**PROJECT REPORT**

Project Semester July – December 2019

# Perception for Autonomous Vehicles

Submitted by

**Gaurav Bhosale**

**PRN-16070123031**

Under the Guidance of

**Prof. Jayshree Pande**
**Assistant Professor**

**Mr. Ninad Pachhapurkar**
**Dy. Manager**


**Department of Electronics and Telecommunication Engineering**

**SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE**

**July to December 2019**

## DECLARATION

I hereby declare that the project work titled Perception for Autonomous Vehicles is an authentic record of my own work carried out at Automotive Research Association of India (ARAI) as requirements of six months internship semester for the award of degree of B.Tech. Electronics and Tele-Communication Engineering at Symbiosis Institute of Technology, Pune, under the guidance of Mr.Ninad Pachhapurkar and Prof. Jayshree Pande, during June to December,2019.

Gaurav Bhosale
16070123031

Date: 20th December 2019

Certified that the above statement made by the student is correct to the best of our knowledge and belief.

**Prof. Jayshree Pande**
**Assistant Professor**
**Faculty Mentor**

**Mr. Ninad Pachhapurkar**
**Dy. Manager**
**Industry Mentor**

# CONTENTS

# ACKNOWLEDGEMENT

I thank Automotive Research Association of India (ARAI) for providing me an opportunity to work on a live project of great significance to the organization with a team of talented professionals and access to their best computing infrastructure and resources.

I am grateful to my Industry Mentors Mr. Ninad Pachhapurkar and Mr. Manish Karle, for extending patient guidance, advice and opportunities to improve my work.

I also express my sincerest thanks to the other interns, engineers, managers and other members of ARAI for their cooperation made this internship enjoyable and memorable.

Finally I would like to thank my faculty mentor Jayshree Pande ,Electronics and Tele-Communication HOD DR. Neela .R and Director of Symbiosis Institute of Technology, Dr. Ketan Kotecha for giving me the opportunity to pursue a semester long internship at Automotive Research Association of India.

# List of Figures

# List of Abbreviations

| Sr. No. | Abbreviation | Stands For |
|---|---|---|
| 1 | ARAI | Automotive Research Association of India |
| 2 | TG | Technology Group |
| 3 | ROS | Robot Operating System |
| 4 | OpenCV | Open Computer Vision |
| 5 | LiDAR | Light Detection and Ranging |
| 6 | PCL | Point Cloud Library |
| 7 | Numpy | Numerical Python |
| 9 | PnP | Perspective N Point |
| 9 | RANSAC | Random Sample Consensus |
| 10 | KD- Trees | K- Dimensional Trees |
| 11 | YOLO | You Only Look Once |
| 12 | IOU | Intersection Over Union |
| 13 | GPU | Graphical Processing Unit |
| 14 | LCD | Liquid Crystal Display |
| 15 | AI/ML | Artificial Intelligence / Machine Learning |

# 1) Automotive Research Association of India (ARAI)

Automotive Research Association of India (ARAI) was established in 1966. It is the pioneer R&D organization of the country set up by the Government of India and has been playing crucial roles assuring safe, less polluting, more efficient, and reliable vehicles.

Fig (1.1) ARAI

It has primary objectives in R&D into building competencies, capacities, and indigenous technologies for the betterment of the Indian Automotive Industry.

The automotive technology, in general, is going a significant facelift for the ever-increasing demand for connected and intelligent vehicles. The technical challenges for this are complicated, and this trend stresses a focus on R&D. There is a need to leapfrog to contest in the current market situation. The motivation of this group is to gear up ourselves to compete and sustain in the global automotive arena.

The technology group at ARAI is a technology innovation center that aims to provide sustainable and smart mobility solutions. Various programs are going at the Technology Group Intelligent Vehicle Controller (iVCON), Intelligent Energy Management System, Green Mobility xEV Program, Intelligent Vehicle Technology, and Clean Energy programs.

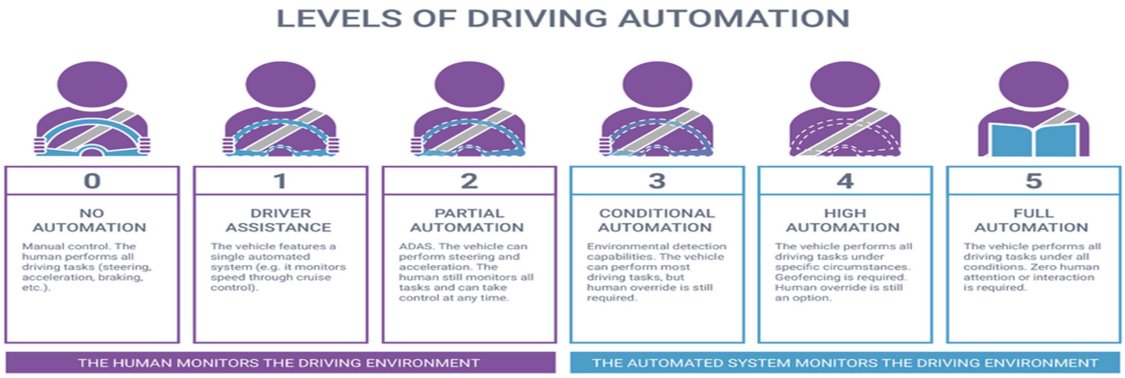

Fig(1.2) ARAI

ARAI has forayed into the development and validation of Advanced Driver Assistance (ADAS) and Automated Driving (HAD) features. As a part of this initiative, ARAI has launched a program named Intelligent Vehicle Technology, where their main focus is to develop and use these advanced technologies for India specific conditions.

# 2) Brief Introduction

The average traffic collision-related death rate is 17.4 deaths per 100,000 people in India. Either you or someone you know has likely been in an accident. Almost 95% of the road-related death are caused by Over speeding, Drunken Driving, Distraction to Driver, etc. The human behind the wheel is almost always responsible for the accidents. A solution to this automating driver behavior. Autonomous Vehicle or Self- Driving cars are vehicles that can sense its environment and navigate safely to their desired destination.

According to the society of Automotive Engineers (SAE) currently defines six levels of driving automation ranging from Level 0 (fully manual) to Level 5 (fully autonomous).




Fig(2.1) Levels of Autonomous Cars

At a commercial level, we already have level 2 automation in most of the cars in the coming years. The leaders in the industry like Tesla, Google, Daimler, etc. are each trying to reach Full automation as soon as possible.

My work in this report describes the perception of Autonomous vehicles. In general, tasks of Autonomous Vehicles can be divided into Perception, Cognition, and Action. Perception for humans is the ability to sense the world around them through their primarily five senses. Similarly, the car needs to sense its surroundings. Most of the cars use cameras, ultrasonic sensors, radars, and LiDARs for gathering information about their environment. My work describes Sensor Fusion for the two most essential sensors, the Camera and LiDAR, giving the car the ability to identify the object and its distance from the vehicle.

## 3) Projects Undertaken

Being a part of Technology Group in ARAI. I got allotted to work on projects building perception pipeline for ARAI's Autonomous Vehicle 'Swayam GO'.

1) Setup and Calibrating LiDAR- Camera
    - Setting up LiDAR
    - Setting up Camera
    - Intrinsic Calibration of Camera
    - Data acquisition from LiDAR and Camera

2) Sensor Fusion
    - Transformation Matrix between LiDAR and Camera
    - Acquiring co-ordinates of board corners from Camera.
    - Acquiring co-ordinates of board corners from LiDAR.
    - Finding the correspondence between LiDAR and Camera.
    - Projecting the 3D LiDAR points to 2D Image.
    - Projecting 2D Image in 3D LiDAR data.

3) 3D Obstacle Detection in LiDAR.
    - Crop Box function
    - Plane Segmentation using RANSAC.
    - Euclidean Clustering
    - Min-Max Boxing

4) Custom Object Detection using YOLO.
    - Data Preproccesing
    - Hyper parameters tuning and Training the Model.
    - Improvement and Detections of the Models.

5) Minor Tasks
    - Turtlebot3
    - Data acquisition for Autonomous Vehicles.
    - Displaying Steering Sensor values on LCD at Car's Dashboard.

# 4) Details of Projects Undertaken

1) **Software Requirements-**

**a) ROS – Robot Operating System**


Fig (3.1) ROS

The Robot Operating System (ROS) is a meta-operating and framework a system for writing software for robotics. It is a set of libraries and tools that are made to ease the task of creating complex and robots across a wide variety of robotic platforms. It aims to solve the problem of creating robot software. ROS is open source platform allowing a collective research and development. For example, one laboratory might have experts in mapping indoor environments, and could contribute a world-class system for producing maps. Another group might have experts at using maps to navigate, and yet another group might have discovered a computer vision approach that works well for recognizing small objects in clutter. ROS was designed specifically for groups like these to collaborate and build upon each other's work.

**b) Python**

**Python** is an high level programming language. It works on different platforms and has easy to read and understand syntax. it offers great code flexibility and is great language for prototyping.

**c) Point Cloud Library (PCL)**

The **P**oint **C**loud **L**ibrary (or **PCL**) is a large scale and open source library for 2D and 3D image and point cloud processing. PCL has various algorithms for 3D point cloud based performance.

**d) C/C++**

It is a general programming language with a strong focus on performance, efficiency and flexibility

## 2) Hardware Requirements –

### a) Velodyne VLP- 16 LiDAR

The Velodyne LiDAR 16 is a 16 channel LiDAR from Velodyne. It is the smallest and most cost-optimised product in Velodyne's lineup. It has a range of up to 100m and accuracy upto+/- 3cm. It mostly used robotics, shuttles, and other autonomous systems.

### b) Logitech c310 Camera

A digital **camera** is an optical instrument which is used to capture digital still images or to record video and stores in digital memory.

### c) TurtleBot 3 – Waffle Pi

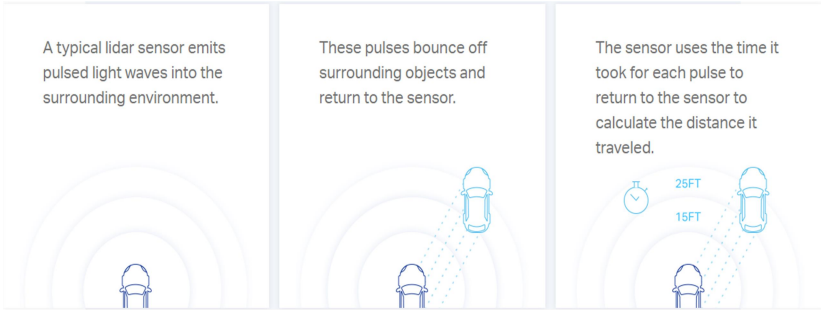
TurtleBot is a low-cost, personal robot kit with open-source software.

TurtleBot3 is a small, affordable, programmable, ROS-based mobile robot for use in education, research, hobby, and product prototyping. This often used to deploy algorithms before deploying on actual hardware.

## 3) Background Information


Fig(3.2) LiDAR

LiDAR stands for “light detection and ranging.” It is also called called “3D scanning” or “Laser scanning.” The LiDAR is an device rotating at high speed and emitting eye-safe beams of laser which are then used to produce a 3D map of surroundings .LiDAR is used for various tasks for mappings and guidance like building maps for geology, archaeology and autonomous vehicles.




Fig(3.3) LiDAR Working

The above process is repeated thousands of times per second, which generates a precise map of the environment. A computer can utilize this map for safe navigation.

The LiDAR technology makes it possible for self-driving cars to create and navigate 3D maps generated by LiDAR. The LiDAR provides the depth of surrounding pedestrians and vehicles as well as other things like road geometry. It also has the benefit of working exceptionally well in low-light environments, unlike cameras.

In self-driving cars, LiDAR is widely used. The sensors are installed on top of the vehicle, which provides us a 360 degrees Field of View. The LiDAR rotates, and the reflected rays are used to map the 360-degree surroundings of the vehicles. Using algorithms, the data received is converted into 3D graphics, which are then displayed as 3D maps, providing us data for navigation.

The LiDAR data is given in terms of radians and distance with respect to the sensor. These are then converted from the Spherical system to the Cartesian system.

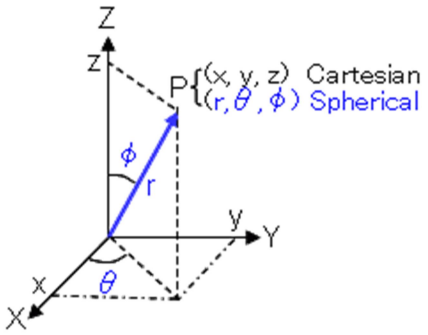
The points are converted from spherical coordinates to Cartesian coordinates, as shown in Fig.

Below are the equations—

$x = r*\cos\theta$

$y = r*\sin\theta*\cos\beta$

$z = r*\sin\theta*\sin\beta$




Fig(3.4) Cartesian and Spherical System

I mounted the LiDAR on top of a specially created bracket on the top of the car. These provide us with a 360-degree view around the vehicle. The LiDAR has a range of up to 100 m on a clear day and +/- 3cm accuracy.

By emitting invisible lasers, we able to create maps. These rays of lasers create “point clouds” that represent the vehicle’s surrounding environment to create the data acquired by sensor data.

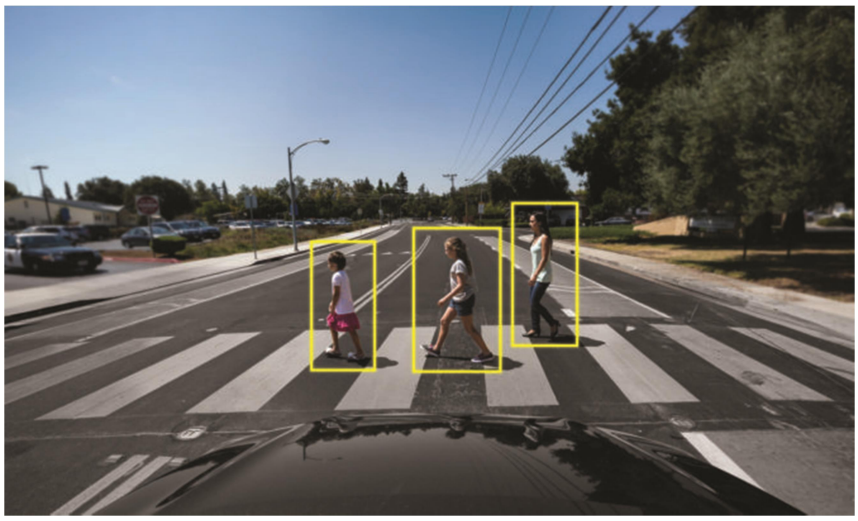
While LiDARs are extremely useful for autonomous vehicles, however, currently there are cost and reliability issues with LiDAR. Radar is cheaper than LiDAR, and LiDAR has more moving parts, which increases the error and a chance of failure. There is always a trade-off between using a radar or LiDAR sensor.

From Videos to Images, cameras are the cheapest way to acquire data about the environment around the car. Cameras provide accurate visuals, but cameras have their limitations. Although cameras are excellent and provide reliable visual information, they do not give us any information for where an object is located in environments. Cameras also fail in low perceptibility conditions, like fog, rain, or night-time.

The Logitech C310 camera is shown in the figure.
The camera provides us with information from the environment.
The camera captures images 720 x 480 as RGB images.
The image resolution obtained is enough for this project.


Fig(3.5) Logitech C310

We mount the camera on the Dashboard of ARAI's Autonomous Vehicles SwayamGO. This gives us the Image data in front of the car, allowing us to gather feature-rich data. Most of the Cars use multiple cameras to get data from all around the vehicle.



Fig(3.6) Object Detection through Camera

Camera and LiDAR sensors provide data about the car's environment. Similarly, as humans process a lot of data through their senses, the autonomous vehicle must be able to massive amounts of data and a continuous stream of information coming from various sensors in the car.

To do this, Self-driving cars use a process called sensor fusion.
Sensor fusion is the idea that if data from various sensors can be combined in the best method, the merged data can be more precise, more reliable, or provide a more excellent knowledge of the context in which the data is collected. It is entirely plausible to combine

data from various sensors to produce extremely rich, context-aware data, which reduces the constraints of singular sensors. Sensor fusion in automobiles is a concept that aims to produce the best information the vehicle with a combination of the best information available from all of its systems.

LiDAR and camera are sensors that are mostly used in autonomous robots, Example: Autonomous cars — combining the information from the LiDAR and camera help in overcoming their individual limitations. The camera is an excellent sensor that has feature-rich and dense data. It is used to detects roads, lanes, and obstacles. The LiDAR gives us depth in 360 degrees around us with a meager resolution.

Sensor fusion of LiDAR-camera is meant to complement each other. The camera is responsible for detection and identification tasks while the LiDAR gives the distance of the detected objects. The fusion of camera and LiDAR can be done in two ways — the fusion of data or fusion of the results.

## 4) Project Details

Being a part of Technology Group in ARAI. I got allotted to work on projects based on ARAI's Autonomous Vehicle. The projects which I worked on all are part of developing a perception pipeline for Autonomous Vehicles. It includes setting up, calibration, sensor fusion, and detection.

### Task 1- Calibrating LiDAR and Camera

#### 4.1.1) Setting up the LiDAR

To set up, configure, and calibrate the LiDAR to develop a sensor fusion system.

The LiDAR was setup in ROS (Robot Operating System) this was done using the VLP-16 drivers provided by the Velodyne.

The LiDAR communicates with the Computer using Ethernet. This required using custom configurations provided by the Velodyne. This established a connection between the LiDAR via Ethernet to our PC.

To capture and to work on LiDAR data via Ethernet. ROS was used due to its hardware abstraction. ROS provided appropriate drivers and software for visualization and recording the LiDAR data.

Below is a Fig. after Installing and Configuring Velodyne LiDAR.

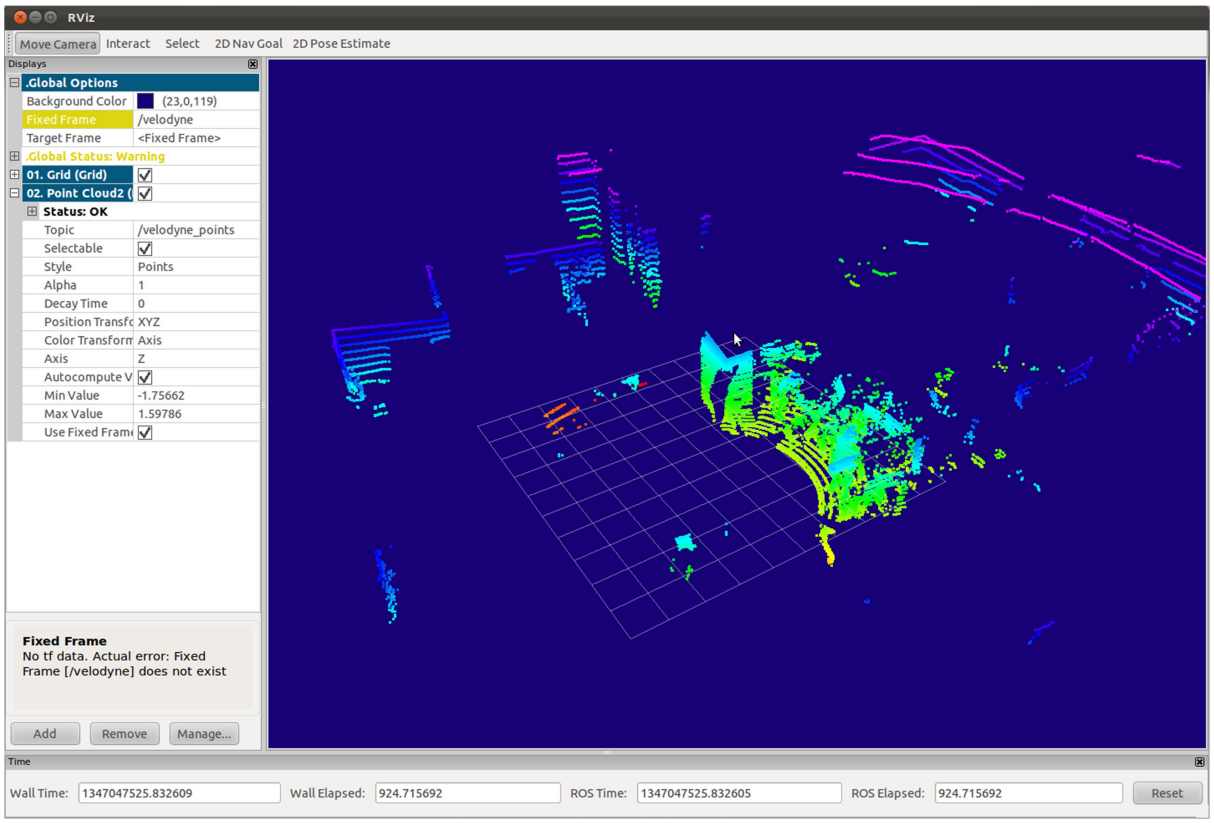



Fig.(4.1) Data from Velodyne LiDAR

### 4.1.2) Setting up the Camera

The Camera used is Logitech C310 for capturing Images and Videos. The Setup for this similar to the LiDAR. Wherein appropriate drivers and Configurations were done to get the Camera functioning.

### 4.1.3) Intrinsic Calibration of Camera

.

Importance of Camera Calibration –

Camera calibration is the process of obtaining the actual parameters of the camera that is used to take videos or images. Some of these parameters are lens distortion, focal length, format size and principal point.

The primary reason for this is because Camera produced at the factor are not perfect with exact specification mentioned. Due to small defects in lens, distance between lenses, etc.

These parameters can be used to adjust for errors in lens distortion, estimate the size of an object in world, or estimate the location of the camera in the environment. These Camera Calibration is necessary for applications such as computer vision to measure, detect and identify objects.

The Camera Calibrator Module was written using the OpenCV library. The Following are the steps taken to calibrate the camera –

I used checkerboard based Calibration with a 8 x 6 checkerboard. This has to be done in a properly lit area clear of any obstruction or check board patterns.

The camera was setup to publish images over ROS.

To start Calibration using the cameracalibrator.py code. In order to get a good calibration,


Fig(4.2) Calibration of Camera

The checkerboard was moved around the room. The X,Y,Size and Skew have to be filled up for calibrating various parameters of the camera. A successful calibration will result in real-world straight edges appearing straight in the corrected image. Failed calibration will result in not straight edges or with black or blank images.

### 4.1.4) Acquiring Data from LiDAR and Camera

To acquire data from LiDAR and Camera. ROS was used by me to publish the LiDAR Data and Camera Data on a ROS topic and subscribe to each of these topics and save the data as ROS bagfiles.

ROS Topic –

The prime method for ROS nodes to communicate is by sending and receiving messages. Messages are broadcasted on a topic. In ROS, every topic has a unique name. If a node wants to share information with any of the topics, it uses a publisher to send data to a topic. If a node wants to receive information from a specific another node, it must subscribe to the exact topic. Also, different messages have different message types. Example - If you want odometry information from your wheels, then that information is sent on a topic like /odometry, any program/node that intends to use odometry information has to subscribe to /odometry.

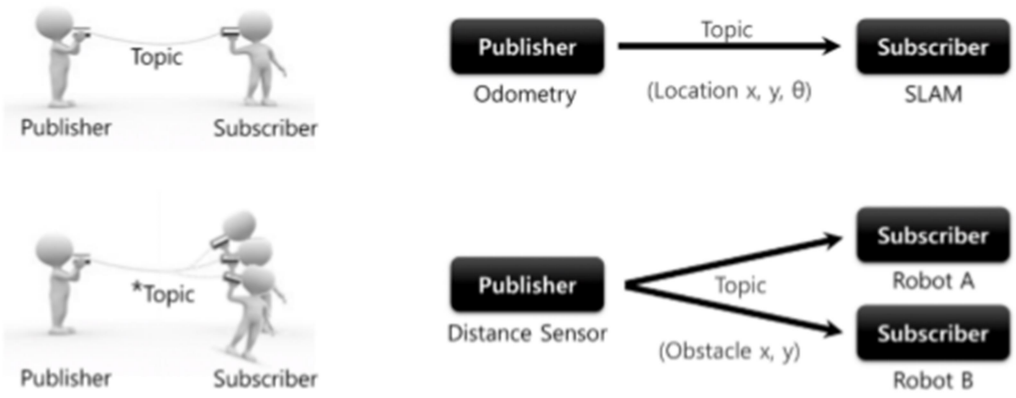



Fig.(4.3) ROS Topics explained

Publisher – It is a node which sends or outputs data. For us, this LiDAR and Camera. Subscriber – It is a node that receives data. For us, this is our rosbag file. Rosbag- In ROS, we can save various messages in bag format and play them back when necessary to reproduce the same environment when data is recorded. Rosbag is a program that creates, plays and compresses bags. These bagfiles are used to rerun the data to train and test the various works done below.

## Task 2- Sensor Fusion

There are many ways of fusing LiDAR with a camera using the extrinsic rigid-body calibration between the data of LiDAR and the camera. To get the transformation matrix between the LiDAR and camera. This transformation matrix gives us the rotation and translations from the 2D camera frame to the 3D LiDAR frame.

Before doing this, it is necessary to get data that has a common target between them. This means a fiducial target is required in front of both the sensors. A planar checkboard that is easy to detect due to its black and white squares by both camera and LiDAR.

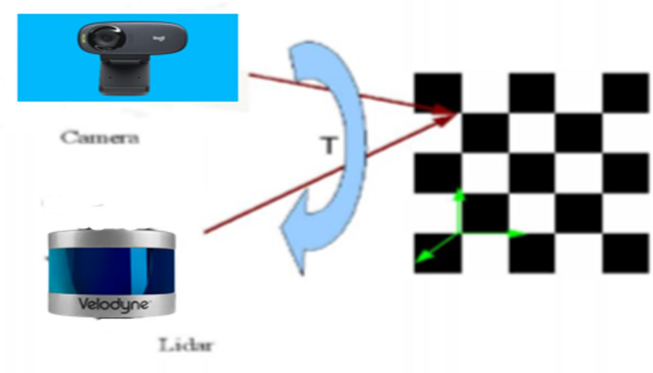
The camera gets the color and texture data from the fiducial checkboard while the LiDAR obtains 3D point cloud data from the target. The points 3D points were converted from LiDAR into 2D points in the camera plane, using camera projection model. We find the correspondence point to point correspondence between 2D and 3D points. This is done by

using extracting features of edges and corners from checkerboard. This was captured the checkerboard pattern from the camera using OpenCV in Python.

### 4.2.1) Getting the Transformation Matrix between LiDAR and Camera

#### 4.2.1.1) Using the Checkboard –

The checkerboard was used  a fiducial target for the extrinsic calibration of the 2D LiDAR sensor and the camera. A fiducial target can be anything like a board or circular board, but it must be distinct from its surroundings because it must be visible for both the LiDAR and the camera. The checkerboard pattern is easy to recognize both the sensors. Since the checkerboard is a unique pattern, it is easier detect the board in Opencv and find the point to point correspondence between the camera and LiDAR data Below Image shows the Camera data on the Left and Right is LiDAR Data.




Fig(4.4) Transformation between Camera and LiDAR

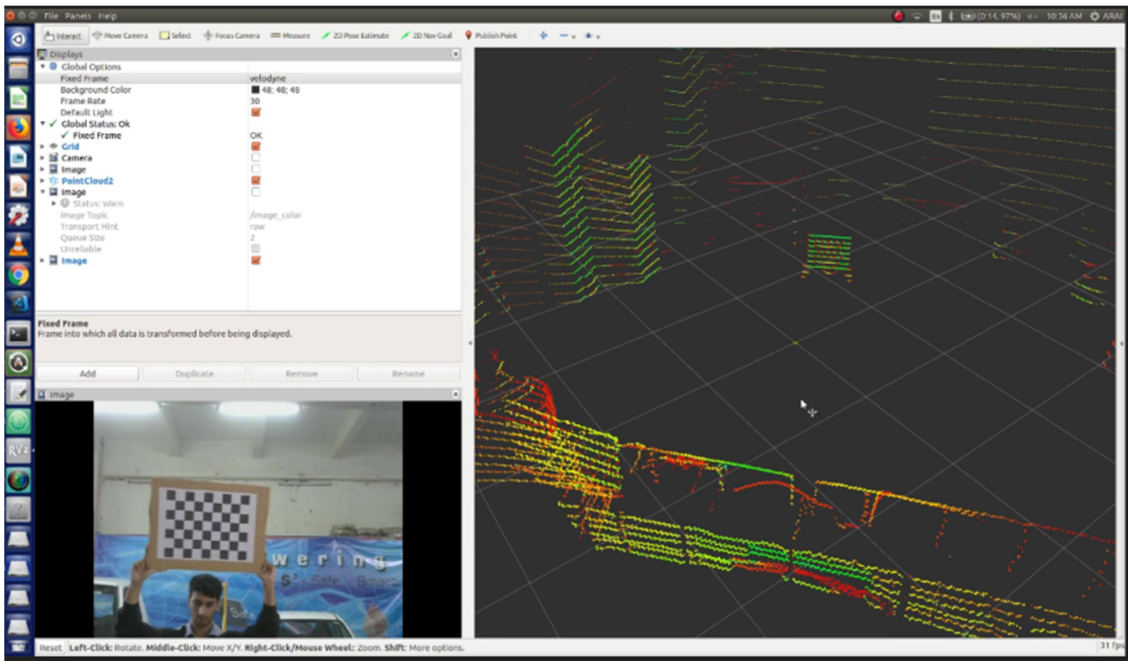


Fig.(4.5) LiDAR and Camera Visualisation

### 4.2.1.2) Getting Co-ordinates of corners of board from Camera-

The checkboard was mounted in front of the Camera and LiDAR to obtain the data between both the sensors. To detect the Check board in Camera Image, The code for check board detection was written. This is done using the OpenCV library and Python. The script is written such that the find Check board in OpenCV function detects the check board.

Since we are using a black and white checkerboard .
I use the function to detect the corners of a Check board.
Thus we can get the Top Left co-ordinates of the Image in (u,v), Right, Bottom Left and Bottom Right.

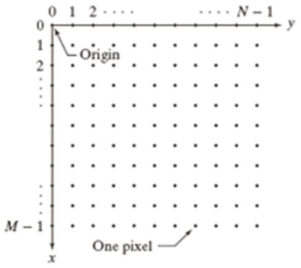
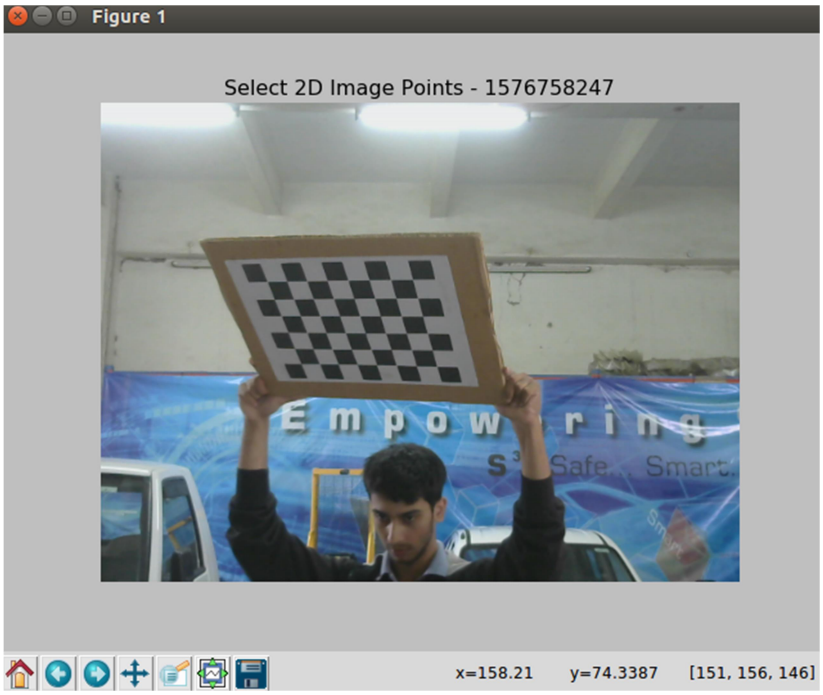



Fig.(4.6) Image coordinates

The GUI was used for manually extracting the points by clicking the corners on the Image. We can extract co-ordinates in Image as u,v for each of the four corners of the Image.The figure denotes the (u,v) coordinates as (x,y). We will be using u,v since we will be indicating the 3D coordinates in x,y,z.




Fig(4.7a) Camera Calibration

Once I got the four co-ordinates, this was stored array in text file for further use for finding the transformation.

#### 4.2.1.3) Getting Co-ordinates of Corners from LiDAR Data-

A similar method was followed but using LiDAR data. Coding an extraction for 3D LiDAR data proved difficult. Several Libraries were used for the same.Python, Numpy, Matplot Lib, PCLROS were used for the project.I run the rosbag file and pause the rosbag file at some moments. As explained above, I store the checkboard co-ordinates for the Image — the same way while the Rosbag demo is paused. A window is opened this window has LiDAR points at the movement at which the rosbag is paused. Then we select four corners of the checkboard in the 3D Image of LiDAR

These Points are then saved as LiDAR datapoints as text files in x,y,z coordinates.

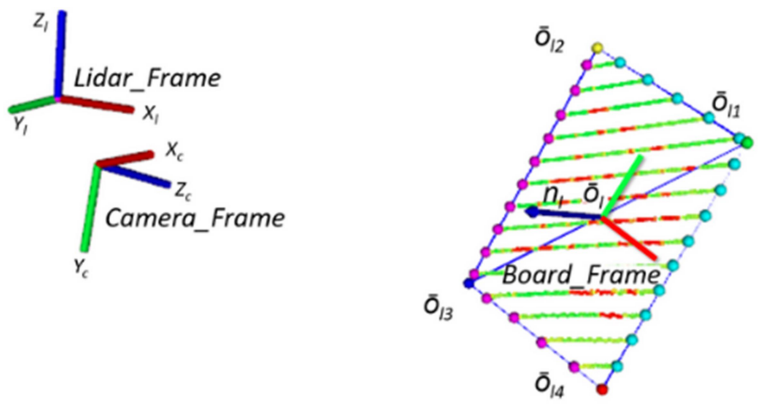

Fig.( 4.7b) LiDAR Calibration

#### 4.2.1.4) Finding the Correspondence between LiDAR and Camera.

Now we have four co-ordinates of the board in Image and four co-ordinates of the board in LiDAR co-ordinates. To find the co-correspondence between these, we use two algorithms –

1) PnP – Perspective N Point

2) RANSAC - Random sample consensus




Fig(4.8) Coordinate frame transformation

Perspective N point is a problem in which we determine the position of the camera from the n different 3D points in the world and their corresponding points in the 2D projection of the Image. The camera position and orientation of the camera are given in 6 degrees-of-freedom (DOF), which are made up of the rotation (roll, pitch, and yaw) and 3D translation (along x,y, and z-axis)of the camera with respect to the world. This is used mostly is robotics, Virtual Reality, and Pose estimation problems.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Fig(4.9)PnP Matrix

The perspective model for cameras is as follows:

s*Pc = K*[R|T] *Pworld where Pworld = [x y z 1]^T is the homogeneous world point from 3D LiDAR co-ordinates.

Pc = [u v 1]^T is the corresponding homogeneous image point and where s is a scale factor for the image point.

The intrinsic camera parameters are denoted by the matrix K, where fx and fy are scaled focal lengths, γ is the skew parameters mostly assumed to be zero, and (u0 and v0) is principal focal point scaled focal length. R and T are the values of 3D rotation and 3D translation of the camera.

PnP is extremely prone to errors if there are outliers in the set of the point correspondence which we have obtained. Thus often, RANSAC is used with different solutions like PnP to make the final solution more robust to outliers.

RANSAC –

Random sample consensus is a repetitive method for predicting a model from data in the presence of many outliers. The algorithm is straightforward. The RANSAC algorithm operates by recognizing the outliers in a given dataset and predicting the required model. It is an outlier detection and removal algorithm. The Ransac algorithm is non-deterministic because it gives results only with some probability, but this probability rises as we allow more iterations.

RANSAC algorithm works by using the below steps-

1. We randomly select a subset of the data set, which we call hypothetical inliners.
2. Then we fit a model that we define to the selected subset.
3. The points that do not fit within our model are outliers. We determine the number of outliers
4. The above steps are repeated for a prescribed number of iterations.
5. The points which fit the model are part of the consensus set.

PnP is prone to errors due to outliers in our corresponds. I decided to use PNP with RANSAC.I wrote the code to retrieve the corresponding data of each of the four corners of LiDAR and Camera and to run the PNP Ransac algorithm. OpenCV's solvePnP algorithm is used with the RANSAC algorithm.The arrays of Image points, LiDAR Points, Intrinsic Camera matrix is passed to the function.

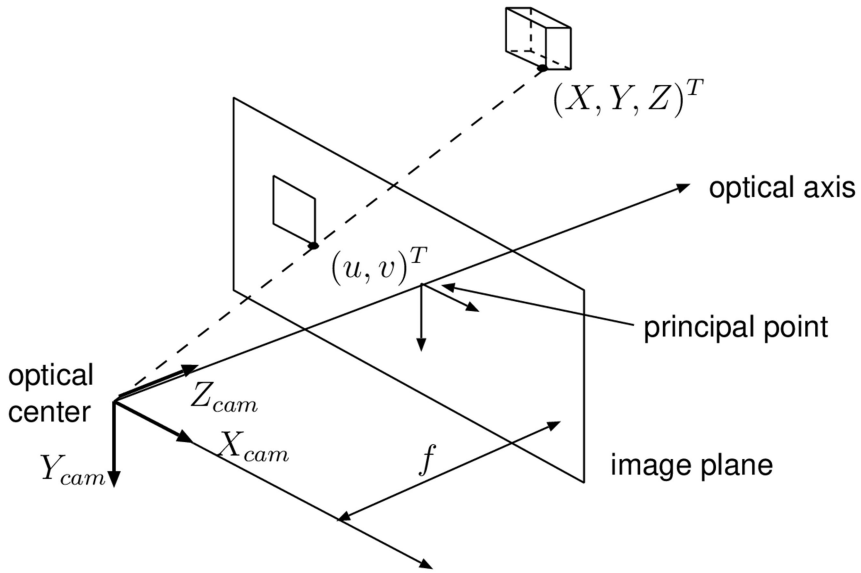
The resulting output we get after solving these matrices is the values of translation and rotation from Camera frame to LiDAR Frame.

The output is given as -Euler angles (RPY rad ) and Translation Offsets( X, Y, Z)Thus, We have obtained the 6 DOF Rigid Body transformation between the LiDAR and Camera.

#### 4.2.1.5) Projecting the 3D LiDAR points to 2D Image-

To test if our sensor data is fused appropriately we need to project these for our convenience as well as to check if there are any discrepancies or if we have fused the data incorrectly.

To project the points I used the Pin-Hole Camera model as shown below which gives us the relationship between 3D world points and 2D points in plane of the camera.




Fig(4.10) Pin-Hole Camera Model

The above relationship can be defined using similar triangle rules. We used the above model to project the LiDAR points on the Image on the Video. The Pin-Hole camera model was used to back project the 3D (x,y,z) to 2D (u,v). Then these were combined using OpenCV functions and published them over ROS.

Below is the output of sensor fusion –

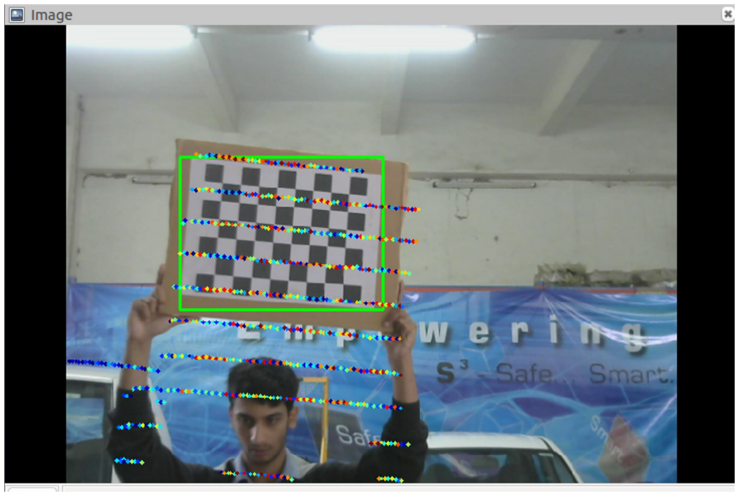
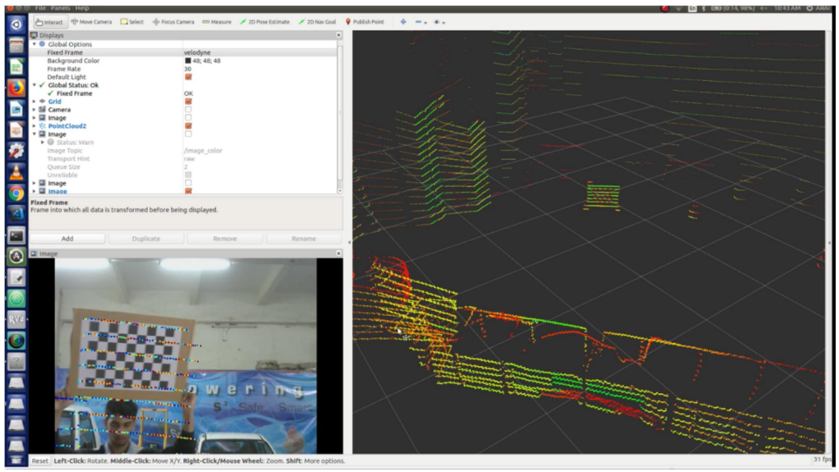



Fig.(4.11) Sensor Fusion (Projecting LiDAR points on 2D image)

The multi-coloured points are the LiDAR points back-projected from 3D to 2D.

As you can see I was able to achieve sensor fusion.



Fig(4.12) LiDAR points on Image giving us depth in Image

Since now I was able to achieve sensor fusion. We can extract the distance of objects from LiDAR which are detected on the Camera. This can be used to develop various algorithms for autonomous vehicles as ARAI can not only identify a object though the camera but now ARAI can tell how far a object from the car.

**Project 2D points on 3D data in LiDAR-**

Performing inverse of the above projection. I projected the outline of board in green on the LiDAR data.

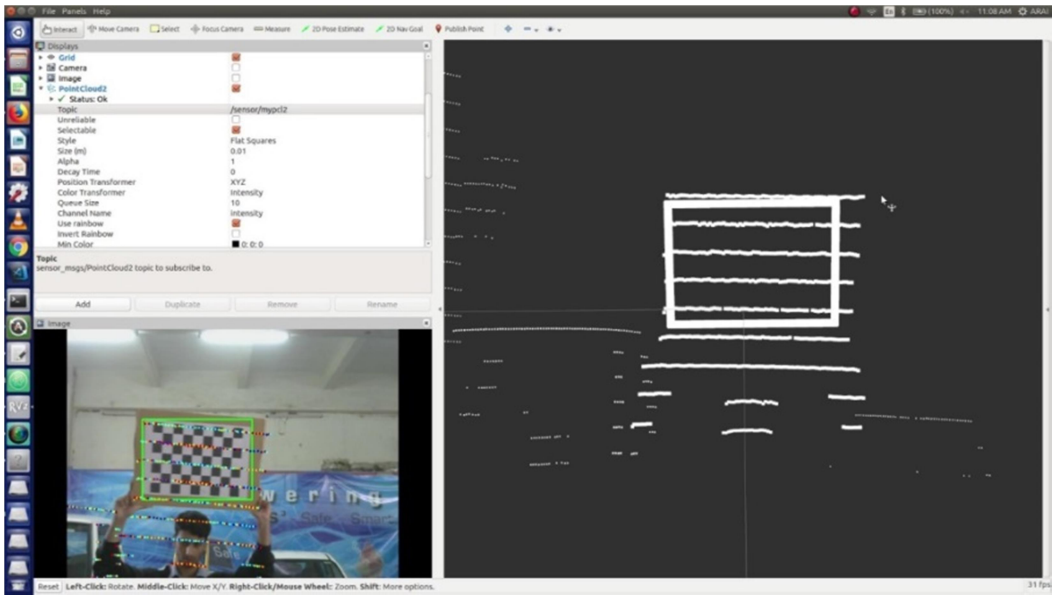


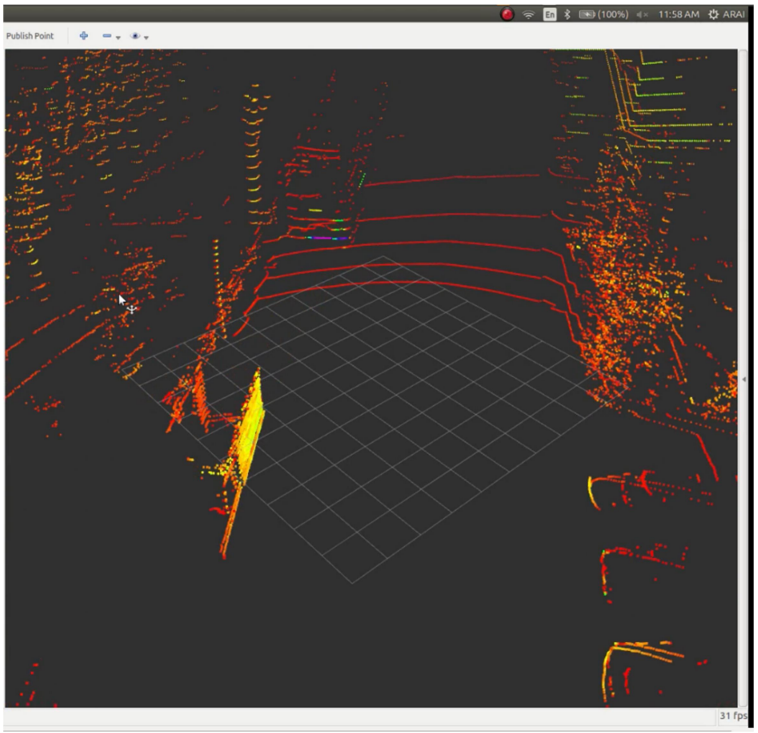
Fig. (4.13) Board Outline in Point Cloud Data

## Task 3- 3D Obstacle Detection in LiDAR

The above fusion, which we performed in the previous task, is known as Output fusion. Still, an alternative way to fuse the data is through output fusion of data where we perform a sensor fusion of the results is when, we perform object detection in the image and object detection in the LiDAR data (point cloud) separately, and we fuse the results to increase our confidence and redundancy.


Fig (4.14) Data Fusion and Output Fusion

In this type of fusion, we process the camera image data independently, get an output from camera detection, and then validate against the processed LiDAR detection output or vice-versa. This fusion method is beneficial since it can improve the reliability of a system. The 3D obstacle detection is done using the Point Cloud Library and C/C++.Here I had to detect obstacles in LiDAR Data. The following Image shows the data which was recorded on the road wherein I had to detect obstacles on the road.




Fig(4.15) A road in Point Cloud Data

The right window shows the LiDAR data on the road, and this data must be processed for detecting obstacles.

To process this data, I will be using PCL and Its functions. The functions which I used for Obstacle detection are shown below.

**4.3.1) Crop Box function –**

Since we want to detect the obstacles only on the road using the crop box function to process the points which are inside the crop box. This not only reduces the computation needed for all the points and since we need only to detect the objects on the road. We box extract the points only on the road.

**4.3.2)Plane Segmentation using Ransac**

We filter the points which are planes in the box, which we defined above. As you observed from the above data, there are various planes in the LiDAR Data, like the road and building, which might be included in our processing box. We do not want to detect them as obstacles incorrectly; thus, we use Plane Segmentation with the Ransac algorithm to remove planes from our processing pipeline.

**4.3.3)Euclidean Clustering –**

If we observe the above Image. You notice then most of the objects in the LiDAR Data are in clusters. We exploit this fact using segmenting point cloud into clusters based on Euclidean distance

Euclidean Distance –The Euclidean distance between any two points in a 2D plane or 3-D space computes the shortest distance between the two points connected by a line segment. It is the most straightforward way of representing the distance between two points.

Euclidean distance formula

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

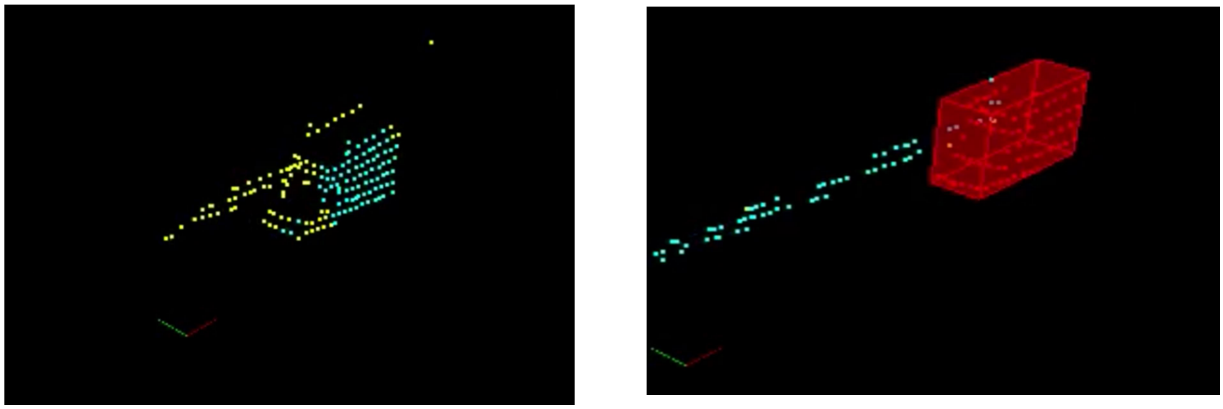
We need to calculate this for thousands of points and their distance with each other. This takes a long time to process. This is a prolonged process, but to speed up this process I used an algorithm called KD-Trees Search.

KD-Tree Search –A k-dimensional tree is a space partitioning algorithm used in various algorithms for creating partitioning a space with k dimensions. It is a binary search tree.

**4.4.4)Min- Max Boxing -**

Clusters were extracted from the above steps. To box these using the Min-Max boxing function for easier identification.

Below is a Car detected in PCL –



Fig(4.16) Obstacle Detection of Truck .

Below are obstacles that are extracted using clustering, and a box is fitted around it.

## Task – 4 Custom Object Detection in Image using YOLO

Humans can quickly identify and detect objects existing in an image. The human perception system is extremely quick and precise and can perform elaborate tasks like identifying various objects and detect obstacles with no thought. The massive amounts of data that we have available and faster GPU's make it possible to train algorithms that can detect objects. Object detection has applications in various areas of machine vision, including robotics and video surveillance.

You Only Look Once (YOLO) is an advanced object detection algorithm. Object detection consists of determining the location of the object in the image, as well as classifying those objects

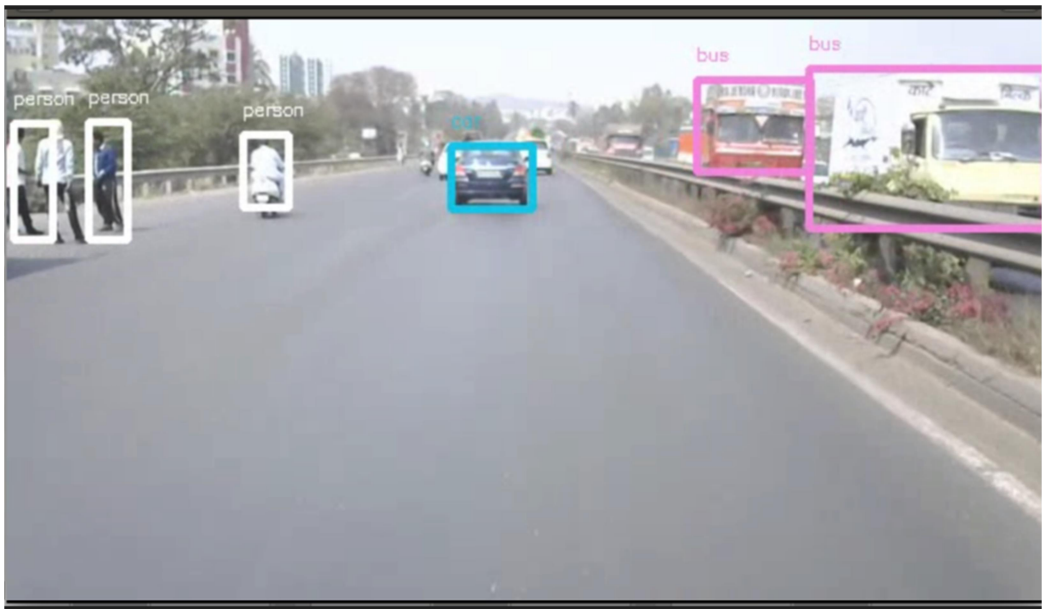
Once you input an image into a YOLO algorithm, it splits the images into an SxS grid. Each of these cells is predicting one object and a bounding box, but before that, it predicts *B* bounding boxes and *C* class probabilities.

The YOLO algorithm uses four components and a confidence score to predict the input. The center of a bounding box which is (bx, by), Height (bh), Width (bw), and the (c), which is the class of the object. The final predicted value is confidence (pc) is the probability that the object is inside the bounding box.

A YOLO network is like a regular convolutional Neural Network, it comprises of convolutional and max-pooling layers and then at last two fully connected CNN layers. The final step is a loss function -

Since each bounding box has multiple predictions and we need only to predict one class, we use the loss function. To do this, the loss function is used to calculate the loss for every true positive. To make the loss function effective, we need to choose the bounding box with the highest Intersection over Union (IoU). This method enhances predictions by making specific bounding boxes, which increases the predictions for different sizes and ratios.

Since ARAI wanted to work on Indian Traffic conditions, and YOLO is trained in western countries. I was given the task of training the YOLO algorithm for Indian Conditions.




Fig(4.17) YOLO Trained in Western countries deployed in Indian Conditions.

As you can see it detects Trucks as Bus and people on bikes as just people instead of bikes.


Fig(4.18) YOLO Trained in Western countries deployed in Indian Conditions(2).

It detects tempos as cars and trucks as bus etc.

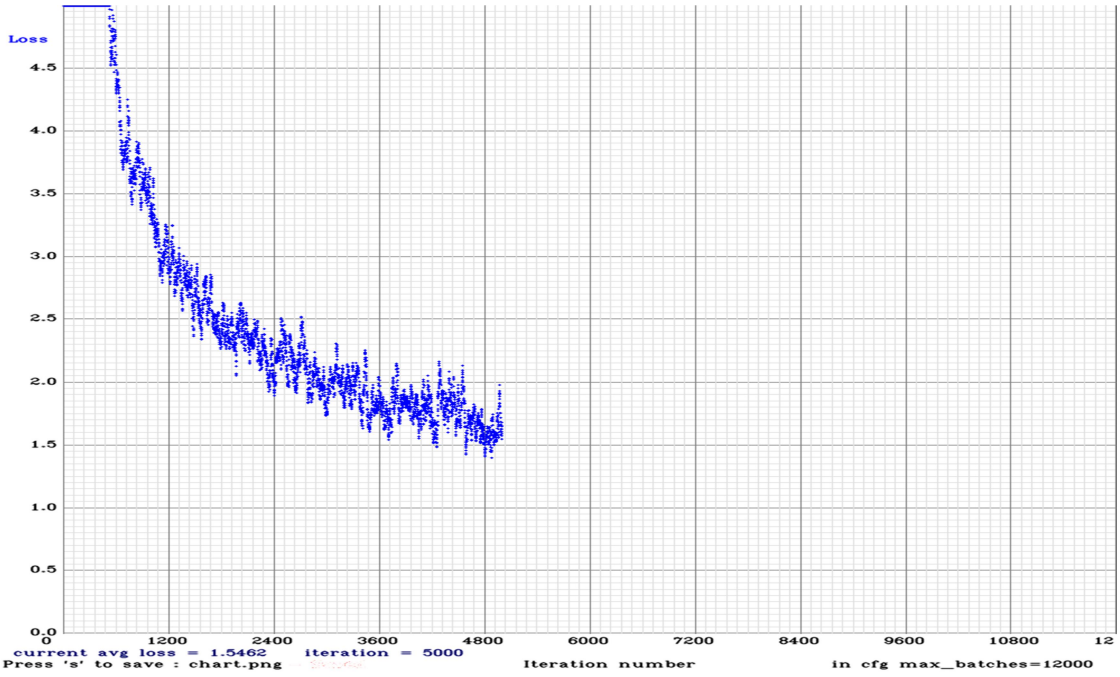
To do this, I had first to install Darknet an Open Source Neural Network framework also had install CUDA, which is a parallel computing program developed by NVIDIA for computing on GPUs. CUDA is helpful for various developers who can speed up their computing on multiple GPUs. This needed since neural networks take a long time to train on GPUS, and without a GPU, it will take a long time to train Neural Network.

I was first given 200 Images to train the network using the YOLO algorithm. The code was written for the config files and hyperparameters needed for the YOLO model.

This was then deployed, although the accuracy was terrible. It was concluded that with more data and longer training time. We would be able to train a model that would be accurate enough to work. So then I was provided with 2000 Images distributed as follows in classes as follows -Trucks, Bikes, Auto-rickshaws, TrafficSigns, traffic lights, and Tempos. Moreover, we wanted to check if augmenting data before using train the algorithm has any benefits or decreases the accuracy. To augment the data, I wrote various scripts using python and OpenCV. I used Image processing filters in OpenCV to augment the data. I wrote scripts to augment the data to gray scale, Increase Contrast, Blur, and sharpen Images.

Each of the Images carries it with an annotation file which describes where and what the image is labeled as. For example – If there is a dog in the image, then the label file contains the class Dog with dogs co-ordinates. Since I was augumenting the Images, I also needed to make appropriate changes to their annotation files. I wrote a code to parse these text files and make the necessary changes in their annotation files for each of the Images.

After doing the augmentation, I had a dataset of around 10,000 Images and their 10,000 annotation files. I trained this model with various Hyper-parameters and over multiple iterations.Hyper-parameters are parameters whose values are set up before training a model, on other settings are values which are derived while training the data. Hyperparameters are – batch size, iterations, learning rate, momentum, etc.




Fig(4.19) Graph of avg loss

Above chart is for model trained for about 5000 iterations and a loss of upto 1.5. This is a lot of loss usually models are not deployed unless the model has a stagnant loss.

The result of training YOLO from scratch for Indian traffic conditions is on left while right side is a pre-trained YOLO model.

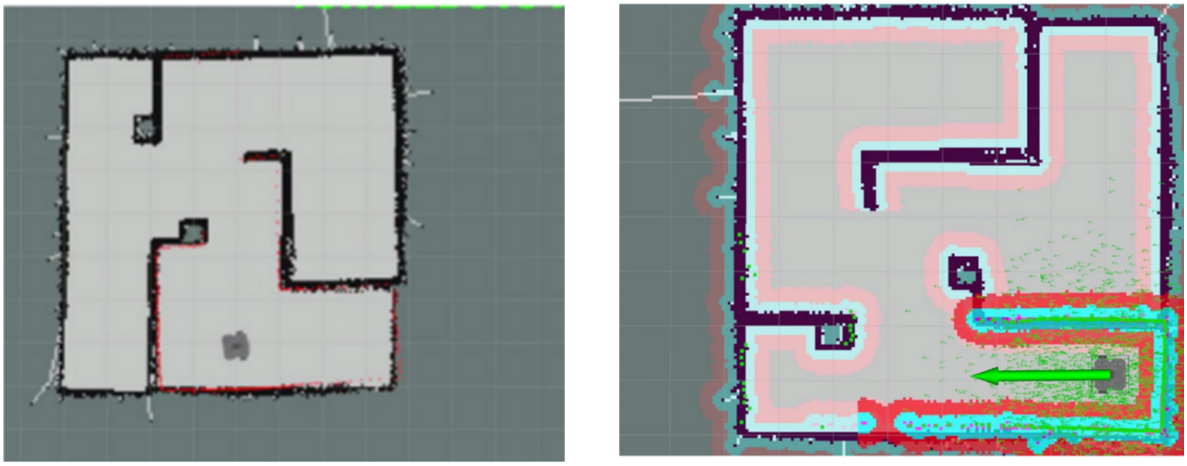

Fig(4.20) My trained model detecting Traffic-signs, trucks, tempo and bikes

A better accuracy can be achieved by using a bigger dataset I had a dataset of only 2000 images. A longer training period until the loss decreases to ideally around 0.1 loss.. It is crucial not over fit the data and use the best validation loss.

**Task- 5 Minor Tasks**

### 4.5.1) Turtlebot-

- Writing Code for Turtlebot to launch 2D LiDAR and Camera of Turtlebot. A strong knowledge of ROS ecosystem and python is required. I built a package which is able to take data simultaneously from camera and 2D LiDAR
- Running Various ROS functions like SLAM and Mapping .



Fig(4.21) Mapping and Navigation on Turtlebot3

- The green dots are available positions which the bot thinks it can reach.
- The Blue square is the range of laser scan.
- The cyan highlight are obstacles which turtlebot can sense and identify as obstacles.
- The cost map is cleared after every scan and the map is updated to remove or add obstacles on the map.

### 4.5.2) Data acquisition using LiDAR and Camera

Configured and setup data acquisition tools on SwayamGo. I did some field work for acquiring data of Pune roads in rainy season.

### 4.5.2) Wrote code for Interfacing Arduino with LCD and steering Sensor on the autonomous Car

Wrote code for Arduino which interfaces with steering sensor and displays the steering angle and wheel angle on the LCD in dashboard depending on the rotation of steering wheel.

## 5) Learning Outcomes and Experience

Internship at Automotive Research Association of India (ARAI) for the past six months has been an incredible experience in terms of my technical and professional development. My work under Mr. Ninad Sir has allowed me to learn and increase my knowledge in the fields of Robotics, AI/ML, Computer Vision, and Autonomous Vehicles. I was able to learn technologies and work on various hardware like LiDAR and Camera-based systems. I was able to learn software and frameworks like OpenCV, Robot Operating Systems, and Python. This internship has developed a keen interest in me regarding automotive technologies. Since I was the only Intern working perception pipeline project, I had flexibility and autonomy in making decisions while also having the leeway to learn from my mistakes. At ARAI, I had to chance to work with other Interns since most of the other interns were either M.Tech or graduated students. Because of this, I was able to learn from interns of different specializations.

A part of Technology Group I was able to learn from other members of the Technology group working on different technologies such as Intelligent Vehicle controller, Intelligent Energy Management System, and Hybrid Electric Vehicle programs. I was able to explore various career alternatives before my graduation and build professional work experience. The internship allowed me to turn theory into practice as some of the subjects which I learned during my college were very helpful in completing my tasks.

The internship helped me pick up new skills for a professional environment like developing healthy work habits, which are necessary for bringing success in a job. I developed communications and interpersonal skills, like having discussions with your manager and having the ability to take criticism well.

The experience that I gained at ARAI is invaluable for my career, and it was a pleasure working with the best professionals in the automotive industry.