TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

PROJECT REPORT

ON

SELF DRIVING CAR: MAPPING, PATH PLANNING AND OBSTACLE AVOIDANCE

SUBMITTED BY:

AARJAN BUDATHOKI(PUL077BEI004)

ABHIGYAN BHUSAL(PUL077BEI007)

MANISH GURUWACHARYA(PUL077BEI021)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

MARCH, 2024

Page of Approval

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **"Self Driving Car: Mapping, Path planning and Obstacle Avoidance"** submitted by **Aarjan Budathoki**, **Abhigyan Bhusal** and **Manish Guruwacharya** in partial fulfilment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

.....
Supervisor
Person A
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
Internal examiner
Person B
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

.....
External examiner
Person C
Assistant Professor
Department of Electronics and Computer Engineering,
Pulchowk Campus, IOE, TU.

Date of approval:

Copyright

Acknowledgments

We would like to express our heartfelt gratitude to all those who have encouraged and inspired us in the work on this project.

We would like to thank the Department of Electronics and Computer Engineering of IOE, Pulchowk Campus for providing us with an opportunity to work on this project. Similarly, we would like to express our sincere gratitude to the Robotics Club for their valuable support. We are especially indebted to our junior Devish Phuyal for his help with fixing the drive-train of our test vehicle and getting our vehicle up and running.

We are thankful to our seniors for their guidance and technical support. Their feed- back in the planning phases has helped us gain clarity in what we are trying to do. The guidance they gave us was instrumental in shaping the scope and design of our project.

Finally, we are thankful to our family and friends for their moral support and love.

Abstract

This project focuses on the development of a Self-Driving Car(SDC). It emphasises mapping, path planning, obstacle avoidance, and sensor fusion. The project forms an extensible and easy-to-understand SDC implementation. This project focuses on the development of a Self-Driving Car(SDC), emphasising Mapping, Path Planning, Obstacle Detection and Avoidance. The initial phase involves the precise tracking of the vehicle using sensor fusion algorithms and advanced Artificial Intelligence(AI) techniques. Additionally, a high-precision mapping system is created using physical systems such as Lidar and Camera, enabling accurate environmental perception.

For the optimal desired motion of the vehicle, the system is implemented with a Proportional-Integral-Derivative (PID) controller as well as Model Predictive Control (MPC). Ensuring the security and safety of the vehicle, the project utilises the Robot Operating System (ROS) framework, along with Real Time Operating System (RTOS) implementation for enhanced system speed.

Afterwards, advanced algorithms process real-time data for path planning, calculating optimal routes while considering pedestrians on the road. The system autonomously navigates, utilising an Obstacle Detection algorithm to avoid objects in its path. To further enhance safety, the vehicle incorporates an emergency override electronic system and other fail-safe devices in case of potential system malfunctions.

Keywords: *Self-Driving Car(SDC), Artificial Intelligence(AI), Proportional-Integral-Derivative (PID), Model Predictive Control (MPC), Robot Operating System(ROS)*

Contents

List of Figures

List of Abbreviations

SDC	Self Driving Car
IMU	Inertial Measurement Unit
PID	Proportional, Integral and Derivative
MPC	Model Predictive Control
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
LoRa	Long Range
RTOS	Realtime Operating System
SLAM	Simultaneous Localization and Mapping
ICP	Iterative Closest Points
ROS	Robot Operating System
STM	ST Microelectronics
DARPA	Defence Advanced Research Project Agency
CMU	Carnegie Mellon University
NNLS	Non-linear Least Squares
PnP	Perspective-n-Points
EDA	Electronics Design Automation
HLoc	Hierarchical localization
MPPIC	Model Predictive Path Integral Control
AMCL	Adaptive Monte-Carlo Localization

1. Introduction

1.1 Background

Over the years, the evolution of transportation has seen a significant increase in road traffic, leading to congestion, accidents, and environmental concerns. The need for a safer, more efficient, and sustainable mode of transportation became evident. The introduction of SDCs emerged as a compelling solution to these challenges, leveraging advancements in robotics and AI. SDCs, equipped with sensors and AI algorithms, have the potential to eliminate human errors, making roads safer. The introduction of self driving cars represents a groundbreaking convergence of robotics and artificial intelligence (AI) technologies, marking a transformative era in transportation.

In the context of Nepal, the vehicle industry hasn't taken off, there has not been any significant efforts in the industry to make it happen either. The implementation of self-driving cars (SDCs) faces challenges in Nepal due to its diverse and unpredictable environment. The country's rugged terrain, lack of standardized road infrastructure, and dynamic conditions make it difficult for SDCs, which rely on precise mapping and AI adaptability. Overcoming these hurdles requires a specialized approach to ensure the effectiveness and safety of autonomous vehicles in Nepal's unique driving landscape. The integration of robotics and AI in SDCs represents a strategic evolution aimed at creating a more efficient, safer, and sustainable future for transportation. The synergy with AI is indispensable, as it empowers these self-driving cars to perceive and interpret their surroundings, make real-time decisions, and adapt to dynamic driving conditions. Machine learning algorithms enable these vehicles to continuously improve their performance by learning from vast amounts of data generated during various driving scenario.

1.2 Problem Statements

Following are the question that drove our interest in the project. Our works will be based on the questions below:

- How does an actual vehicle system work?
- How can we control the basic locomotion of a car?
- How does the system perform using feedback mechanism?

- How can we make a system reactive to its environment?
- Can the system be reliable enough to handle its errors on it own?

1.3 Objectives

- To build an autonomous vehicle capable of moving from one location to another location, stay on route and make decisions about changing routes that cause least damage to itself.
- To build the autonomous vehicle capable of obstacle detection and obstacle tracking.
- To design a robust electronic system.
- To show an appropriate project management scheme covering all the important aspects of project management and work record keeping.

1.4 Scopes

For making the project well defined and achievable, we define the coverage and limit sets for our goals which are listed below:

- Working environment for the system will have well knowledge about track/path.
- Resetting the PID controller's integral term to zero in the presence of high output values to enhance stability.
- Include a dedicated emergency stop switch as a critical safety feature in the event of a vehicle malfunction.

2. Literature Review

2.1 A Brief History of Autonomous Vehicles

The first traces of modern autonomous navigation technology can be found in the rovers, space probes and missile systems starting from the late 1960s. The technologies developed in this era, no doubt accelerated by the Cold War and space race, permeated slowly to cars as well. In the 1980s, Ernst Dickmanns, a German pioneer, and his team from the University of Bundeswehr retro-fitted a Mercedes Benz van into VaMoRs, an autonomous vehicle that used computer vision to analyze road-facing camera images [4]. Dickmanns' success led to the large PROMETHEUS project which led to a decade of steady improvements, culminating in the VITA vehicle, which set the then record for autonomy by completing an almost 1000-mile trip at 95% autonomy. Across the pond, the NavLab 5 vehicle from Carnegie Mellon University also demonstrated autonomous driving, setting a 98.2% autonomy record [5].

A distinctive inflection point in the history of self-driving cars is the DARPA challenge [6] held in 2003, 2004, and 2007. The 2003 edition concluded with none of the participants able to complete the challenge. 2004 saw Stanford's Stanley [1] win, led by Sebastian Thrun. CMU won the 2007 challenge.

Figure 2.1: Stanford's Stanley (Image from [1])

A more complete survey of recent methods can be found in [7].

SAE International, formerly the Society of Automotive Engineers, defines six levels of driving automation, ranging from no driving automation (level 0) to full driving automation (level 5) [8]. This taxonomy is widely used throughout the industry to classify current methods and compare them against one another.

2.2 State Estimation

An important part of every self-driving system is the fusion of data from multiple sources to estimate the state of the vehicle. Sensor fusion is usually done using filters.

2.2.1 Kalman Filter

The Kalman Filter [9] is a Bayes optimal filter for the estimation of random variates that follow the normal distribution, and whose process, control, and sensor models are linear.

The Kalman Filter assumes a Markov model for the evolution of the state, such that the state of the robot, x_k is dependent on the last state $\hat{x}_{k-1}$ and the control inputs u_k. The process model relates $\hat{x}_{k-1}$ with x_k, and the control-input model B_k relates u_k to a corresponding change in the state brought about by the input. Thus, the state evolution model is as follows, including a process noise w_k to model process uncertainty.

$$x_k = F_k \hat{x}_{k-1} + B_k u_k + w_k$$

Readings from the sensors are integrated into the estimate using the observation model corresponding to the sensor. An important detail to note is that this integration is done not in the state space of the system we are tracking, but in the space of sensor readings. A Kalman Gain is computed by comparing the sensor reading to a predicted sensor reading, which is then used to produce a new estimate.

$$\hat{x}_k = (I - K_k H_k) x_k + K_k z_k$$

Two assumptions lie at the heart of the Kalman Filter:

- The state variables and sensor readings are normally distributed, with some known mean and covariance.
- The process model, sensor-input model, and the control-input model are all linear transformations.

These assumptions make the update and estimation steps tractable since the normal distribution has several properties that favor its use. Chiefly, normal variates are closed

under addition, multiplication, and affine transforms, and the conjugate prior of a normal distribution's mean is another normal distribution itself. Hence, all models used in Kalman Filters are linear.

2.2.2 Non-Linear State Estimation

Nonlinear systems pose a problem because a normal variate transformed through a nonlinear function does not result in a normal variate. For nonlinear systems, two main extensions to KF are popular: the Extended KF [10] which uses locally linearized versions of all non-linear models using Taylor Series extension upto the first-order terms, and the Unscented KF (UKF) [11] which transforms a small set of so-called sigma points through the non-linear models, and then fits a normal distribution on the resulting transformed points.

2.3 Robot Operating System(ROS)

2.3.1 Nodes

Nodes are computational units in ROS Graph. In a working robotic system multiple nodes work together.Each node can send and receive data from other nodes via topics, services, actions, or parameters.

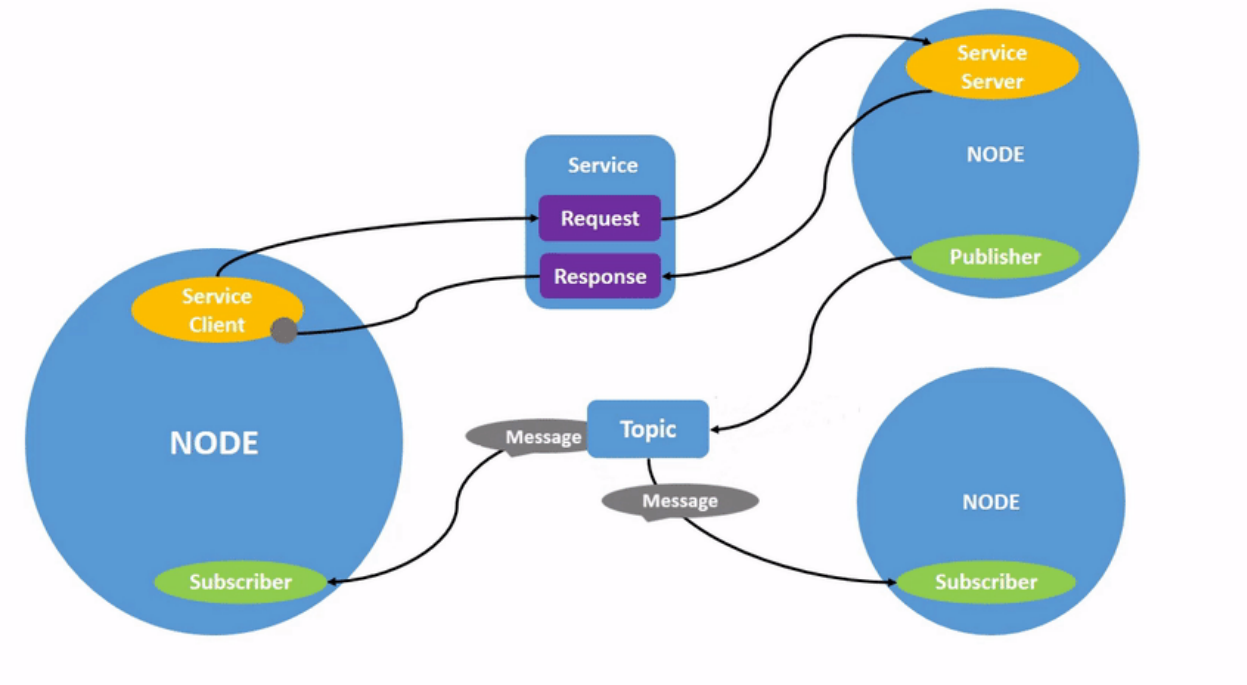


Figure 2.2: ROS Nodes

2.3.2 Topics

Topics are intermediate for connection between multiple nodes. Nodes associating with a topic can either publish or subscribe it. A node may publish data to any number of topics and simultaneously have subscriptions to any number of topics. Topics can establish one-to-many, many-to-one or many-to-many relationships.

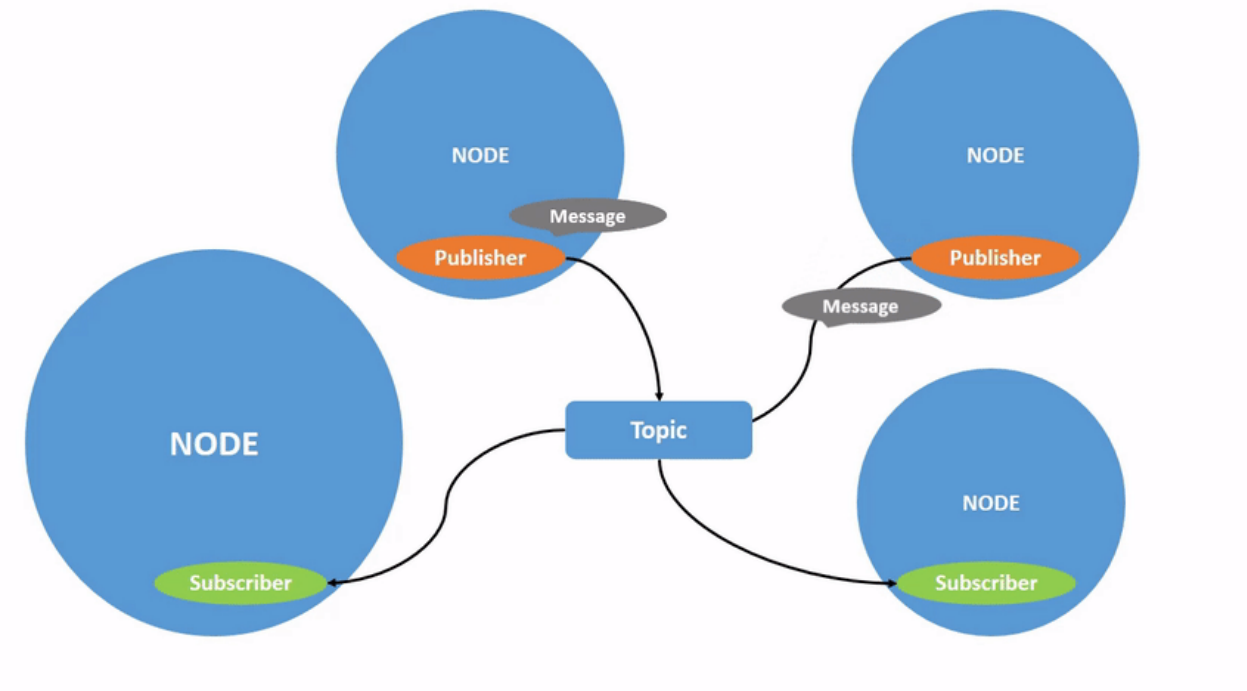


Figure 2.3: ROS Topics

2.3.3 Services

Service is another method of communication between nodes. Its works on Request-Response model. By the use of service the nodes provide data only on request from other nodes.

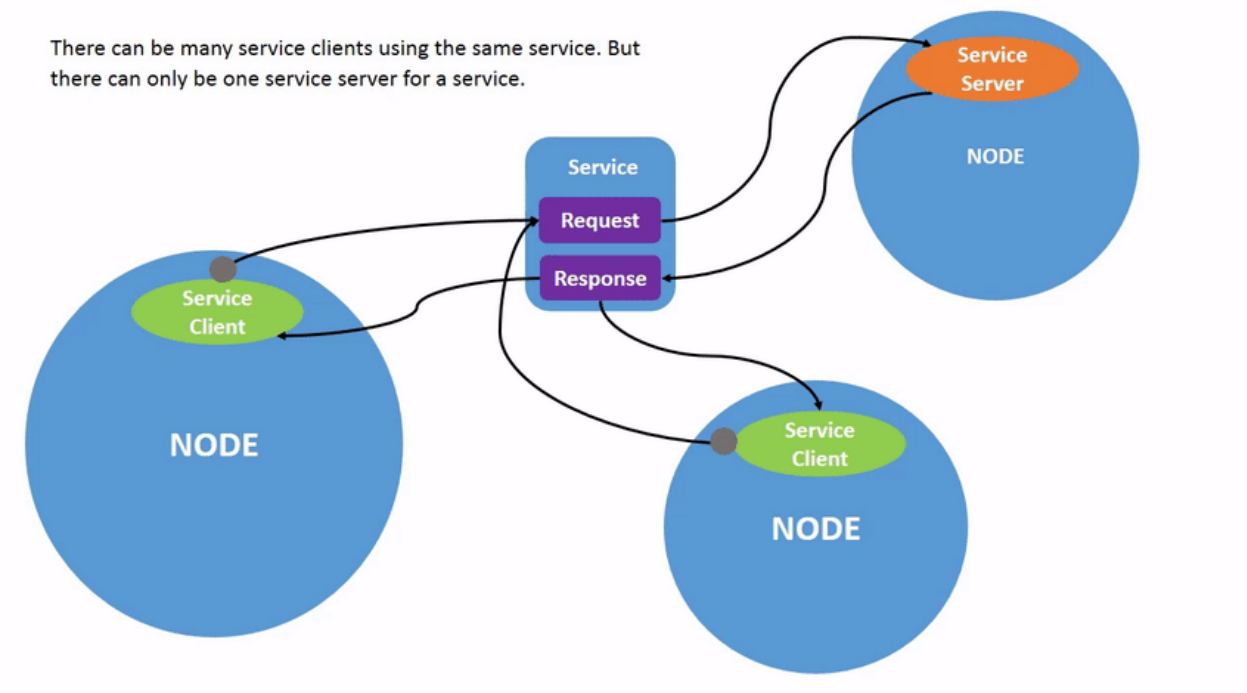


Figure 2.4: ROS Service

2.3.4 Parameters

A node is associated with its individual parameters. Parameter is used to configure nodes at startup without changing the code of the Nodes. The lifetime of a parameter is tied to the lifetime of the node.

2.3.5 Actions

Actions are another types of communication which are used for long running tasks. It consists of 3 parts: a goal, feedback, and a result. Actions are based on topics and services. Actions use a client-server model, similar to the publisher-subscriber model. An "action client" node sends a goal to an "action server" node that acknowledges the goal and returns a stream of feedback and a result.

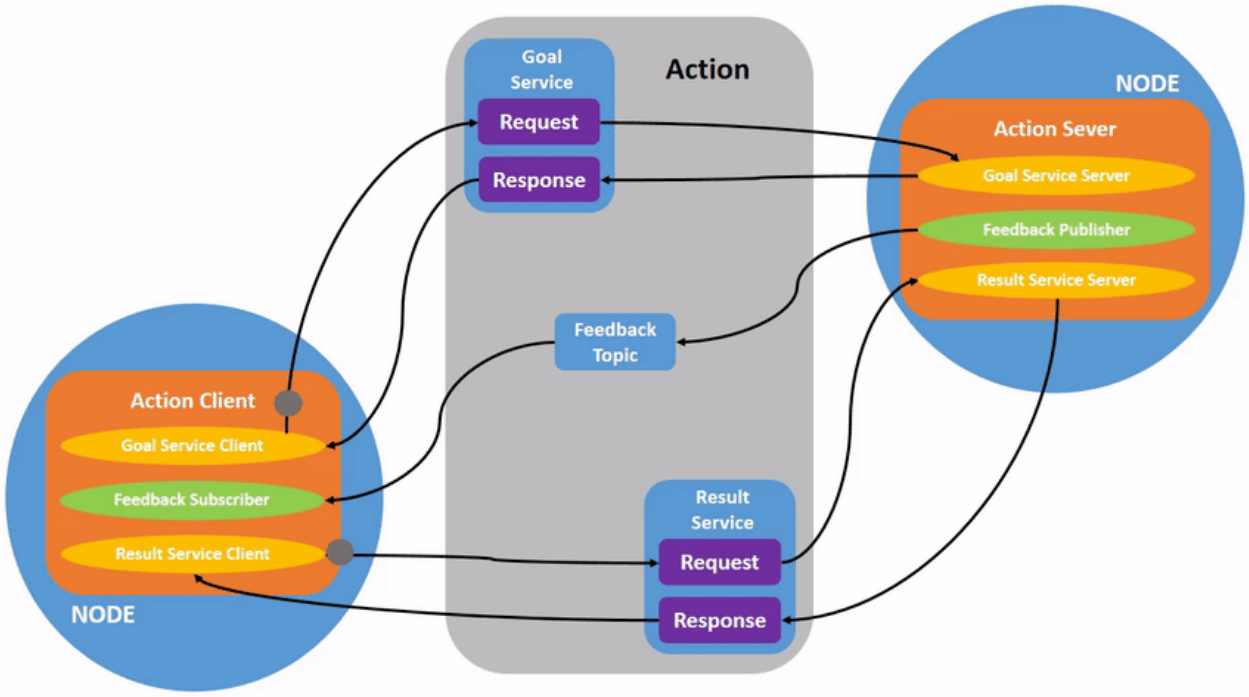


Figure 2.5: ROS Actions

2.3.6 Packages

A package is a bundle of ROS codes. It consists of multiple node and the nodes outside the package are connected through various means describes above with a specific message format enabling the package to operate for specific task in the system. The packages can also be released to allow others to build and use it easily.

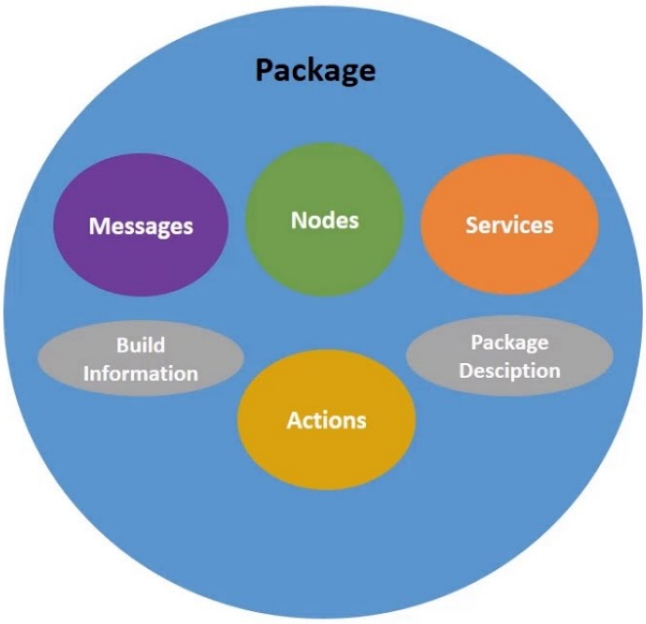


Figure 2.6: ROS package

2.3.7 Perception

Localization, mapping, and perception have been very active areas of research in the last few decades. For localization using a 2D lidar, representative methods are outlined in [12] [13]. Visual localization, and the more general problem of Simultaneous Localization and Mapping, have been approached through multiple ways, but a popular and robust method based on matching features in images that have some desirable invariant properties, is ORB-SLAM [2].

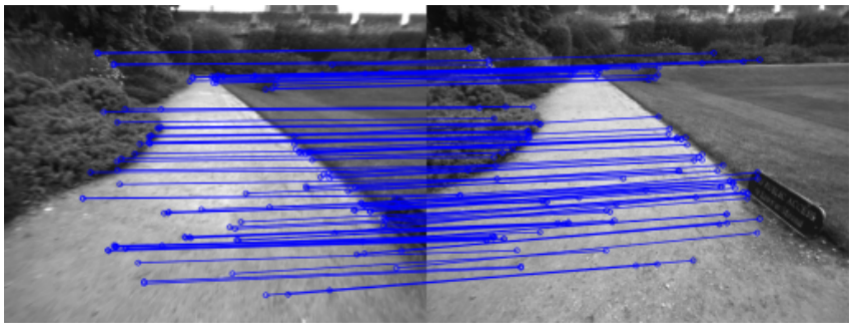


Figure 2.7: Matching ORB keypoints between images (Image from [2])

For applications where an *offline* map of the world, built and processed in a distinct phase before actual autonomous driving, is enough, a more robust and large scale mapping

algorithm may be used, which typically uses a Structure from Motion (SfM) pipeline [14]. SfM typically involves joint optimization of a large scale non-linear least squares (NNLS) problem defined on camera poses (rotation and translation) and a sparse set of 3D points in the world [15]. A common approach to define this problem is to match a sparse set of so-called keypoints between images using a keypoint description and matching algorithm, and then initializing the 3D location of these points using simple two-view triangulation [16] [17] [18] [19]. Localizing on an offline SfM map then boils down to simple robust search, possibly augmented with a multi-step hierarchical localization approach [3].

In particular, [3] outlines a state-of-the-art hierarchical localization method. The method has two steps. The first step involves the use of an image-level descriptor, for example [20], to localize the camera at a coarse level. The second step then searches for keypoints between the image and the map, and optimizes the pose of the image.

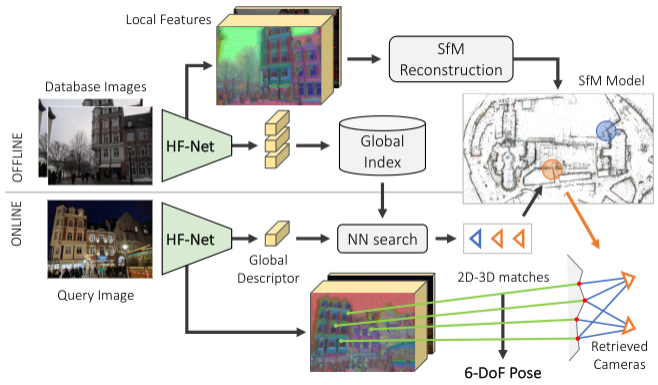


Figure 2.8: Hierarchical Localization with HF-Net (Image from [3])

Perception involves the detection of obstacles, classification of drivable areas in images, and scene understanding in general. We refer the reader to [21] [22] [23] for a few representative methods that are in use in the Autonomous Vehicle industry.

2.4 Control

Good control of actuators like steering and torque on the wheels is crucial to the technology. Two popular approaches are the Proportional-Integral-Derivative (PID) controller [24] and Model Predictive Control [25].

2.4.1 PID Control

PID is a simple and perhaps the most widely used controller for linear systems. It has three terms, corresponding to the error, its integration over time, and its time derivative. The weights for each of these terms are used to control their impacts on the overall control

output. The overall control output function of the PID controller in continuous form is

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau \mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

In practice, a discrete formulation is used more often, which approximates the integral using quadrature and the derivative using finite difference. The selection of the three coefficients is done through a process called PID tuning. There is a range of techniques for tuning PID controllers, from manual tuning to simple rules on observed system response, to automatic in-loop algorithms that select the best parameters using genetic optimization or fuzzy search [26].

2.4.2 MPC

Model Predictive Control is a controller that uses a model of the system for predicting the system states up to a finite time horizon. It then computes an optimum control output for achieving the defined set point, subject to constraints.

The general MPC framework involves minimization of the following cost function

$$J = \sum_{t=k}^{k+p} W_s(x_t - r_t) + W_c \Delta u_t^2$$

W_s weights the tracking error between reference state r_t and the predicted state x_t, while W_c controls the strength of regularization on the control inputs $u(t)$ by penalizing the derivative. Other terms may be added to this formulation to control required variables, such as jerk and path curvature. p is the time horizon. The states x_t are predicted using the model, which may be linear or non-linear. Depending on the convexity and linearity of the terms, various optimization methods may be used.

2.5 Social Impacts and Ethics

As with any new technology, it is important to consider the social impacts and ethics of SDC technology. Keeping in mind the small scope of the project, we defer the discussion of a more general social and regulatory view on SD technology to [27].

2.6 Methodology

2.7 Scale of the Project

We have developed a small electric toy car of dimension 100cm, 55cm, and 45cm. The vehicle is equipped with DC motors to drive the rear wheels, while the front wheels is connected through a bar linkage system controlled by a servo for steering. To power the motors, we had used the Lippo batteries, securely positioned in the central area of the chassis, specifically in the back seat of the car.

The vehicle design takes a lot of research and work. The difficult aspect before any mass production is creating the vehicle itself. We are not looking to mass-produce this vehicle and neither we want to spend time on any aesthetics of the system. So taking that out of the equation, still the main aspect of any vehicle is the chassis. Initially, the chassis contained a steering mechanism issue, and the existing chassis was equipped with a small low speed DC motor with no feedback. To address this, we replaced the current motor with a 775 motor with seperated 3D printed gear attached to the optical rotary encoder for the feedback. Additionally, we designed a steering mechanism to be controlled by servo motor.

As listing our tasks, we have completed the tasks from Printed Circuit Boards(PCB) designing and circuit layout to the intricate aspects of sensor placement, communication design, and embedded firmware development. The project further entailed expertise in real-time operating systems, proficiency in CAN protocols, and the intricate process of system integration. Moreover, our scope extended into the realms of Visual SLAM (Simultaneous Localization and Mapping), leveraging technologies such as Hector SLAM. The integration of mapping techniques and the utilization of ROS(Robot Operating System) frameworks was pivotal in orchestrating a cohesive and efficient self-driving car system. It is sure that this is a very engaging project and we populated most of our effort in it.

2.8 Autonomous Region and Environment

For the vehicle to navigate within the confines of our campus specifically along the road encircling the Robotics Club block, we accounted for various environmental factors. This includes negotiating shaded areas created by trees, sunlight stretches, and the presence of students strolling along the road, with bicycles parked on one side. Operating autonomously during the daytime, our SDC heavily relied on Camera data, incorporating visual slam technology to ensure precise navigation. We addressed crowded scenarios, particularly when roads are densely populated with pedestrians, our SDC will implement strategic U-turn maneuvers. Our approach encompasses robust solutions to effectively navigate these diverse environmental conditions, ensuring the safe and reliable operation of the autonomous vehicle.

2.9 Absolute and Relative Positioning of the System

We need to keep track of our vehicle in order to command it to where to go in real-time. For this, we need to know the position of the vehicle. By fusing information from the IMU and rear wheel encoders and applying the Unscented Kalman Filter [28], we achieve precise and instantaneous positioning feedback. In the long run, utilizing Lidar and Camera data, we will implement Visual SLAM technology to establish and maintain the vehicle's position accurately over time. But instead of doing this, we integrate local planning techniques like road segmentation, and obstacle detection using YOLOv8 and Lane-Keeping to generate a map which will be used for the navigation of the robot.

2.10 Team and Work Division

- Mechanical Work

  Each and every modification of hardware chassis of the car was accomplished with the tools and resourcees available in the Robotics Club. Junior members from the Robotics Club offered assistance in the system modification process.

- Electronics and Embedded Systems

  This part mostly consisted of electronics hardware design like PCB designing and fabrication. Our team used PCB Design software: Altium for PCB Designing. Fabrication of PCB will be done by outsourcing it to PCBWay. Component placement on the printed board will be done with the tools available in the Robotics club.

- Control System Design

  The Control System Design was used in the velocity tuning of the motor used in the wheels of the car for vehicle control and stability.

- Perception Implementation

  Perception problem requires training of the model and data selection to suit our environment and system. Optimization of the model is crucial for fast inference. We collected different datasets of the track around the Robotics Club and annotated the datas and trained to the yolov8 model and this model will be used for detecting the obstacle and segmentation of the road for finding the drivable region.

2.11 Equipment, Tool and Devices

- **Microcontroller** We used STM32F407 microcontroller based on ARM Cortex-M processors having best performance levels and peripherals for our needs of different appli-

cations.

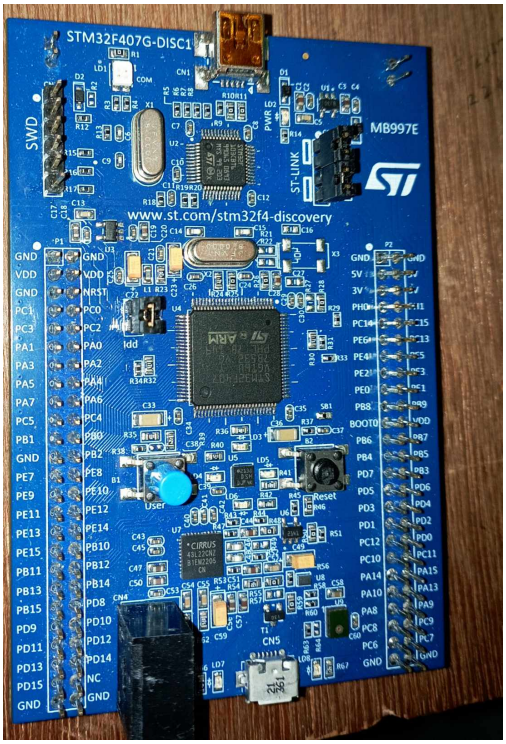


Figure 2.9: STM32F407 Discovery

- **Raspberry Pi**

  We used Raspberry Pi 4 model B, a single-board computer used for most of the major operations of the system like perception of the car and supporting role for estimating the pose of the vehicle.

Figure 2.10: Raspberry PI

- **Inertial Measurement Unit**

We used Inertial Measurement Unit (MPU6050) for the estimation of the vehicle. In our setup, we have IMU sensors such as MPU6050.

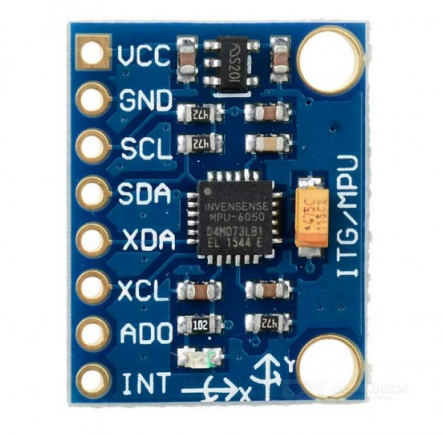


Figure 2.11: MPU6050

- **Camera**

  For perception in a self-driving car (SDC), we choosed the Logitech C270 HD Webcam. This webcam is capable of recording videos at 720p resolution with a frame rate of 30 frames per second (fps).

Figure 2.12: Logitech C270 Camera

- **Lidar**

  For perception in an SDC, we have chosen the A1 RPLidar. The A1 RPLidar is a lidar sensor known for its reliability and cost-effectiveness in providing essential environmental data for autonomous vehicle systems.

Figure 2.13: A1 RP Lidar

3. Experimental Setup

3.1 Mechanical Hardware Design

The toy car available in the Robotics Club is used as the frame of our system. The frame was modified with the addition of motors, a steering mechanism, encoders, a camera, and a lidar. The steering mechanism was designed according to the Ackermann steering geometry [29].

Figure 3.1: Mechanical Hardware

3.2 Electronics System Design

The bulk of the electronics system runs on one of two processors: a Raspberry Pi for high-level tasks, and an STM32 microcontroller for low-level, time-critical tasks. Custom printed circuit boards were made to help the two processors interface with sensors and power management circuits.

3.3 PID Tuning and System Calibration

The controllers were tuned very precisely. Also, any model that we had assumed needed to be calibrated so that it was in accordance with the actual physical system. This was an ongoing process since the physical system was not stationary, and any changes made to the system warranted re-tuning and calibration.

3.4 ROS Implementation

For high-level tasks, a fast, standard message bus was needed to facilitate easy and natural division into independent processes. For this, we used the Robot Operating System [12]. ROS is a mature and popular set of libraries, packages, and middleware that includes drivers, implementations of algorithms, and message-passing infrastructure. Using ROS allowed us to standardize on a fixed set of types for communication between the high-level nodes of the system.

3.5 Mapping, Localization, and Perception

Perception, mapping, and localization together formed a system for understanding the scene that the vehicle was in. Mapping refers to the creation and maintenance of a map of the environment. The map was created using lidar using an iterative closest points (ICP) based algorithm. Mapping for the camera was done online, using a visual Simultaneous Localization and Mapping (SLAM) algorithm. Localization was done using a particle filter-based approach using the map and readings from the sensors. Finally, perception was done using an object detection and semantic segmentation model, based on the YOLOv8 package [21] [30]. Distinct obstacles were tracked using a Kalman filter implementation. The node's perception was run on laptop computers in order to compensate for the low computing power of the Raspberry Pi.

3.5.1 Mapping

For mapping, a two-tiered mapping strategy was followed. A 2D lidar was used with the SLAM Toolbox implementation in ROS2. The lidar gave a laser scan, which gave range readings in all directions around the robot. A map of the target area was constructed using Iterative Closest Points and smoothing. The map obtained from 2D lidar was an occupancy grid, showing solid obstacles on both sides of the road.

While it is possible to localize using the 2D lidar map only, in practice, range readings in the road when actual driving are very different from the range readings in the mapping phase. So, a visual map was also built. HLoc [3] was used to build a hierarchical map of the target environment. NetVLAD [20] was used to produce image descriptors for coarse localization, and DISK features [31] were used for fine-tuning poses. Matching was done using LightGlue [19]. Producing a map involves matching descriptors between images, and the map was then optimized using COLMAP [32] [33].

3.5.2 Localization

Real-time localization was achieved by fusing the position estimates from the visual positioning system into the EKF node. This data was fused with higher-frequency estimates from the wheel odometry, estimates from the IMU, and the lidar map.

3.5.3 Perception

Perception used a webcam for Object detection, Image segmentation, and Visual SLAM. The system required a pretrained model to run, for which a dataset was curated for a local environment. The dataset was composed of pictures with the obstacles in the real environment. The same dataset was also used for image segmentation.

Figure 3.2: Dataset Images

4. System design

4.1 System Overview

The system is designed around two controllers: a main controller based on a Raspberry Pi, and a sub-controller based on an STM32 Microcontroller. The controllers are connected to each other via a UART bus. The main controller hosts the processes for perception and localization.

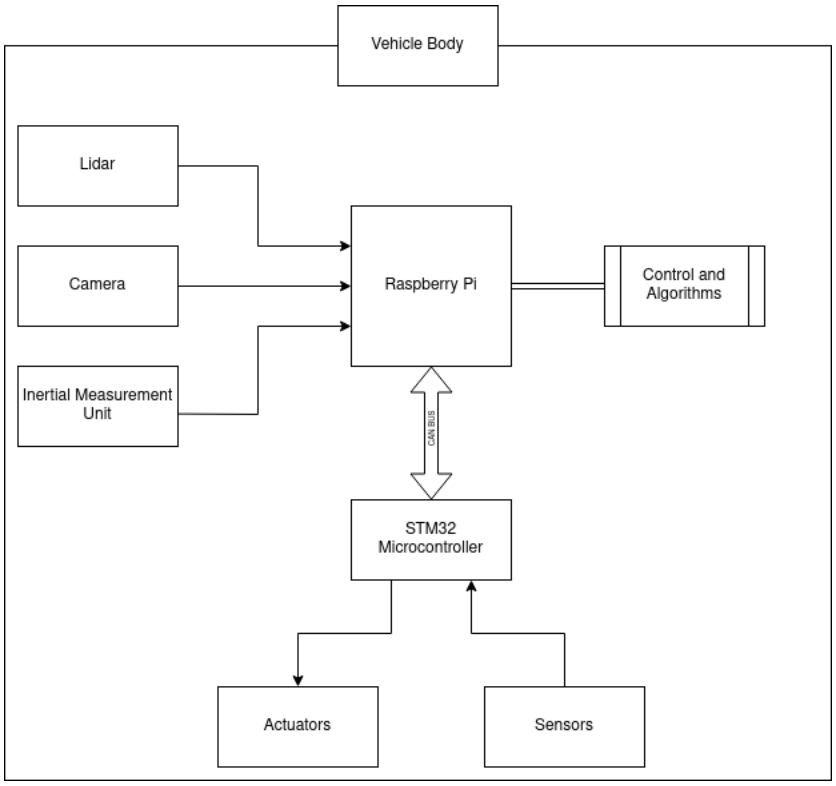


Figure 4.1: System Overview

The high-level design of the system was inspired by Stanley [1]. The system was composed of 4 layers: Sensor layer, Perception Layer, Control layer, and Monitor layer. These layers were logical, and each was made of several independent, concurrently running nodes. The life-cycle of these nodes, and communication between them, was facilitated by ROS [12]. ROS acted as a middleware for these nodes to communicate, while also providing a standard programming model for creating and running nodes. Nodes could be written in a variety of languages, but Python and C++ were popular.

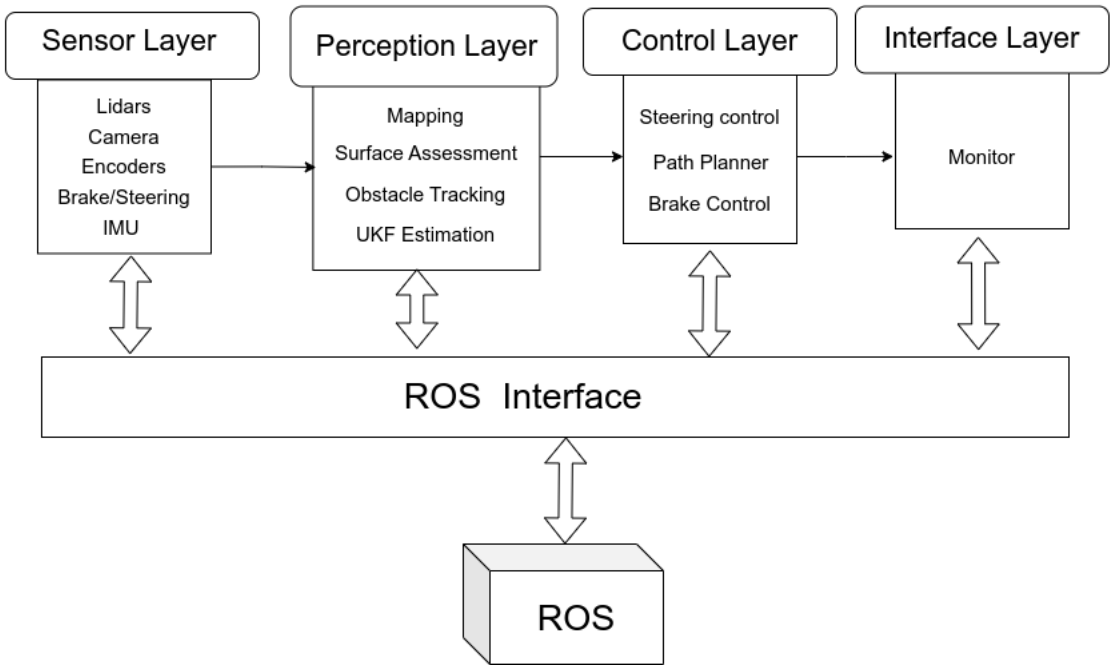


Figure 4.2: System Layers

4.1.1 Main Controller

The system incorporated Raspberry Pi as the main controller. It ran Robot Operating System (ROS) for managing and monitoring the overall operation of the system. Devices like Lidar, Camera, and Inertial Measurement Unit (IMU) were connected to the Raspberry Pi. Their outputs were consumed via ROS drivers, which published the received data to topics using standard ROS data types. For perception tasks that required more computation, we supplemented the Pi with a laptop computer with a GPU.

The ROS nodes could be as granular as required. Some of the parts of the system that were implemented as nodes included:

- Localization
- Mapping
- Perception and Scene Understanding
- Obstacle Detection and Tracking
- Path Planning
- Control Algorithms
- Sensor fusion Algorithm

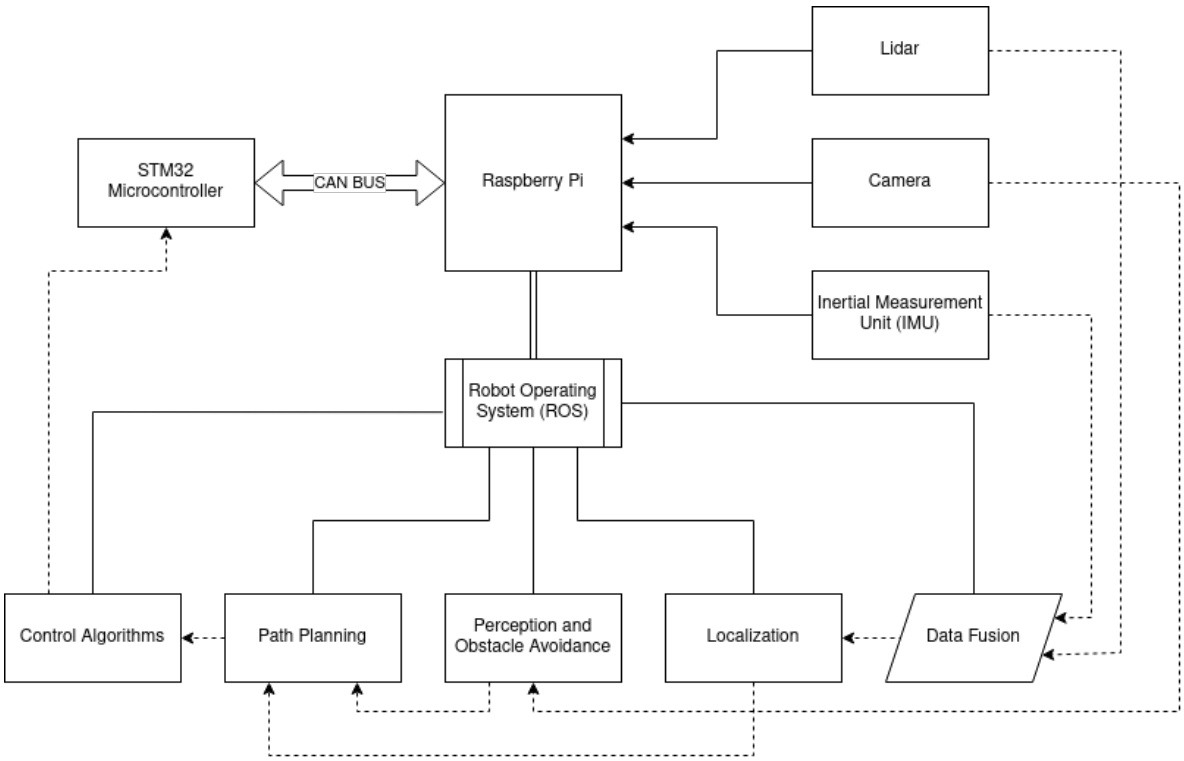


Figure 4.3: Raspberry Pi as Main Controller

4.1.2 Sub-Controller

The STM32 Microcontroller was used to control drive-train actuators for the motion of the system. The motor was connected to the motor driver controlled by STM32 using a PID controller, which took feedback from the encoder and additional inputs from the UART bus. The FreeRTOS system was implemented in STM32 for task management. Individual algorithms were threaded in order to create a stable and responsive real-time system.

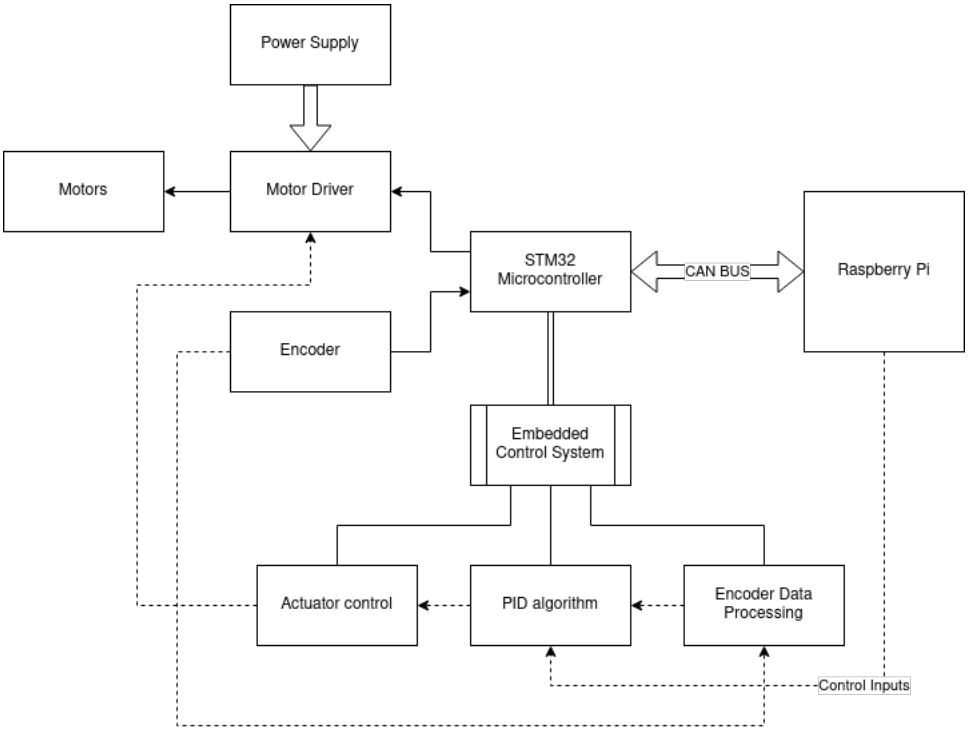
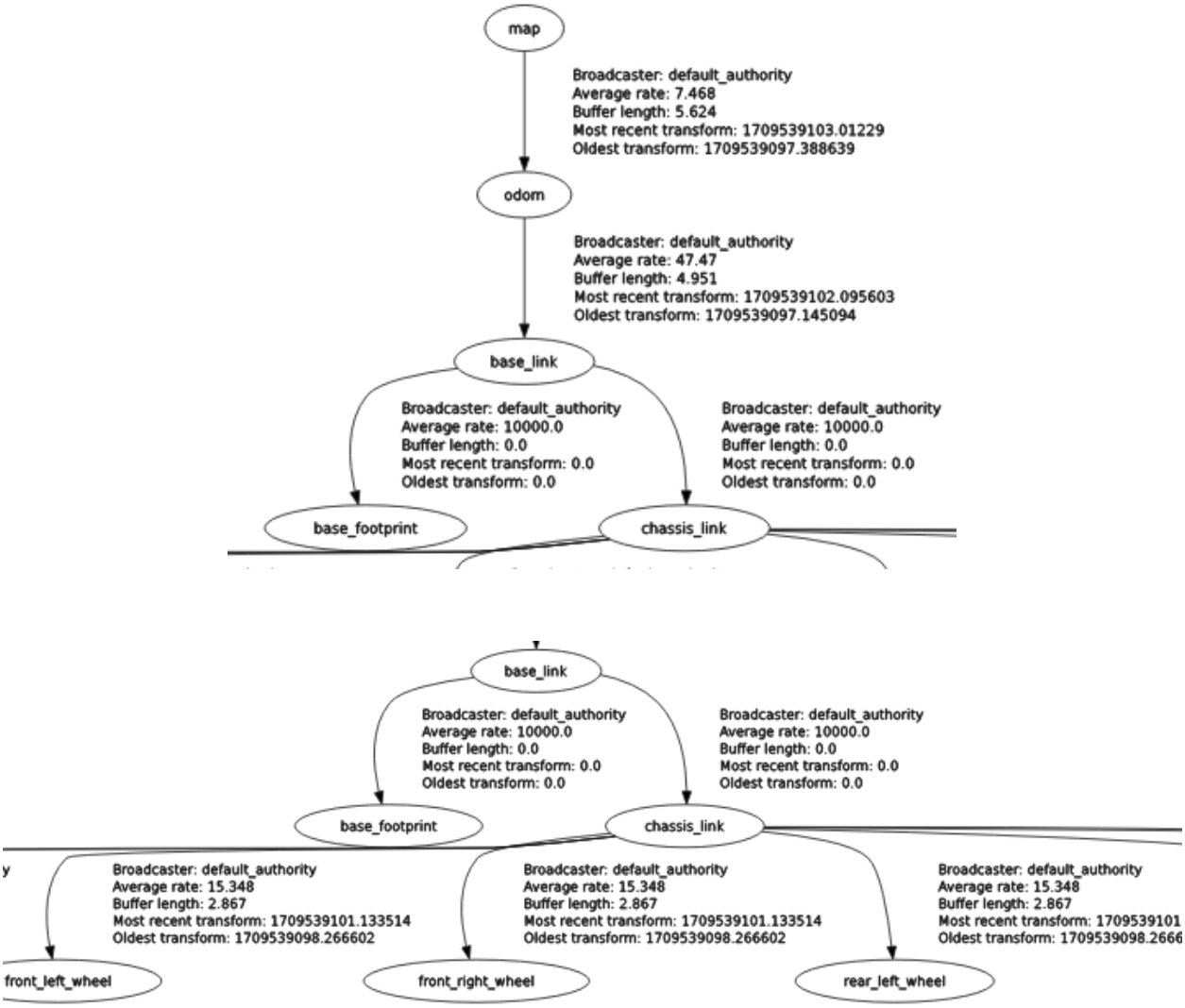


Figure 4.4: STM32 as Sub Controller

4.1.3 Carlikerobot Description



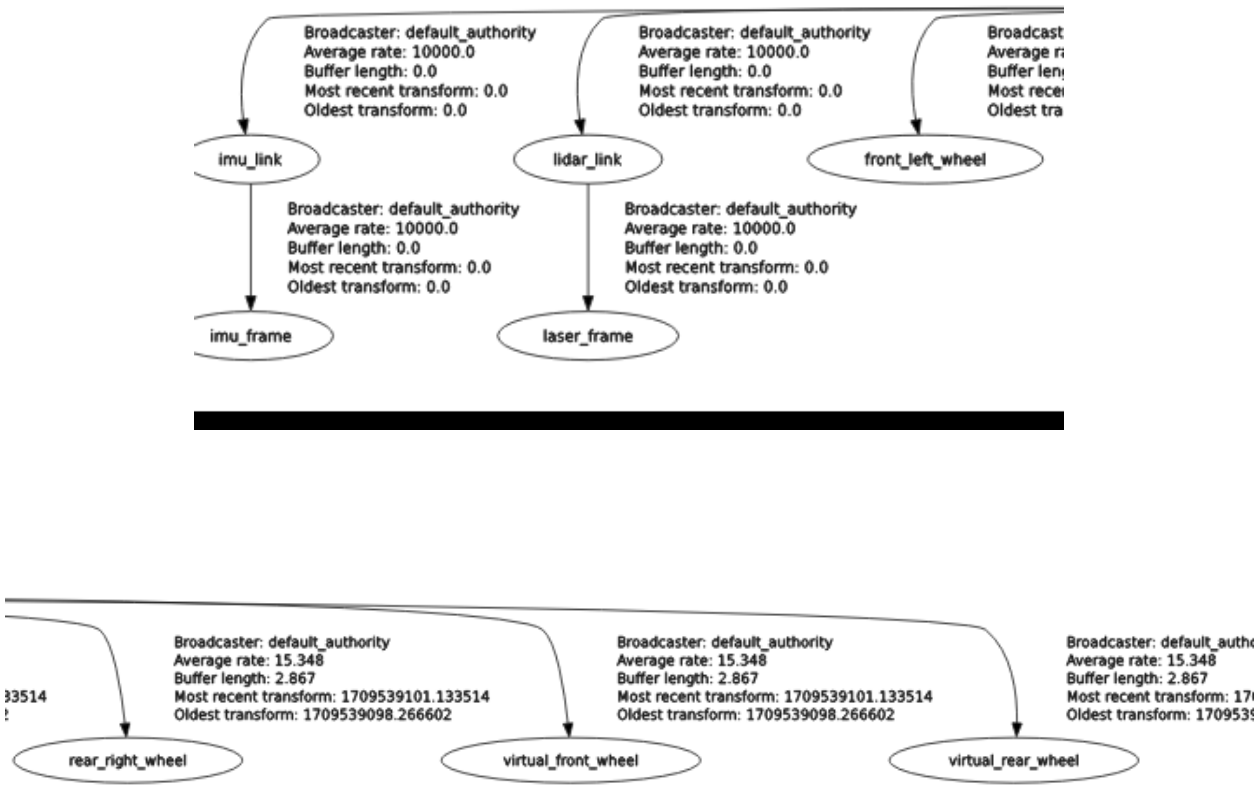


Figure 4.5: TF Tree of hardware car

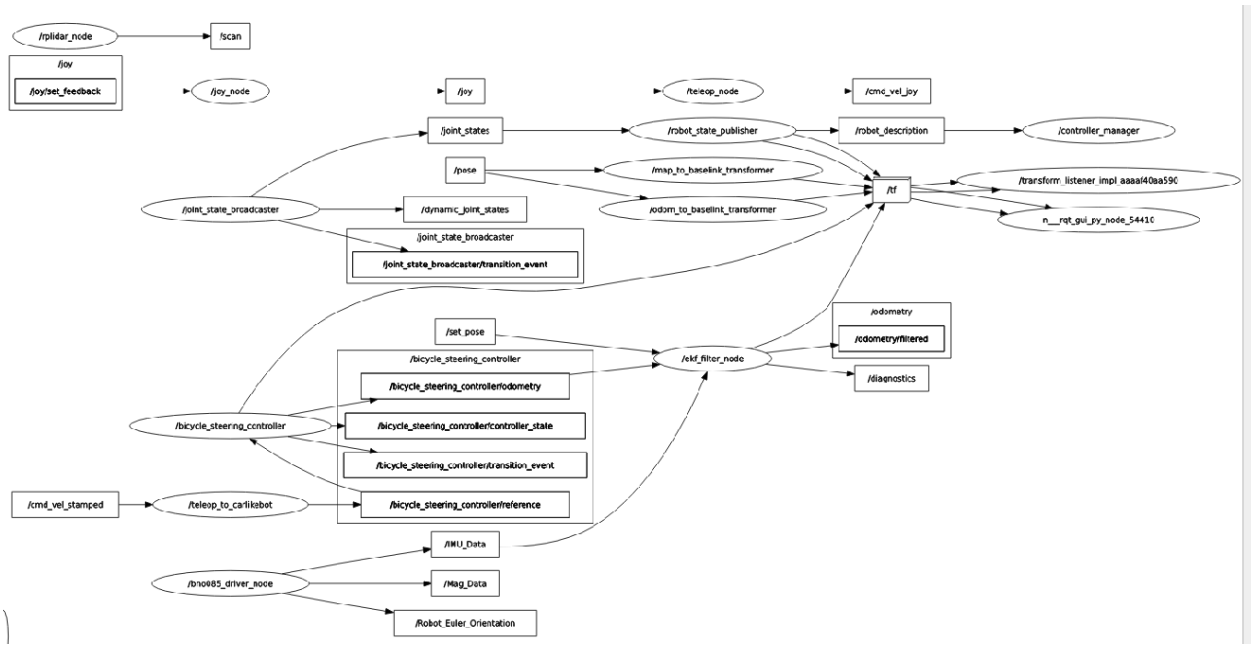


Figure 4.6: Node and Topics of the Car

5. Results & Discussion

5.1 Mapping

We mapped a target environment using SLAM Toolbox [34]. The approach uses Iterated Closest Points in order to match lidar scans across time, which makes it prone to failure in areas with very little geometry to match. The overall mapping of the environment was reasonably good, and the world geometry was very well reflected in the maps.

The mapping and navigation pipeline was tested in an indoor test environment, and the figure 5.1 shows the map produced by the pipeline.

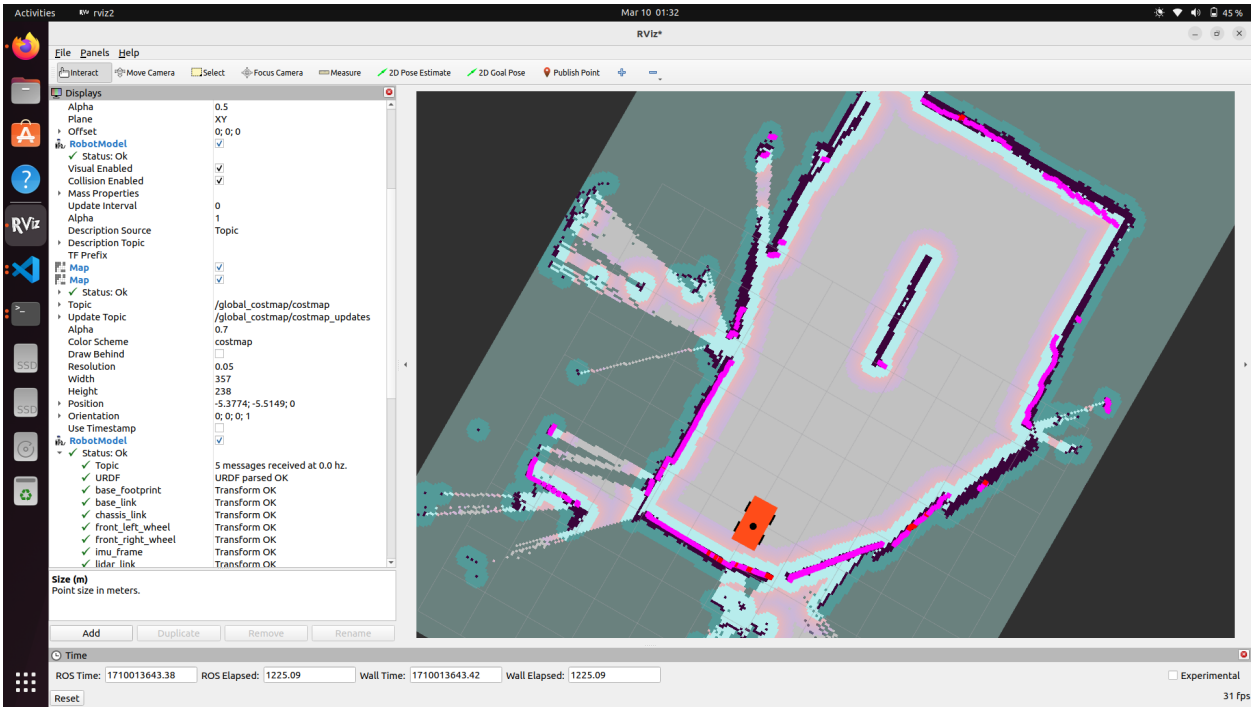


Figure 5.1: Indoor Mapping

5.2 shows a map of the road around the Robotics Club and Machine Workshop buildings.

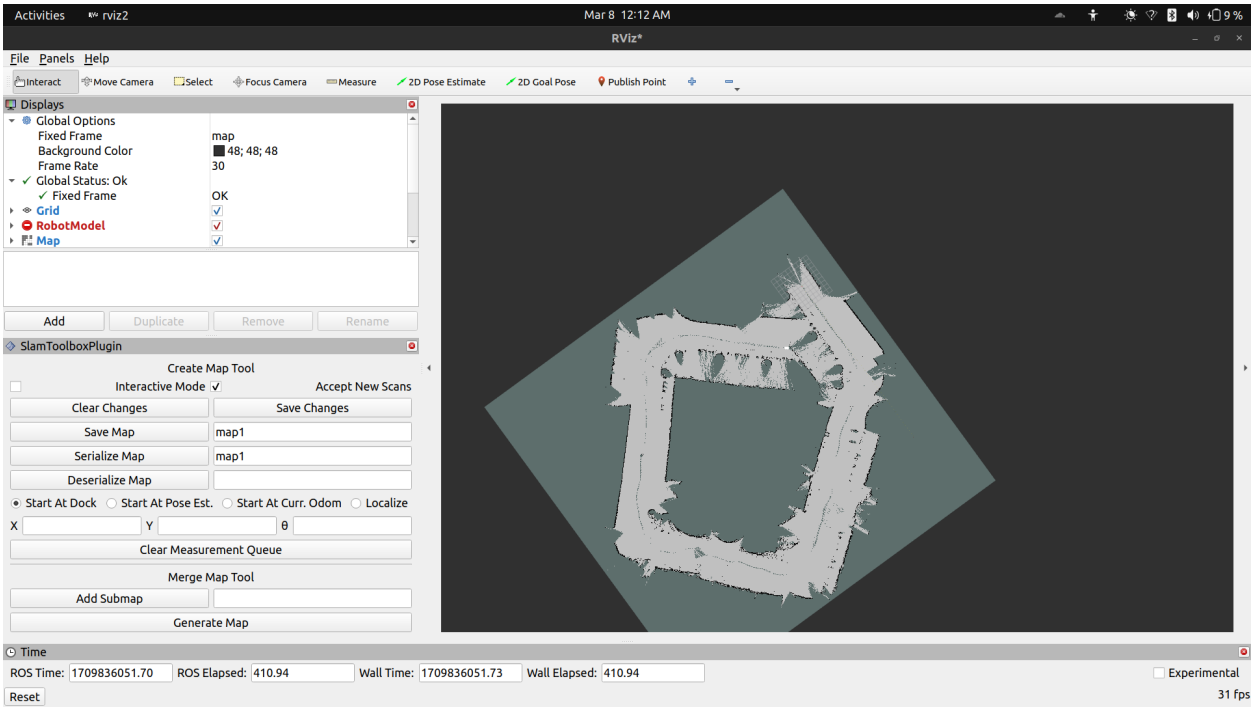


Figure 5.2: Outdoor mapping using Lidar

The lidar required an environment with sufficient features and geometry in the surroundings. The location with minimal feature geometry was very prone to error and drift, as shown in 5.3. This issue was fixed using manual insertion of geometry into the environment using plywood boards.

Figure 5.3: Map with Sparse Features

5.2 Perception

The results from segmentation and object detection on our test dataset are shown in 5.4 and 5.5 respectively. The object detection model produced bounding boxes for each car, bike, road, tree, and person, and the results were verified on our test dataset.

Figure 5.4: Object Detection

The segmentation model produced bounding boxes for each distinct road region, along with a binary mask for each road instance.

Figure 5.5: Object Segmentation

5.3 Visual Mapping

The map produced by HLoc [3] is shown in 5.6. The construction of the map was largely automatic, and required a single directory of images of the target environment. We used 400 images of the target environment. The mapping process was found to be computationally intenstive but parallelizable through CUDA. The map construction was done on a server with AMD Ryzen 7 5700X 8-Core processor, 32GB of RAM, and an NVIDIA GeForce RTX 2080 Super GPU with 8GB of VRAM.

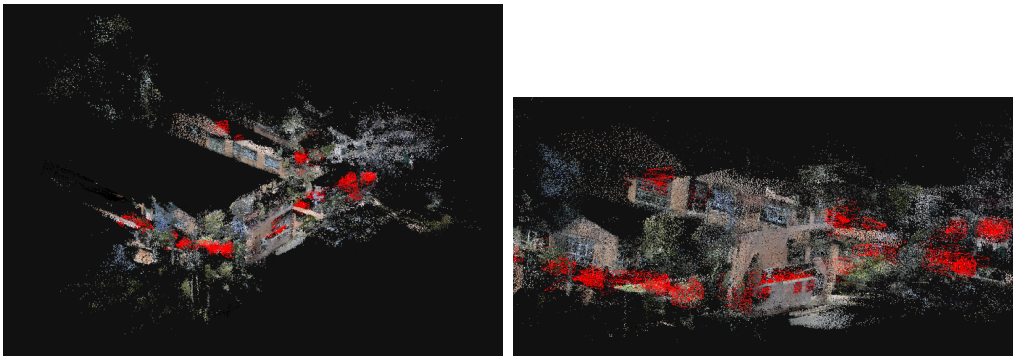


Figure 5.6: Visual Map Generation using HLoc

5.4 Navigation

The navigation stack uses a global occupancy grid obtained from lidar map, a local costmap from instantaneous lidar readings for obstacle avoidance, and an inflation layer to account for the large geometry of the car in order to produce a path to the goal point. The path is obtained by keeping in mind the geometry of the vehicle so that paths that cannot be executed by the car are not produced. Figure 5.7 shows the planned path to a goal pose, and the rotated square around the car is the local costmap built by the inflation layer.

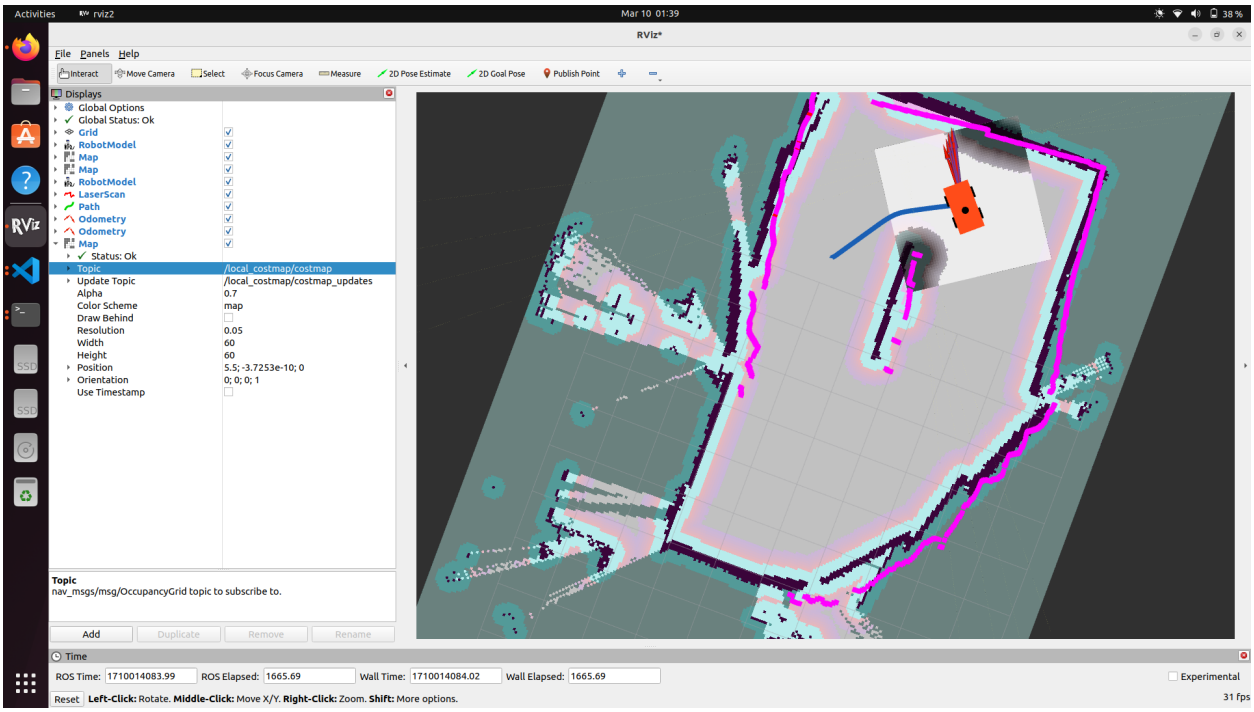


Figure 5.7: Navigation, Localization and Path Planning

6. Conclusion

We created a self-driving car system that can navigate from point A to point B in an environment. The system was built using commodity components that we obtained for free or were cheaply available. The system was reasonably robust against different lighting conditions and low-speed obstacles through the use of different advances in sensor fusion and perception. While we purposefully limited the scope of the project to adhere to budget and time limitations, the system we built is easy to extend, for example to multiple cars in a collaborative environment, or via the use of better sensors. The system has a distributed architecture, and the processing loads are divided across multiple nodes in different computers. As such, scaling the system in the future could be quite natural and straightforward.

The main goal of the project was for us to learn various concepts, some of which are control, navigation, path planning, visual mapping and localization, lidar mapping and localization, obstacle avoidance, among others. This goal was achieved to our satisfaction, and hence the project was successful in that regard.

7. Limitations and Future enhancement

7.0.1 Limitations

- **ROS2_control for Ackermann steering controller**

  We did not found any ROS2 control demos on Ackermann steering controller Despite weeks of searching for alternatives, at last we had to implement available bicycle steering controller inherited from steering controllers library.

- **Hardware Resources & Budget**

  The absence of essential hardware components such as **GPS, 3D LIDAR and depth cameras** can severely hinder the data acquisition process. Without these sensors, the SDC lacks crucial sensory inputs necessary for accurate mapping, path planning and navigation.

- **Remote Communication**

  Unavailability of a reliable wireless communication medium throughout the traversal path resulted in frequent disconnection with local machine which highly affected the system performance.

7.0.2 Future Enhancements

- Navigating using GPS Localization

  GPS provides global positioning data, enabling for precise location on earth's surface. By combining GPS data with other sensor inputs such as LiDAR, cameras, and odometry, vehicles can achieve more precise localization, reducing errors and improving overall navigation accuracy.

- Navigating with speed limits

  For SDC to fully reliable we can desgin SDC to limit the maximum speed of car in speed restriction areas marked on a map.

- Dynamic Object Following

  We can use navigation stack to track and follow moving objects in real time, enhancing ability to navigate safely and efficiently in dyanmic environments.

References

[1] Sebastian Thrun, Michael Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, and Pamela Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *J. Field Robotics*, 23:661–692, January 2006.

[2] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163, October 2015. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[3] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From Coarse to Fine: Robust Hierarchical Localization at Large Scale, April 2019.

[4] Oral-History:Ernst Dickmanns, December 2020.

[5] CMU. Navlab 5 Details.

[6] DARPA. The Grand Challenge.

[7] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Paixão, Filipe Mutz, Lucas Veronese, Thiago Oliveira-Santos, and Alberto Ferreira De Souza. Self-Driving Cars: A Survey, October 2019. arXiv:1901.04407 [cs].

[8] On-Road Automated Driving (ORAD) Committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. April 2021.

[9] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[10] Extended Kalman filter, November 2023. Page Version ID: 1185257110.

[11] E.A. Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, October 2000.

[12] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[13] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-Time Loop Closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[14] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 2017.

[15] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment — A modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–372, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description, April 2018.

[17] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning Feature Matching with Graph Neural Networks, March 2020.

[18] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[19] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed, June 2023.

[20] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition, May 2016.

[21] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement, April 2018. arXiv:1804.02767 [cs].

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241, Cham, 2015. Springer International Publishing.

[23] Fei Liu, Zihao Lu, and Xianke Lin. Vision-Based Environmental Perception for Autonomous Driving, December 2022. arXiv:2212.11453 [cs].

[24] N. Minorsky. DIRECTIONAL STABILITY OF AUTOMATICALLY STEERED BODIES. *Journal of the American Society for Naval Engineers*, 34(2):280–309, 1922. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1559-3584.1922.tb04958.x.

[25] Carlos E. García, David M. Prett, and Manfred Morari. Model predictive control: Theory and practice—A survey. *Automatica*, 25(3):335, 1989.

[26] Hari Om Bansal, Rajamayyoor Sharma, and PR Shreeraman. PID controller tuning techniques: a review. *J. Control Eng. Technol*, 2(4):168–176, 2012.

[27] Scott McLachlan, Evangelia Kyrimi, Kudakwashe Dube, Norman Fenton, and Burkhard Schafer. The Self-Driving Car: Crossroads at the Bleeding Edge of Artificial Intelligence and Law, February 2022. arXiv:2202.02734 [cs].

[28] Fredrik Matsson. Sensor fusion for positioning of an autonomous vehicle, 2018.

[29] Ackermann steering geometry, September 2023. Page Version ID: 1177050788.

[30] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, January 2023.

[31] Michał J. Tyszkiewicz, Pascal Fua, and Eduard Trulls. DISK: Learning local features with policy gradient, October 2020.

[32] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[33] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6(61):2783, 2021.

Appendices

It should contain the information that is somehow missing in the main body and it presents extra information about your project like program source code, raw data, and so on.