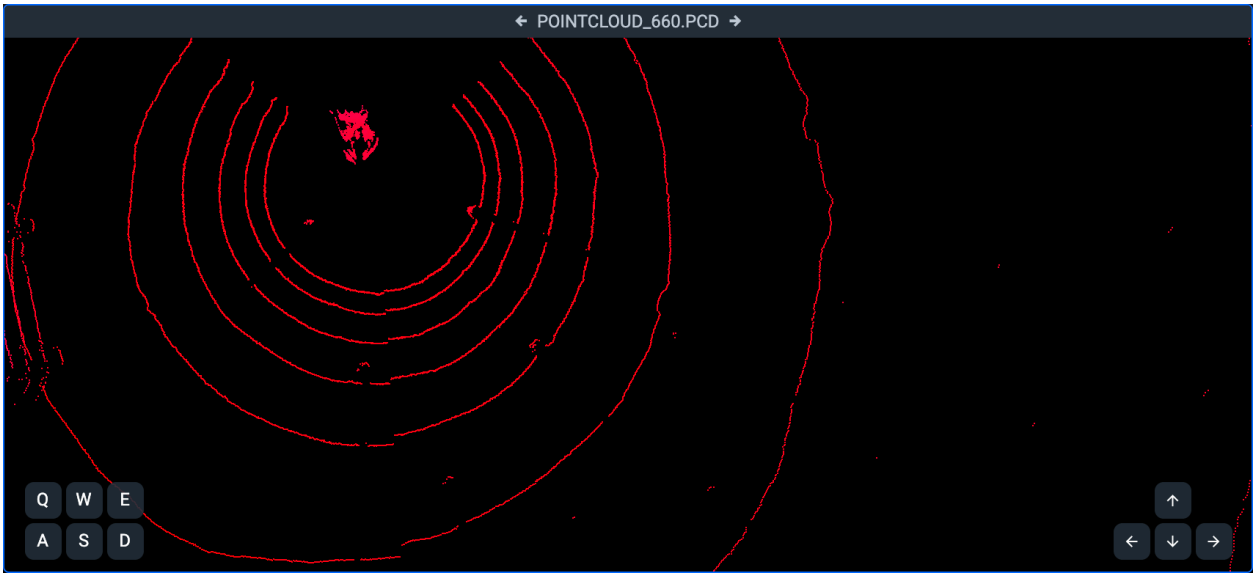


Figure 6.1.1. Visualisation of a LiDAR point cloud data

In *figure 6.1.2*, the car is labelled by annotation tool with tag *Car*, three example cones in different distances are tagged with *Cone 1, Cone 2, Cone 3.* I will zoom out those cones to clarify how the cones look in point clouds.

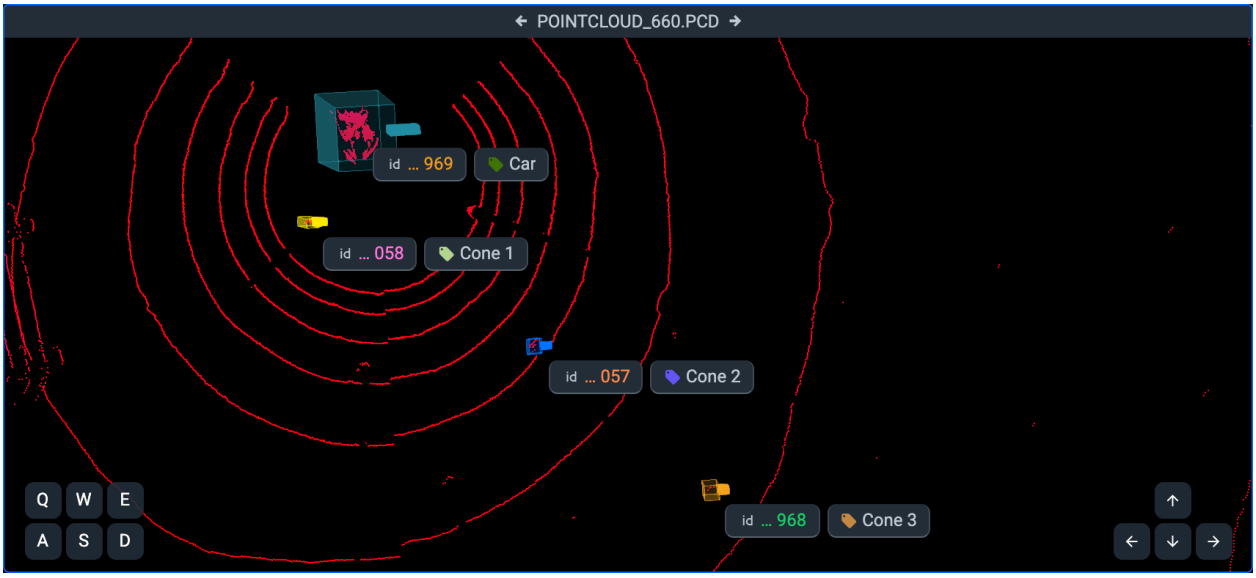


Figure 6.1.2. Visualisation of a LiDAR point cloud data

Three cones are zoomed on top and side views in *table 6.1.3*, information about distance to the car is given by geometry information from its belonging point, number of points for each cone is counted manually.

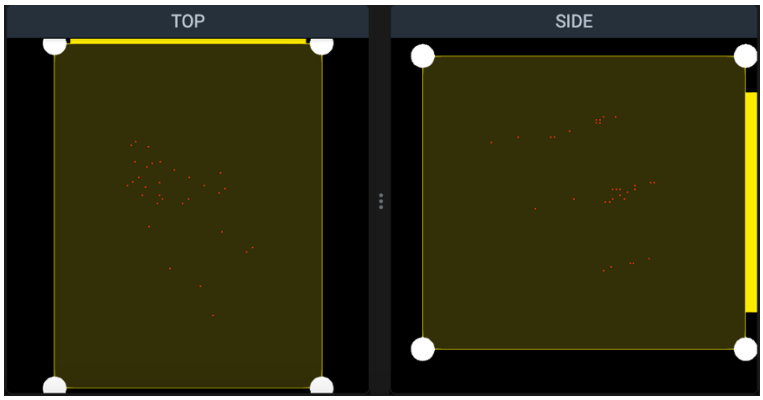
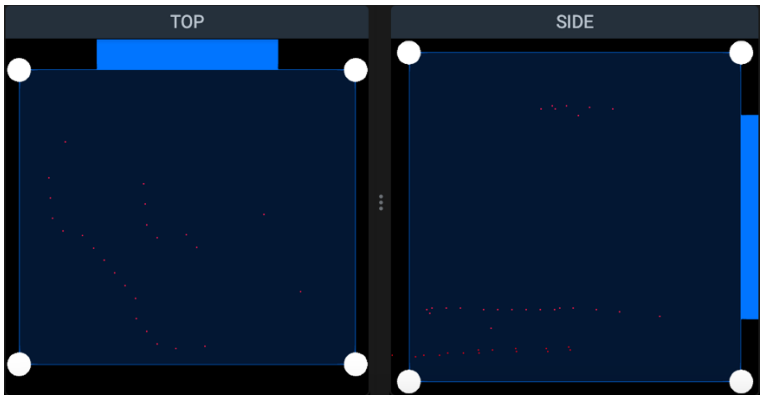
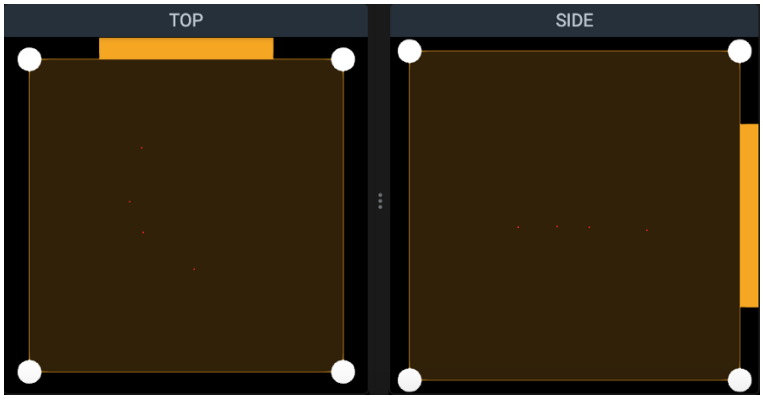
Cone	Top and side views of the cones found in a LiDAR point cloud	Distance to the car (m)	Number of points
Cone 1		1.417	31
Cone 2		4.361	24
Cone 3		8.528	4

Table 6.1.3. Zoomed-in top view and side view of each of the cones denoted in figure 7 using Supervise.ly

From the figure and table above, notice how the number of points describing a cone object decreases as the distance increases. For a cone that is 8.5m away, only 4 points are present, all in one laser level. Cone that is 4.4m away, 2 rows of points are available and it's quite clear that it's a cone. The closer cone, despite having a fair number of points, is very noisy. The shape of the point cluster does not resemble a cone shape.

Since the RS-16 LiDAR returns around 300,000 points per second [12], lowest compared to other LiDARs (RS-32 with scan rate 600,000pts/s or RS-Ruby with 2,304,000pts/s, more information [57]), the cloud is quite sparse, point cloud representation will not drastically decrease the efficiency (as fast as) compared to other methods (considering the computational cost of converting from one representation to another). However, feature-based approaches might not work well because of the cloud sparsity. There is not enough data to suggest a line of a surface. Grid-based could potentially work but the advantages over the raw data representation are not as clear in this case. LiDAR only method in this case (sparse point cloud RS-16 LiDAR) seems to be not enough to detect the small cones.

In order to evaluate which sensor should be used for object localisation, an experiment will be conducted. Using the OBR Autonomous sensor plate *(figure 6.1.3)*, some experimental data with cones will be recorded on both sensors. Since the software that comes with the ZED camera can produce a 3D point cloud representation of the observed scene, I will visualise 3D point cloud based on ZED camera by Steriolabs [9] and compare the point cloud outputs of both sensors.

Firstly, the sensors must be spatially aligned in software to match the point clouds that they produce.

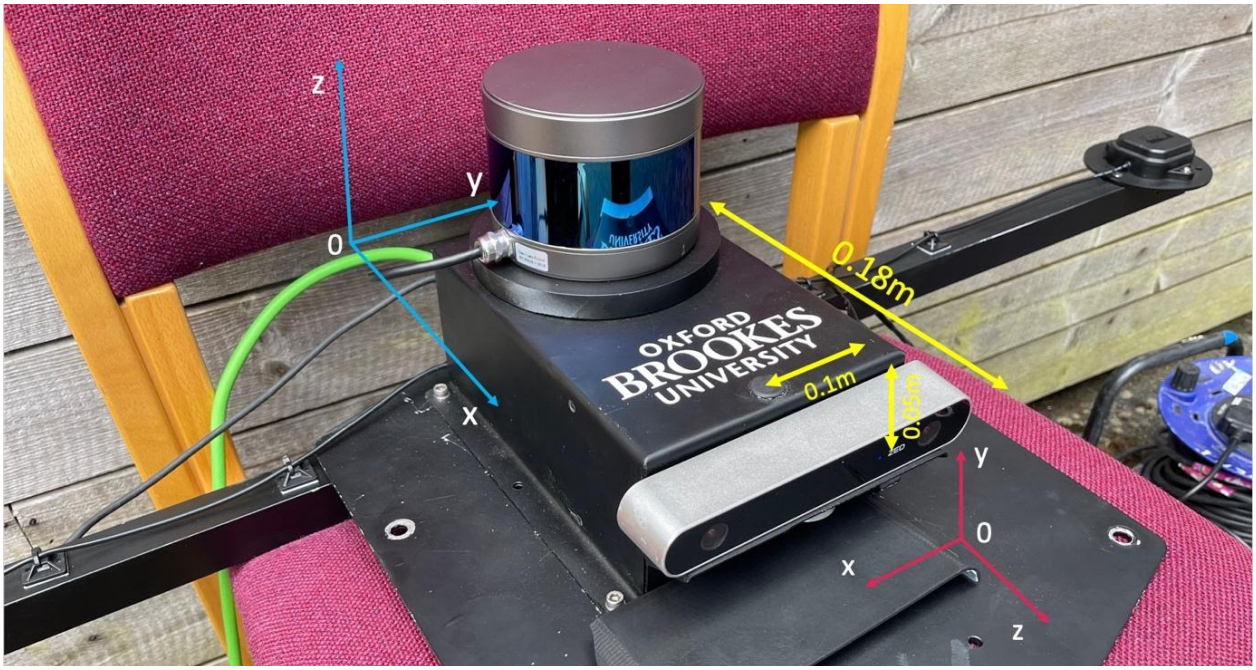


Figure 6.1.3. The OBR Autonomous sensor plate

The first step is to match the coordinate systems of the two sensors by Matlab [45]. After visualising both sensor's point clouds *(figure 6.1.4),* it shows that LiDAR returned data in right-handed z-up format, while the camera was left-handed x-up. Since the OBR Autonomous used the former format, the camera's point cloud was transformed.

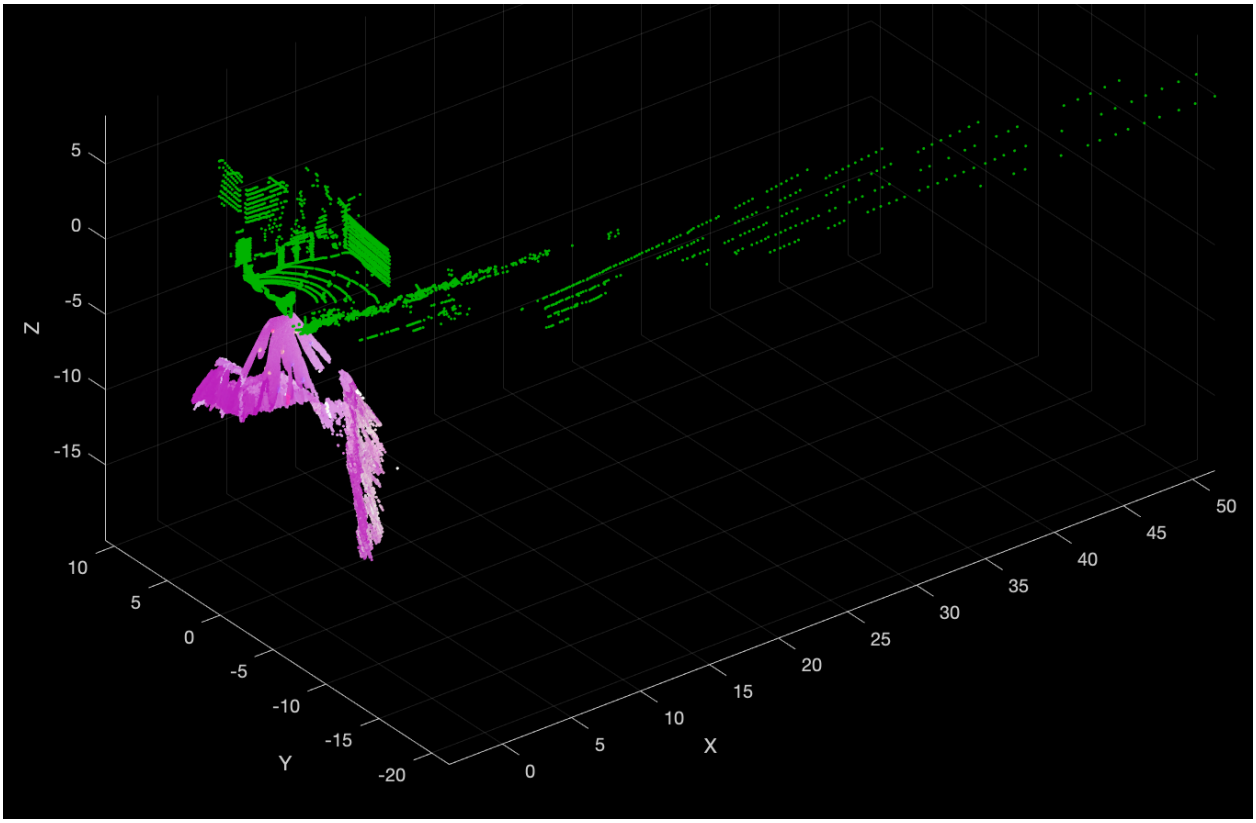


Figure 6.1.4. ZED (pink) and LiDAR (green) point clouds before rotation.

The vehicle sensor plate has a physical offset between the sensors in 3 dimensions *(figure 6.1.3)*. To match the point clouds, the camera's cloud should hence be translated. Orientation of the sensors was not the same, and therefore a 3D rotation matrix was used to align the point clouds. Because the LiDAR is a cylinder which cannot tell exactly the front, it is hard to measure the orientation offset of a physical plate, I had to adjust the values in software and decide the rotation angles based on the visualizations. Those angles that align the clouds best were chosen *(figure 6.1.5)*. The position of the sensors is fixed on the plate. Therefore, once these translation and rotation parameters were found, they shall be used for the rest of the time.

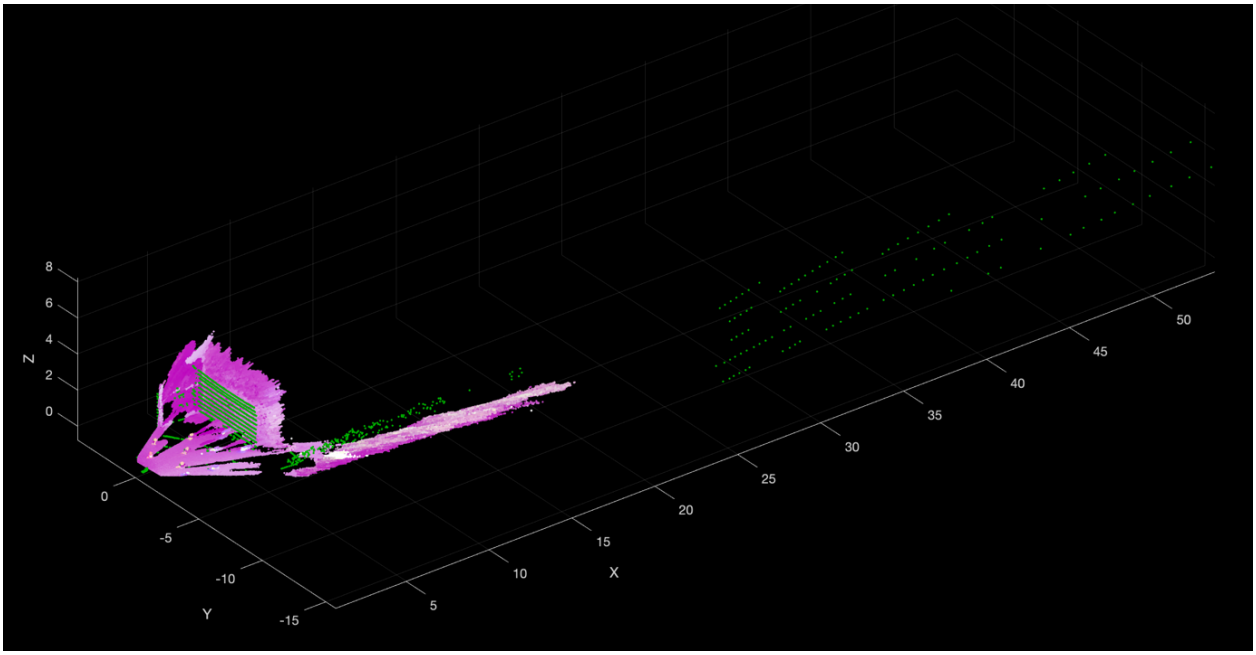


Figure 6.1.5. ZED (pink) and LiDAR (green) point clouds after rotation.

I used Zed camera and LiDAR point clouds with cones from both sensors taken at the same time. Since the cones are quite small and can be spread across the surface far away, the ZED's point clouds make good data for aligning the point clouds.

The point cloud data is a very large dataset. Moreover, the part of the point cloud that is useful is only the front one (where the camera is looking). Also, the top part of the point cloud is omitted because there are no objects. It could also be useful to remove the ground from the point cloud, but it will require extra processing time and might actually remove the points that are important [5].

Since ZED camera has a limited horizontal field-of-view compared to LiDAR, it made sense to crop the output of LiDAR to save space for processing and match it with camera's FOV. According to ZED's documentation [9], the camera's horizontal FOV is 90 degrees. However, the point cloud suggests a smaller FOV: 80.54 degrees, this is due to the image rectification process. Such small FOV might be a downside of the current object localisation system. After cropping *(figure 6.1.6),* the size of the LiDAR data decreased from 12,000 points to about 1,300 points. Cameras generate much more data in the point cloud, therefore using calibrated LiDAR can be more time- and memory-efficient.

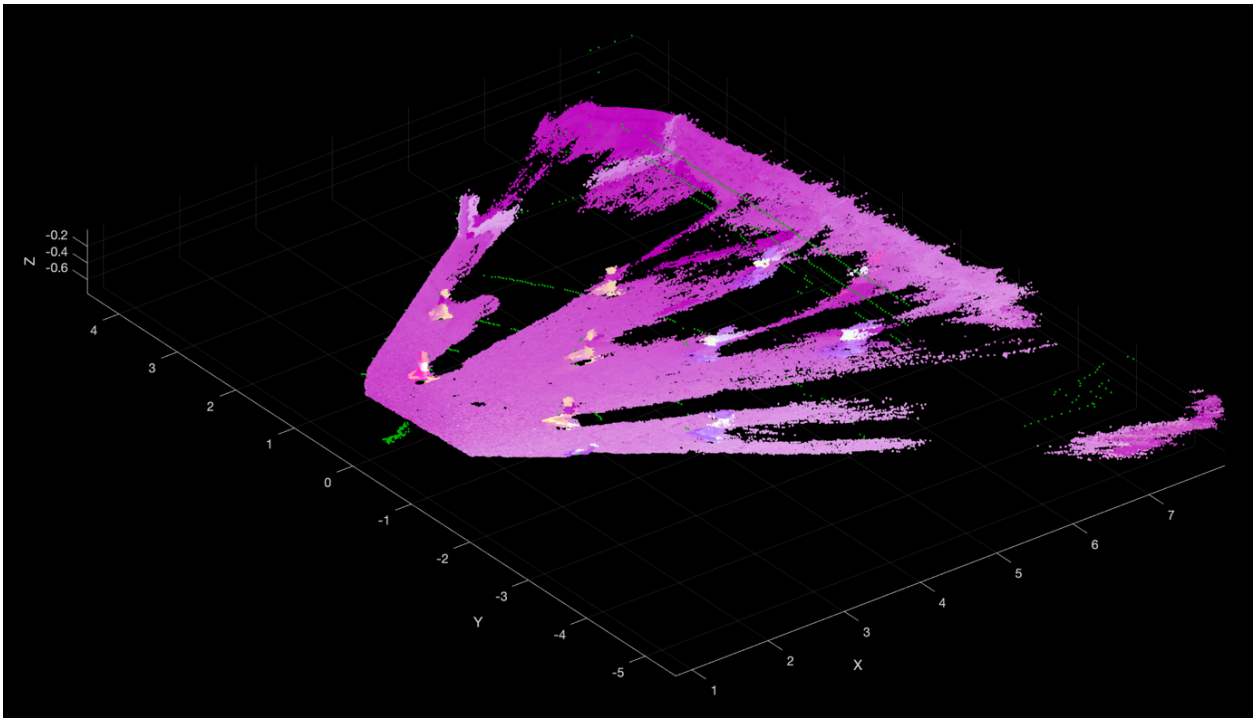


Figure 6.1.6. ZED (pink) and LiDAR (green) point clouds after rotation and cropping

Point cloud alignment allows us to measure the distances to cones from both sensors and compare the results between one another and against the true measurement.

I set up and recorded some data from OBR car *(figure 6.1.7).* The cones are measured by a wheel and a tape measure. The limitation of this experiment is a potentially large error in measuring the real distance by humans.

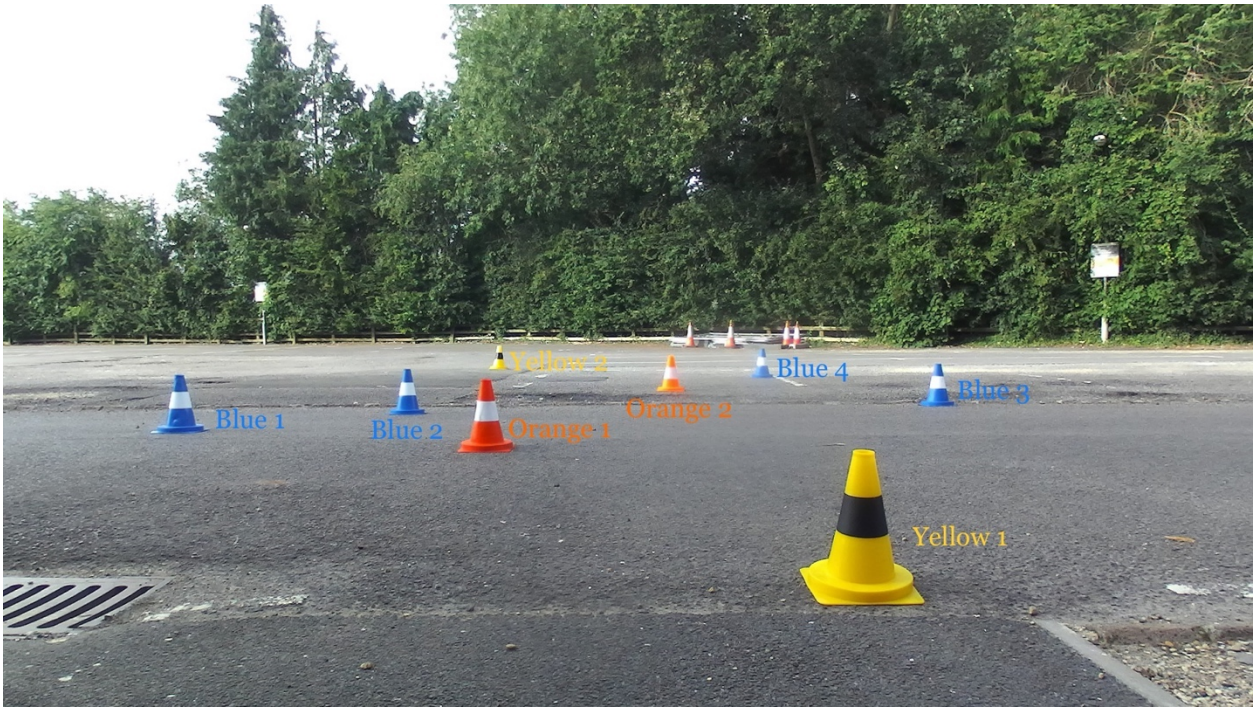


Figure 6.1.7: Example of cones are set up for comparison of ZED and LiDAR measurement errors

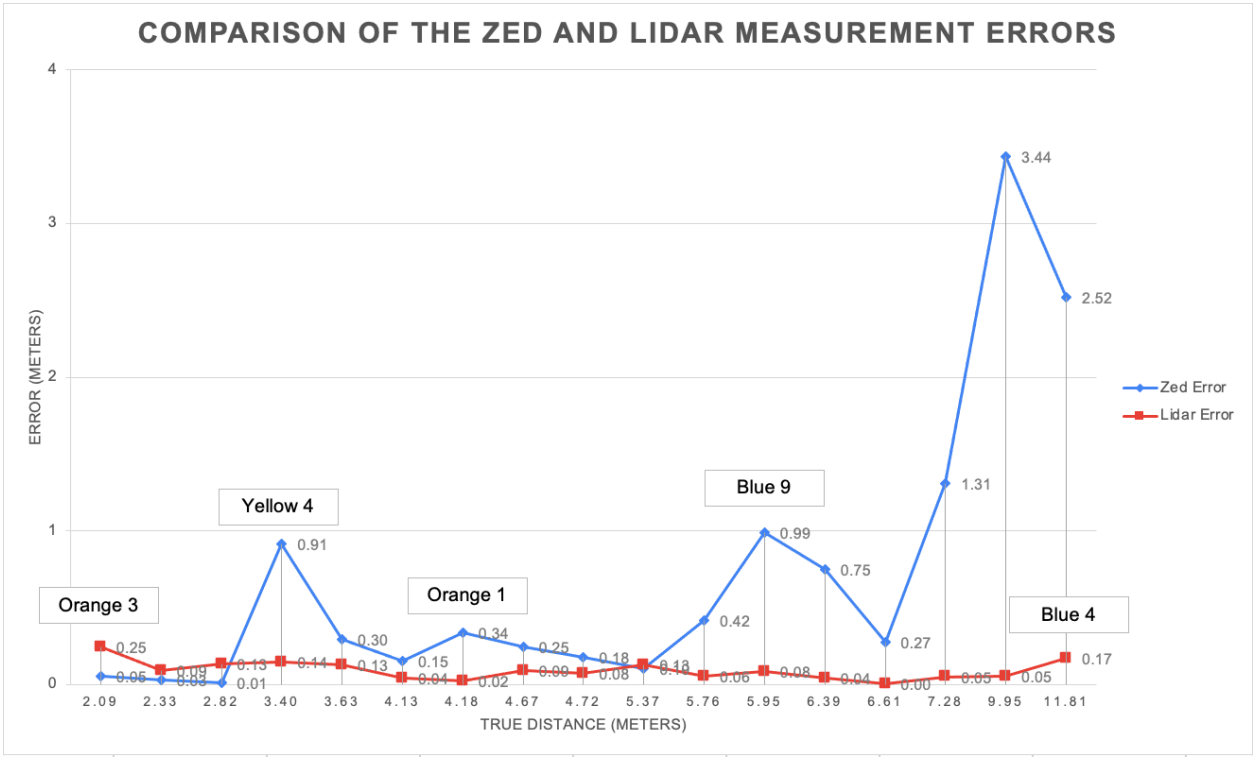


Figure 6.1.8. Distance measurement error for ZED camera and LiDAR (Appendix A)

In *figure 6.1.8*, the vertical axis shows distance measurement errors for ZED camera and LiDAR and horizontal axis shows real distance. Based on the collected data of 19 cones *(Appendix A)*, LiDAR is more accurate on longer distances. However, the LiDAR we are currently using omits some cones because it is limited to 16 channels. The average error for ZED is 0.63 m, and for LiDAR is 0.08m. The ZED stereo camera is not as accurate as the RS-16 LiDAR. The LiDAR provides quite a constant error independent of the distance.

This experiment clearly shows the advantage of LiDAR but also suggests that LiDAR is less accurate than its datasheet mentions (2cm accuracy of measurements is mentioned but the average error of measurement from the experiments is 8cm).

To integrate LiDAR with the current vision system, there must be a method to map a 2D bounding box from object detection to the 3D points. With a ZED camera it is easy: the point cloud is structured, and every point corresponds to a pixel on an RGB image. Hence looking at a point cloud is like looking at an image where each pixel has an extra value: depth. Therefore, if we wanted to map a pixel from a 2D image to a point on a 3D point cloud, we could take the (x, y) 2-D coordinates (indices) of the pixel and extract a point with the same indices. This point will bear 3-D coordinates for the object from the 2D camera image. LiDAR is not structured, and such mapping will not work. So, the solution is to use a calibration method. Calibration is a process of calculating the right numeric values of a transformation matrix which is used to map (i.e., transform) a 2D pixel of a camera image to a corresponding 3D point of a LiDAR point cloud using linear algebra [45].

6.2. Fusion Method

Sensor fusion is the task of putting the data of all sensors into the same space. The method that I think is effective is the one based on the open-source work [48] provided in Udacity. The starter sensor fusion code is gradually developed in C++ as part of the Sensor Fusion Nanodegree course. I acknowledge Elif Ayvali for reimplementing the code in python [49].

The method is to perform a 2D object detection on images, map the bounding boxes to the 3D data returned by LiDAR, and extract the 3D location to the detected object. The method is split into 3 steps [50]:

1. Extraction of 2D bounding boxes from an image using YOLOv3
2. Given the bounding boxes and the calibration matrix, cluster the points that belong to the same object in a LiDAR point cloud.
3. Extraction of bounding boxes with information of camera distance and LiDAR distance.

To perform object detection on images, an object detector will be trained. YOLOv3 was chosen in the first step. The model used in this study was pretrained on a COCO (Common Objects in Context) dataset that features 91 object types including many roads user and road scenario objects like person, bicycle, car, motorcycle, bus, train, truck, horse, traffic light, and stop sign [46].

If a custom dataset should be used, it can be used to train YOLOv3 again and use the same method. OBR Autonomous has a dataset of cones, I trained some images from OBR car as demo *(figure 6.2.1)*. The system will be written in Python 3, this is the language that OBR Autonomous uses. The trained file is saved as yolov3.weights format for step 1.

Figure 6.2.1. OBR cones data is trained by YOLOv3

Step 2 is to cluster point clouds from LiDAR with bounding boxes from YOLO by calibration matrix. This matrix is attached in KITTI dataset which I will use to demo the results. However, with a custom data for OBR car, we need to find the calibration matrix, this will be described in 6.3. Camera LiDAR calibration.

In step 3, given 2 consecutive image frames, calculate keypoint descriptors (distinctive parts of an object on an image) for each bounding box based on the image data. This step is used for object tracking and an alternative camera-based distance estimation. The LiDAR distance can later be compared to camera-based approach to demonstrate the difference in accuracy. This step will be applied automatically for custom dataset.

6.3. Camera LiDAR Calibration

Lidar camera calibration estimates a rigid transformation matrix that establishes the correspondences between the points in the 3-D lidar plane and the pixels in the image plane [58]. After that, the rigid transformation matrix is applied in sensor fusion code to put all the outputs from ZED camera and LiDAR in the same space.

MATLAB for example provides a calibration tool for that, as well as a number of open-source projects. To estimates a rigid transformation matrix, normally a checkerboard pattern is used. The checkerboard (a rectangular plane with a black-and-white checkerboard pattern on it) can be recognised on images and point clouds using established computer vision algorithms [47] because of its geometrically structured appearance. Knowing the real size of the squares on the board and using a checkerboard-detection based on camera image and rectangle board detector for LiDAR, it is possible to estimate the position and orientation of the same board from both sensors and produce a transformation matrix for fusing the sensor data. As stated previously, a camera is not ideal for 3D estimation, and LiDAR point cloud is sparse, therefore there is a certain amount of error when it comes to such calibration.

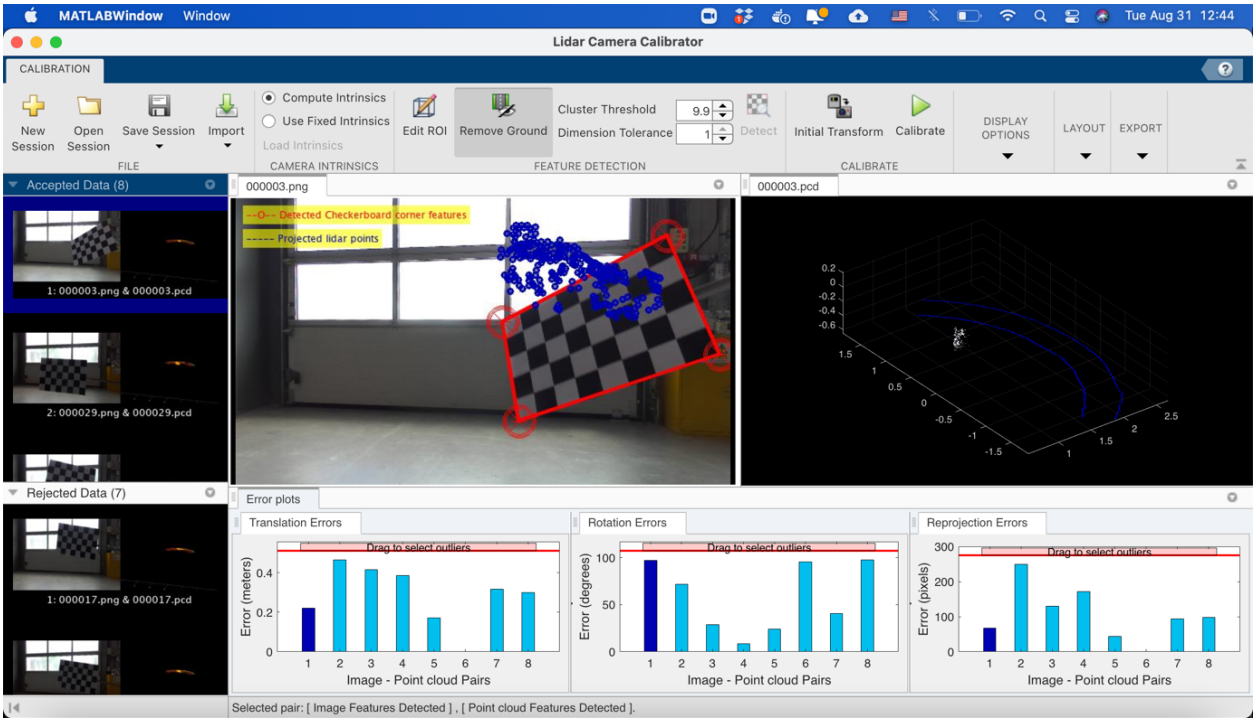


Figure 6.3.1. Checkerboard image and point cloud are collected by OBR camera and LiDAR on Matlab calibration app

In *figure 6.3.1*, you see an example of such a calibration process using MATLAB calibration app and a printed checkerboard pattern. 15 pair frames are recorded by camera and LiDAR, 8 pair frames are accepted and 7 frames are rejected (left panel). The blue points on the right are the point clouds mapped to an image.

Cluster Threshold is set to 9.9 the Euclidean distance threshold for differentiating point cloud clusters to 9.9 world units instead of default 0.5 to get more accepted data. Higher values of the best-fitting cluster mass indicate a lower scatter in the mass relation [59], this related to a bad quality of input data.

Dimension tolerance is the maximum acceptable deviation between the known standard and the calibrated device [60], is also set up to maximum of 1. A higher Dimension Tolerance indicates a more tolerant range for the rectangular plane dimensions [61].

The bar plots at the bottom of the figure describe 3 types of errors to evaluate the accuracy of calibration. They are produced by MATLAB for each image & point cloud pair. Three types of errors are defined to estimate the accuracy of the calibration [45]:

Translation Error: Mean of difference between centroid of checkerboard corners in the lidar and the projected corners in 3-D from an image.

Rotation Error: Mean of difference between the normal of checkerboard in the point cloud and the projected corners in 3-D from an image.

Reprojection Error: Mean of difference between the centroid of image corners and projected lidar corners on the image.

The translation, rotation and reprojection errors evaluate how precisely the LiDAR points and estimated locations of the checkerboard corners align in 3D (translation and rotation errors) and 2D (reprojection error) spaces [45]. In this figure, for example, the translation error is about 0.2 meters, rotation error is about 100 degrees, and reprojection error is about 70 pixels *(figure 6.3.1).* This is a large error that will not allow precise calibration and, hence, fusion of sensors.

7. Results

Because the camera-LiDAR calibration error was so large *(figure 6.3.1 Error plots),* I did not use the OBR Autonomous sensors and data to test fusion method. Instead, to test and evaluate the sensor fusion of 3D object detection, I used the KITTI dataset. This dataset has both image and point cloud data from road traffic environments, and valid calibration parameters that will allow sensor fusion.

YOLOv3 produces 71.83% mAP (the mean average precision) detection quality in the KITTI dataset, as evaluated by another research [51]. The mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections [62]. The point clouds in the KITTI dataset were generated by Velodyne HDL-64E LiDAR [52]. According to the sensor specifications, it can produce measurements 20 times a second with the accuracy < 2cm [53]. Therefore, fusing the sensors for this public road scenario can ensure competitive object detection quality and their high-accuracy localisation in 3D.

The mentioned-before open-source work was used to create a 3D object detection and localisation pipeline. The images in *figures 7.1, 7.2 and 7.3* illustrate the result of this

implementation. *Figure 7.1* is bounding boxes from YOLOv3 with their confidence. *Figure 7.2* is point clouds are clustered into the bounding boxes. *Figure 7.3* displays bounding boxes for each detected object (road vehicles) with measurement distance. The images were produced using the same program but replacing measurement units from TTC (time to collision) to distance (meters). As an example, the distance information for one of the cars was extracted and displayed. There are two measurements, the first measurement comes from the LiDAR information, the second measurement is a camera-based distance estimation using image keypoints descriptors.

Figure 7.1. Step 1 – YOLOv3 result on KITTI dataset

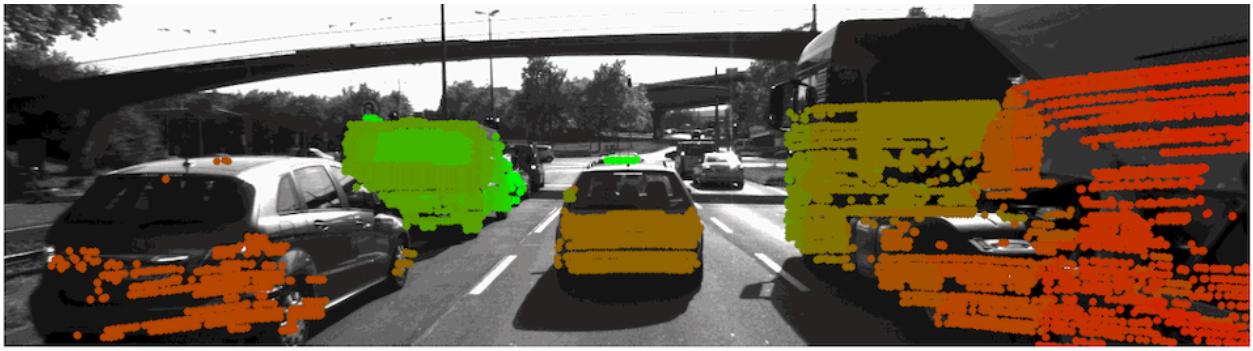


Figure 7.2. Step 2 - LiDAR point clouds mapped to the camera pixels that belong to the same objects

Figure 7.3. Step 3 - 3 consecutive camera frames processed by the software.

In *figure 7.3*, you can see that all the vehicles that are relevant for the autonomous driving task are detected (enclosed by a bounding box). The distance information was extracted from the corresponding LiDAR and camera data. Looking at the 3 consecutive frames, it's clear how the LiDAR measurements correctly describe that the car in front is getting closer.

8. Conclusion

The object detection and localisation solution has been produced. Two experiments were performed to answer two questions for OBR car:

1) Is that more accurate when using LiDAR to measure the distance than ZED camera? The average distance measurement error for OBR camera is 0.63m, and for OBR RS-16 LiDAR is 0.08m with small cone objects. Fusion of sensors might potentially lead to better performance if camera data is used for 2D object detection, LiDAR is used for distance measurement, and the data is combined for 3D object localisation. LiDAR data also takes less space since it's much sparser than the ZED camera 3D data and effective cropping in runtime.

2) Is RS-16 LiDAR good enough for sensor fusion method? The sensor fusion does not require a high dense point cloud data from LiDAR to measure distance to the car because the distance is measured by geometry information of the points. However, to use the fusion, an accurate camera LiDAR calibration matrix is needed, it also means that a denser point cloud - a better quality LiDAR is needed. After the calibration file is produced, the output of the OBR Autonomous cone detector and the LiDAR data can be fused using the code produced during this project. However, as described above, OBR RS-16 LiDAR is not proper for an accurate calibration matrix. Hence, the sensor fusion method worked for KITTI dataset, but it failed when applying on OBR car. If OBR wants to use sensor fusion method for cone detection, then a more channels LiDAR should be used.

9. Further work

To find which LiDAR should be used to detect cone object, Gazebo simulation environment can be used to produce LiDAR point cloud data from different LiDARs, and then repeat calibration method to find if the quality of point cloud data is enough for an accurate Camera – LiDAR calibration.

One important problem that has not been thoroughly discussed is time-based synchronisation of sensors. Because the two sensors have different output rates (0.05 sec

for LiDAR and 0.04 sec for the ZED camera. Source: OBR Autonomous), and because the YOLO-based cone detector has an average latency of 0.045 seconds (Source: OBR Autonomous), the 2D and 3D data should be matched in time to produce correct measurements. This can be handled in future work.

References

[1] Sae.org. (2019). SAE J3016 automated-driving graphic. [online] Available at: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic.

[2] Campbell, S., O'Mahony, N., Krpalcova, L., Riordan, D., Walsh, J., Murphy, A. and Ryan, C. (2018). Sensor Technology in Autonomous Vehicles: A review. 2018 29th Irish Signals and Systems Conference (ISSC).

[3] www.imeche.org. (n.d.). FS AI. [online] Available at: https://www.imeche.org/events/formula-student/team-information/fs-ai/ [Accessed 14 Sep. 2021].

[4] Strobel, K., Zhu, S., Chang, R. and Koppula, S. (2020). Accurate, Low-Latency Visual Perception for Autonomous Racing: Challenges, Mechanisms, and Practical Solutions. arXiv:2007.13971 [cs]. [online] Available at: https://arxiv.org/abs/2007.13971.

[5] Kabzan, J., Valls, M. de la I., Reijgwart, V., Hendrikx, H.F.C., Ehmke, C., Prajapat, M., Bühler, A., Gosala, N., Gupta, M., Sivanesan, R., Dhall, A., Chisari, E., Karnchanachari, N., Brits, S., Dangel, M., Sa, I., Dubé, R., Gawel, A., Pfeiffer, M. and Liniger, A. (2019). AMZ Driverless: The Full Autonomous Racing System. arXiv:1905.05150 [cs]. [online] Available at: https://arxiv.org/abs/1905.05150.

[6] Fursa, I., Fandi, E., Musșat, V., Culley, J., Gil, E., Teeti, I., Bilous, L., Sluis, I., Rast, A. and Bradley, A. (2021). Worsening Perception: Real-time Degradation of Autonomous Vehicle Perception Performance for Simulation of Adverse Weather Conditions. [online] Available at: https://arxiv.org/pdf/2103.02760.pdf [Accessed 14 Sep. 2021].

[7] Supervise.ly. (2019). Supervisely - Web platform for computer vision. Annotation, training and deploy. [online] Available at: https://supervise.ly.

[8] Kok, K.Y. and Rajendran, P. (2020). A Review on Stereo Vision Algorithm: Challenges and Solutions. ECTI Transactions on Computer and Information Technology (ECTI-CIT), 13(2), pp.112–128.

[9] www.stereolabs.com. (n.d.). ZED Stereo Camera. [online] Available at: https://www.stereolabs.com/zed/ [Accessed 14 Sep. 2021].

[10] Ortiz, L.E., Cabrera, V.E. and Goncalves, L.M.G. (2018). Depth Data Error Modeling of the ZED 3D Vision Sensor from Stereolabs. ELCVIA Electronic Letters on Computer Vision and Image Analysis, 17(1), p.1.

[11] Pendleton, S., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y., Rus, D. and Ang, M. (2017). Perception, Planning, Control, and Coordination for Autonomous Vehicles. Machines, 5(1), p.6.

[12] www.robosense.ai. (n.d.). RS-LiDAR-16 - RoboSense LiDAR - Autonomous Driving, Robots, V2R. [online] Available at: https://www.robosense.ai/en/rslidar/RS-LiDAR-16/ [Accessed 14 Sep. 2021].

[13] Asvadi, A., Premebida, C., Peixoto, P. and Nunes, U. (2016). 3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes. Robotics and Autonomous Systems, 83, pp.299–311.

[14] Fusion of 3D-LIDAR and camera data for scene parsing. (2014). Journal of Visual Communication and Image Representation, [online] 25(1), pp.165–183. Available at: https://www.sciencedirect.com/science/article/pii/S1047320313001211 [Accessed 14 Sep. 2021].

[15] Moosmann, F., Pink, O. and Stiller, C. (2009). Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/5164280?arnumber=5164280 [Accessed 14 Sep. 2021].

[16] Zhou, Y. and Tuzel, O. (2017). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. arXiv:1711.06396 [cs]. [online] Available at: https://arxiv.org/abs/1711.06396 [Accessed 14 Sep. 2021].

[17] Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J. and Beijbom, O. (2019). PointPillars: Fast Encoders for Object Detection from Point Clouds. arXiv:1812.05784

[cs, stat]. [online] Available at: https://arxiv.org/abs/1812.05784 [Accessed 14 Sep. 2021].

[18] Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y. and Bai, X. (2019). TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. arXiv:1912.05163 [cs]. [online] Available at: https://arxiv.org/abs/1912.05163 [Accessed 14 Sep. 2021].

[19] Chen, Q., Sun, L., Wang, Z., Jia, K. and Yuille, A. (2020). Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots. arXiv:1912.12791 [cs]. [online] Available at: https://arxiv.org/abs/1912.12791 [Accessed 14 Sep. 2021].

[20] Oleksiienko, I. and Iosifidis, A. (n.d.). Analysis of voxel-based 3D object detection methods efficiency for real-time embedded systems. [online] Available at: https://arxiv.org/pdf/2105.10316.pdf.

[21] Sack, D. and Burgard, W. (2004). A comparison of methods for line extraction from range data. [online] CiteSeer. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.629 [Accessed 14 Sep. 2021].

[22] Oliveira, M., Santos, V., Sappa, A.D. and Dias, P. (2015). Scene Representations for Autonomous Driving: An Approach Based on Polygonal Primitives. Advances in Intelligent Systems and Computing, pp.503–515.

[23] Wang, M. and Tseng, Y.-H. (2011). Incremental segmentation of lidar point clouds with an octree-structured voxel space. The Photogrammetric Record, 26(133), pp.32–57.

[24] Asvadi, A., Premebida, C., Peixoto, P. and Nunes, U. (2016). 3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes. Robotics and Autonomous Systems, 83, pp.299–311.

[25] Octree-based region growing for point cloud segmentation. (2015). ISPRS Journal of Photogrammetry and Remote Sensing, [online] 104, pp.88–100. Available at: https://www.sciencedirect.com/science/article/pii/S0924271615000283 [Accessed 14 Sep. 2021].

[26] Maturana, D. and Scherer, S. (n.d.). VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. [online] Available at: https://www.ri.cmu.edu/pub_files/2015/9/voxnet_maturana_scherer_iros15.pdf [Accessed 14 Sep. 2021].

[27] Qi, C.R., Su, H., Niessner, M., Dai, A., Yan, M. and Guibas, L.J. (2016). Volumetric and Multi-View CNNs for Object Classification on 3D Data. arXiv:1604.03265 [cs]. [online] Available at: https://arxiv.org/abs/1604.03265 [Accessed 14 Sep. 2021].

[28] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. and Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/7298801 [Accessed 14 Sep. 2021].

[29] Ren, S., He, K., Girshick, R. and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), pp.1137–1149.

[30] Cai, Z., Fan, Q., Feris, R.S. and Vasconcelos, N. (2016). A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. arXiv:1607.07155 [cs]. [online] Available at: https://arxiv.org/abs/1607.07155.

[31] Xiang, Y., Choi, W., Lin, Y. and Savarese, S. (2017). Subcategory-aware Convolutional Neural Networks for Object Proposals and Detection. arXiv:1604.04693 [cs]. [online] Available at: https://arxiv.org/abs/1604.04693 [Accessed 14 Sep. 2021].

[32] Brownlee, J. (2019). How to Perform Object Detection With YOLOv3 in Keras. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/.

[33] Jeong, M., Park, M., Nam, J. and Ko, B.C. (2020). Light-Weight Student LSTM for Real-Time Wildfire Smoke Detection. Sensors, 20(19), p.5508.

[34] Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. [online] Available at: https://arxiv.org/pdf/1804.02767.pdf.

[35] Meyer, G., Charland, J., Hegde, D., Laddha, A. and Vallespi-Gonzalez, C. (n.d.). Sensor Fusion for Joint 3D Object Detection and Semantic Segmentation. [online] Available at: https://arxiv.org/pdf/1904.11466.pdf.

[36] Du, X., Ang Jr., M.H., Karaman, S. and Rus, D. (2018). A General Pipeline for 3D Detection of Vehicles. arXiv:1803.00387 [cs, eess, stat]. [online] Available at: https://arxiv.org/abs/1803.00387 [Accessed 14 Sep. 2021].

[37] Mousavian, A., Anguelov, D., Flynn, J. and Kosecka, J. (2017). 3D Bounding Box Estimation Using Deep Learning and Geometry. arXiv:1612.00496 [cs]. [online] Available at: https://arxiv.org/abs/1612.00496 [Accessed 14 Sep. 2021].

[38] Qi, C.R., Liu, W., Wu, C., Su, H. and Guibas, Leonidas J (2017). Frustum PointNets for 3D Object Detection from RGB-D Data. [online] arXiv.org. Available at: https://arxiv.org/abs/1711.08488 [Accessed 18 Dec. 2019].

[39] Xu, D., Anguelov, D. and Jain, A. (2018). PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. arXiv:1711.10871 [cs]. [online] Available at: https://arxiv.org/abs/1711.10871 [Accessed 14 Sep. 2021].

[40] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A.C. (2016). SSD: Single Shot MultiBox Detector. [online] Available at: https://arxiv.org/abs/1512.02325 [Accessed 14 Sep. 2021].

[41] Chen, X., Ma, H., Wan, J., Li, B. and Xia, T. (2017). Multi-View 3D Object Detection Network for Autonomous Driving. arXiv:1611.07759 [cs]. [online] Available at: https://arxiv.org/abs/1611.07759 [Accessed 14 Sep. 2021].

[42] Ku, J., Mozifian, M., Lee, J., Harakeh, A. and Waslander, S. (2017). Joint 3D Proposal Generation and Object Detection from View Aggregation. [online] arXiv.org. Available at: https://arxiv.org/abs/1712.02294 [Accessed 14 Sep. 2021].

[43] Liang, M., Yang, B., Wang, S. and Urtasun, R. (2020). Deep Continuous Fusion for Multi-Sensor 3D Object Detection. arXiv:2012.10992 [cs]. [online] Available at: https://arxiv.org/abs/2012.10992 [Accessed 14 Sep. 2021].

[44] www.cvlibs.net. (n.d.). The KITTI Vision Benchmark Suite. [online] Available at: http://www.cvlibs.net/datasets/kitti/raw_data.php.

[45] uk.mathworks.com. (n.d.). Plot 3-D point cloud - MATLAB pcshow - MathWorks United Kingdom. [online] Available at: https://uk.mathworks.com/help/vision/ref/pcshow.html [Accessed 14 Sep. 2021].

[45] uk.mathworks.com. (n.d.). Lidar and Camera Calibration - MATLAB & Simulink - MathWorks United Kingdom. [online] Available at: https://uk.mathworks.com/help/lidar/ug/lidar-and-camera-calibration.html.

[46] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Lawrence, Z.C. and Dollár, P. (2014). Microsoft COCO: Common Objects in Context. [online] arXiv.org. Available at: https://arxiv.org/abs/1405.0312.

[47] Duda, A. (n.d.). Accurate Detection and Localization of Checkerboard Corners for Calibration. [online] Available at: http://bmvc2018.org/contents/papers/0508.pdf [Accessed 12 Sep. 2021].

[48] www.udacity.com. (n.d.). Become a Sensor Fusion Engineer. [online] Available at: https://www.udacity.com/course/sensor-fusion-engineer-nanodegree--nd313.

[49] GitHub. (n.d.). GitHub - eayvali/SFND-3D-Object-Tracking: KITTI LiDAR and Camera Fusion. [online] Available at: https://github.com/eayvali/SFND-3D-Object-Tracking.

[50] Edge AI Guru. (2020). Tracking 3D objects with Lidar and Camera. [online] Available at: https://edgeaiguru.com/3D-Object-Tracking [Accessed 14 Sep. 2021].

[51] Li, Y., Tong, G., Gao, H., Wang, Y., Zhang, L. and Chen, H. (2019). Pano-RSOD: A Dataset and Benchmark for Panoramic Road Scene Object Detection. Electronics, 8(3), p.329.

[52] Geiger, A., Lenz, P. and Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/abstract/document/6248074/ [Accessed 30 Apr. 2020].

[53] pdf.directindustry.com. (n.d.). HDL-64E Datasheet - Velodynelidar - PDF Catalogs | Technical Documentation | Brochure. [online] Available at: https://pdf.directindustry.com/pdf/velodynelidar/hdl-64e-datasheet/182407-676099.html [Accessed 14 Sep. 2021].

[54] Robotsense (n.d.). RS-LiDAR-16 User Manual. [online] Robotsense. Available at: https://www.robotshop.com/media/files/content/r/rse/pdf/robosense-rs-lidar-16-laser-rangefinder-150-m-user-guide.pdf.

[55] Lambert, J., Carballo, A., Cano, A.M., Narksri, P., Wong, D., Takeuchi, E. and Takeda, K. (2020). Performance Analysis of 10 Models of 3D LiDARs for Automated Driving. IEEE Access, 8, pp.131699–131722.

[56] Zhao, Z.-Q., Zheng, P., Xu, S. and Wu, X. (2018). Object Detection with Deep Learning: A Review. [online] arXiv.org. Available at: https://arxiv.org/abs/1807.05511 [Accessed 30 Dec. 2019].

[57] autonomoustuff.com. (n.d.). Lidar Comparison Chart. [online] Available at: https://autonomoustuff.com/lidar-chart [Accessed 18 Sep. 2021].

[58] uk.mathworks.com. (n.d.). What Is Lidar Camera Calibration? - MATLAB & Simulink - MathWorks United Kingdom. [online] Available at: https://uk.mathworks.com/help/lidar/ug/lidar-camera-calibration.html [Accessed 18 Sep. 2021].

[59] Reyes, R., Mandelbaum, R., Hirata, C., Bahcall, N. and Seljak, U. (2008). Improved optical mass tracer for galaxy clusters calibrated using weak lensing measurements.

Monthly Notices of the Royal Astronomical Society, [online] 390(3), pp.1157–1169. Available at: https://academic.oup.com/mnras/article/390/3/1157/1066260 [Accessed 18 Sep. 2021].

[60] Metal Cutting Corporation. (n.d.). What Is Calibration Tolerance? [online] Available at: https://metalcutting.com/knowledge-center/what-is-calibration-tolerance/ [Accessed 18 Sep. 2021].

[61] uk.mathworks.com. (n.d.). Detect rectangular plane of specified dimensions in point cloud - MATLAB detectRectangularPlanePoints - MathWorks United Kingdom. [online] Available at: https://uk.mathworks.com/help/lidar/ref/detectrectangularplanepoints.html?searchHighlight=dimension%20tolerance&s_tid=srchtitle [Accessed 18 Sep. 2021].

[62] Paperspace Blog. (2020). Mean Average Precision (mAP) Explained. [online] Available at: https://blog.paperspace.com/mean-average-precision/.

[63] Yeong, D.J., Velasco-Hernandez, G., Barry, J. and Walsh, J. (2021). Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review. Sensors, 21(6), p.2140.

Appendices

Data from real world experimentation A (source: OBR)

Cones	True x	True y	True Distance	Zed x	Zed y	Zed Distance	Zed Error	LiDAR x	LiDAR y	LiDAR Distance	LiDAR Error
Yellow 1	1.98	0.66	2.09	2.14	0.021	2.14	0.05	2.23	0.68	2.34	0.25
Orange 1	2.3	0.4	2.33	2.30	0.023	2.30	0.03	2.19	-0.45	2.24	0.09
Bue 1	2.64	-0.99	2.82	2.83	0.028	2.83	0.01	2.82	-0.88	2.95	0.13
Blue 2	2.97	1.65	3.40	4.31	0.043	4.31	0.91	3.15	1.62	3.54	0.14
Blue 3	3.63	0	3.63	3.92	0.039	3.93	0.30	3.76	0.07	3.76	0.13
Orange 2	3.63	-1.98	4.13	3.98	0.040	3.98	0.15	3.76	-1.83	4.18	0.04
Blue 4	4.1	0.8	4.18	4.52	0.045	4.52	0.34	4.13	0.80	4.20	0.02
Yellow 2	4.62	-0.66	4.67	4.91	0.049	4.91	0.25	4.72	-0.60	4.76	0.09
Orange 3	4.62	0.99	4.72	4.91	0.049	4.91	0.18	4.69	1.01	4.80	0.08
Blue 5	4.7	2.6	5.37	5.27	0.053	5.27	0.10	4.87	2.56	5.50	0.13
Yellow 3	5.61	-1.32	5.76	6.18	0.062	6.18	0.42	5.58	-1.18	5.71	0.06
Yellow 4	5.94	0.33	5.95	6.94	0.069	6.94	0.99	5.85	0.40	5.87	0.08
Yellow 5	6.2	1.53	6.39	7.14	0.071	7.14	0.75	6.19	1.72	6.43	0.04
Blue 6	6.6	-0.33	6.61	6.33	0.063	6.33	0.27	6.61	-0.17	6.61	0.00
Yellow 6	7	2	7.28	8.59	0.086	8.59	1.31	6.91	-2.13	7.23	0.05
Blue 7	9.9	1	9.95	13.39	0.134	13.39	3.44	9.82	-1.23	9.90	0.05
Blue 8	11.6	2.2	11.81	14.33	0.143	14.33	2.52	11.81	1.96	11.98	0.17
Blue 9	1.98	0.66	2.09	2.14	0.021	2.14	0.05	2.23	0.68	2.34	0.25
Orange 4	2.3	0.4	2.33	2.30	0.023	2.30	0.03	2.19	-0.45	2.24	0.09

Data from real world experimentation B (source: OBR)

True Distance	Error (Calculated - True)	True Distance	Error (Calculated - True)
2	0.24	6	-2
2	0.28	7	2.25
2	0.28	8	3
2	2.5	8	2.8
2	0.15	8	4.15
3	0.27	8	3.6
4	0.7	9	3.5
4	0.25	10	5
4	0.8	10	6.65
5	1.35	10	-6
5	1.35	11	7.5
5	1.3	11	6.3
6	1.52	12	8.7
6	1.68	13	10.3
6	1.9	14	11.4
6	2	14	-10
6	-2	15	19

Code link:

https://drive.google.com/drive/folders/13HAIHlr_4NDj87dLsBtu7nDZv2izdReu?usp=sharing