Indian Institute of Technology, Kanpur

Dr. Gaurav Pandey, Dr. Mangal Kothari, Dr. S R Sahoo

Report By: Harsh

I. **The Team**

Third Year Students :

- Animesh Shashtry
- Deepak Gangwar
- Harsh Sinha
- Shubh Gupta
- Swati Gupta

Second Year Students :

- Aalap Shah
- Bhuvi Gupta
- Hardik Maheshwari
- Hemanth Bollamreddi
- Nishkarsh Aggrawal
- Pritesh

First Year Students :

- Anay Mehrotra
- Anav Prasad
- Ashish Bhatti
- Ashish Patel
- Siddharth Srivastav
- Vaibhav Aggrawal
- Yash Chandnani

Contents

Abstract

In this report we present the techniques involved in applying the concepts of Localization, Mapping, Path Planning on a Ground Vehicle platform for autonomous navigation in unexplored territories. Further we also deal with the vision methodologies involved for enabling the robot to traverse within lanes, avoid obstacles and maintain its heading while navigating in the said unfamiliar territories.
The Project tackles the requirements for participating in the Intelligent Ground Vehicle Competition

Keywords: Localization, Mapping, SLAM, Motion Planning, Intelligent Ground Vehicle, Waypoint Navigation, Vision

II. **Introduction**

Gone are the days when giant hulking robots were fixtures on factory floors and rarely encountered any situation out of their predetermined environments. Today we are having a revolution in the field of autonomous robotics with Google's self driving cars raking up more hours in California than an average human driver there, to the robots that are exploring the sea bed, quad copters and ground based robots that are exploring unknown environments. Robots today need to able to deal with unexpected environments and challenges. In this project we enable a robot to do waypoint navigation in unknown environments while creating maps of the same environment all the while avoiding obstacles and also move in between lanes drawn on the ground. We divided this problem in three parts namely :

- Vision
- Localization and Mapping
- Motion Planning

This project deals with the implementation of Simultaneous Localization and Mapping(SLAM) and Vision techniques on a ground based vehicle and the use of Motion planning techniques on the maps to enable the aforementioned vehicle to do way point navigation in unknown environments and to maintain the vehicle in between lanes drawn on the ground, all of this is done online.
The major challenges here are to get correct data from every sensor in the common frame of reference, create a map using both vision and LiDAR data, use of correct vision algorithms to obtain the information from the images and then in the correct amount of time find intelligent paths for the robot to travel along. We got very good results indoor but couldn't get a map outdoors.
This report has been divided into two parts one dealing with the Vision aspects of the robot and the other deals with the Localization Mapping and the Motion Planning aspects. In both the sections we flow with the data from the sensors on-board; we start with explaining our methodology for getting the information form the sensors, follow it up with the expansion on the algorithms used. To end the report we present the results that we have obtained.

III. **Vision**

A. Methodology

The hardware used for the image capture is a web cam called Genius wide cam, with 120° view of the environment and the processing is done on Intel NUC on the robot itself, although the option to do the processing on any laptop of choice is there. The primary objectives of the vision algorithm are as follows -

- Obstacle Removal
- Lane Detection
- Local and global Occupancy Map updation

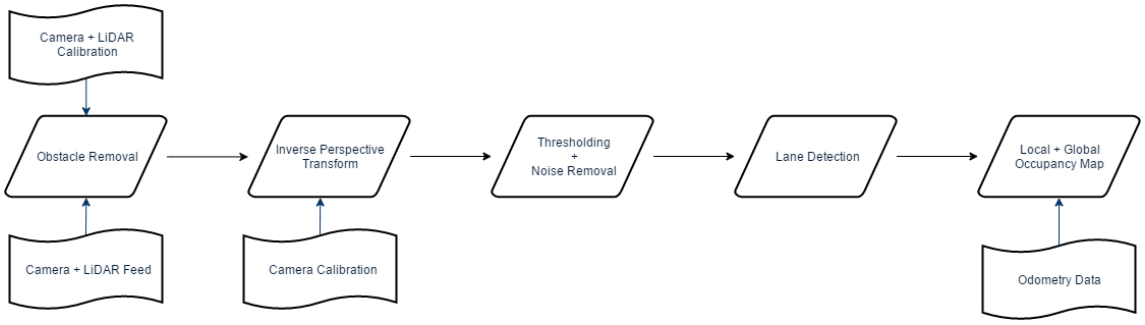


Figure 1: Algorithm of the vision stack

The vision stack is divided into following subsections, which are detailed upon in the following sections:

- Camera undistrotion and Odometry and association
- Lidar-Camera Calibration and Obstacle Elimination
- Overhead view transformation
- Lane detection
- Heading determination
- Global Map generation

B. Camera undistortion and Odometry association

To begin with, we need to procure an image of the space in front of the robot and convert it into a suitable format for further processing. We have used a wide angle camera for this purpose with a field-of-view of 120 degrees. With such a wide field of view, the image produced by the camera is distorted at the edges and hence,

needs to be undistorted. Radial distortions arise as a result of the shape of lens, whereas tangential distortions arise from the assembly process of the camera as a whole. Radial correction is of the form

$$x_c = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_c = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Tangential correction is of form

$$x_c = x + 2p_1 xy + p_2(r^2 + 2x^2)$$
$$y_c = y + 2p_1 xy + p_2(r^2 + 2y^2)$$

Also, the camera intrinsic matrix [1] needs to be determined which is of the form

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where x, y are camera pixel coordinates and w is the homogenizing term and X, Y, Z are real world coordinates. We use the classical black-white chessboard approach using opensource openCV codes to obtain the distortion coefficients as well as camera intrinsic matrix, and save them to a file. We then use this data while capturing the image to undistort it.
Since the further processing of the image passes it through various nodes, the image might be appreciably delayed when it is to be combined into the global map, and would hence be correlated to false odometry. To avoid this issue, we combine the image with the correct odometry at a previous stage and pass it along as a hybrid message of image data and odometry.

C. Lidar-Camera Calibration and Obstacle Elimination

Obstacle removal is an important step because obstacles act as a hinderance to the lane detection algorithm. We perform this step by mapping the data obtained from a 2D LiDAR on the video feed from the camera. In order to do so, we need a transformation matrix from the LiDAR reference frame to the camera reference frame. This transformation depends upon the placement of the sensors on the robot. Hence, we first fix the sensors on the robot and then calibrate them to obtain the required transformation. The rough values of rotation and translation matrices are calculated by physically measuring these quantities. These values are then fine-tuned manually by ensuring that there is a proper overlap of the LiDAR points on the video feed.
Using this transform, we find points on the image corresponding to the obstacles. Using a small patch around these points as an input for the watershed algorithm [2], the foreground is separated from the background. The foreground is assumed to contain all the obstacles. Hence, we replace the entire foreground with zero valued pixels so that they do not interfere with the lane detection algorithm.

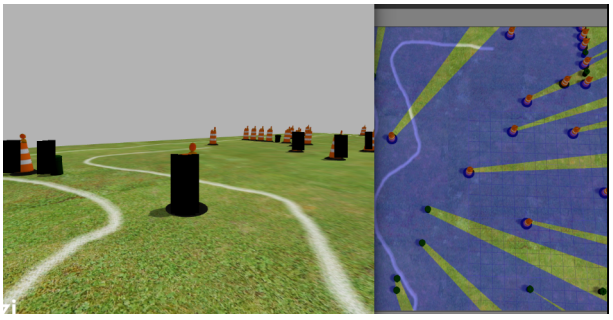


Figure 2: Left - Obstacle removed image, Right - LiDAR scan

D. Overhead view transformation

The obstacle-free image has a perspective error because of which the lanes appear to be converging towards each other which makes separation between the right and left lane difficult. Hence, this error is removed by an inverse perspective transform of the image to give a bird's eye view. This transformation is important as it makes path planning easier after the lanes have been detected.
The homography matrix for this transform is obtained by placing a chessboard on the ground plane and obtaining the transform by detecting chessboard corners [3] and comparing them with their true dimensions which correctly aligns corner points of the chessboard as they should appear from an overhead view.
The Overhead view transformation is a rotation about the camera center which converts the real world ground plane into a plane directly in front of the camera and parallel to the image plane. Hence, first all image pixels are mapped to the ground plane, followed by a rotation and conversion back into pixel coordinates. This process is simplified to a multiplication with a perspective matrix.

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

here, x, y, w are homogeneous pixel coordinates and X, Y, Z are mapped real world coordinates of the ground in front of camera.

Figure 3: Left - Original image, Right - Inverse Perspective Transform (Bird's eye view)

E. Lane detection

The lane detection algorithm makes two assumptions about the lane - 1) The lane is white in colour, 2) The degree of curvature of the lane is very less hence it can be approximated by a straight line.
Using the first assumption, a simple thresholding of pixels based on RGB values is used to filter out background points and keep only the lane points. Erode and dilate functions are used to remove noise from this filtered image. Assuming our lanes are straight, Principal Component Analysis [4] is applied on the lane points to find the best fit line passing through them.
In order to make the above algorithm more accurate, lane points close to the previous location of the lane are used for PCA because the frame rate is considerably high as compared to the speed of the robot. Hence, we can assume that the lane will move a very small distance in successive frames.

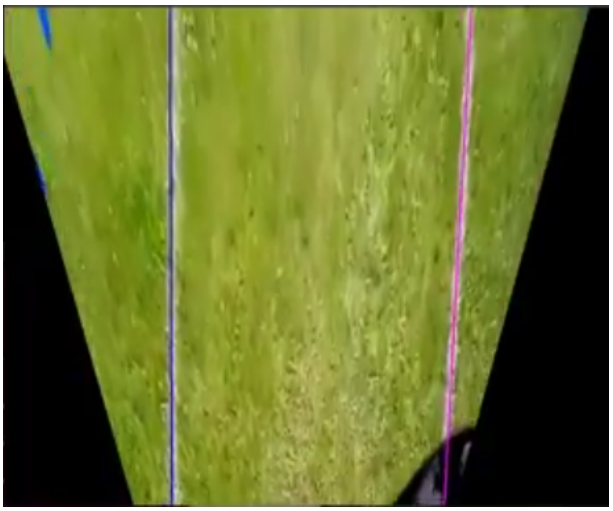


Figure 4: Left lane - Blue line, Right Lane - Pink line

F. Heading determination

Motion Planning algorithms require the vision stack to return an approximate heading direction for the robot, so that an efficient path can be planned. This heading can be assumed parallel to the lane direction, as the robot must roughly move along this direction at all times.
Using the output from the lane detection algorithm, we apply canny edge detection [5] , which detects all edges in the frame, in terms of rho(edge length) and theta(slope angle). Assuming that the frame only contains the lanes(as it comes from the lane detection algorithm), edge detection returns edges roughly parallel to the lanes, and an average of all the theta's obtained can thus be taken as the approximate heading. A further averaging with the values obtained in the last 20 frames(sliding window), ensures that the algorithm returns a smooth and jerk invariant output.

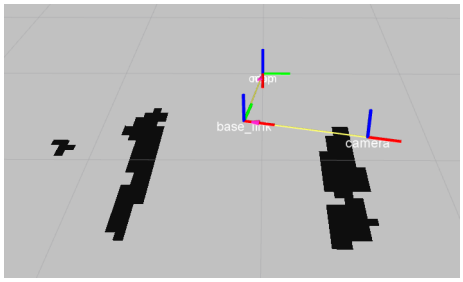


Figure 5: Occupancy Map

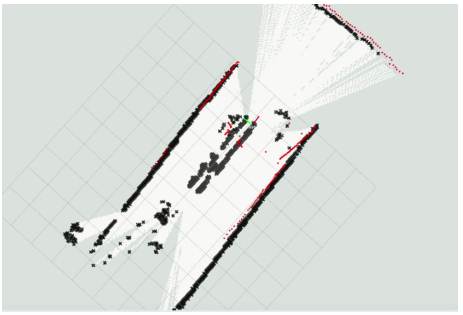


Figure 6: Global map

G. Global Map generation

Once the lane lines are seperated from the rest of the image by the image by the lane detection node, the obtained image must be fused with the occupancy grid containing the lane map. Here we utilize the combined and corrected odometry generated by fusing IMU data and encoder data and corrected by scan matching of lidar data. This odometry had been fused with the image message during an earlier stage and passed along the nodes so as to avoid propagation delays. Utilizing this odometry as well as appropriately scaling the image to the resolution of odometry (determined using intrinsic parameters of camera), we calculate the corresponding rotation and translation matrices and use them to overlay the image over the occupancy grid [4]. Each stack of the image increments that point's probability value, which has a resolution of 0.01.

IV. **Localization Mapping and Motion Planning**

A. Methodology

In this project we didn't follow any one paper fully, instead we worked on combining various methods for autonomous navigation of ground vehicles and implementing them on the hardware available to us. The methodology used is listed in order below :-

1. Getting Sensor data.
2. Conditioning the data for further use.
3. Fusing the data from various sensors using EKF [6].
4. Creation of maps using Slam Gmapping [7].
5. Inflation of map elements for motion planning.
6. Motion Planning using RRT [8].
7. Testing indoors and outdoors.

(a) Fire Bird VI (b) Laser Scanner

Figure 7: Hardware Setup

B. Hardware Setup

We used a NeX Robotics' Fire Bird VI as our robotic platform shown in figure 7a. It is a two wheel drive robot with a built in low level controller that handles the direct input to the motor controllers. The following sensors are mounted or embedded on it :

1. Wheel encoders: Embedded on the robot itself there are two, one on each wheel, optical encoders which give a tick count on how much the wheels have rotated. The resolution of the encoders is 3600 ticks per revolution.
2. IMU: SparkFun Razor 9DOF, has 3-Axis Accelerometer with ±16g and 13-bit resolution, 3-Axis Gyroscope and 3-Axis Magnetometer. With an onboard 8 MHz ATmega 328P, this IMU is capable of calculating the angles onboard itself. This has a prebuilt driver in ROS.

3. Laser Scanner: We used a 360° RP LiDAR Laser Scanner that provides an angular resolution of 1° and rotates at 5.5 Hz. The maximum and minimum range for this device are 6 and 0.2 meters respectively, in between it provides ranges with an accuracy of 1%. RP LiDAR, shown in figure 7b, also comes with a prebuilt driver in ROS.
4. GPS: We used a Ublox Neo M8 GPS module, for outdoor navigation. The accuracy with this module was 4 meters. Although inside we found it not possible to get any fix(link from satellites) outdoors after few minutes we get the fixture.

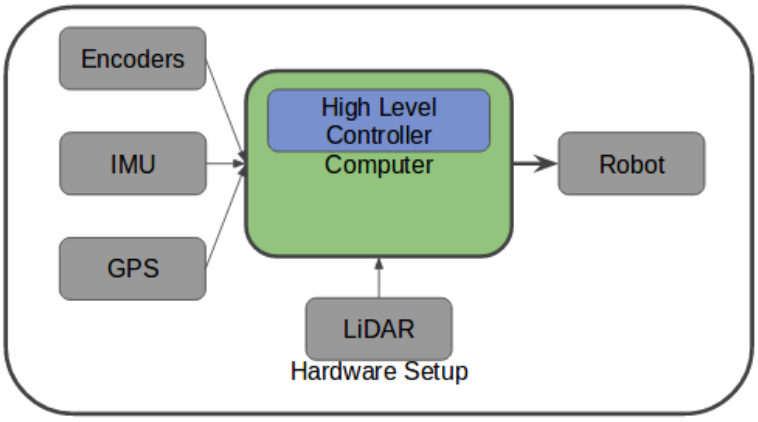


Figure 8: The overall hardware architecture

In addition to these, we also used any laptops at our disposal running Ubuntu 14.04 on mostly sporting an Intel i5 chip, for computing.
The overall hardware architecture is shown in the figure 8.

C. Software Setup

We have used Robot Operating System (ROS) Indigo running on Ubuntu 14.04. We opted to use ROS due to the these reasons :

- Fire Bird had ROS integration available along with most of the sensors.
- ROS enabled us to go for a very modular program structure where every sensor, every task could be easily handled separately.
- ROS lets one manage difference in data rates of various sensors and the code execution frequency by creating buffers of data in form of topics.
- ROS is now a very popular tool among the roboticists, as a result it has a great abundance of resources for various tasks. For instance we used Slam Gmapping which is based on Openslam Gmapping, the original implementation of the paper "Improved Rao Backwellized Particle Filter[7]".

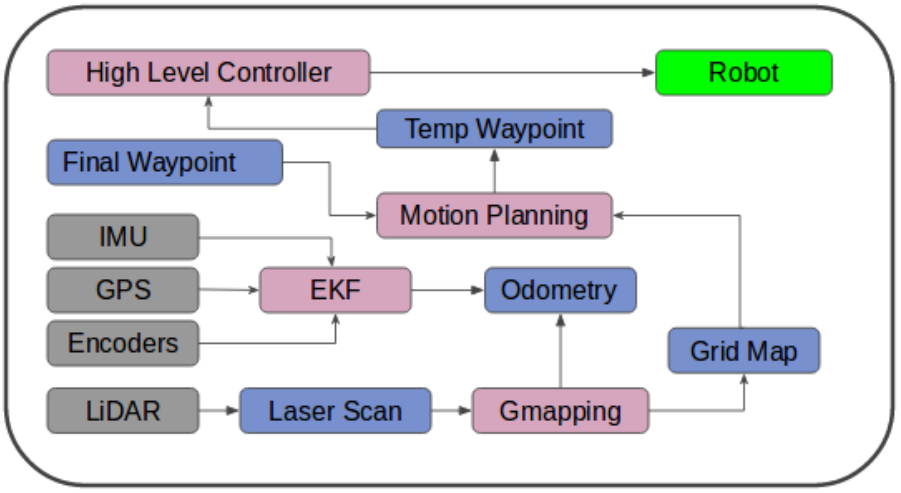


Figure 9: Overall software architecture.

The full software architecture is depicted in the figure 9. The sensor driver nodes are depicted by gray boxes. Every node(pink) in the graph communicates by passing messages. A message is simply a data structure, comprising typed fields. A node sends out a message by publishing it to a given topic(blue). The topic is a name that is used to identify the content of the message. ROS nodes are based on a publish / subscribe model which is a very flexible communication paradigm. Any output that needs to be supplied to a node can be *published* to a topic and then the node that is interested in the data will *subscribe* to it.

D. Algorithms

Namely, we have used following 3 algorithms with some customization according to our test robot :

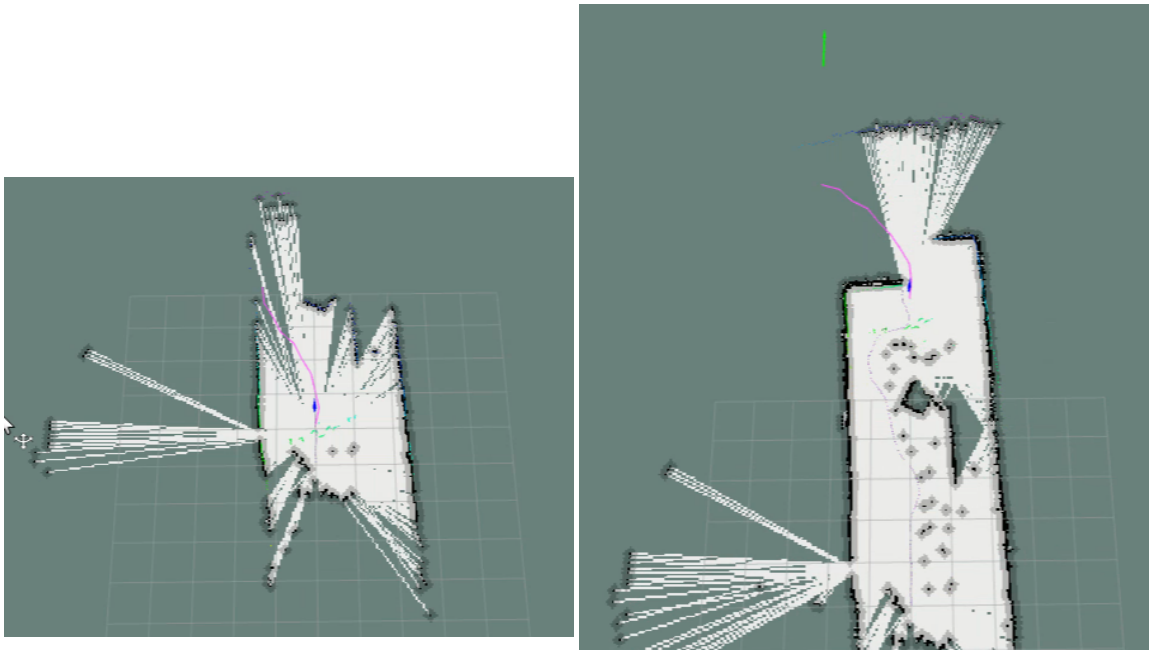
1. **Extended Kalman Filter :** [6] EKF is a very widely used standard algorithm to fuse data from different sources with the assumption that the model is linear and all measurements, positions etc are gaussian in nature. As our test robot uses a differential drive mechanism to traverse, we used the following motion model :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

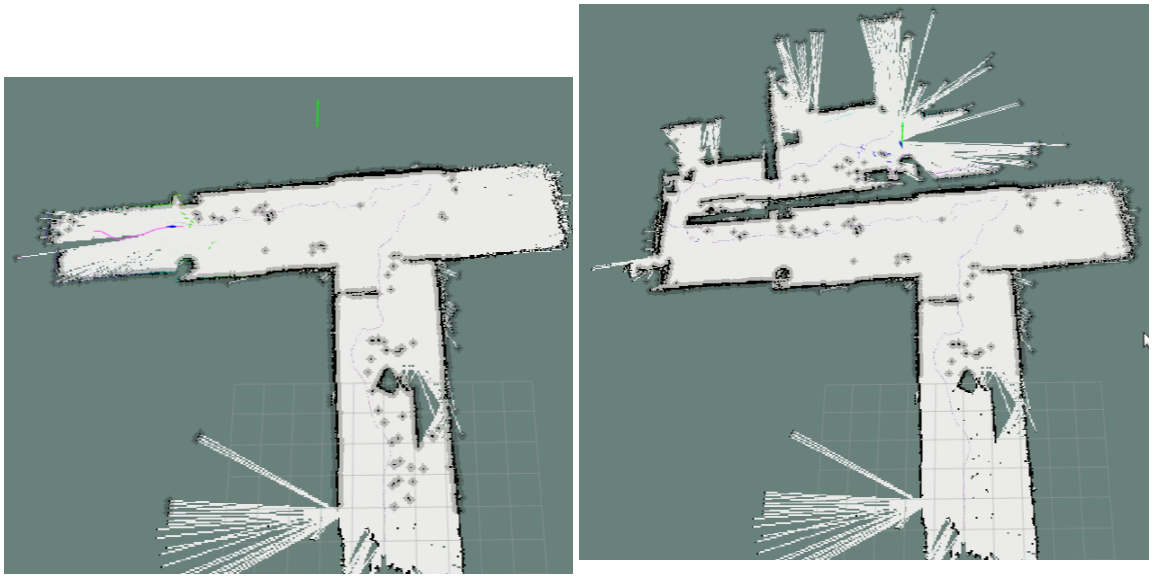
   Where $v, \omega, (x, y, \theta)$ represents linear velocity, angular velocity and position of the robot respectively.
2. **Improved rao-blackwellized particle filters :** [7] Generally particle filter slam is too computationally extensive. It can be made efficient by reducing number of particles but this results in lesser accuracy. So the paper proposes

an adaptive re-sampling method to reduce number of particles in case of grid mapping. In our indoor implementation we used 3 particles.
We used a RosNode[9] developed by ros community based on the paper. The node generates a map as well as improves the position of robot via scan-matching. As to avoid conflict EKF and this node should not publish on the same topic it introduces a new reference frame MAP(earth fixed) and it publishes Transformation matrix between map and odom frame while EKF publishes between odom and robot(base_link) frame.[10]

3. **Rapidly Exploring Random Tree :** [8] RRT is an online fast algorithm to plan path for non-holonomic robots. We implemented RRT in a RosNode and did the following 2 key customization :
   - Put a constraint on the angle of robot turn so that robot may not waste more time in turning. This results in straighter paths and faster navigation of robot.
   - Moreover, to decrease computation we update the path only when the map is updated. Map updation rate is slow so it saves a lot of computation power.



(a) Robot's navigation starting



(b) Robot's navigation ending

Figure 10: Rviz output of Robot's Indoor navigation

V. **Experiments and Results**

We test our vision algorithms in two phases, we test them in a gazebo simulation and then on the robot itself. The details are in the following two subsections. Then we proceed to demonstrate the performance of our software for the localization mapping and motion planning part on Fire Bird VI robotic platform.

A. Gazebo simulation of Vision Stack

All of the codes were tested on a simulated environment developed in gazebo prior to the live run and were found working as per expectations.
NOTE: The gazebo environment has been developed by first year students in Team IGVC and not by students in the course.

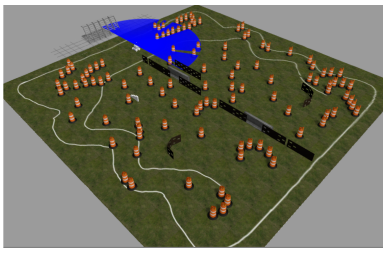


Figure 11: Gazebo simulation

B. Live Run of Vision Stack

The robot has been tested multiple times on surfaces such as green grass, yellow patched grass and cemented surface and gives satisfactory results in an evenly lighted environment. In case of lighting condition changes, primitive thresholding has to be manually updated to give decent results. In case of large variations in lighting conditions during the run (like sudden shadows or glare from sun), the results are not very satisfactory and work needs to be done on improving that aspect. Factors like camera jerks due to uneven surface also lead to unfavorable results, but error created by jerks is mitigated once the robot becomes stable again.

C. Indoor Testing of Localization, Mapping and Motion Planning

A fixed global destination was given to the software and the robot maneuvered itself to the goal in an unknown indoor environment. The results are depicted in the figure 10. It shows the map generated by laser scan (black dots), the inflated map/configuration space(dark gray dots), the explored areas(light gray region), the final goal(green arrow), the robot's current pose(blue arrow), the path generated by RRT(pink line), the robot's optometry (small maroon arrows).

D. Outdoor Testing of Localization, Mapping and Motion Planning

The outdoor tests were conducted in an open ground with nearly no buildings and trees. The robots mapping as well as localization algorithms fail due to the absence of laser scan points. As result of which we weren't able to do the way point navigation outdoors.

VI. **Conclusions**

The technology of autonomous navigation and exploration has an immense range of applications in many fields, and a tremendous scope of further growth and improvement. This project is a small step in this undertaking, with focus on Vision based sensing systems, Localization, mapping and Motion Planning.
Our vision stack has an inherent advantage over other traditional sensing systems like IR, and ultrasonics in that cameras can be used to reduce the overall cost, maintaining high degree of intelligence, flexibility and robustness. We have covered the whole vision stack, starting from undistortion of live video feed, to successful extraction of important parameters like traversable area between the lane, heading, obstacle avoidance and elimination in real time as well as converting them to a form easily integrable with further nodes in the sequence (Mapping, Localisation and Motion Planning).
In this project we successfully demonstrated the autonomous navigation of a ground vehicle in an indoor unknown territory and have developed and easily integrable system for Localization Mapping and Motion Planning on ground vehicles. The paths planned are accounted for the size of the robot and also take into consideration that the general robots are non-holonomic. Our system can work on robots which can provide the odometry information and a LiDAR data only.
Rigorous testing and a complete practical implementation on an actual robotic system implies that all algorithms are feasible, robust and directly applicable in the real world.

VII. **Future Work**

Our future work will focus on extending the approach to provide fully automated camera and LiDAR to camera calibrations, as well as further generalizing the lane detection algorithms, making them light and weather invariant and robust to a wide variety of environments. Further testing of the algorithms as well as a full integration with motion planning also needs to be performed in order to detect any possibilities of failure in certain boundary conditions, and debug them.
Indoor environment presents large number of points which is great for scan matching, so the maps are generally correct but as we reduce the number of the points available for scan matching the quality keeps on reducing. In an out door environment we might need to use some other SLAM algorithm entirely or may have to sack scan matching. In the coming time we plan to do the same.

VIII. References

[1] Monocular Camera Calibration, http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration, 2010. [Online; accessed 2017-04-20].

[2] S. Beucher, F. Meyer, The morphological approach to segmentation: the watershed transformation, OPTICAL ENGINEERING-NEW YORK-MARCEL DEKKER INCORPORATED- 34 (1992) 433–433.

[3] OpenCV Chessboard vision tools, http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html, 2011. [Online; accessed 2017-04-27].

[4] G. Pandey, Probabilistic mobile robotics, Lecture Slides, EE698A, 2017.

[5] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8 (1986) 679–698.

[6] W. Meeussen, Robot pose ekf, https://github.com/ros-planning/navigation/tree/kinetic-devel/robot_pose_ekf, 2009.

[7] W. B. C. Stachniss, Giorgio Grisetti, Improved techniques for grid mapping with rao-blackwellized particle filters, IEEE Transactions on Robotics 23 (2007) 34–46.

[8] S. M. Lavalle, Rapidly-Exploring Random Trees: A New Tool for Path Planning, Technical Report, 1998.

[9] ROS-Perception, Slam gmapping ros node, https://github.com/ros-perception/slam_gmapping, 2009.

[10] W. Meeussen, Coordinate rep 105: Frames for mobile platforms, http://www.ros.org/reps/rep-0105.html, 2010.