Vellore Institute of Technology, Vellore

AutoZ - Pratham

Figure 1. Pratham

Date Submitted : 15th May 2023

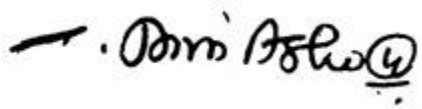
Team Captain

Shashank M

Team Members

Neeraj Kumar Saini | Hrishikheshan Ghiridharan |Magapu Eswar Krishna Chaitanya | Akash K S | Deepesh padala | Gokul Krishnan S | Hariharan S | Harikrishna U Kamath | Karthik Krishnan | Nadheem Nassar Matara | Nandha kishore balraj | Sahishnu Raju K | Shashank M | Harinarayan Ananthakrishnan | Hemanth vasireddy | Kashish Singh gaharwar | Manan Gupta | Arya Agrawal Asmi Dhull | Bhuvan Sundar R | Chaitanya Kandpal | Dhanush Srikanth | Prabhav Gupta | Rahul Rajesh | Satvik Jain | Umar khan | Yashas katte

I certify that the design and engineering of the vehicle Pratham by the current student team has been significant.



Prof. Denis Ashok Sathiaseelan, Faculty Advisor

Introduction

Team AutoZ is an undergraduate student team from VIT, Vellore. We focus our work on autonomous technologies and in the field of robotics. Our end goal is to create an autonomous mobile robot that can be retrofitted with different sensors and actuators as per the application such as Delivery, Agriculture, Mine exploration etc.

Organization

The team comprises undergraduate students under the guidance of our faculy advisor Prof. Denis Ashok. The members are part of 4 working groups; Mechanical, Electronics & Electrical, Software and Social Media.

Role	Name	Domain (s)	Major
Captain	Shashank M	Electronics	ECE
Vice Captain	Nadheem Nasser	Mechanical	MECH
Embedded Systems Lead	Akash KS	Electronics	EEE
Autonomous Software Lead	Deepesh P	Electronics, Computer Sc	ECE
Design Lead	Hariharan	Mechanical	MECH
Manufacturing Lead	Nandakishore	Mechanical	MECH
Social Media Lead	Harikrishna	Social Media	CSE
Machine Learning Lead	Karthik	Computer Sc	CSE
PCB Design Lead	Sahishnu Raju K	Electronics	ECE
Power Electronics Lead	Gokul Krishna	Electronics	EEE
Embedded Systems	Rahul	Electronics	ECE
Software	Hemanth	Computer Sc	ECE
PCB	Arsalan	Electronics	ECE
Safety systems	Manan	Electronics	CSE
Social Media	Chaitanya	Social Media	ECE
Social Media	Arya	Social Media	CSE
Safety Systems	Asmi	Electronics	EEE
Software	Yashas	Computer Sc	CSE
Mechanical	Umar	Mechanical	MECH
Mechanical	Harinarayan	Mechanical	MECH
Mechanical	Dhanush	Mechanical	MECH
Computer Vision	Prabhav	Computer Sc	IT
Mechanical	Bhuvan	Mechanical	MECH
Embedded systems	Kashish	Electronics	ECE
Signal Processing	Satvik	Electronics	ECE

Table 1. Team Composition and Roles

Design Assumptions and Design Process

Along with adhering to the IGVC Rules and Regulations we emphasized on engineering a bot that is modular with easy disassembly and reassembling capability. During the design and prototyping phase we relied on rapid prototyping and testing to reach an effective solution.

Figure 1. Design Process

Cost Estimate

The cost of the robot has been split into two parts. The first column represents the Retail Price of the component. The second column represents the cost to the team for the year. Sponsorships and previously used components helped us significantly reduce costs. This allowed us to create a low cost yet efficient system.

ITEM	Retail cost	Qty	Raw Cost	Team Cost
Motor Driver	$ 85	1	$ 85	$85
RC Transmitter and Receiver	$ 75	1	$ 75	$0
High Current Power Distribution	$ 360	2	$ 720	$100
STM32F103 Bluepill	$ 7	1	$7	$7
ESP32 WROOM	$ 5.50	1	$5.5	$5.50
Realsense D415	$ 450	1	$450	$0
Jetson TX2	$ 450	1	$450	$0
LiPo Batteries (6s)	$ 380	1	$380	$380
Transmission	$ 31	1	$31	$31
Xsens Mti - 670G	$ 2970	1	$2970	$1485
Aluminum Extrusion	$ 185	1	$185	$0
Motors	$ 75	2	$150	$150
Low Current Power Distribution	$ 135	1	$135	$135
LiDAR	$ 105	1	$105	$105

Asus ROG Strix G15	$ 1000	1	$1000	$0
Wiring and Framing	$ 100	1	$100	$100
Total Cost				$2,524

Table 2. Cost Breakup of the Bot

Mechanical Design

1. **Overview**
   Prathams design was formulated with a strong focus on modularity to achieve faster assembly and disassembly. We implemented the "**Switch and Run**" concept to create a bot capable of various functions such as delivery, mine detection, agriculture etc. With minor changes to control systems and sensor stack, we can adapt the same chassis to all the aforementioned purposes. For the competition we have gone for a design capable of running efficiently on road.

2. **Structural layout**
   Our bot's design concept of "Switch and Run" is the basis for the frame, which is constructed using 40x40 T slot aluminum extrusion profiles. The aluminum alloy used in the frame is as strong as some medium-grade steel alloys but is three times lighter than steel. The frame is designed for easy use of different aluminum joint profiles in the building of the bot. The dimensions of the bot meets the requirements of the competition
   Vertical support has been provided where more load is applied. We have used the L bracket and inside corner bracket to join different supporting rails. The structure is made into 3 compartment:

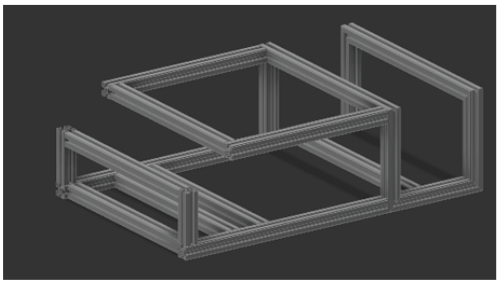


Figure 2. Frame Design

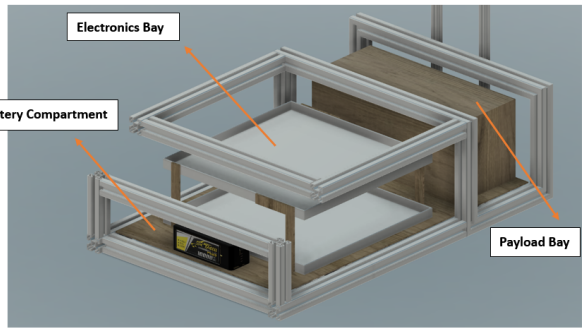


Figure 2.1. Compartments of inside Pratham

   a. **Payload Bay** - where payload is placed for optimum distribution of weight and secure it during transit.
   b. **Electronics Bay** - Where we place our electronics Box and it placed in for easy access for troubleshooting
   c. **Battery Compartment** - Where we place our batteries for easy equipping/swapping.

   The Intel RealSense sensor has been mounted at the rear of the structure using two 20x20 aluminum extrusions, and a horizontal mount for the RealSense camera clamp which allow the sensor height and the angle to be adjusted based on the bot's computation and terrain requirements.
   The RP-Lidar is mounted to the front of the bot using a custom 3D-printed mount and shield to get a precise required field of view for the sensor.

3. **Drive Train:**

Pratham is propelled by a two-wheel powered drive with a rear dual caster. The motor is connected to a shaft, which is mounted on a sturdy metal sheet using clamps and rubber bushings to absorb minor vibration shocks. A custom made protective aluminium casing is provided for the motor. The shaft is connected to two RBI bearings to prevent damage from vibration and torsional stress on the motor. The wheel is connected to the shaft, and the power from the motor is transferred to the wheel. The wheel has a 13-inch diameter and is made of rubber polyurethane.

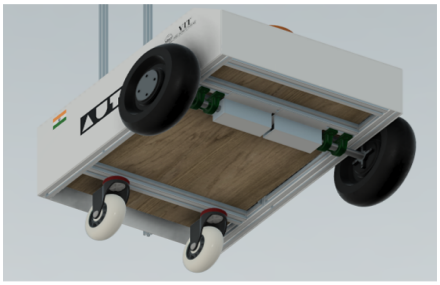


Figure 3. Drive Train

4. **Steering:**

Pratham incorporates the **"Differential Steering"** mechanism to facilitate the smooth maneuverability of our robot around tight corners. This is done by varying the velocities of the two forward drive wheels, which changes the trajectory of the bot.

We can also perform yaw motion, where the bot rotates along the Instantaneous Centre of Curvature.

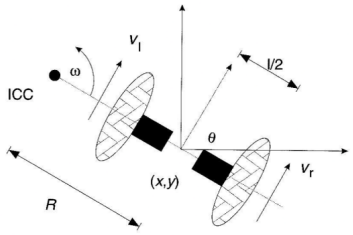


Figure 4. Steering Working Diagram

5. **Analysis:**

FEA can help optimize the rover's weight, strength, stiffness, and other factors critical to its functionality and efficiency. FEA models can simulate the rover's behavior under various conditions, including different terrains, obstacles, and payloads. This analysis allows engineers to make informed decisions about the rover's design and geometry, leading to an optimized and efficient design. Overall, FEA is an essential tool in the development of Pratham, ensuring that it meets the necessary performance requirements and delivers reliable and efficient operation.

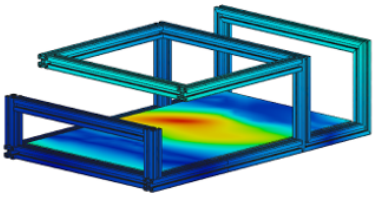


Figure 5. Frame Structural analysis

6. **Body and Weatherproofing:**

Pratham's frames and components are enclosed using multiple sheets of **PVC foam board**. Using these foam boards has a lot of advantages. Water Resistance helps us to waterproof the inner part of the bot, Corrosion Resistance protects the body from chemical reactions, Fire Resistance protects the body and the components from fires. It has a very high strength to its size and its ease of machinability has helped save tool and time costs and it is cheaper than other alternatives, like mild steel and aluminum sheets. In addition to that we wrapped it up with vinyl wrap to endure it from damages and for it to look good aesthetically. The foam board was also used as a base platform to attach all the electrical components of the Electronics Box.

7. **Electronics Box:**

In Pratham, the electronics and electrical components are housed in a **"Electronics Box"**, to make it easier for the team to integrate the electrical and electronics components into Pratham and for ease of access.It is constructed using two Aluminium plates stacked on each

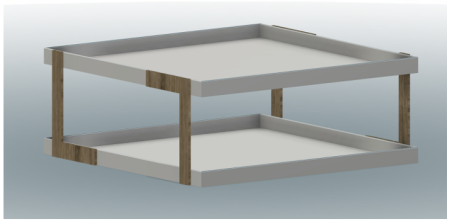


Figure 6. Electronics Box Structure

other with four wooden L-Brackets bolted to support the two layers on top of each other.
The first layer houses all the power distribution and control system components, and the second layer houses all the computation and sensor components.

8. **Implementing Additive manufacturing\ Techniques:**
Additive manufacturing i.e, 3D printing is being used in Pratham for various purposes. We custom designed mounts for the electronics and sensors and used our inhouse Creality Ender 5 Plus 3D Printer to get all the required parts printed. We used Ulitmaker Cura slicing software to get our part compatible to get it 3D printed as shown here.

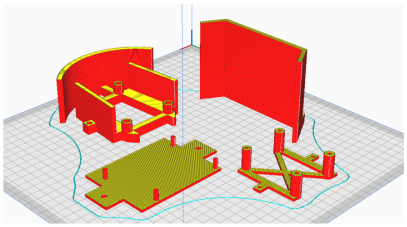


Figure 7. Preview of Slicing for a 3D print

Electronics and Power Design

1. **Power Distribution System (PDS)**
Pratham is powered with a 22Ah 6s 24V LiPo , it has a low current PDS with an array of voltage options (12V, 5V and 3.3V) the ESP32 is powered by a 5V supply from the PDB whereas a 3.3V supply powers the onboard STM32 Bluepill. There are two 18V high current PDS used on the bot. One supplies power to the Motor controller which connects to each of motors through a 12V 20A relay, the other one powers the Jetson TX2.

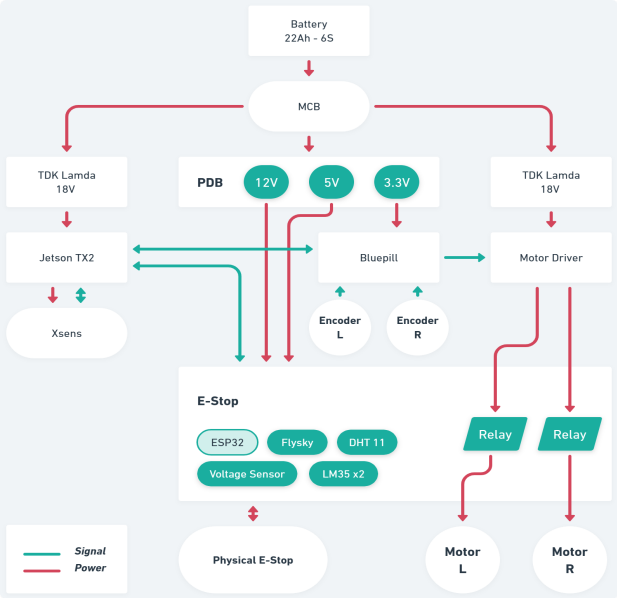


Figure 8. Power Distribution System

2. **Electronics and Sensor Suite**
Pratham uses a Jetson TX2 development kit and an AMD Ryzen 9 5900HX Laptop as its computing system for prognostic calculations. It's connected to the following list of sensors .The XSens IMU is connected to Jetson using USB used as a sensor input for localisation. The Jetson communicates the STM32 Bluepill over USB publishing speed commands which runs the Fuzzy PID algorithm for motor speed control and in turn communicates the feedback of RPM from encoders back to the Jetson. Pratham contains two Rhino RMCS-2083 motors with inbuilt encoders to enable speed control. Xsens MTi-670G which encompasses IMU and GPS along with inbuilt encoders in the motors were used for the wheels to

provide imperative feedback necessary for localisation of the bot. Perception sensors include an RPLidar and a realsense depth camera which are connected to the laptop.

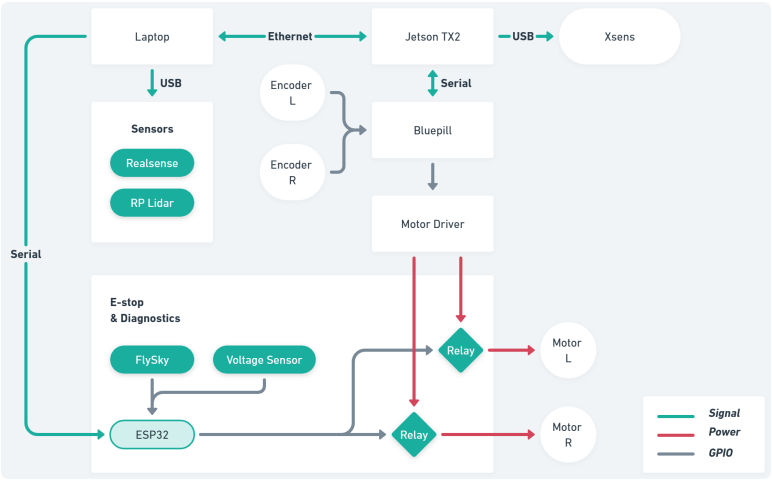


Figure 9. Signal and Communication Diagram

3. Emergency Stop System

Pratham implements a wired and wireless e-stop. It is implemented as a standalone board. A FlySky transceiver is used for the wireless estop. ESP32 is employed as the MCU for the ESTOP system. The signal received from the receiver triggers electromechanical relays to shut off power to the motors. The ESTOP is also triggered when the battery charge is too low to run the bot to protect the LiPo Battery. The ESP32 also communicates the status of the ESTOP to ROS2 running on the TX2 using USB serial communication.

Autonomous Software

1. Software Strategy

The robot's autonomous navigation software is onboard Jetson TX2 and a laptop with an AMD Ryzen 9 5900HX, 16 GB DDR4 ram, and an RTX-3050. The entire software stack for the robot was built and tested in a span of 2 months (April, May). It is built on top of the DDS middleware provided by ROS2 (Robot Operating System) incorporating; localization, perception, path planning and following. A custom navigation stack was built from scratch with C++/Rust instead of utilizing available navigation stacks for better customizability and versatility.

Coordinate frames used in algorithms:

a. /odom
   This is the frame which is representative of the starting point of the robot. For all means and purposes, it is an origin for the state estimation node to use as a reference frame for the robot.
b. /base_link
   The current position of the robot, with respect to the odom frame. This is used by the path planner, costmap generator and path follower to make decisions/get feedback and send control signals.
c. /map_link
   The map frame is a copy of the base_link frame at the instant the map is updated with data from the lidar and other sensors. It serves as a temporary odom/origin for the path planner and follower.

Using the map_link as a temporary origin everytime a new map is generated enables the robot to avoid integral drift and other losses which may be present in the state estimation algorithm.

The ROS2 middleware is being used for its robust implementation of nodes, and topics, which are used to exchange and share data across different nodes which facilitate their respective algorithms.

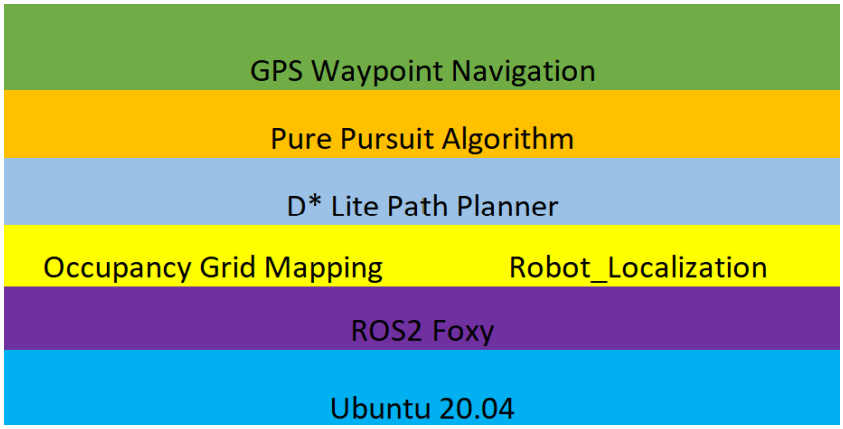


Figure 10. Software Stack

2. Localisation

An EKF (Extended Kalman Filter) was used to fuse data from all the sensors on board the robot and perform state estimation.

The state vector consists of:

- linear position
- linear velocity
- linear acceleration
- orientation
- angular velocity

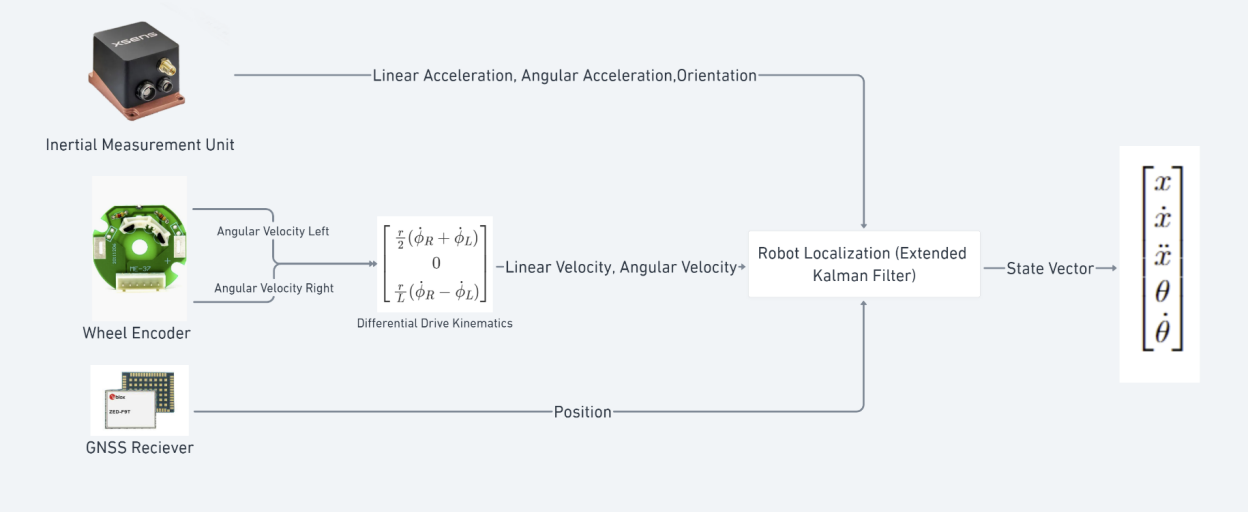


Figure 11. Localization

The robot receives data regarding **linear acceleration, orientation and angular velocity** from the accelerometer, magnetometer, and gyroscope present in the XSens GNSS.

Linear velocity in the x direction, along with **angular velocity in the yaw direction** (z) is obtained from the wheel encoders on the left and right wheels, by passing it through a differential drive dynamics solver. **Position data** is acquired from the GPS sensor on the XSens GNSS.The XSens GNSS provides covariance matrices for all its measurements, and the wheel encoders' covariances were calculated manually and used for the update step of the EKF. State estimation is a very important step and is used across all algorithms which facilitate autonomous navigation.

3. Obstacle Detection and Avoidance

Component	Use
Realsense Depth Camera	It is used to detect lanes and potholes and publish their respective pointcloud
2D LiDAR	Used to detect obstacles and then add them to the map

Table 3. Roles of Perception Sensors

3.1 Lane and Pothole Detection

1. Using HSV Filtering filter out white pixels from the image
2. Detect Potholes using HoughCircles function
3. XOR between HSV image and Pothole image to obtain only Lanes.
4. Apply the masks on depth image
5. Convert depth images into point clouds and add them to the map in their respective layers.

4. Map Generation

A custom bird's eye view environment model was built which performs sensor fusion to fuse the data from the lidar and depth camera with specific post processing methodologies such as inflation and extension. Subsequent costmaps such as the one shown above are collected and arranged according to the position of the robot when it was generated and are used to make a global map of the IGCV course. Map is updated every 0.15s with an inflow of new data. The map contains data of all obstacles in an 8x8m window in front of the robot. As shown in figure 13, the map always lies in front of the robot, and is updated similarly.

The perception nodes provide point-cloud data of all the obstacles to their respective topics, which is then added to their respective layers in the costmap. The point-cloud data is placed in tiny cubes called voxels, where the number of points lying in the voxel decides the weight of the corresponding grid in the costmap. Each layer has its own post-processing steps based on the type of obstacle and finally, they are all aggregated using a bitwise or function to be published for the path planner to use.

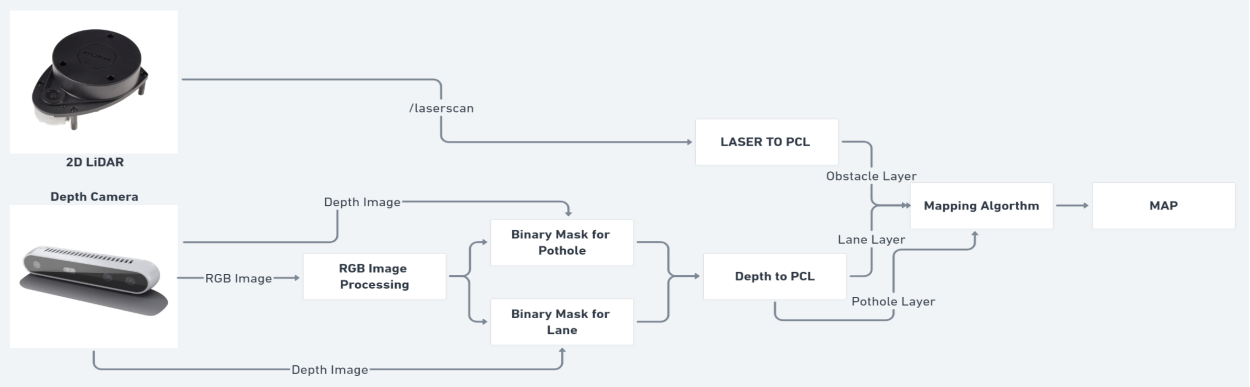


Figure 12. Map Generation

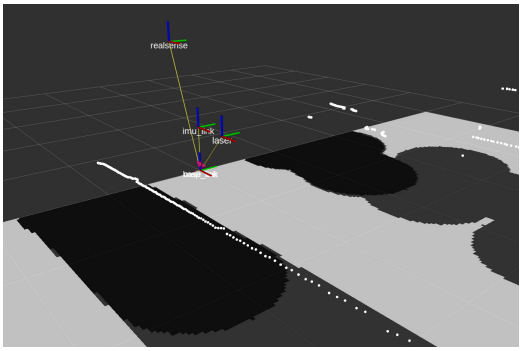


Figure 13. Map with Lanes and Obstacle Layers

4.1 Post Processing

The potholes and lidar-obstacles(barrels) layers are just inflated with the help of erosion operations with a kernel size. The kernel size of the erosion was set to ensure that the area encompassed within 3 feet of an obstacle detected by either the depth camera or a lidar is set to non-traversable. This will ensure that the path planner leaves a 1 feet distance while planning paths around obstacles.

Lane Extension

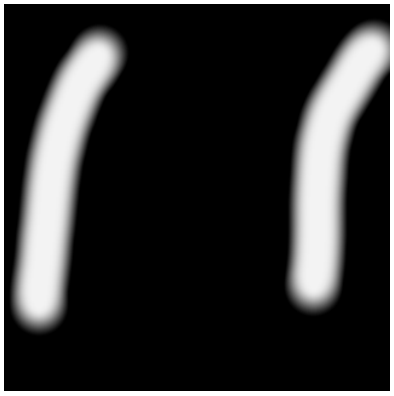


Figure 14. Lane layer of the costmap before post processing

There was a need for heavy post processing due to the limited FOV of the depth camera. Lanes right next to the bot were not detected resulting in an incomplete lane layer as shown in Figure 14.

We observe that the part of the lane right next to the robot is not added. A similar problem is faced when dashed lines are present. This problem is alleviated by dividing the image into 4 segments of equal length along the height, and extending the lanes along the direction cosines of the minarea rectangle with centroids superimposed with the blob as long as the angle made by the lane and the vertical axis was either < 15 degrees or in between 87 - 93 degrees, to accommodate all use-cases. The segments shown below are for if the lanes are directly parallel to the robot.

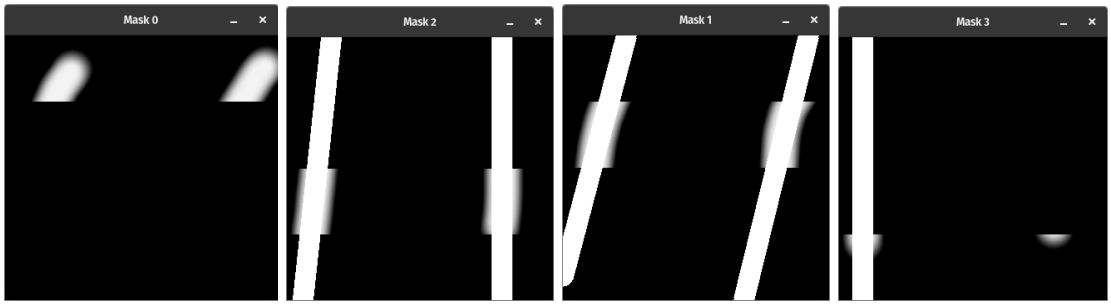


Figure 15. Sliced Images for Post Processing

Figure 16. Post Processed Image

The same post processing technique works well with Dashed Lanes as shown in Figure 17.

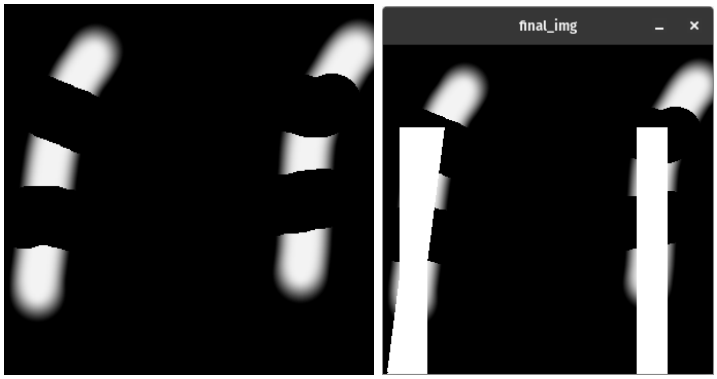


Figure 17. Dashed Lanes

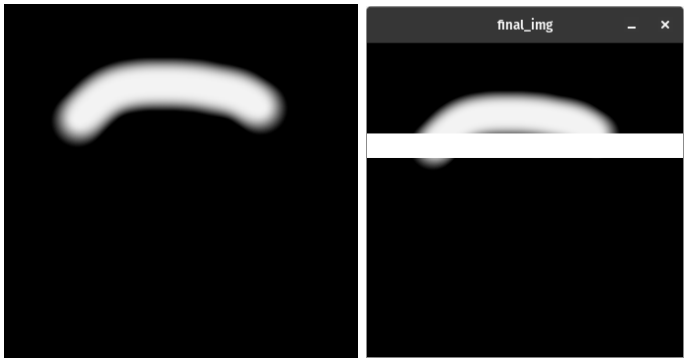


Figure 18. Lane horizontal to the bot

5. Path Planning and following

The local map generated gets updated every 0.15 seconds, and moves along with base_link. The path planner node, which uses the D* Lite algorithm, subscribes to the map, and generates a new path every time it receives a new map. The D* lite algorithm accepts goal coordinates in the XY-plane with respect to base_link. Goal selection using GPS is described in more detail in the next section. The D* lite algorithm was chosen over A* as it planned paths which preferred staying away from obstacles over creating the shortest path, and was computationally inexpensive unlike RRT*, and produced an optimal path unlike RRT.

Path following is done with an algorithm inspired by total pure pursuit. Where the nearest points in the path are compared to the current state, as well as what the state will be after propagating it to the future by *lookahead_time* seconds. *lookahead_time* is a tunable constant. A fuzzy proportional control algorithm was implemented which takes the current error and the propagated error to generate velocity commands which will be given to the control systems.
The maximum linear velocity that the path follower will request depends on the curvature of the path. I.e. higher curvature would require a lot of turns, during which the algorithm has been designed to request slower speeds to maintain stability.

6. Goal selection and path generation

The robot transforms given goal GPS coordinates to UTM and adds it to a queue. The initial coordinates of the bot are added to the end of the queue.
To generate a path, the GPS navigation node listens to the GNSS topic which gives the current latitude and longitude everytime the GPS receiver gets a ping. The current UTM coordinates are subtracted from the current UTM coordinates to attain the goal vector. The goal vector, rotated by the heading of the robot (angle between x-axis of the base_link and the “north pole”) gives the goal which should be given to the path planner.

The path planner has been designed to accept goals which lie outside the costmap. If a goal is given outside the costmap, it plans a proper path till the edge of the map, and then takes the shortest distance to the goal. Once the robot starts following the path, it sees a new part of the course and generates a new map.

The goal UTM coordinates are stored in a queue. When the robot reaches within 0.4m of the current goal, it is popped out of the queue and the next one is used for all the subsequent path generation.

Failure Modes, Failure points and Resolutions

1. Mechanical

Sl. No.	Failure Mode	Resolution
1	Tyre Wobbling	The issue could be because of the damage to the wheel or the tyre separation. This could be solved by leveling the wheel and shaft position or permanent damage to the wheel requires replacement.
2	Bolts loosening leading to vibrations	Use of spring washer and thread locking adhesive solution to prevent further loosening.
3	Non-uniform Wheel traction results in variable power distribution to the wheel.	Could be resolved with an adaptive control system or proper weight distribution.

Table 3. Mechanical failure and resolutions.

2. Electrical

Sl. No.	Failure Mode	Resolution
1	Loss of communication with E-Stop transmitter	E-Stop detects loss of signal and cuts off power to motors.
2	Overheating may cause malfunction	Fans have been placed to dissipate heat from the chassis
3	Sensor failure (Encoder, IMU)	Power to motors will be cut if odometry from Encoders and IMU have deviations.

Table 4. Electrical failure and resolutions.

3. Software

Sl. No.	Failure Mode	Resolution
1	Failure of perception sensors	Behaviors are programmed to compensate for the loss of data and to Stop in case of LOS from multiple sensors
2	Odometry frame drifting away from the actual state	The measurements fill up the state vector completely, reducing the errors due to integral drift. Unreliable sensors have inflated and/or dynamic covariances to ensure that the EKF doesn’t rely on them much for the updation step

3	The DDS network is open to anyone in the same LAN	The domain_id has been changed. The DDS network can be shielded off from the LAN during deployment.
4	Potholes might get added to the lane layer and get extended	The minarea rectangle drawn is checked to determine whether it is close to a square i.e. check if abs(length-width) >= tolerance before extending the lanes

Table 5. software failure and resolutions.

Simulations

A simulation test environment was created using Blender and exported to Gazebo. This involved recreating the features of the IGVC track as per the provided IGVC Rulebook and from previous reference. The ground vehicle was simulated with the sensor stack which eventually got implemented on the final vehicle. The simulation also involved retreating a digital analogue of the vehicle (differential drive, size constraints, speed and other limits etc.).The sensors implemented in the simulation are :

a. Realsense depth camera
b. Lidar

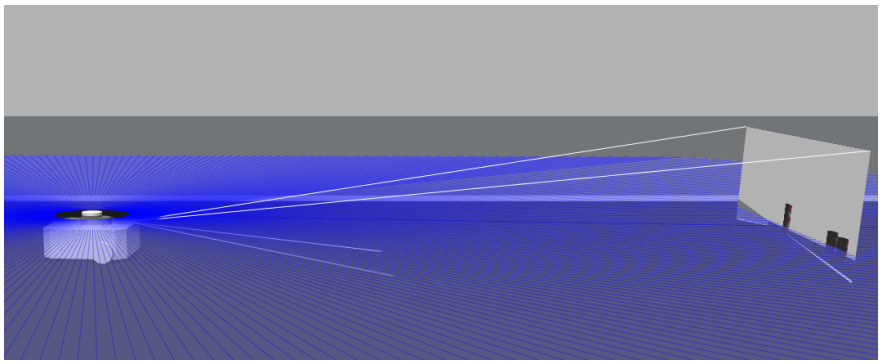


Figure 19. Realsense simulated image

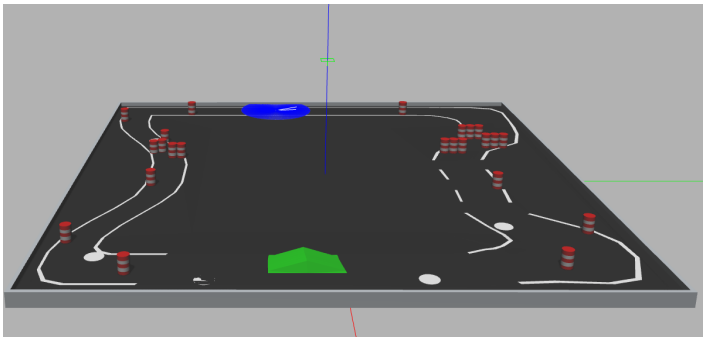


Figure 20. Raw RGB image

Testing and Performance

Apart from the simulation testing performed we have isolated out testing into 5 major sections as per the flow diagram provided.

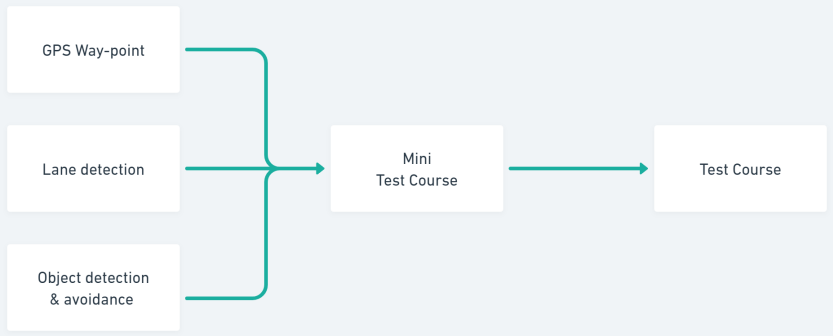


Figure 21. Testing Procedure

Initial Performance Assessments

Parameter	Result	Details
Max speed	5mph	Hardware limited
Max Acceleration	5 m/s²	Software limited
Approach angle	45°	Design parameter
Max Incline	25°	Maximum angle tested
Battery Life	1.5 hrs (approx.)	Approximation based on testing

Table 6. Test data.

Future Prospects

Our current bot started with a focus on modularity and this concept can be taken further by adding hot-swap capability to most components and sub systems. The electronics box and components within can be designed with custom connectors to enable us to remove the box or component without removing or connecting any cables or connectors manually.

To make the bot safe without human intervention a Diagnostics and BMS system can be implemented to notify the user or shut down the bot in case of an anomaly. This can be used in combination with or integrated with the remote E-stop system to ensure both user and bot safety.

Converting the bot to a 4 wheeled rover will allow the bot to traverse harsher terrain unlocking more possibilities to use the bot in the real world. This will also enable better payload capacity and stability compared to the current setup.Software can be written to utilize GPU acceleration, this will reduce the complexity of the systems, and be more power efficient, in addition to operating faster. The perception and path planning pipeline can be done on 3D data instead of 2D. This can accommodate a traversability layer to include ramps, and banked roads as well. With a 3D perception pipeline, reinforcement learning techniques can be used instead of the pure-pursuit controller to make the robot more robust in its path planning