

# Autonomous Navigation and Obstacle Avoidance.

EE 698G

Group 2 & 3

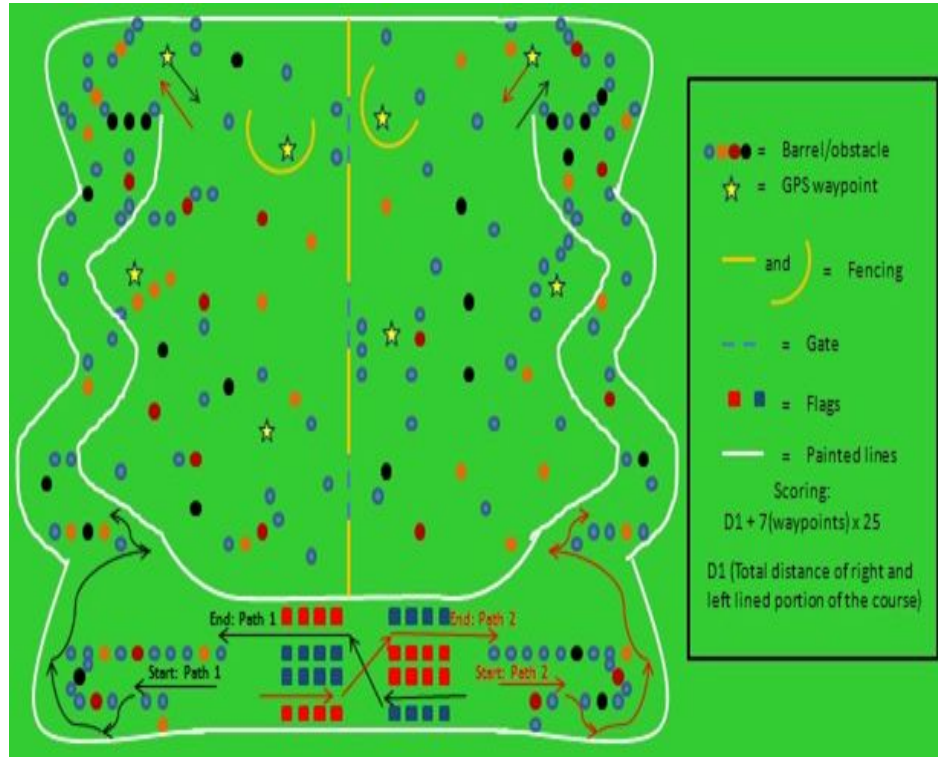
Animesh Shastry  
Deepak Gangwar  
Harsh Sinha  
Swati Gupta  
Shubh Gupta  
Shubham Jain

# The Problem statement



Autonomous navigation  
and obstacle avoidance.

# Objectives



To automate a semi rugged robot to navigate around an outdoor obstacle course similar to the one shown. Thus, the tasks involved are:

- Localization with respect to the lanes as well as the obstacles.
- Get the vehicle motion model and corresponding odometry using Wheel encoders as well as GPS.
- Obstacle avoidance.
- Optimal path finding.

# How is this relevant ?

The technologies involved will essentially result in building of a high-end autonomous robot capable of navigating in outdoor terrains. With a sophisticated lane detection system in place, a whole line of autonomous bots guided by lanes can also be brainstormed upon. Some applications involve:

- Intelligent Transport Systems
  - Self-Driving, Street Legal vehicles
  - Automated Highway Systems
  - Automated Taxi Service
  - Automated Delivery Systems
  - Autonomous Parking
  - Unmanned Maintenance Vehicles
- ❖ Disaster Response
- ❖ Environment Mapping
- Military Mobility
  - Mine Detection
  - Leader - Follower (Autonomous Convoy Operations)
  - Surveillance Systems
  - Manned-unmanned Teaming
- ❖ Manned-unmanned Teaming
- ❖ Navigating in risky environments hazardous to humans.

# Our approach

We divided our work into broadly three divisions, namely

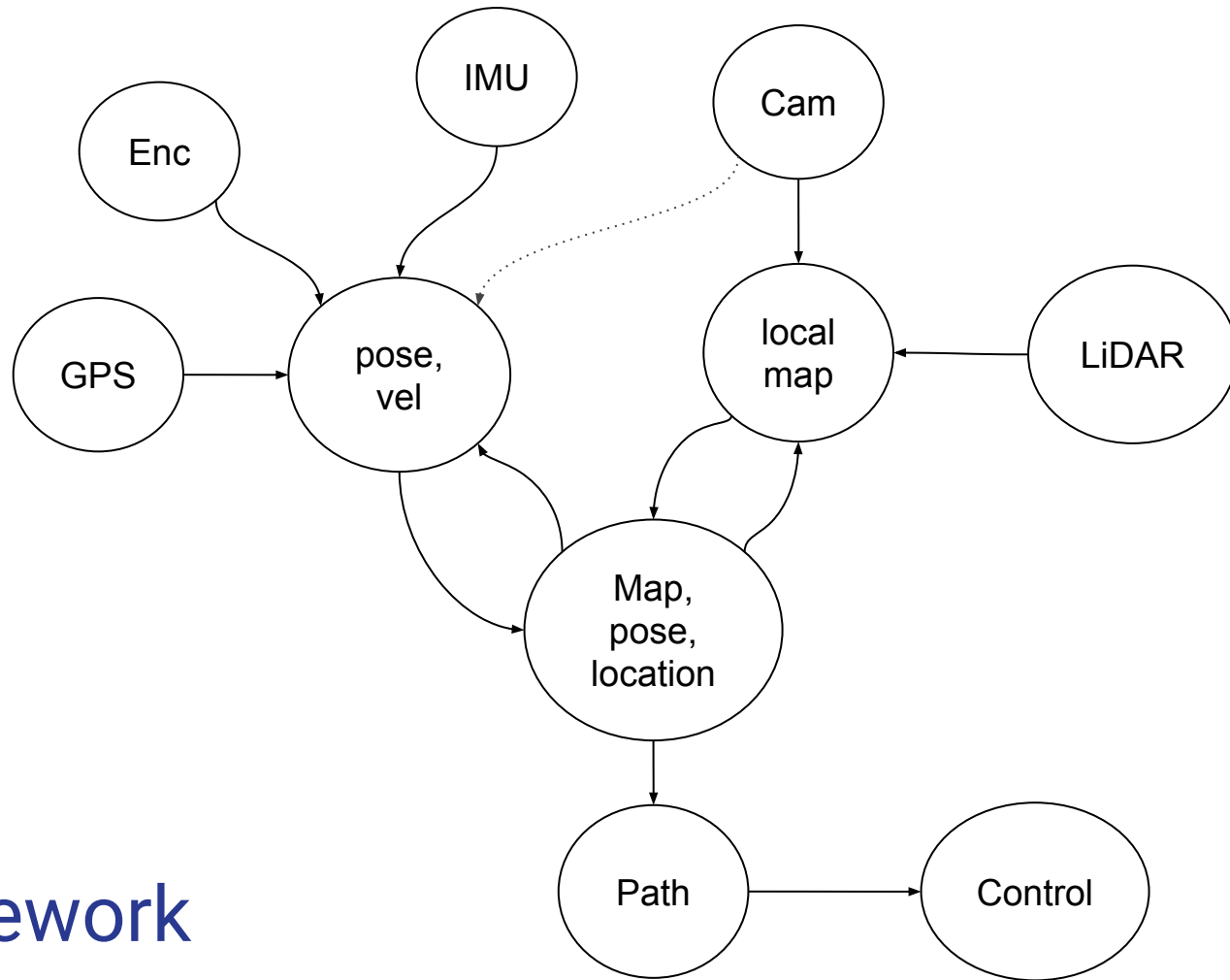
1. Vision
2. Localization and Mapping
3. Motion Planning



# Getting Started

- We have used the ROS (Robot Operating System) framework for our implementation along with C++, python and Matlab.
- From here on we shall use the word 'node' which can be taken similar to a thread launched to do specific tasks.
- The following two slides show us the framework and the work distribution.
- We have showcased how we approached various tasks, and all the results obtained as well as the problems faced if they arose.





# Framework

Vision	Localization and Mapping	Motion Planning
<ul style="list-style-type: none"> <li>→ Obstacle Detection and elimination.</li> <li>→ Lane Detection</li> <li>→ Visual Mapping sans scan matching</li> <li>→ Visual odometry</li> <li>→ Intrinsic and Extrinsic camera calibration.</li> <li>→ Laser Camera calibration</li> </ul>	<ul style="list-style-type: none"> <li>→ Frame transformations between all the sensors and maps.</li> <li>→ Mapping sans scan matching.</li> <li>→ EKF for fusion of sensor data from IMU, GPS, Wheel Encoders <i>etc.</i></li> <li>→ Grid Mapping using the Rao Blackwellized particle filter.</li> </ul>	<ul style="list-style-type: none"> <li>→ A*</li> <li>→ D* lite</li> <li>→ Potential Field (Gradient Based)</li> <li>→ RRT</li> <li>→ Customized RRT</li> </ul>





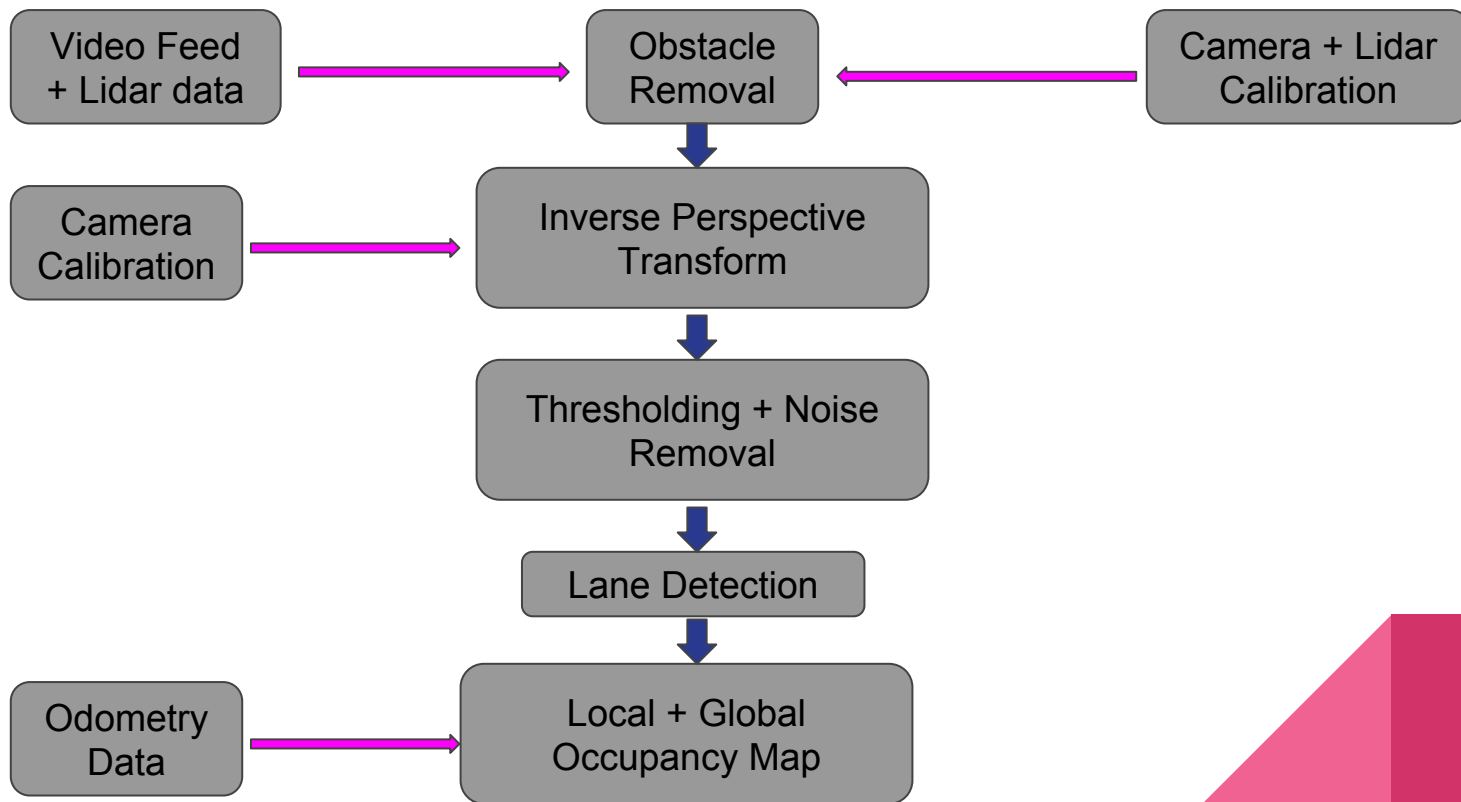
# VISION

# Aim

- The main objective of Vision is to provide the lane information at all times, in a form that can be utilized easily by the motion planning algorithm to ensure that the robot always stays within the detected lanes.
- For this, we need to design an algorithm that detects lanes efficiently.
- Since obstacles constitute noise in the image feed, we need to remove them using fused data from a camera and 2D Lidar, before processing the feed for lanes.
- Finally, a global lane map needs to be updated which is further used by the motion planning algorithm.



# Algorithm



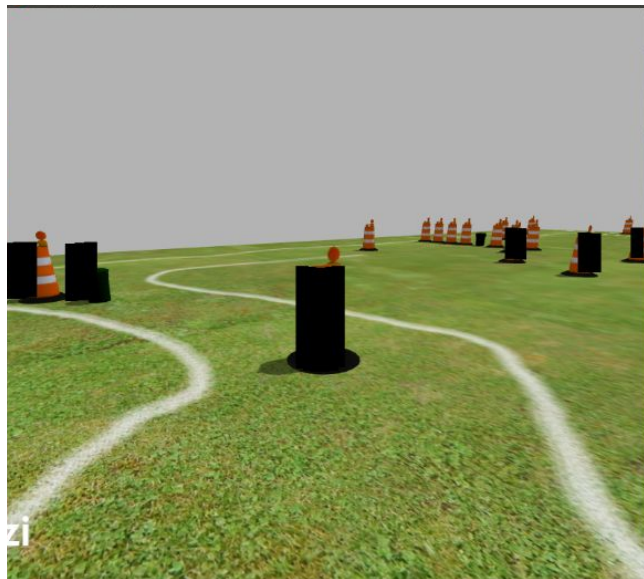
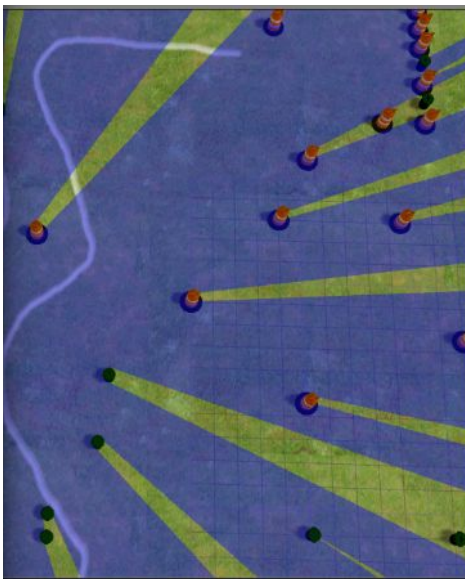
# Obstacle Removal

- Transform from Lidar to Camera
  - Rigid body transform from Camera to world frame and Lidar to world frame obtained by physically measuring translation and roll, pitch, yaw of sensor mounted
  - Using these extrinsic matrices, a transformation matrix from Lidar to camera frame is obtained

$$\begin{array}{c} \text{Extrinsic Matrix} \\ \left( \begin{array}{c|c} I & \mathbf{t} \end{array} \right) \times \left( \begin{array}{c|c} R & 0 \\ \hline 0 & 1 \end{array} \right) \\ \underbrace{\hspace{10em}}_{\text{3D Translation}} \quad \underbrace{\hspace{10em}}_{\text{3D Rotation}} \end{array}$$

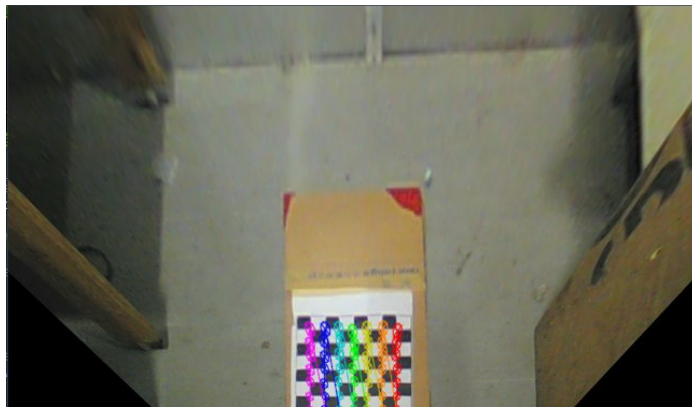
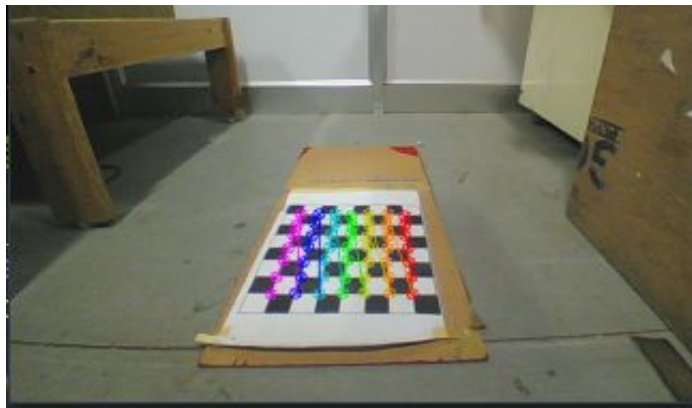
# Obstacle Removal

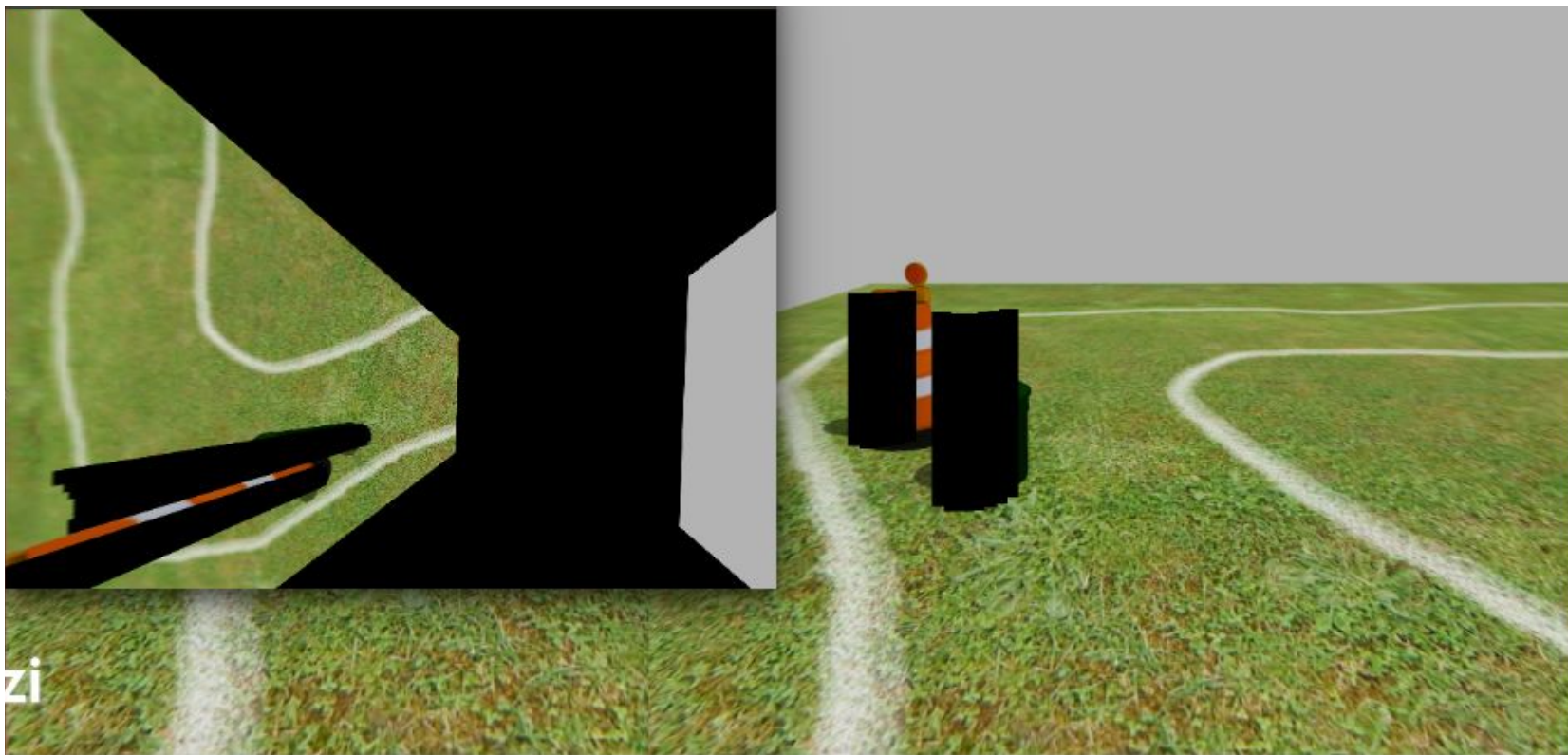
- Using the Lidar to Camera transformation matrix, we project the Lidar point cloud on the image.
- Using Watershed Algorithm, expand over the obstacle points to get the entire obstacle in the image.



# Inverse Perspective Transform

- Camera calibration to obtain bird's-eye view
  - Place a chessboard of known dimensions in front of the camera (fixed orientation)
  - Detect chessboard corners using harris corners detection.
  - Determine the extreme corners, and find positions of these corners in topview using the actual dimensions
  - Compute the homography matrix between the two views.
  - This matrix can now be used to transform the video feed into top view by multiplying with the original pixels.





# Lane Detection

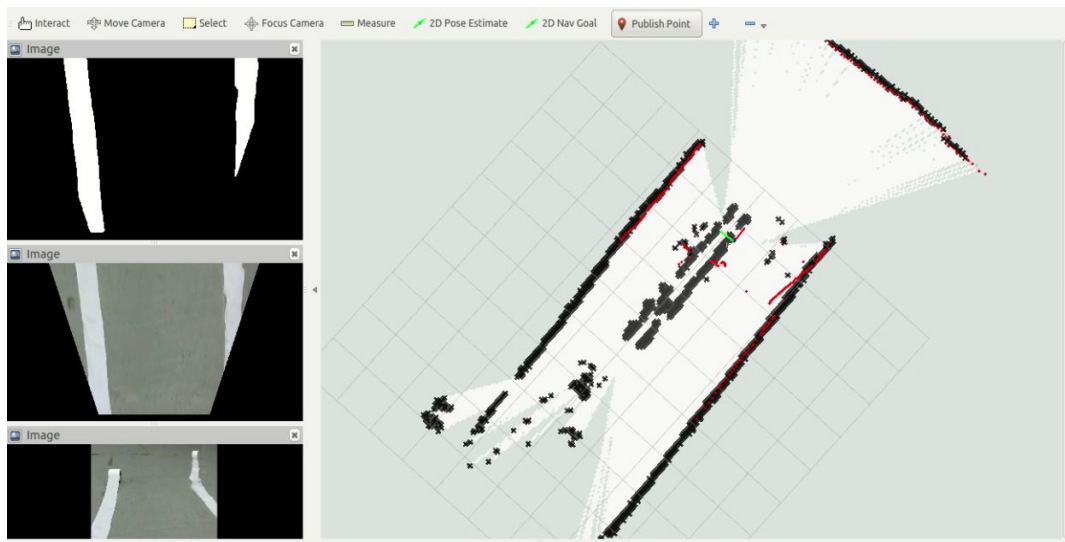
- Thresholding based on RGB values of first frame to keep only lane points
- Noisy points are removed to some extent using erode and dilate (morphological operations)
- Thresholded image is split into two halves for the 1<sup>st</sup> frame and PCA applied individually on both, to determine the best fit lines corresponding to the lanes
- Region centred around detected lanes is used as predicted area for the lane to exist in the next frame
- Next iteration of the algorithm uses this region and applies PCA to determine the new lane positions and the process repeats

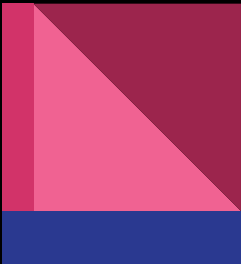




# Lane Map

- A local occupancy map is created using the lane data obtained
- This map is transformed to world reference frame using odometry data and used to update the global occupancy map





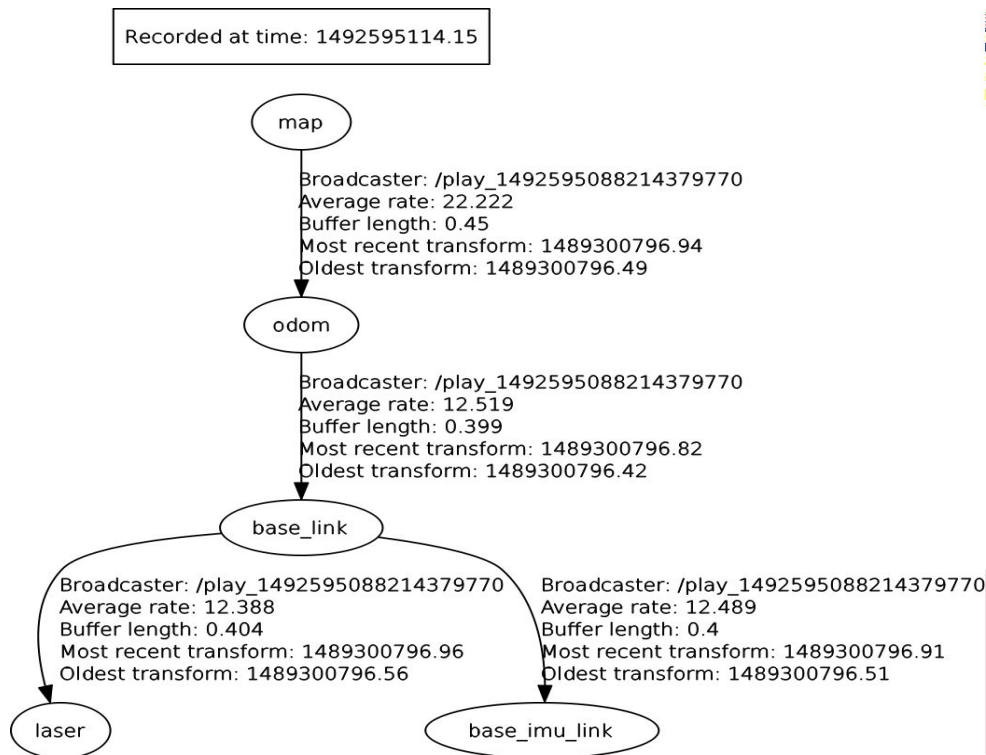


# LOCALIZATION AND MAPPING

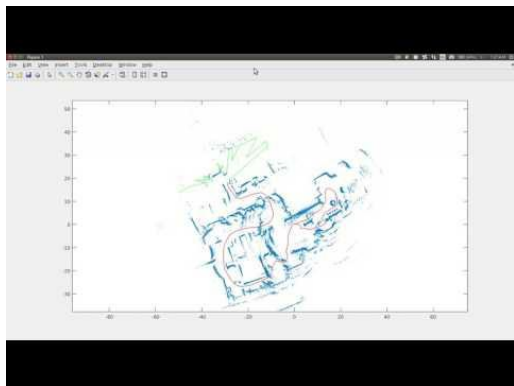
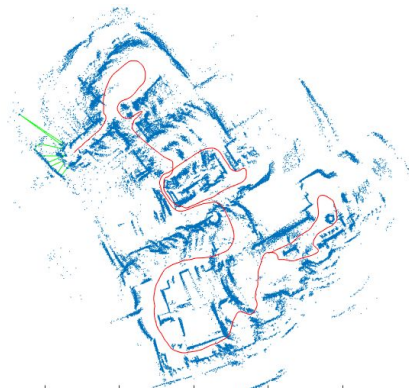
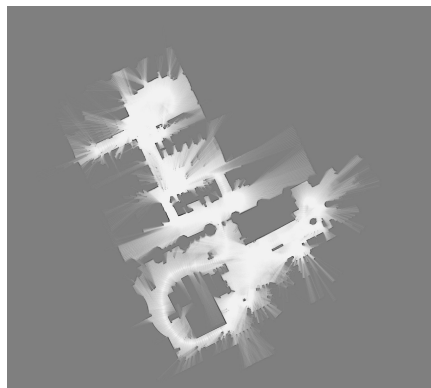
# Frame transformations

We have several frames between which we dynamically calculate and maintain transformations with the help of **tf** feature in ROS. The frames are :-

1. Base\_link
2. Map
3. Odom
4. Camera
5. Laser (for LiDAR)
6. IMU + GPS



# Mapping sans scan matching.



- The images show the expected and the generated maps for the Edmonton dataset
- The datasets contain only the encoder values and LiDAR data the generated map tends to distort when the robot takes sharp turns although this is more pronounced in the Malaga dataset.
- We implemented ICP for correcting this error but the computation times for it were very large.

# EKF


- The motion model of the robot used was :
  - ◆ Velocity model, i.e it was assumed that the robot was given linear and angular velocity commands which remained constant for a sampling period.
  - ◆ Wheel encoder odometry was also available, this calculated from encoder ticks as linear velocity (average of left and right wheel speeds) and angular velocity (difference of the aforementioned ones).
- The data from IMU is a set of three vectors  $[a_x, a_y, a_z]$ ,  $[g_x, g_y, g_z]$  and  $[m_x, m_y, m_z]$ . On this data then William Premerlani 's DCM algorithm is used to get the orientation in quaternions. Thus the measurement model for this in the EKF is [yaw, pitch, roll].
- The model for pose is  $[x, y, z, \phi, \theta, \psi]$ .
- On these parameters a standard EKF has been implemented.

# Grid mapping

- For the grid mapping using rao blackwellized particle filter we employed gmapping a tool readily available in ROS.
- Gmapping is probably the most used SLAM algorithm currently and is a standard on PR2.
- Gmapping requires tf between laser, odom and base frames. Observations :  $z_{1:t} = \{z_1, z_2, \dots z_t\}$  and Odometry :  $U_{2:t} = \{U_2, U_3, \dots U_t\}$ .
- It returns a posterior :  $p(x_{1:t}, m \mid z_{1:t}, U_{2:t})$ .



# Gmapping (Key ideas)

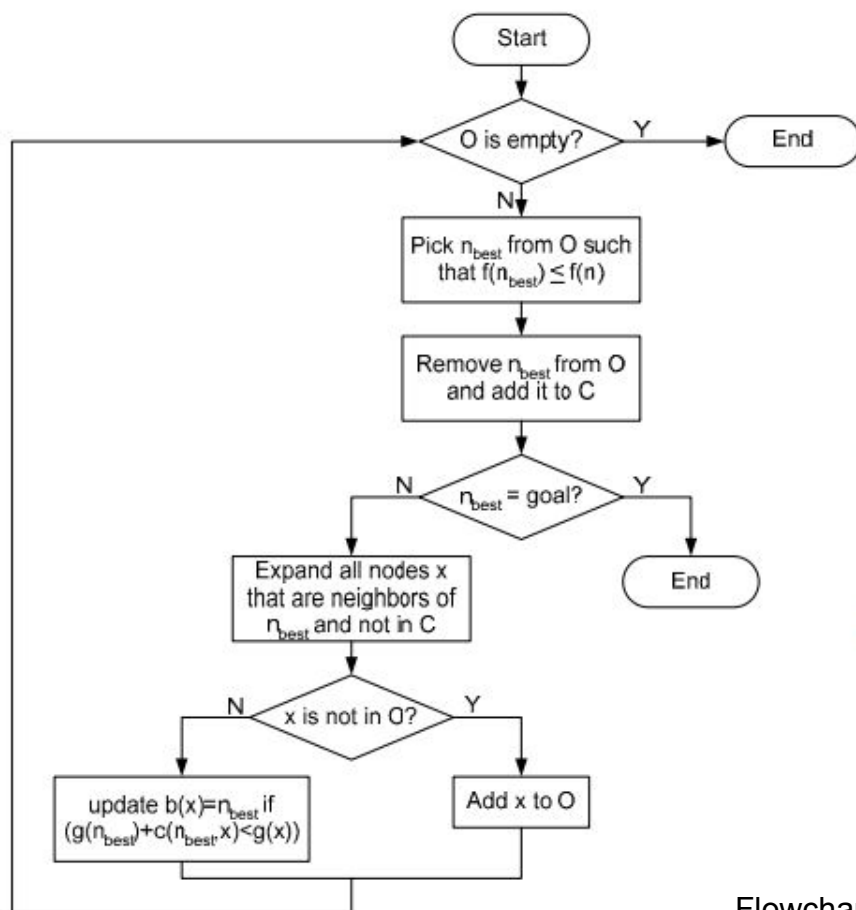
- Rao Blackwellized Particle Filter :
    - Each Particle carries with it a sample history of the robot poses, posterior over the maps given the sample pose history and the most likely map.
  - Proposal Distribution :
    - Sample from an approximated optimal proposal distribution.
    - Weight update and resampling based on effective sample size.
  - Scan Matching
    - Standard ICP for scan matching based on the beam endpoint model for the laser range finder is used.
  - For more details, see paper: “Improved Techniques for Grid Mapping with Rao Blackwellized Particle Filters” by Giorgio Grisetti, Cyrill Stachniss, Wolfram Burgard, IEEE Transactions in Robotics, 2006
- 





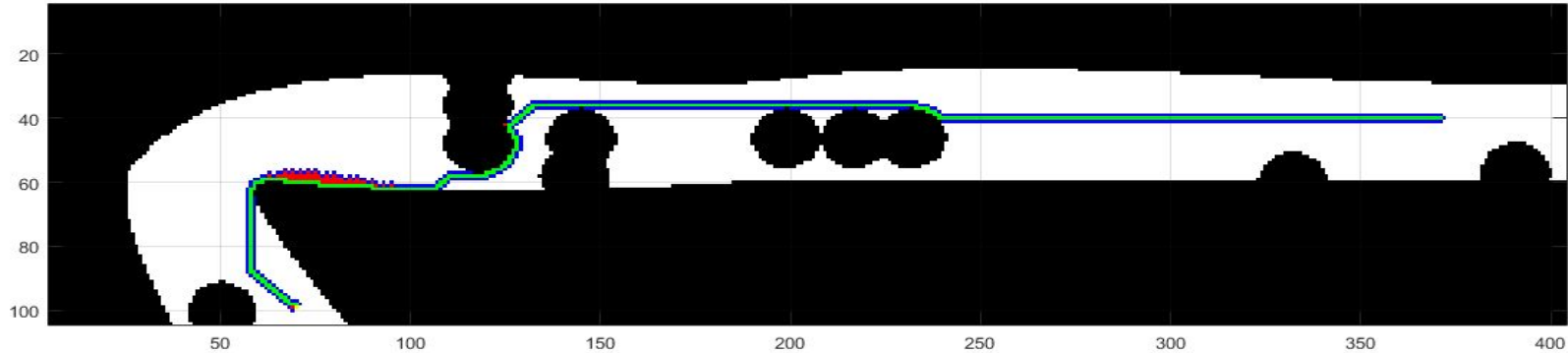
# MOTION PLANNING

A\*



Flowchart credit : Howie Choset

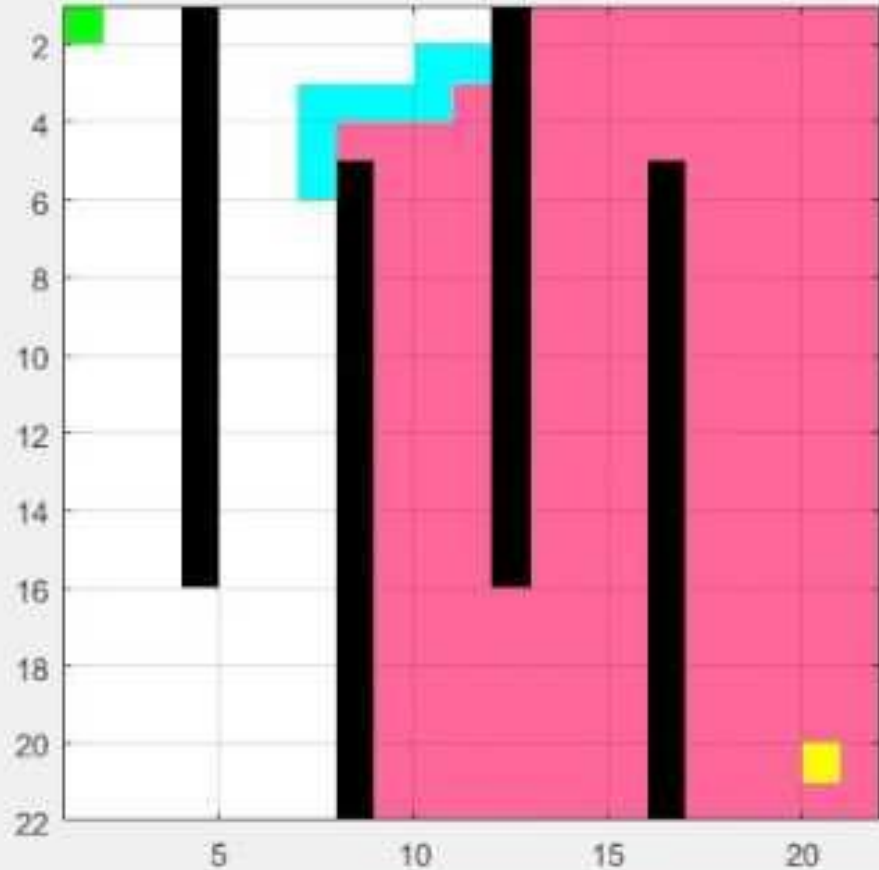
A\*



- Gives shortest path but not feasible in real world scenario because the paths are too close to the obstacles and the mandatory requirement that the map must be static.
- Holonomic constraint.

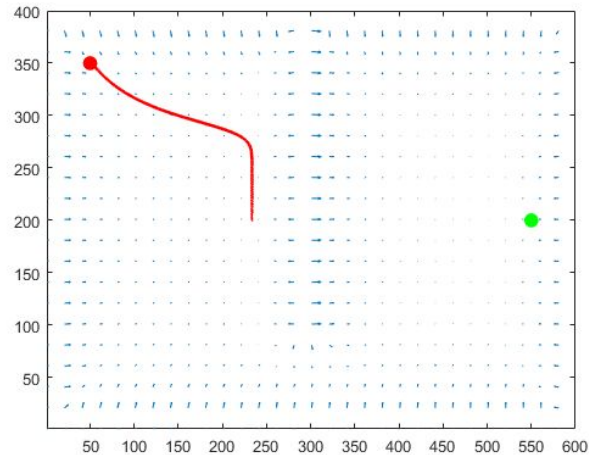
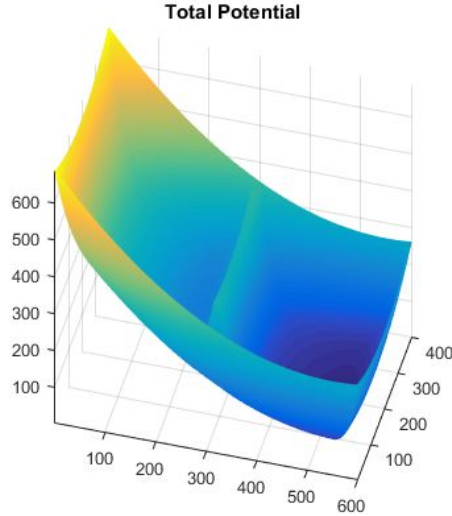
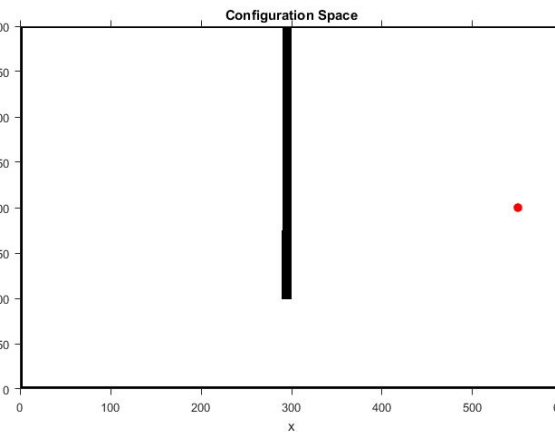
# D\* Lite

- Algorithm is similar to A\*.
- The advantage is that it can dynamically adjust its path if the map changes.
- Computationally less expensive than A\* in dynamic environment.
- Holonomic constraint.

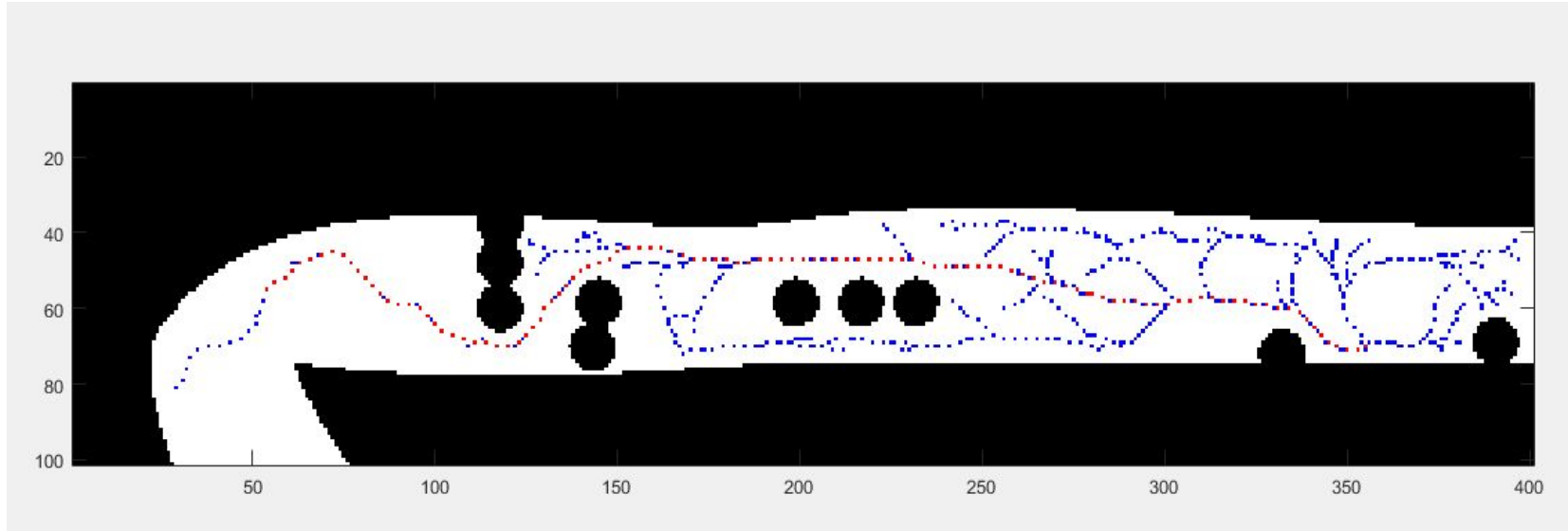


# Potential Field (Gradient Based Planner)

- Algorithm includes creating repulsing and attracting potential and then guiding the robot through the potential terrain.
- The advantage is that the paths are very smooth and the obstacle avoidance is optimal
- Computationally similar to A\*
- No Holonomic constraint.
- The major disadvantage is local minima in potential field in which the robot can easily get stuck



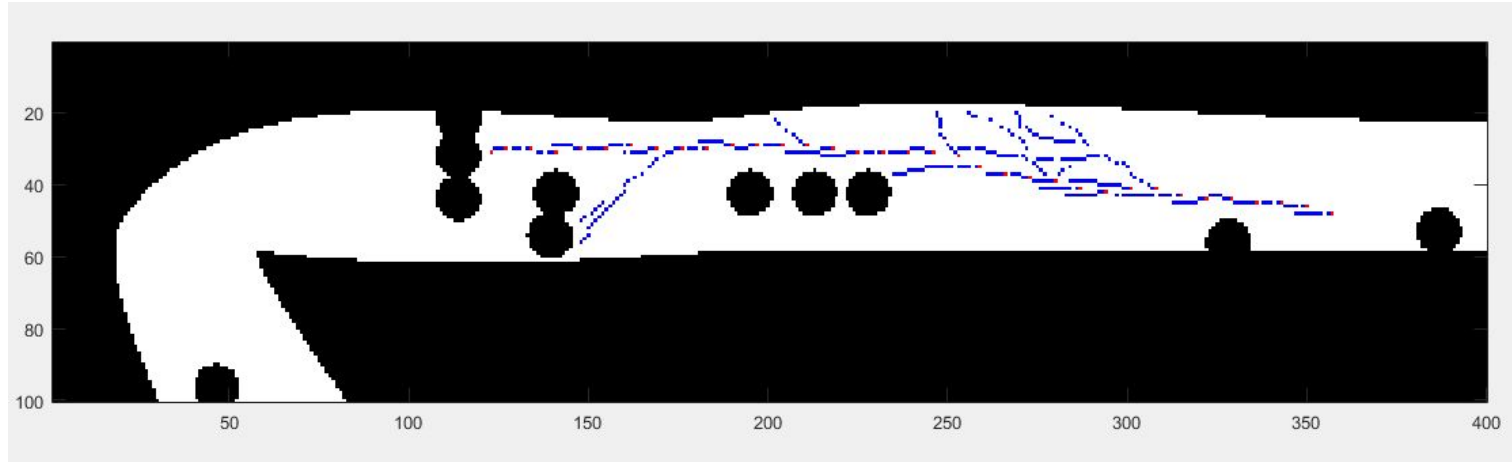
# RRT



- Computationally much less expensive than A\* or D\*
- Better feasible paths in real world scenario.
- No holonomic constraint.

Credit : Aalap Shah

# Customized RRT

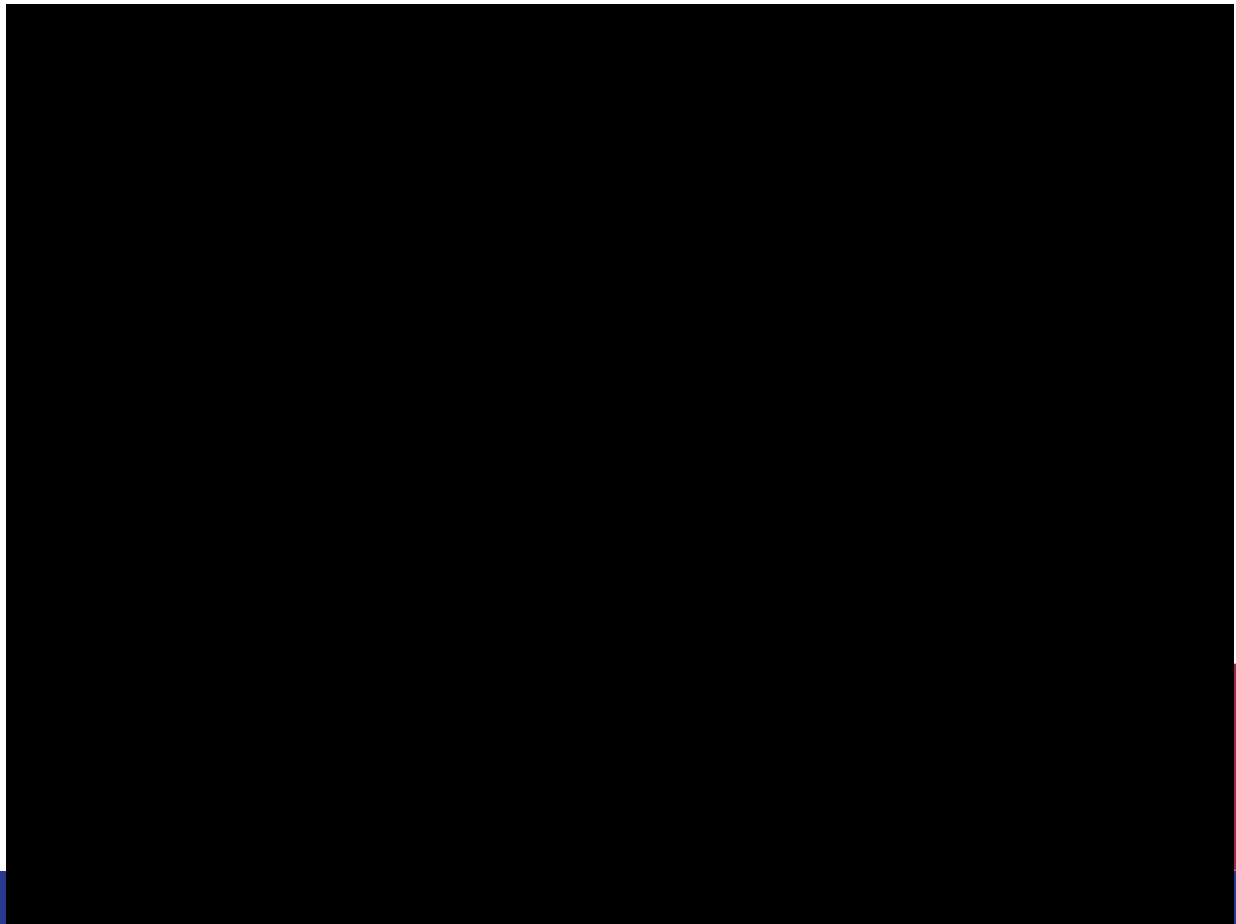


- Placed a constraint on the node generation to adjust to the turn angle limits of the robot.
- Adaptively increasing the turn angle limits if straight paths are not found.
- Using lanes to calculate a heading that is then used to generate a temporary goal.

Credit : Aalap Shah

# The result : Waypoint navigation

Using all the features other than vision we were able to do waypoint navigation, where the waypoints were odometry based.





# Future

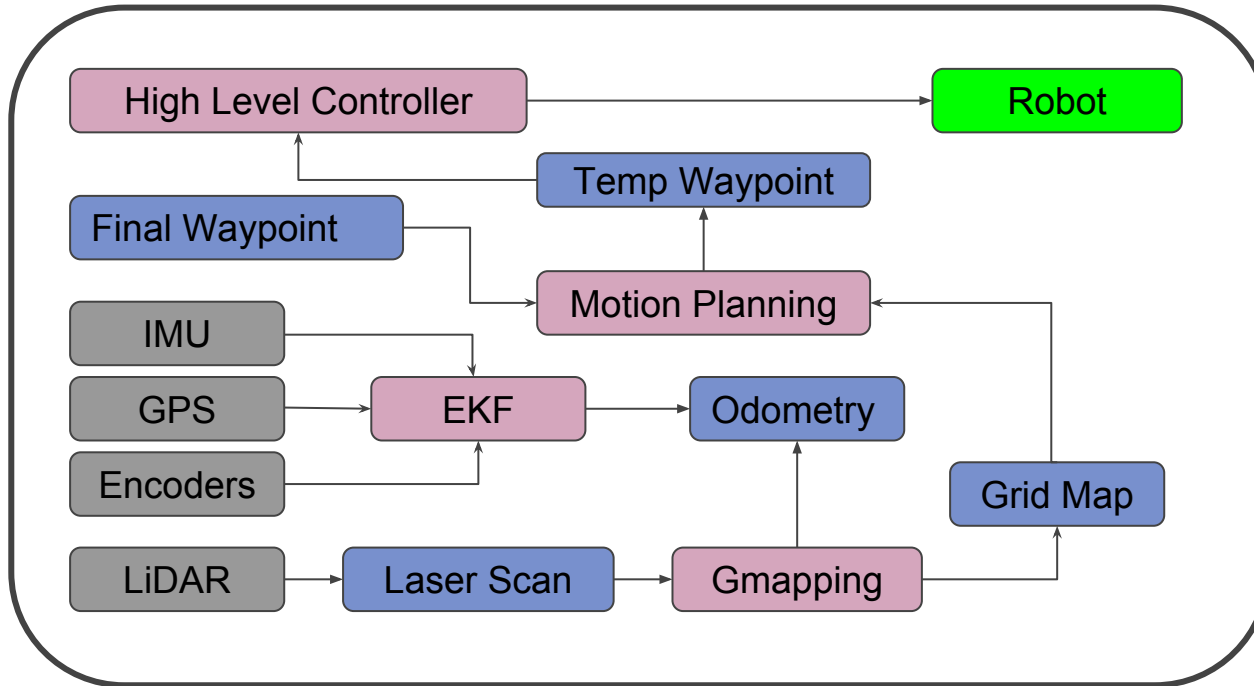
- We were able to achieve indoor localization to a great extent, we are currently trying to get our system working outdoors, in environments much more sparse.
- The work we did here is part of our efforts towards the Intelligent Ground Vehicle Competition, an annual competition for autonomous robots.



# References

1. G. Grisetti, C. Stachniss and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," in *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34-46, Feb. 2007.
2. Robot-pose-ekf: <https://github.com/ros-planning/navigation>
3. Slam-Gmapping: <https://www.openslam.org/gmapping.html>
4. Camera-calibration: [http://wiki.ros.org/camera\\_calibration](http://wiki.ros.org/camera_calibration)







Any Queries ?



Thank You