Undergraduate (B. Tech) Project Thesis End -Term Report

Autonomous Control for Driverless Car

(Project Code: PI 1)

Submitted by

Akash Pardasani - 2015ME10629
Dhruv Talwar - 2016ME10670

Supervisor

Prof. Sunil Jha

Department of Mechanical Engineering
IIT Delhi

November 2019

Acknowledgement

We express our deep gratitude to Professor Sunil Jha, our project supervisor for his patient guidance, enthusiastic encouragement, useful critiques for this project as well as for helping us carry out extensive data acquisition and testing of experiments.

Abstract

In this report, we have discussed our objectives, approach, experiments and results for our senior undergraduate thesis on Autonomous Control of Electric Car. We have extensively discussed the background as well as our approach to solving the crucial problems for autonomous control: localization and perception. We have used ROS (Robot Operating System) as the platform to perform simulations as well as test our autonomous driving algorithms for the Mahindra e2o electric car at our disposal as a part of the Mahindra Driverless Car Challenge.

Keywords

Driverless Car, Localisation, Perception, ROS

1. Introduction

An autonomous car, also known as a self-driving car or a driverless car refers to a vehicle which relies on a variety of sensors such as camera, lidar, GPS, Inertial Measurement Unit (IMU), and sonar to perceive its surroundings and uses the input from the above sensors to navigate and drive to a desired destination without any human intervention.

With great potential to alleviate or eradicate fatalities of road accidents arising from human error, an estimate of 4.67 lacs accidents leading to 1.47 lacs deaths in 2017 [1], as well as reduce commute time on account of intelligent coordination between vehicles, autonomous cars have become a topic of great research interest across the world with universities and companies contributing to solving the technological hurdles in the way of achieving a fully self-driving car.

Autonomous vehicles can help solve major challenges faced in India, where on an average 100 minutes are spent behind the steering wheel[12], also it will lead to better safety and more efficient cars. Adaptability to Indian roads is a difficult challenge as the technology would need to cater to the difficult condition of roads, potholes, flooded roads and pedestrians, cyclists and animals. The enormous amount of data could present challenges of data security, privacy, ethics and policy dilemma.

The functioning of a driverless car be understood by drawing parallels to the role of a human driver who effectively senses the environment by visual perception and sound. He then processes the visual data by classifying the road, pedestrians, trees, pavement, weather, other vehicles and other . Based on his driving experiences in the past, a driver predicts the future state of the environment which includes the static and dynamic obstacles and takes appropriate actions.

There are a number of systems at interplay in a self-driving car control system where the sensor data is mapped to an internal state estimation via a pipeline which is then fed into the vehicle control system which handles the navigation, path-planning and obstacle avoidance aspects while using the machine learning models and data readily available through cloud access.

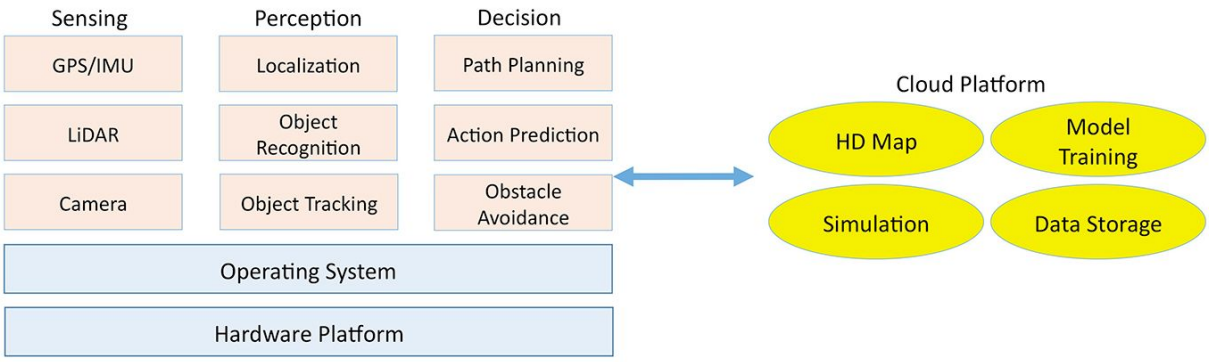


Figure 1: Autonomous Car Control Pipeline
(Image Source: https://www.oreilly.com/radar/creating-autonomous-vehicle-systems/)

Extensive work has been done in the areas of LiDAR (laser sensors) and camera based environment sensing through proximity estimation from laser reflectivity, object recognition, instance and object segmentation. The localization subtask is performed either through global localization which requires generation and maintenance of Dense 3-D Maps (for LiDAR based sensing) or Occupancy Grids and using a particle filter for real-time position estimation.

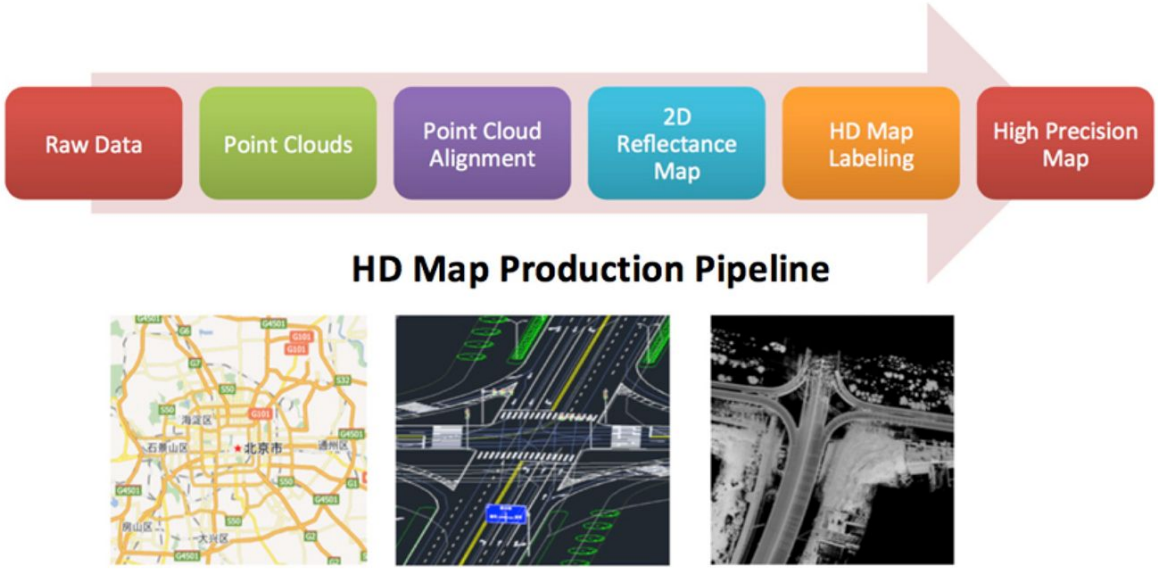


Figure 2: Localisation Pipeline
(Image Source: https://www.oreilly.com/radar/creating-autonomous-vehicle-systems/)

In parallel, motion planning in dynamic environments shared with other agents and obstacles has been a major robotics research area in the fields of decision making, route planning and feedback control.

While the research in each of the subsystems started as probabilistic decision making modeling or geometric approaches (in case of computer vision), recent advances in deep learning has directed the efforts into the development of end to end modules, requiring a properly curated set of training data examples and candidate actions to be taken in each of the scenario, and the car then uses the real-time sensor information to determine the appropriate action on its own.

The problem of control of an autonomous vehicle consists of various sub-problems or subsystems and a linear approach could lead to catastrophic results as the failure in one aspect could render the entire code non-functioning. A modular approach with proper safety checks for each subsystem in place would help in easy development and maintenance of the code. Robot Operating System which is a peer-to-peer, tools based, multi-lingual open source framework serves this purpose and it can be used for implementation of commonly used functionality, message passing between processes, and package management. [2]

With the project being a part of Rise Prize Driverless Car Challenge, we have to use the Mahindra e2o Electric Car at our disposal as the testbed. The car is installed with a secure gateway unit which uses CAN (Controller Area Network) commands to communicate with the controller and the car.

2. Literature Survey

2.1 Environment and Setup

2.1.1 Robot Operating System or ROS

Robot Operating System is a Linux based software, has many prebuilt libraries and packages which can be modified and changed accordingly to build robot functions' its framework, it used nodes, packages, messages topic and services[3]. Any executable code written which takes data from any of the robot's sensor and passes the same or the output value to another code is termed as a ROS Node. The ROS system is like a send and receive system, the data from the sensors, which are called messages are sent to nodes from ports called topics. A node that sends messages on a topic is called a publisher node and the node which subscribe to the messages sent via another topic is termed as the subscriber node. All nodes are combined in a ROS package which can very easily be combined on any Linux software with ROS installed.

2.2 3D Mapping using Cartographer

Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations[4]. SLAM (Simultaneous Localization And Mapping) enables accurate mapping where GPS localization is unavailable, such as indoor spaces or in cases requiring high precision.
It is a surveying system which measures distance to a target by illuminating the target with laser light and measuring the reflected light with a sensor, in this case a 3D LiDAR. The generated cost map or point cloud outputs from the lidar sensor provide the necessary data for robot software to determine where potential obstacles exist in the environment and where the robot is in relation to those potential obstacles[5]. SLAM algorithms use LiDAR and IMU data to simultaneously locate the sensor and generate a coherent map of its surroundings.

2.3 LIDAR for Autonomous Cars

Recent advances have led to increased interest in the 3D Maps based localization method by leading companies and research groups using a 3D LIDAR sensor. Methods include local position estimation relative to the lane edge and center marking, using variation in reflectivity between road and paints used for lane segmentations, and global position estimation using a particle filter approach with reference to a previously generated dense 3D Map [6]. The latter approach has demonstrated high accuracy and robustness for a well maintained map and is the most deployed localization method.

2.4 Localisation

Autonomous vehicles require precise knowledge of their position and orientation in all weather and traffic conditions for path planning, perception, control, and general safe operation. GPS measurements may have an errors from 1 to 10 meters. This error is too important and can potentially be fatal for the passengers or the environment of the autonomous vehicle. We reviewed the two major types of localisation - active sensor based and passive sensor based[7]. The localisation and mapping capabilities are required to model the surrounding environment and to plan a safe and efficient route for the autonomous driving.

2.5 Experience from DARPA Challenge Reports

A review of the reports from DARPA Grand Challenge 2004 and DARPA Urban Challenge 2007 helped us to understand the hierarchical architecture of a self-driving car control system and further develop and refine our approach as well [8][9]. The modular approach followed by the teams and their experience helped us to understand the problems faced in developing the control system architecture and find their prospective solutions.

3. Project Objectives and Work Plan

3.1 Problem Definition and Motivation

As a part of the Mahindra Driverless Car Challenge, this project aims at contributing to the tasks and challenges involved in developing a fully autonomous self-driving car. The project is a part of Team dLive of IIT Delhi which is participating in the Mahindra Driverless Car Challenge.

3.2 Objectives of the Work

The current objective of the project is to achieve autonomous control of the Mahindra e2o electric car by solving the following challenges and tasks:

1. Generation of a dense 3D Map for IIT Delhi Campus using LIDAR data to be used for localization using particle filter approach
2. Accurate Localization of the vehicle in the IIT Delhi Campus map
3. Development of a Qt based desktop application for real time analysis of the information from the various sensors on the car
4. Segmentation of 2D LIDAR data to detect roads and lanes

3.3 Methodology

We started with familiarizing ourselves with ROS packages for Velodyne data capture, analysis and manipulation. Then we focused on getting familiar with the Google Cartographer system for generation of 3D Maps from LIDAR data and used the data generated through previous test drives of the Mahindra e2o electric car as well as acquired fresh data through further test drives for the same. After obtaining the 3D maps and occupancy grids, we implemented the ROS packages for AMCL localisation as well as Cartographer's native localisation on the same. We, parallely, also worked on implementing the Qt based GUI and tested it's efficiency by analysing the data captured during test drives. Upon successfully obtaining localisation the 3D maps for different parts of the campus, we familiarized ourselves with Python Matplotlib libraries to analyse the data captured by 2D LIDAR for detecting drivable roads for the car and implemented the same.

We also familiarized ourselves with the ROS and Gazebo environment to simulate the environment for a driverless car and test out our mechanisms and controllers.

The methodology can be primarily broken down as:

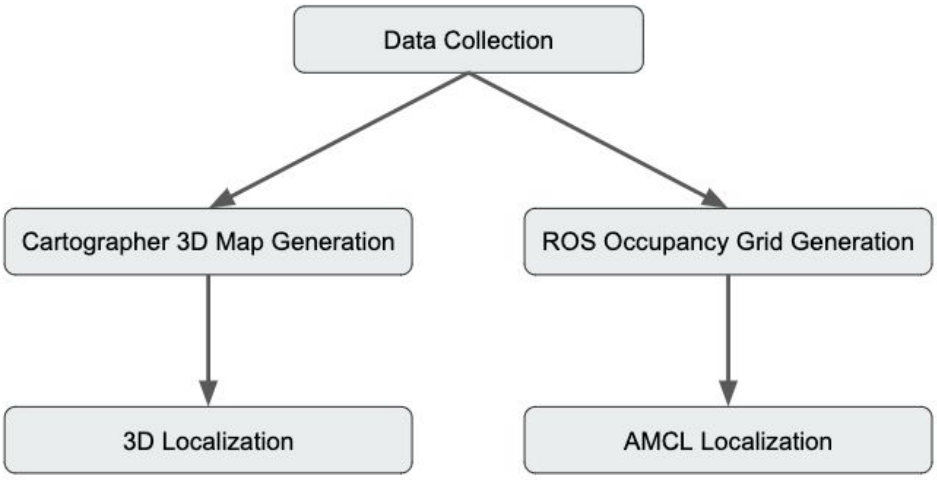


Figure 3: Methodology

4. Work Progress

4.1 Relevant Theory

4.1.1 Particle Filter

In probabilistic robotics belief is represented by conditional probability distributions. It assigns for every possible state a probability. To determine the belief of state bel(x_t) we need to consider all sensor and action data that happened before time t. This is denoted in the following equation:

$$bel(x_t) = p(x_t, |z_{1:t}, u_{1:t}) \quad (1)$$

Here $z_{1:t}$ is all the sensor data from time step 1 until time step t. The control data, actions by which the robot changes the world, is denoted as $u_{1:t.\ The}$ key idea of the particle filter is to represent the belief bel(x_t) by a set of samples drawn from bel(x_{t-1}) . The samples of a distribution are called particles and are denoted as:

$$X_t = x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \quad (2)$$

In equation each particle $x_t^{[m]}$ with (1<m<M) is a hypothesis to what the true world state may be at time t. X_t is the set of all M particles. Because a particle filter is to approximate the belief bel(x_t) by the set of particles X_t, the equation is as follows:

$$x_t^{[m]} \sim p(x_t, |z_{1:t}, u_{1:t}) \quad (3)$$

The particle filter algorithm constructs the belief bel(x_t) recursively from the belief bel(x_{t-1}) one timestep earlier. So, since the belief is represented by the particle set as in equation 9 the particle filter construct a new set of particles X_t from the set X_{t-1}. The particle filter has two important parts. The first part generates a hypothetical state $x_t^{[m]}$ on time t based on the particle $x_{t-1}^{[m]}$ and control u_t. This step involves sampling from the next state distribution $p(x_t, |u_t, x_{t-1})$. Then the importance factor $w_t^{[m]}$ is calculated based on measurement z_t and the calculated particle $x_t^{[m]}$.

The importance factor is interpreted as the weight of each particle. The resulting $\bar{X}_t$ is a temporary set with weighted particles. The second part is called the importance resampling and redraws a new set X_t with replacement from the temporary set $\bar{X}_t$ and contains only the particles with a high probability. So over time the set is converge to the hypothesis that produces the most likely output. The particle approximation is the core part of the MCL algorithm, which itself came by with some new improvements. The advantages of the algorithm is that it copes well with noisy data, it concentrates only in interesting space regions[10].

4.1.2 Adaptive Monte Carlo Simulation

GPS alone is not accurate enough to be used for localisation of self-driving cars as it is noisy in urban environments on account of bouncing of signals from the buildings as well as an accuracy of 5 metres under an open sky is insufficient for solving crucial problems like path planning and obstacle or pedestrian avoidance. We decided to use a landmark based particle filter localization approach for precise tracking of the vehicle using the information from the Velodyne VLP-16 LIDAR mounted on the car.

In this study we have used the most popular localization technique which is by the Adaptive Monte Carlo approach. Monte Carlo localization (MCL), also known as particle filter localization, is an algorithm for robots to localize using a particle filter. Given a map of the environment, the algorithm estimates the position and orientation of a robot as it moves and senses the environment. The algorithm uses a particle filter to represent the distribution of likely states, with each particle representing a possible state, i.e., a hypothesis of where the robot is. The algorithm typically starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about where it is and assumes it is equally likely to be at any point in space. Whenever the robot moves, it shifts the particles to predict its new state after the movement. Whenever the robot senses something, the particles are resampled based on recursive Bayesian estimation, i.e., how well the actual sensed data correlate with the predicted state. Ultimately, the particles should converge towards the actual position of the robot.

These particles each have their own coordinate values and orientation just like the actual robot along with a given weight. The weight value (w_t) is defined as the absolute difference between the actual pose of the robot and the predicted pose by that specific particle. The bigger or larger the weight of the particle, the more accurately it defines the pose of the robot. Whenever the robot moves in the environment and gives new sensor data the particles are re-sampled. With each re-sample, particles that have low weights perish and the particles with high weights survive. After many iterations of the AMCL algorithm, the particles will converge and evaluate an approximate of the robot's pose This algorithm estimates the robot's orientation and position based on the sensor's input to the algorithm[10].

TableI AMCL Pseudo code -Algorithm

1: **procedure** AMCL(x_{t-1}, v_t, s_t)

2: $X_t \leftarrow \phi$

3: for n =1 to N loop:

4: $x_t^{[n]} \leftarrow$ Motion Update(v_t, $x_{t-1}^{[n]}$)

5: $w_t^{[n]} \leftarrow$ Sensor Update (s_t, $x_t^{[n]}$)

6: $X_t \leftarrow X_t + < x_t^{[n]}, w_t^{[n]} >$

7: end for

8: for n=1 to N loop:

9: draw $x_t^{[n]}$ with probability $\propto w_t^{[n]}$

10: $X_t \leftarrow (X_t + x_t)$

11: end for

12: return X_t

The algorithm can be broken down into 2 steps Motion movement and sensor update and resampling Process. In this algorithm the robot's pose is represented by a belief factor (X_t). Initially, all the particles have equal weight (w_t)and the belief state are estimated by randomly generating N particles. The Algorithm takes into consideration the previous belief state values X_{t-1}, the movement command V_t, and the sensor measurement S_t as the initial starting input. In the first iteration of the AMCL algorithm, a hypothetical state is calculated as the initial position of the robot, As the robot's position is updated the laser scanner sends different and updated measurements to the ROS nodes, using these measurements, the particles' weight is recalculated. The result of this iteration are added to the previous belief state and thus getting a new belief state which corresponds to the location of the robot. [10][11]

In the other part of the algorithm, resampling of the particles are done, as each belief state has a weight value associated with it, after each resampling iteration the particles that have a high weight value survive and are then used in the next iteration while those particles with a low weight value perish during the resampling. Finally, with the new sensor measurements the algorithm estimates the position of the robot by calculating the belief state. [10]

4.2 Experiments and Setup

4.2.1 Sensor Data Acquisition

We conducted 5 test drives of the Mahindra e2o car in the IIT Campus for gathering sensor data from an array of sensors: 3D Lidar, 2D Lidar, IMU and GPS. The data was consolidated into rosbag files which were later analysed and implemented upon. For some aspects like point cloud map generation, we had to split as well as filter the bag files to work within the memory limits of Cartographer.

4.2.2 3D Map Generation

With the 3D velodyne point cloud data collected from the test drives, we generated dense 3D maps as well as 2D maps or occupancy grids of different sections of the IIT campus. We obtained pbstream files for the same, ply files for point cloud data visualisation and pgm files for ROS compatible map formats (occupancy grids) for performing localisation in ROS. We also performed extensive tuning of the Cartographer platform for adopting it for the specific sensors in our case as well as obtaining high quality SLAM performance.

4.2.3 AMCL Localisation

We implemented AMCL localisation package on the ROS compatible maps we obtained using Cartographer to obtain the car's pose relative to the map frame. We also compared the performance of AMCL package with Cartographer's inbuilt localisation for determining the best tool for localisation for our project. We also tuned the parameters for both, amcl localisation node and cartographer localisation node in ROS, for obtaining greater precision and accuracy. Using the dead reckoning position returned by the amcl node, we calculated the car's location and compared it with the true value captured by the GPS and IMU during the test drives. Also, we calculated the variance between the calculated trajectory returned by Cartographer localisation node for comparison with actual trajectory captured during map generation.

4.2.4 Qt based Sensor Analysis Desktop Application

We built a PyQt5 based sensor data visualization GUI for real-time analysis of various sensors information. We used GPS coordinates in (Latitude, Longitude) and converted it into Universal Transverse Mercator (UTM) system coordinates for accurate plotting of points on Google Map of IIT Delhi Campus. We did the same using utm libraries in Python and incorporated the Python application with ROS for acquiring the Pixhawk GPS coordinates and plotting it using a PyQt5 application on a Matplotlib window.

4.2.5 Road Detection using 2D LIDAR

We mounted a 2D LIDAR on the bonnet of Mahindra e2o car and used the variance in the intensity at different angles to determine the position of the car relative to the road. We tested our hypothesis of obtaining significant difference between intensity in reflections from the road surface, curbs and surrounding ground. We obtained the 2D LIDAR data from the /scan topic in the rosbag files and filtered the intensity readings for plotting it in Matplotlib window for different time instances using Python.

4.3 Results and Discussion

4.3.1 Google Cartographer 3D Maps Generation

We tested the Google Cartographer platform with the data acquired from the several test drive to generate 3D maps. We successfully generated the 3D map along with XY, YZ and ZX projections of the map in the respective planes. We also got the height variations of the road via the map. The route that the car took while recording this bag file was clearly visible in the map generated. Using the data, we generated the maps of different locations in the campus. The Blue line in each map indicates the path followed by the car on the map, which is very precise to the actual path taken. In the map we are able to visualize the stationary cars, the infrastructure and the minor details of the environment. Cartographer was able to make better maps when the environment had a lot of features such as buildings, walls nearby. The following show the different maps that we generated.

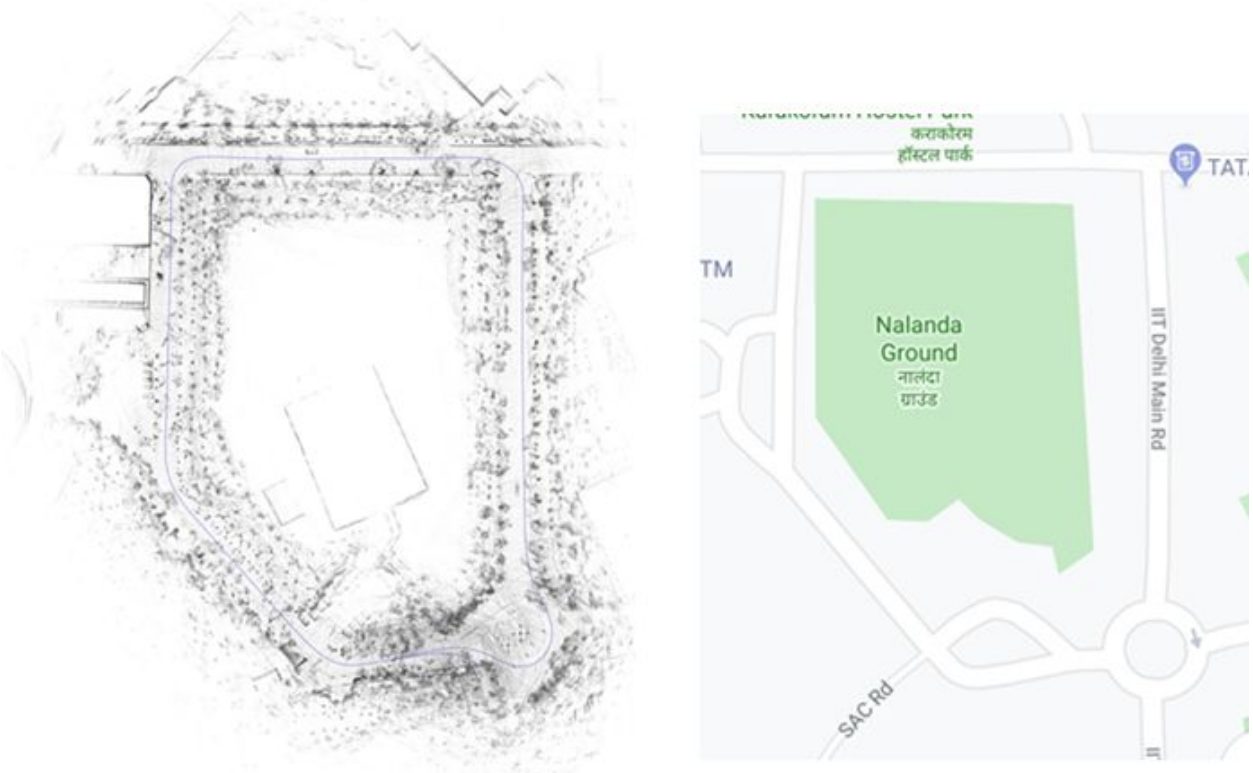


Figure 4: Cartographer Map of IIT Delhi, SAC Circle Area (Google Maps for comparison)

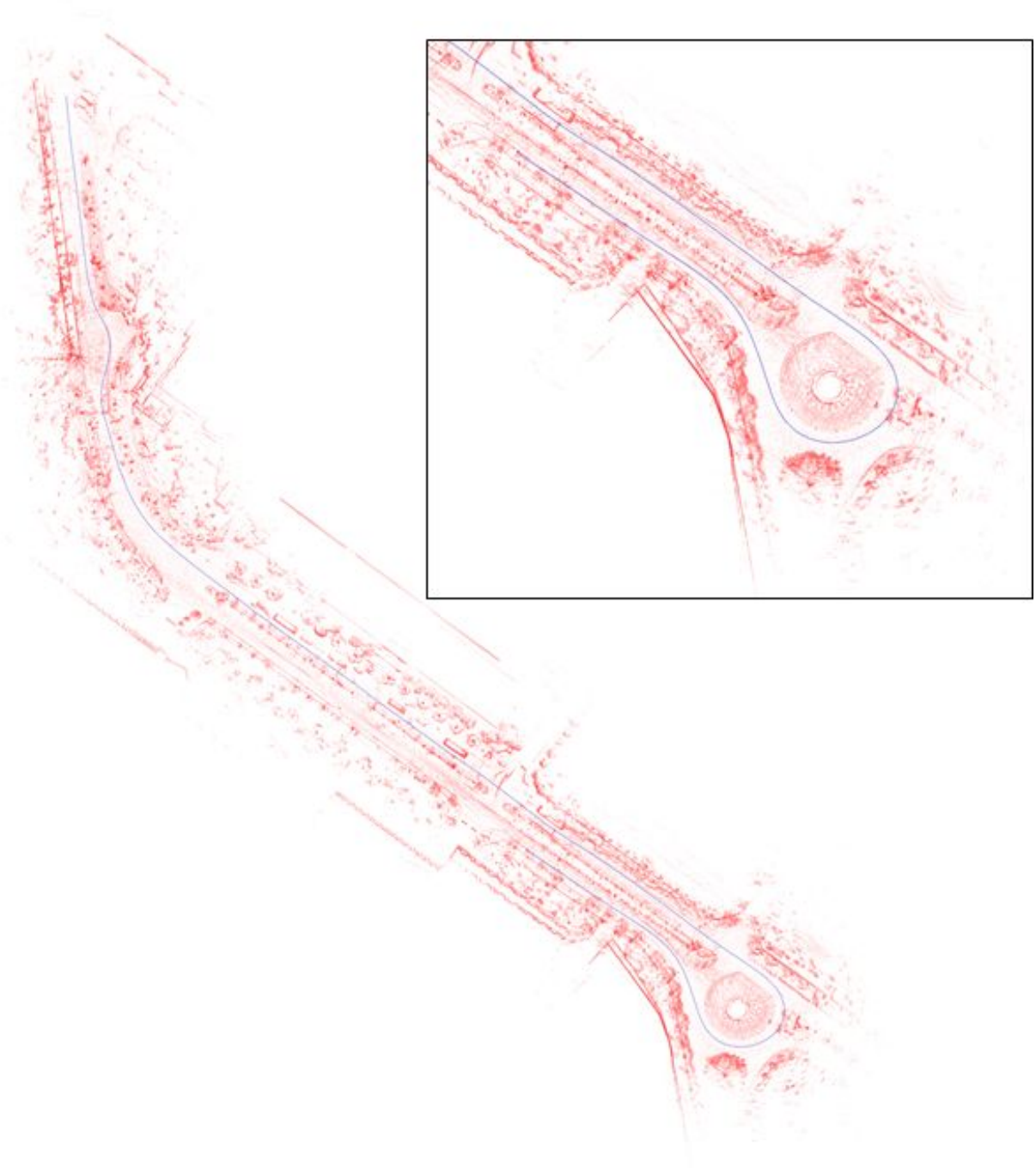


Figure 5: Cartographer Map, IIT Delhi Campus, Himadi Circle

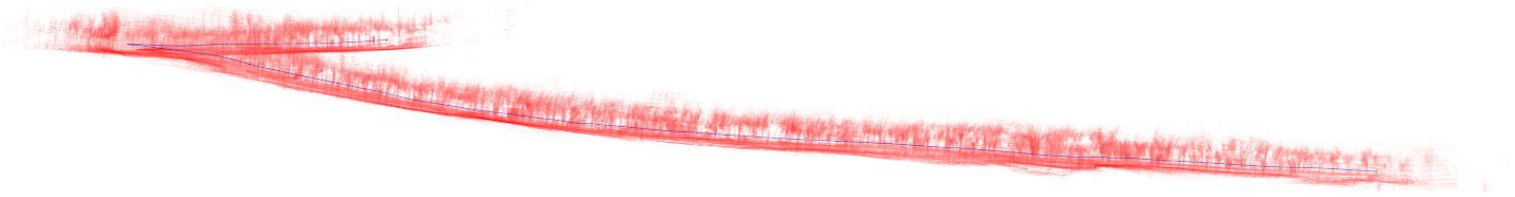


Figure 6: Cartographer Map, XZ orientation

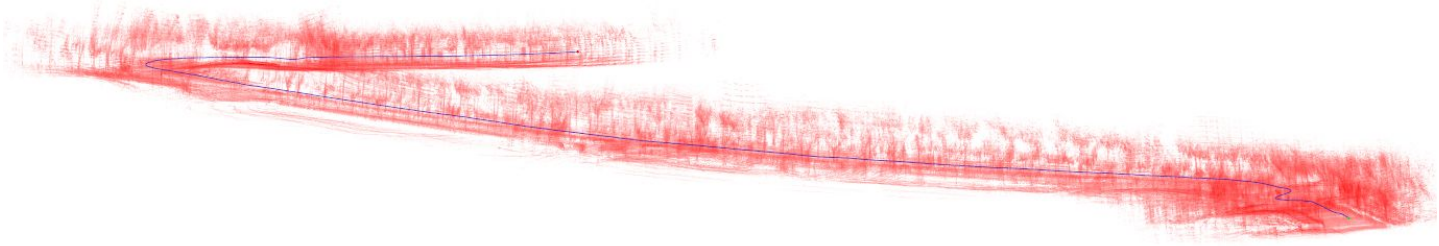


Figure 7: Cartographer Map, YZ orientation

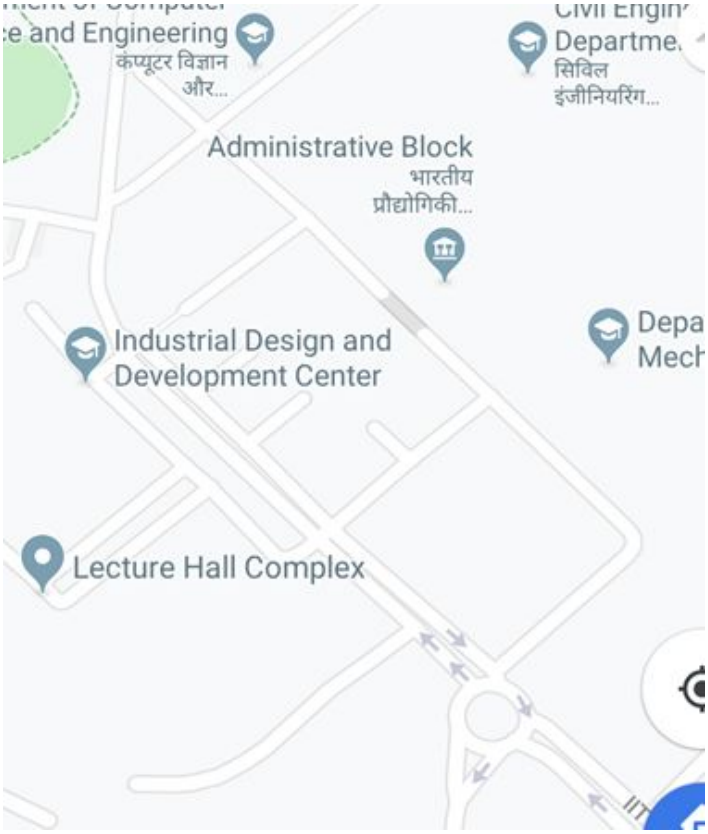


Figure 8: Google Map, IIT Delhi Campus, Himadi Circle

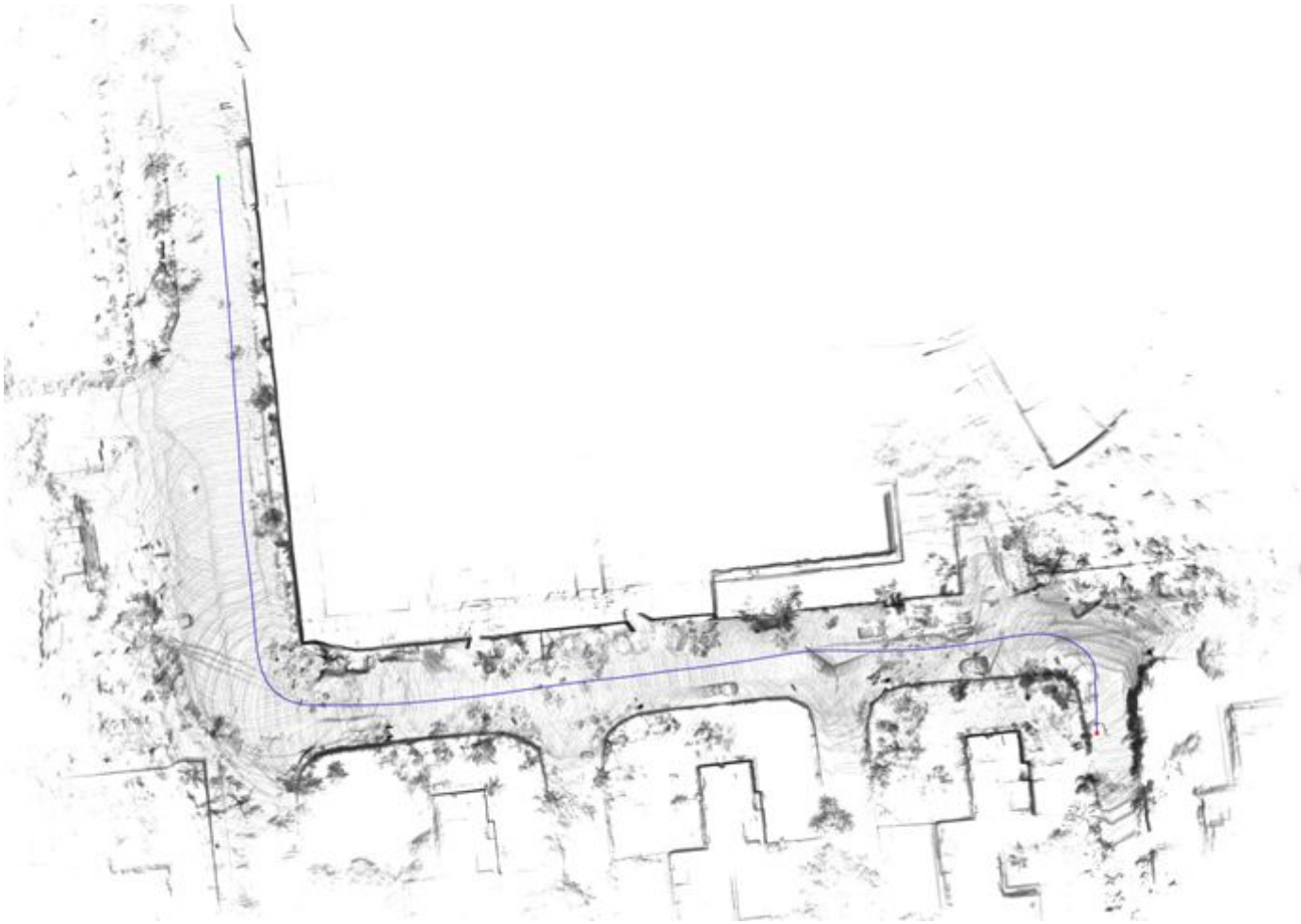


Figure 9: Cartographer Map, IIT Delhi, New Campus Area

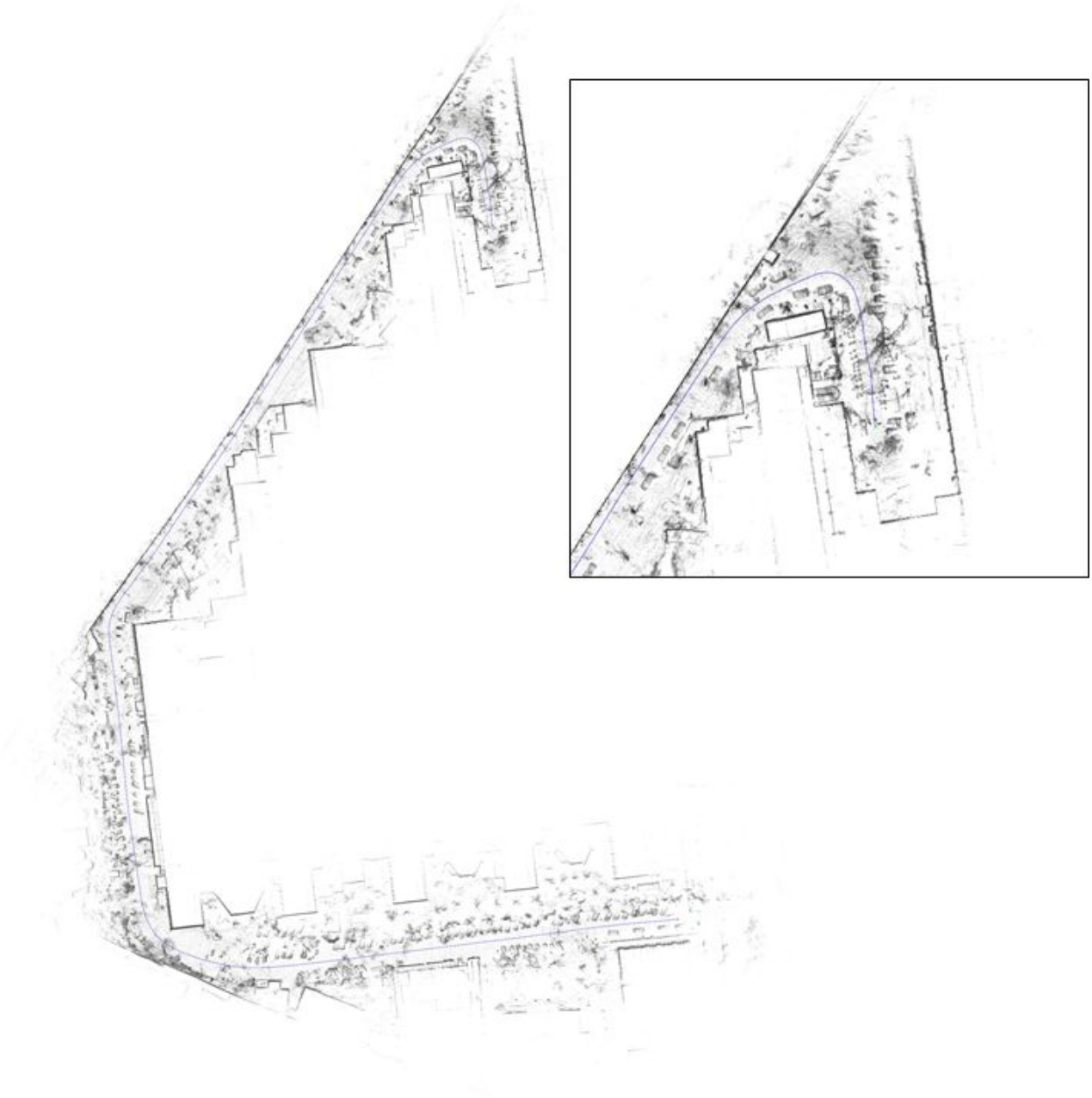


Figure 10: Cartographer Map IIT Delhi Campus, Blocks Area

4.3.2 ROS Map generation

Using the same test bag files we were also able to build ROS compatible maps.
These are 2D maps or occupancy grids which do not show height variation but show the obstacle boundaries in the environment. The map describes the occupancy state of each cell of the world in the color of the corresponding pixel. In the standard configuration, whiter pixels are free, blacker pixels are occupied and the pixels in between are unknown. The ROS occupancy map is not as accurate as the google cartographer map but it is used as a simple representation or blueprint of an environment.

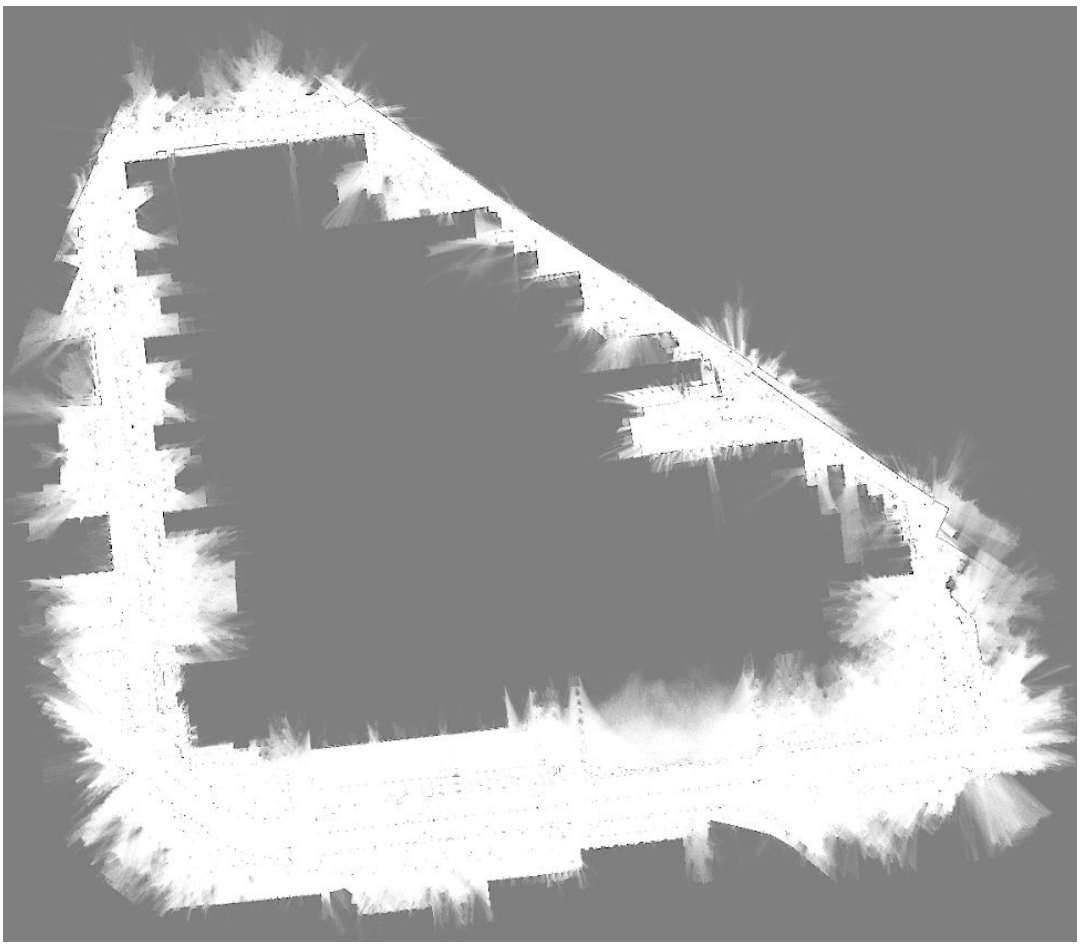


Figure 11: ROS Map showing IIT Delhi Campus, Blocks Area

Figure 12: ROS Map showing IIT Delhi campus, Main Road

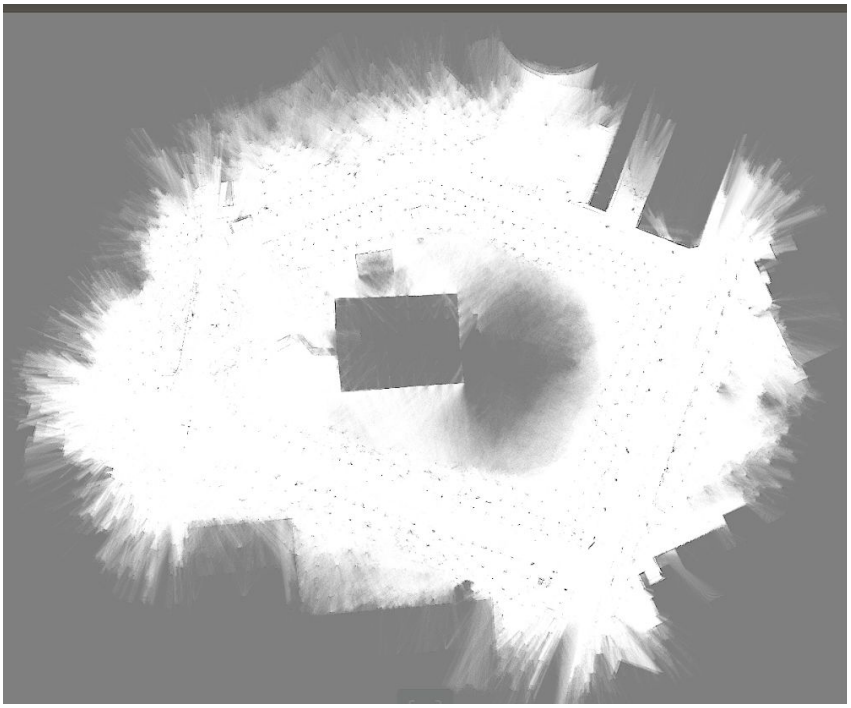
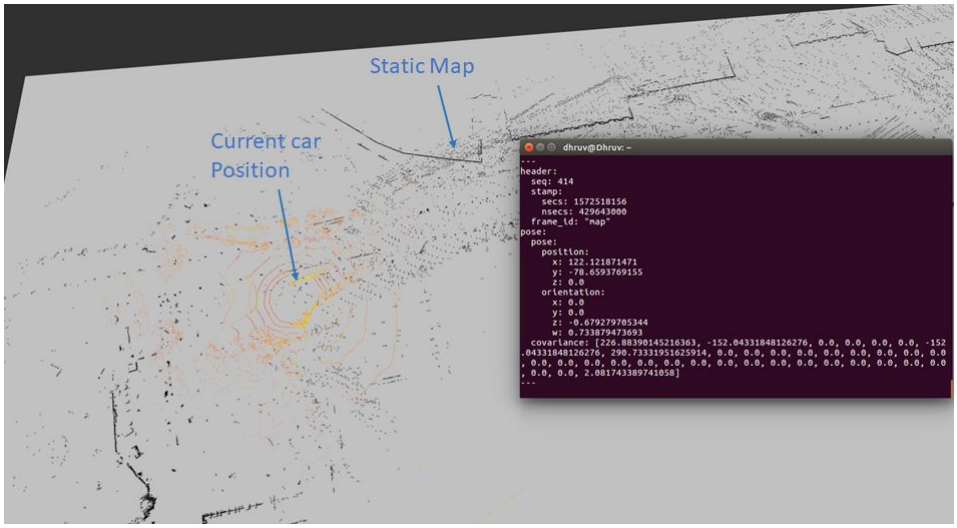


Figure 13: ROS Map showing IIT Delhi Campus, SAC Circle

4.3.3 AMCL Localisation

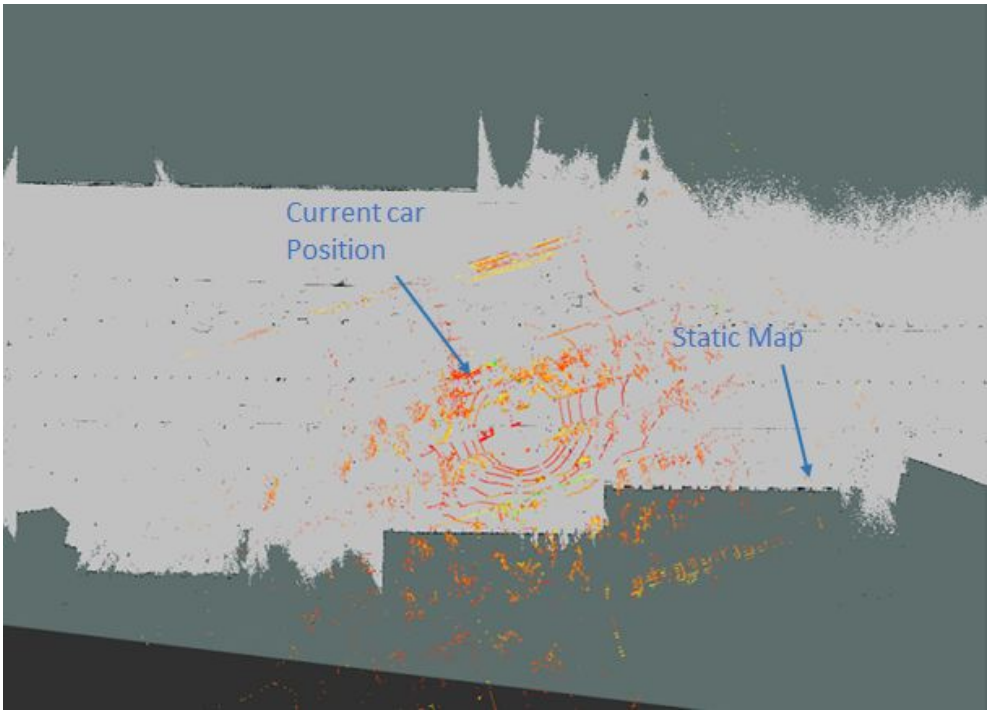
Using the ROS compatible maps obtained in the previous step, we implemented Adaptive Monte Carlo Localisation (AMCL) localisation package which is based on the Particle Filter approach to localise the car in the maps of the campus. We were able to obtain the X, Y and Z coordinates of the car in the map frame and the quaternion using which we could find the roll, pitch and yaw of the car.

(a)

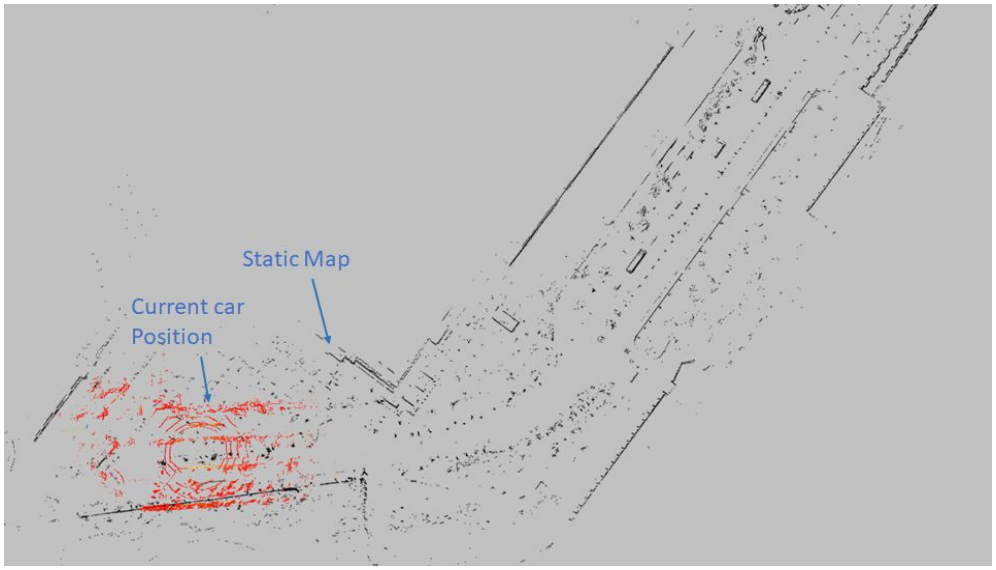


(b)

Figure 14: Pose of the car obtained from AMCL algorithm



(a)



(b)

Figure 15: Current car location with respect to the static ROS map

4.3.4 Cartographer Localisation

We also tested Cartographer's inbuilt localisation package on the bag files for offline localisation of the car in the 3D maps. The blue coloured trajectories denote the actual trajectory of the car during the course of the experiment and the green trajectory represents the output trajectory of the Cartographer localisation package. We were also able to obtain the coordinates of the points of the trajectory as it is generated along the way with respect to the map frame.

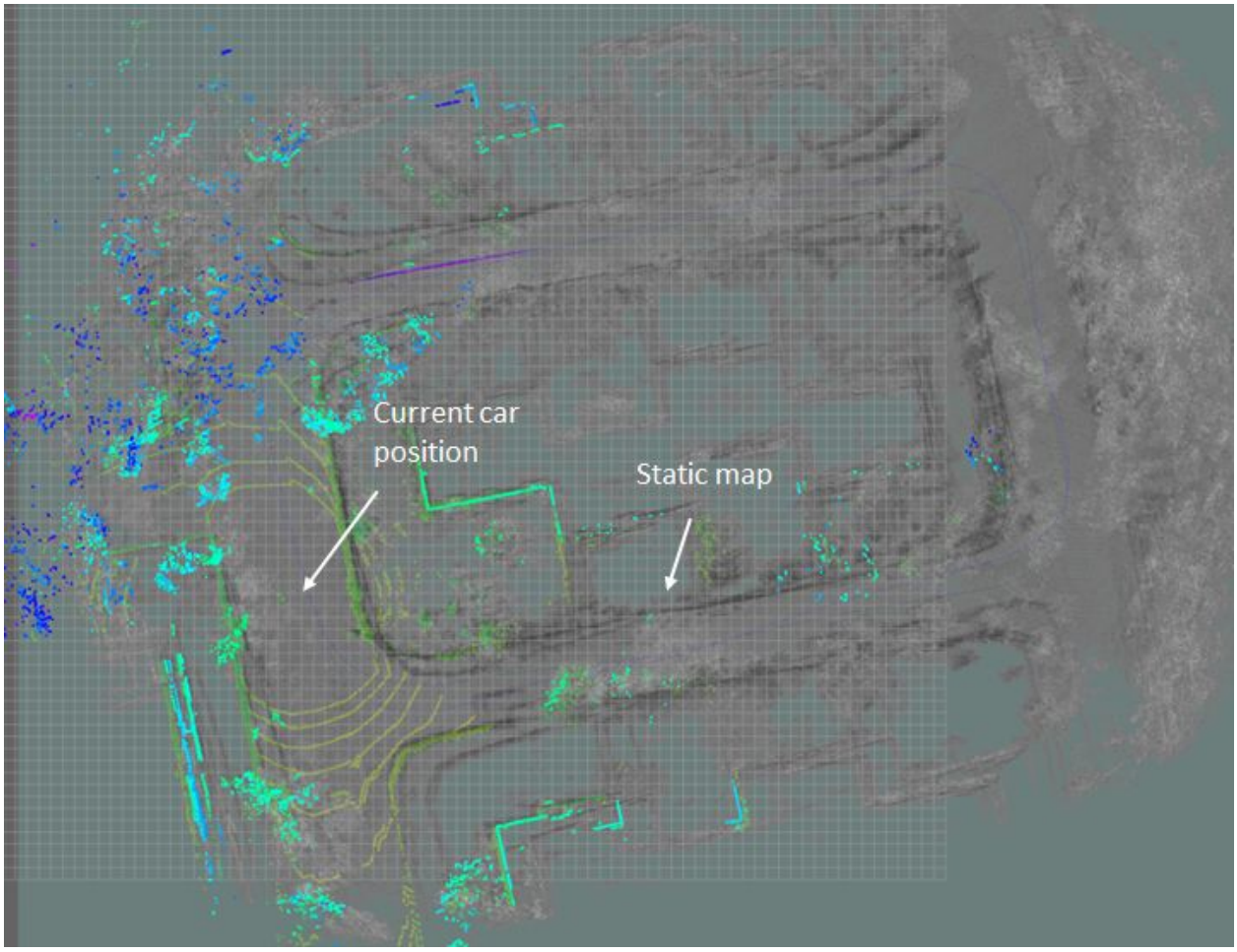
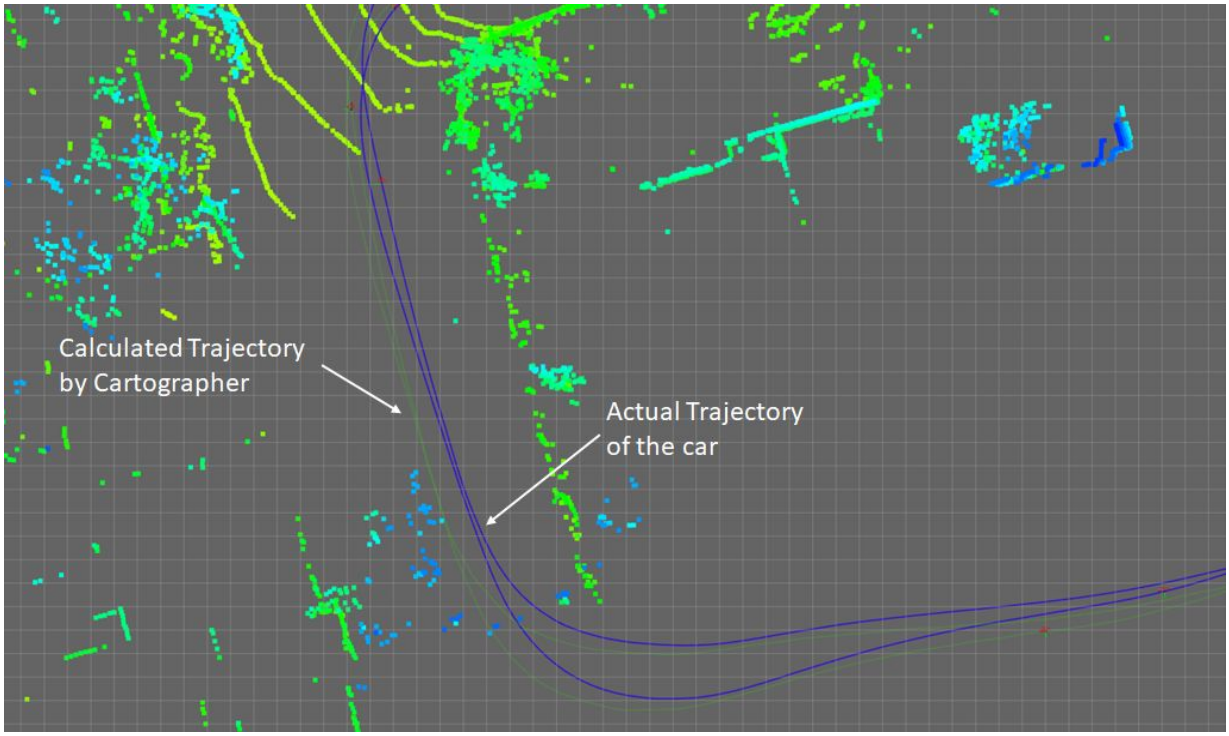


Figure 16: Car's position in static Cartographer map in Rviz (IIT Delhi, New Campus Area)


Calculated Trajectory
by Cartographer
Actual Trajectory
of the car

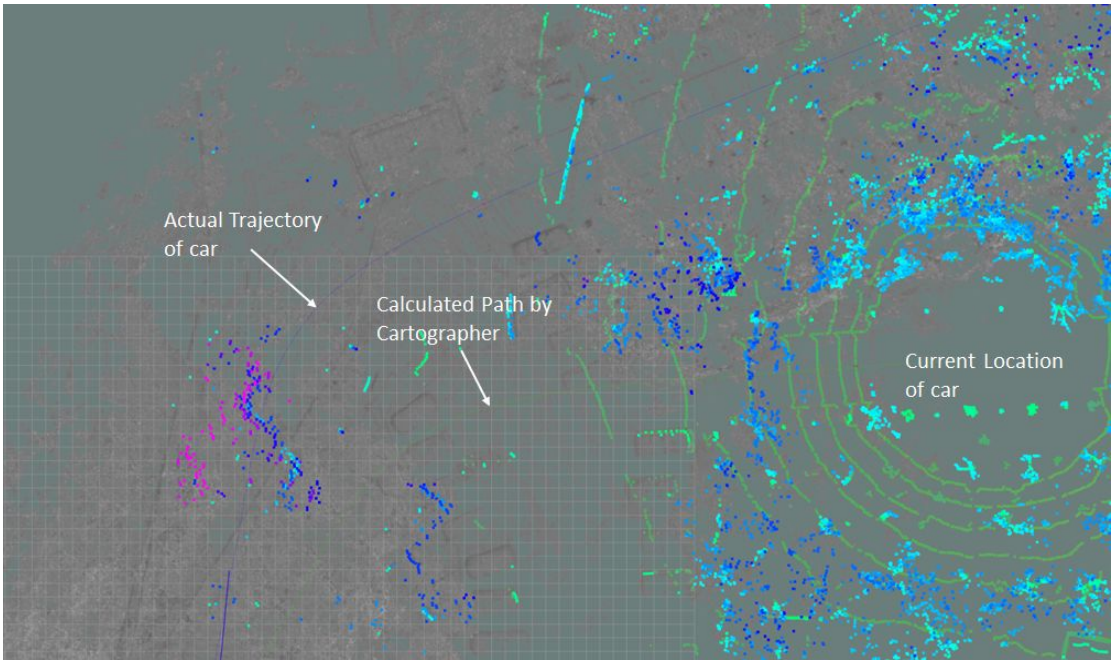
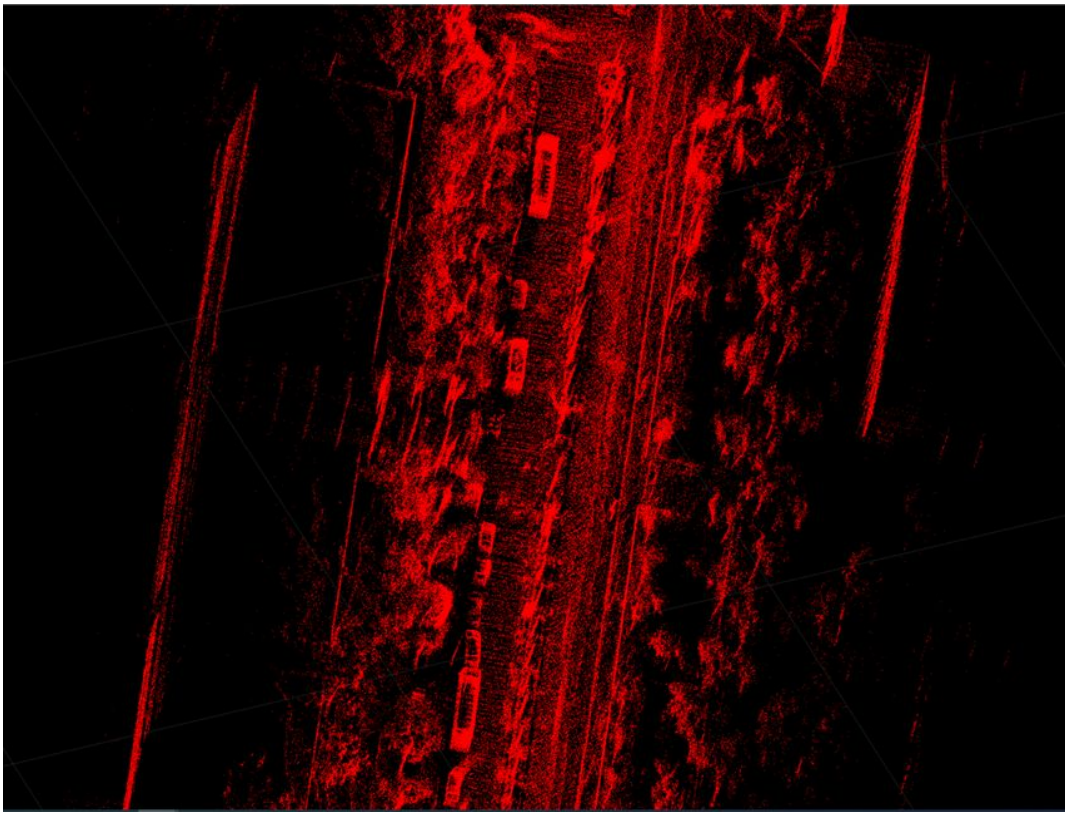

Actual Trajectory
of car
Calculated Path by
Cartographer
Current Location
of car

Figure 17: Comparison of Actual Trajectory and Calculated Trajectory of the car in Rviz

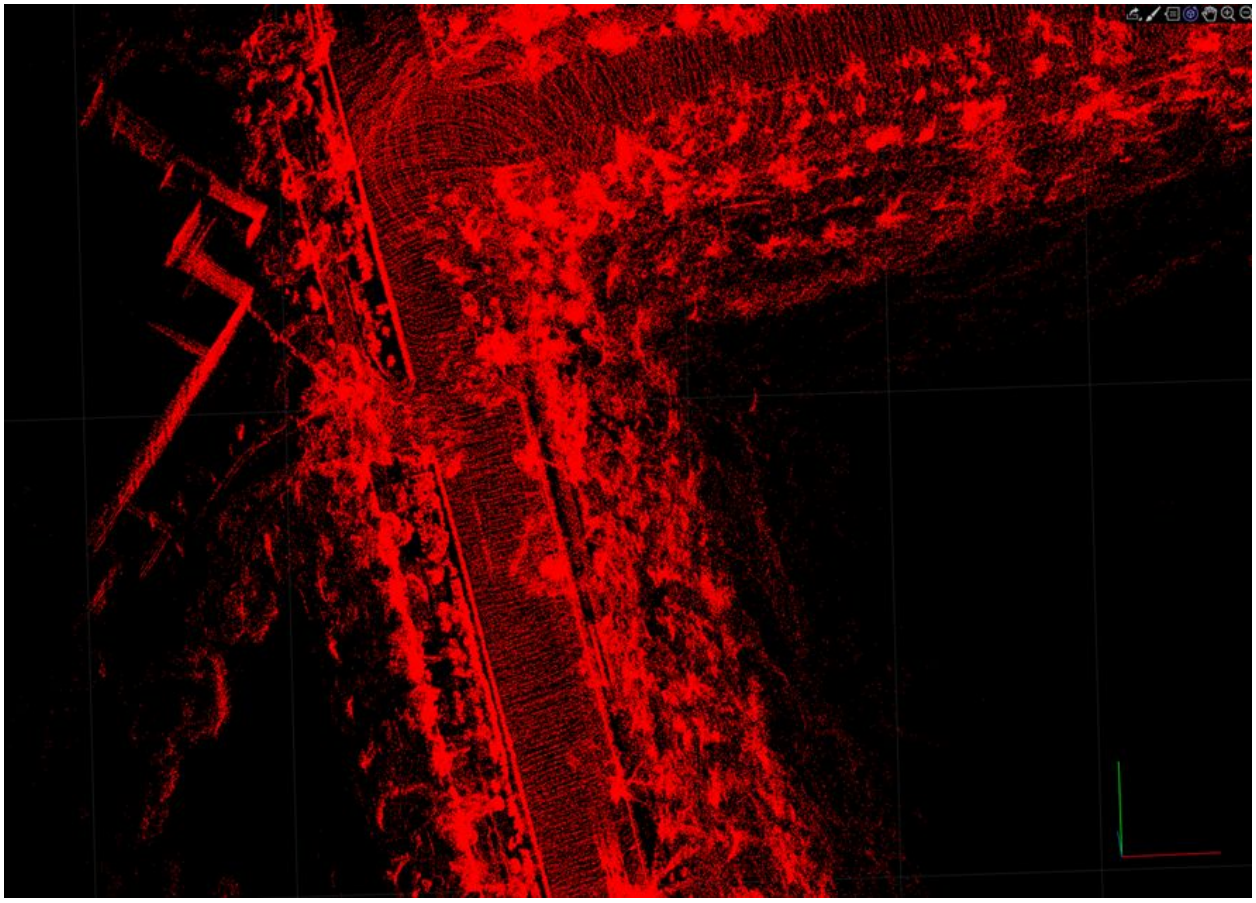
Figure 18: Rviz Output of Cartographer Localisation

4.3.5 3D Visualization of Point Cloud Maps

We obtained ply files using Cartographer and visualised the same using Matlab. .ply files are the representation of the point cloud map generated by the Velodyne LIDAR during the course of data collection. We obtained 3D maps with information of intensity recorded after laser reflection in each direction of the LIDAR.



(a)

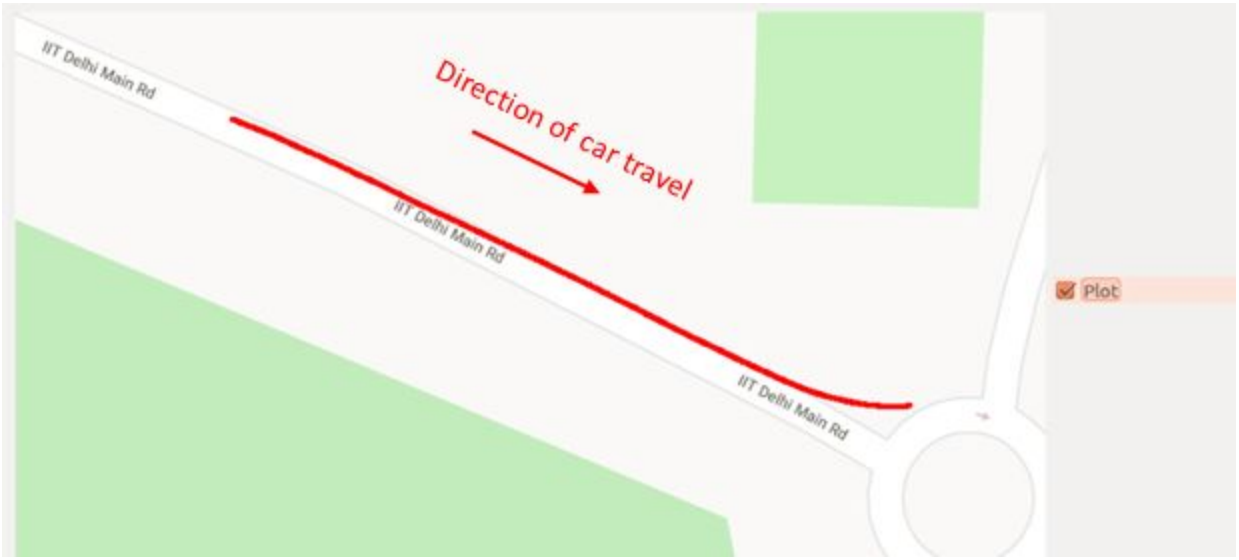


(b)

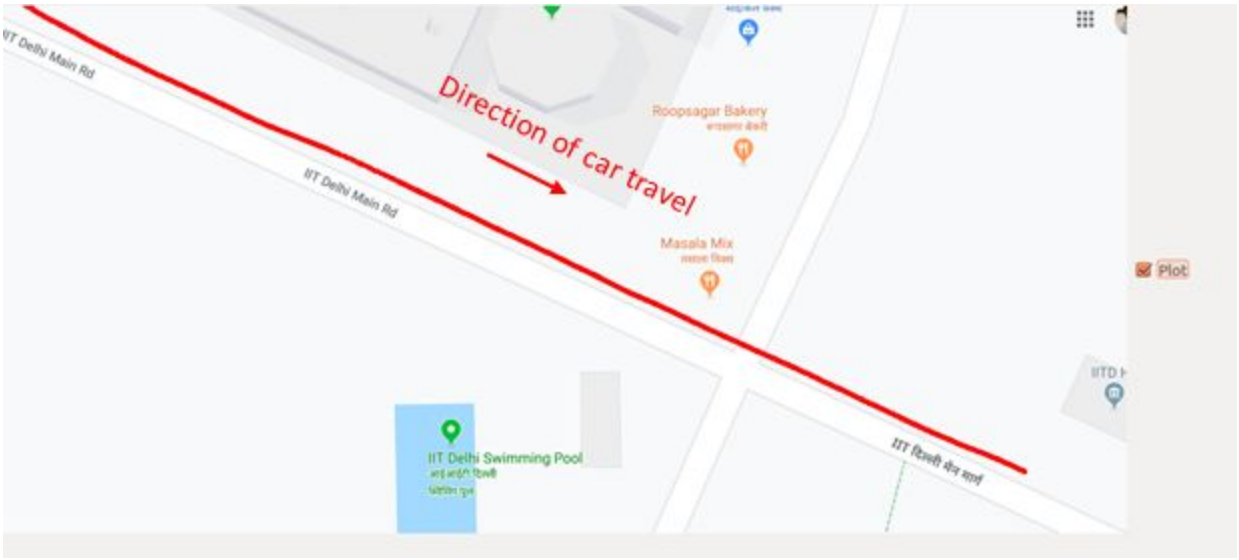
Figure 19: Visualisation of ply files in MATLAB

4.3.6 Qt based Sensor Analysis GUI

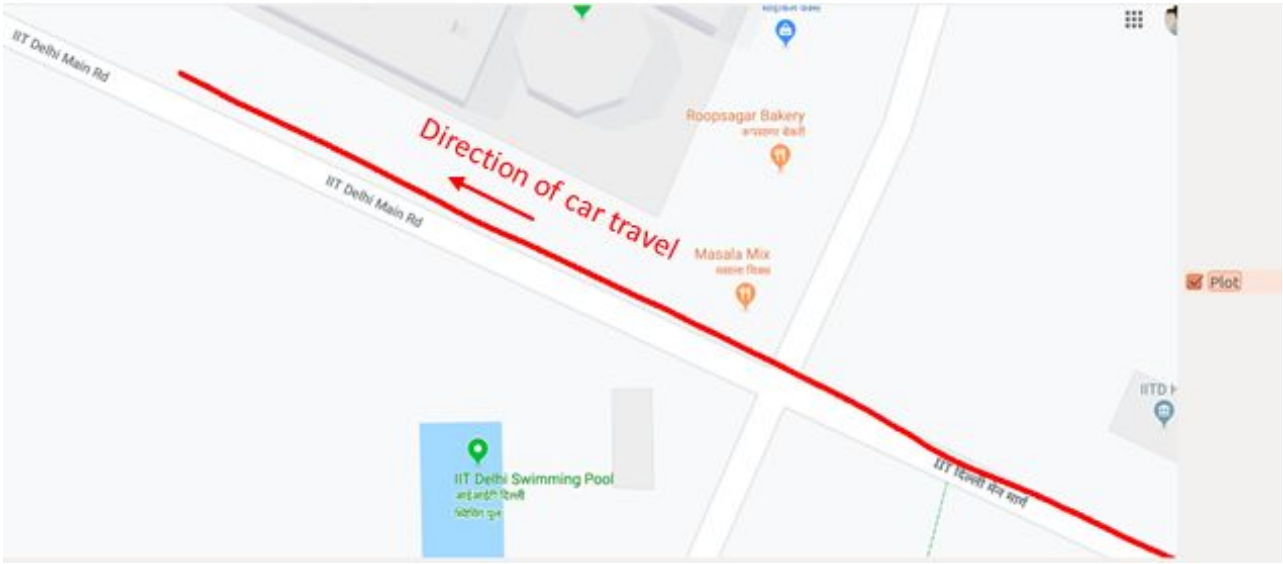
We plotted the GPS coordinates published by the Pixhawk IMU mounted on the car. We used Python libraries to convert the (Latitude, Longitude) data in UTM coordinate system for plotting on the map. In order to analyse the accuracy of the data, we used the Google Map of the specific parts of the campus set to maximum zoom level as the background tile.



(a)



(b)



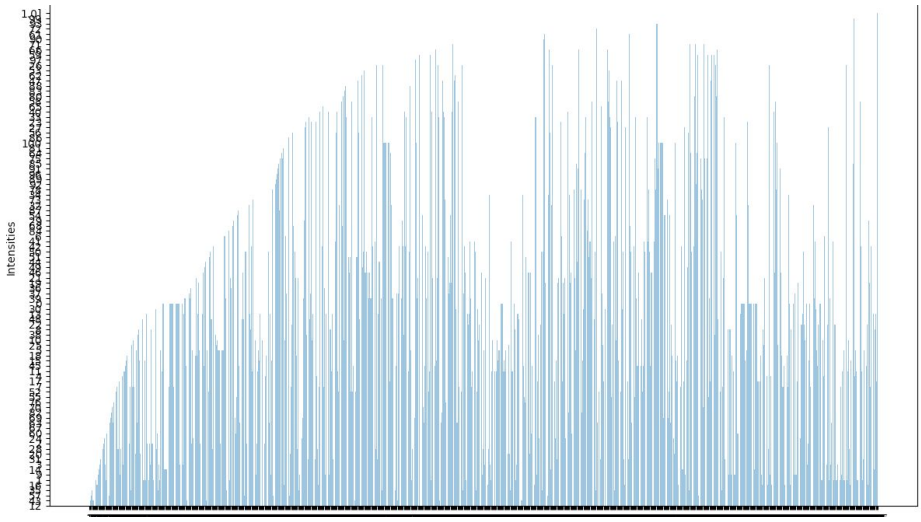
(c)

(d)

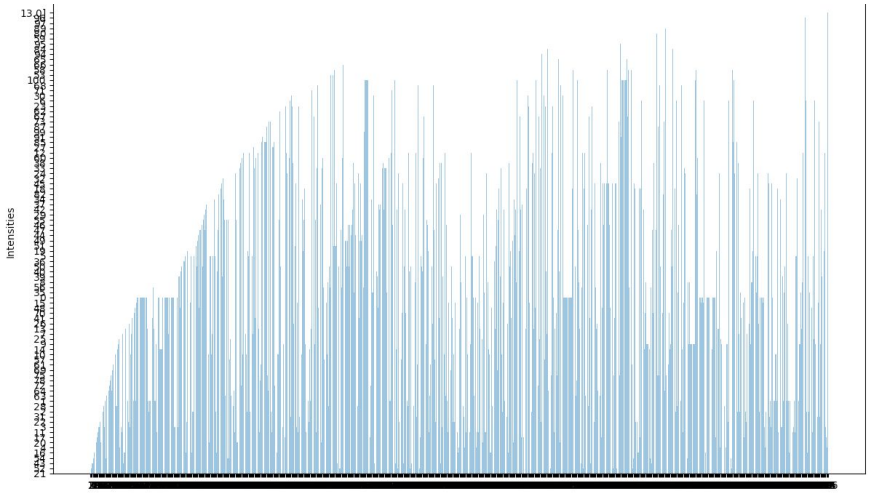
Figure 20: PyQt5 GUI with GPS Plot on Google Maps Tile

4.3.7 Road Detection using 2D LIDAR

We filtered the raw intensity values from the rosbag files recorded by the 2D LIDAR during the test drives and plotted the same for the duration of the test drive on a Matplotlib window using Python. The x axis denotes the range of the 2D LIDAR and the y axis represents the intensity recorded in the particular direction.



(a)



(b)

Figure 21: 2D Lidar Intensity Plot

5. Conclusions

With the experiments targeted to solve each of the subproblem of the Mahindra Driverless Car Challenge, we demonstrated our overall approach and documented the results of each of the experiment. Based on the results of 3D map generation described in the Results chapter, we conclude that the same approach should suffice for highly accurate localisation of the car and further tuning of Cartographer localisation as well as AMCL localisation for occupancy grid would provide results which can then be integrated with motion planner algorithms. Although the experiment for road detection using 2D LIDAR did not provide conclusive results, further testing of the hypothesis in conjunction with the vision based module for the same should be adequate for integration with motion planner modules. We suggest rigorous testing of both the subsystems, localisation and drivable road detection, in simulation and experimental settings for identification and resolution of any errors pertaining the same. The Qt based GUI setup has been done and information from other sensors for visualisation can be easily done as and when needed. We have also documented our approach for 3D LIDAR based localisation as well as 2D LIDAR based localisation for future use.

6. References

[1] India way behind 2020 target, road accidents still kill over a lakh a year
http://timesofindia.indiatimes.com/articleshow/65765549.cms?utm_source=contentofinterest&utm_medium=text&utm_campaign=cppst
[2] Morgan Quigley et. al "ROS: an open-source Robot Operating System", 2009, IC
[3] About ROS
https://www.ros.org/about-ros/
[4] Cartographer
https://google-cartographer.readthedocs.io/en/latest/
[5] Lidar
https://en.wikipedia.org/wiki/Lidar#Autonomous_vehicles
[6] Liang Wang, YihuanZhang and JunWang - Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR, 2017
[7]Javanmardi, Ehsan, et al. "Autonomous vehicle self-localization based on abstract map and multi-channel LiDAR in urban area." *IATSS research* 43.1 (2019): 1-13.
[8] John Leonard, David Barrett, Jonathan How, Seth Teller, Matt Antone, Stefan Campbell, Alex Epstein, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Troy Jones, Olivier Koch, Yoshiaki Kuwata, Keoni Mahelona, David Moore, Katy Moyer, Edwin Olson, Steven Peters, Chris Sanders, Justin Teo, and Matthew Walter - Team MIT Urban Challenge Technical Report, 2007
[9] Stanford Racing Team - Stanford Racing Team's Entry In The 2005 DARPA Grand Challenge, 2005
[10] Hiemstra, P. and Nederveen, A., 2007. Monte carlo localization
[11] Lima, O. and Ventura, R., 2017, April. A case study on automatic parameter optimization of a mobile robot localization algorithm. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)* (pp. 43-48). IEEE.
[12] Indians spend more time behind the wheel than Chinese, Aussies: Survey
https://timesofindia.indiatimes.com/india/Indians-spend-more-time-behind-the-wheel-than-Chinese-Aussies-Survey/articleshow/46337734.cms