IGVC IITK

Date: Jan 14, 2017

Progress

Last evaluation

Discussion of problem statement and team structure

Current Status

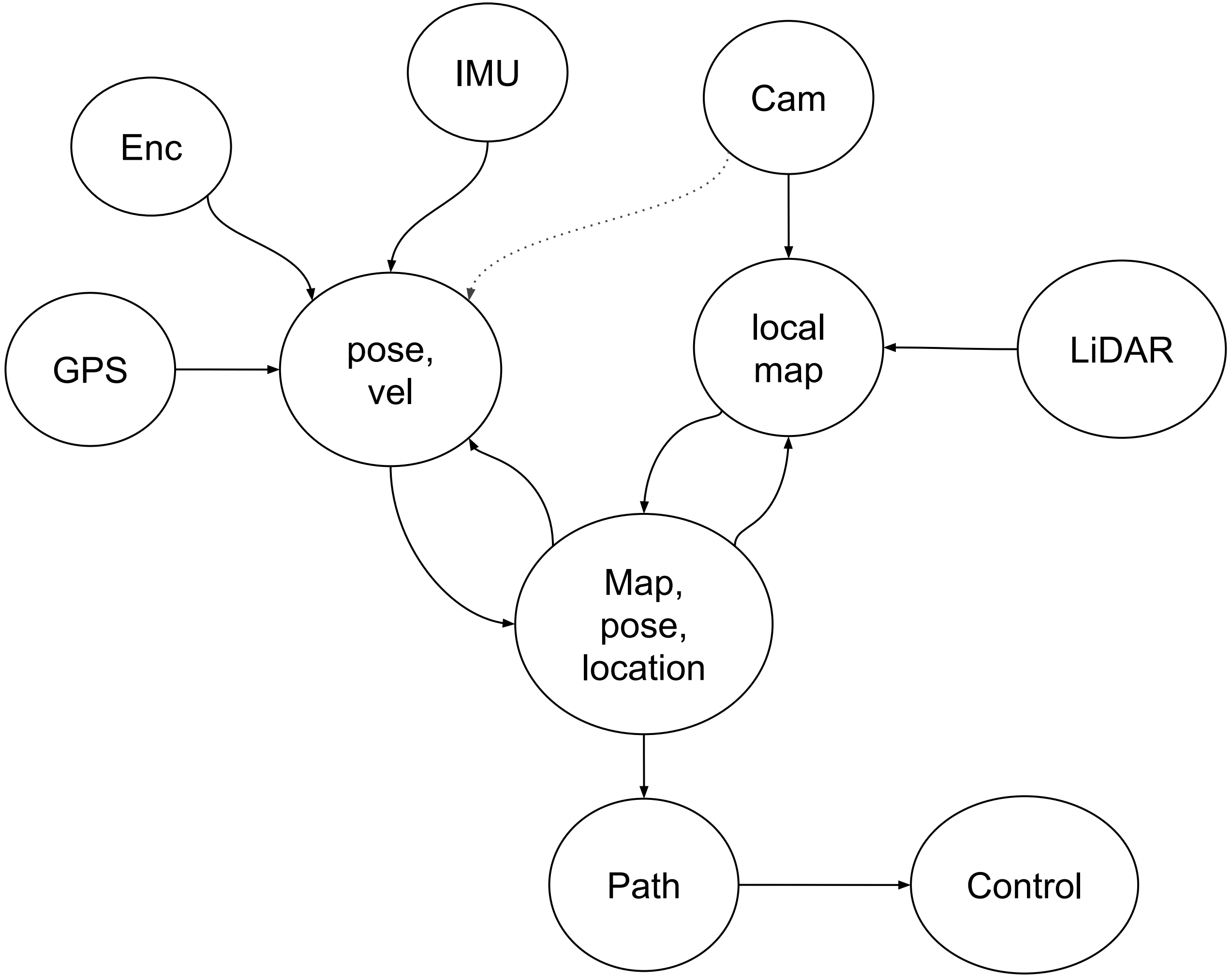
Lecture series

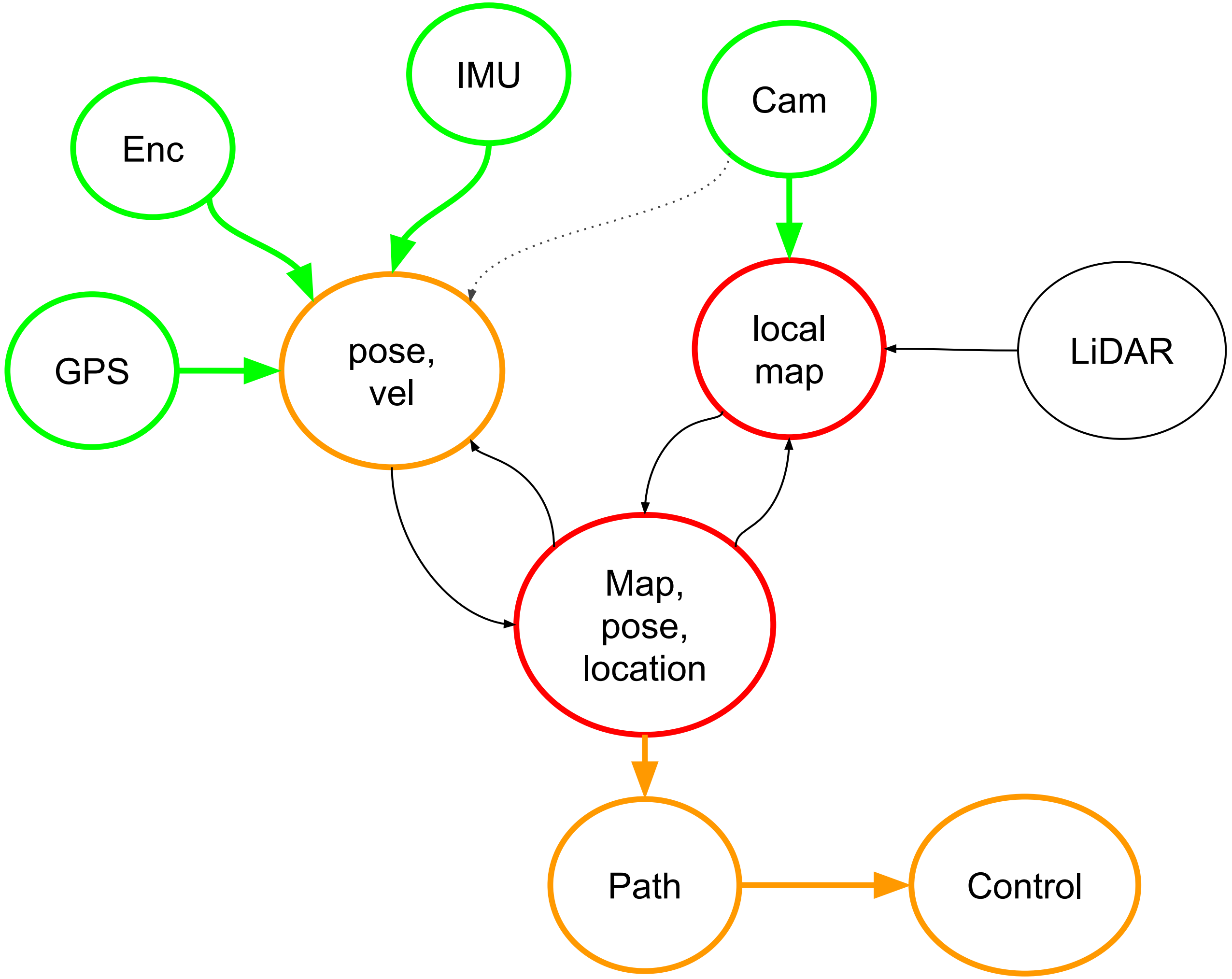
Modular work:

- Image Processing
- Odometry
- Control
- Motion planning

Future plans

- Combine odometry and control for firebird robot
- Generate map of environment using RPLidar


IMU
Cam
Enc
GPS
pose, vel
local map
LiDAR
Map, pose, location
Path
Control


IMU
Cam
Enc
GPS
pose, vel
local map
LiDAR
Map, pose, location
Path
Control

Lane Detection and Obstacle Elimination

IGVC Problem Statement poses a requirement for the robot to stay within two white lane lines, drawn over grass using chalk powder. Hence, there is a need to **detect the lines and separate them from the rest of the environment**.

Algorithm

- HSV over RGB space.
- Lower Gaussian pyramid constructed.
- Erosion using a cross shaped kernel of 3x3 pixels twice.
- Sobel filter applied in x direction.
- Inverse Perspective transformation applied.

Shortcomings

- Edges of obstacles remained
- Not invariant to lighting changes

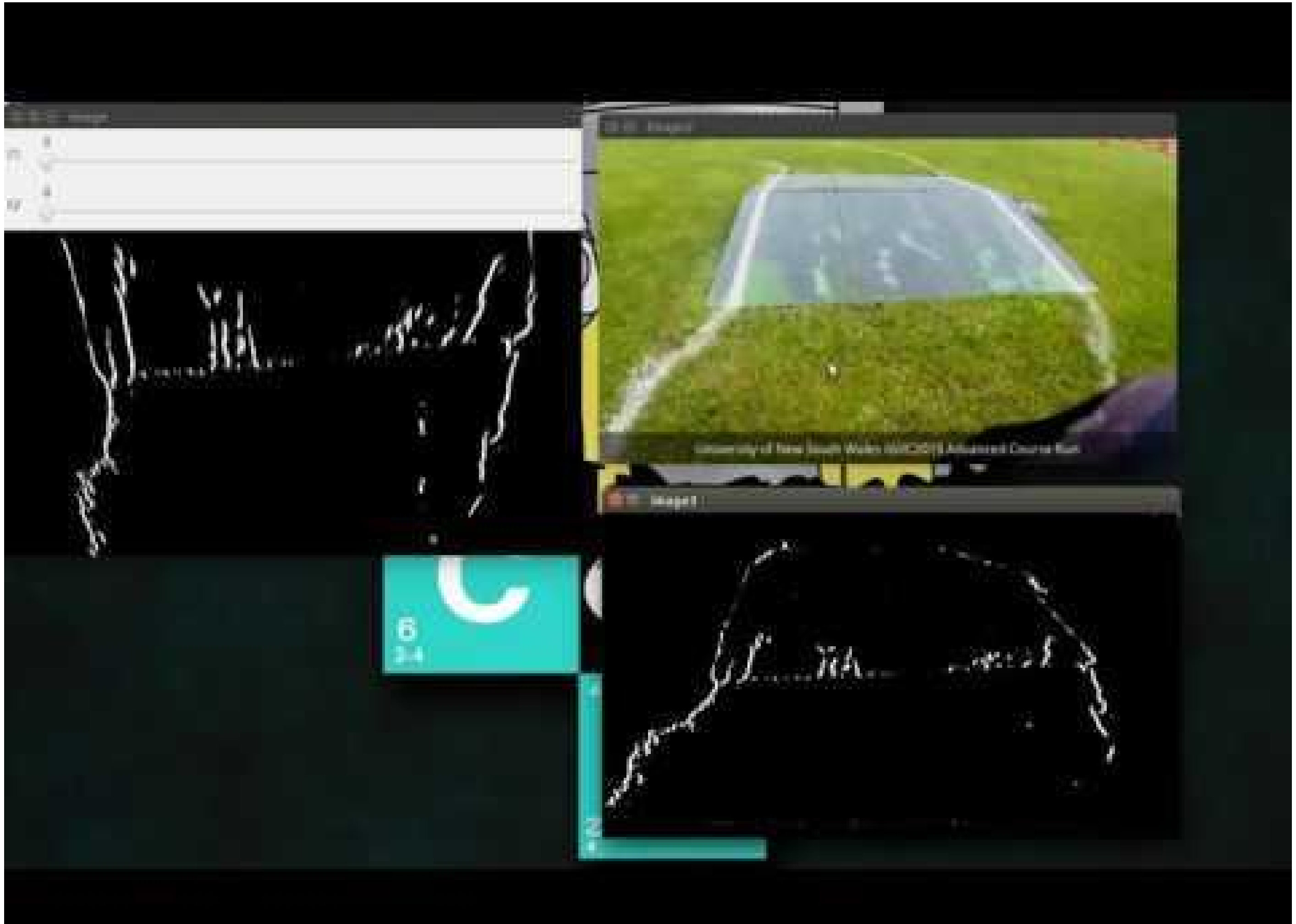


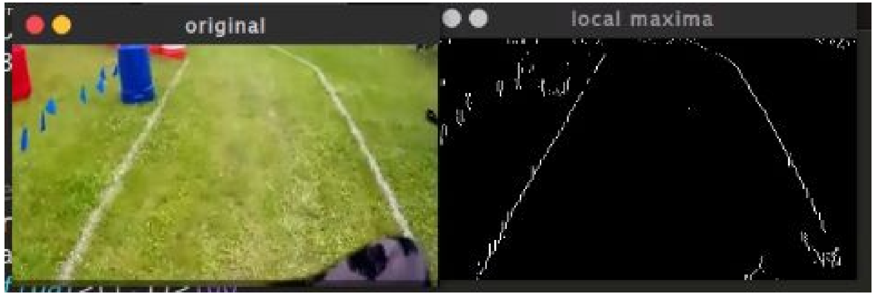
image
image2
University of New South Wales IGVC2015 Advanced Course Run
image1
6

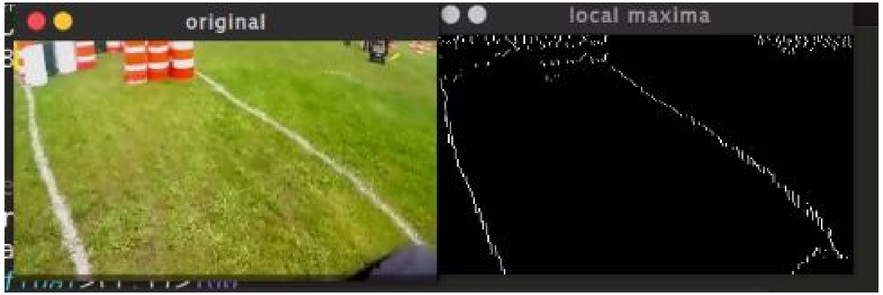
Kernel Approach

[*-1-1* ***1 1 1 1*** *-1 -1*] for lane of width four px.

Linearly varying lane width, from top to bottom

Select local maxima

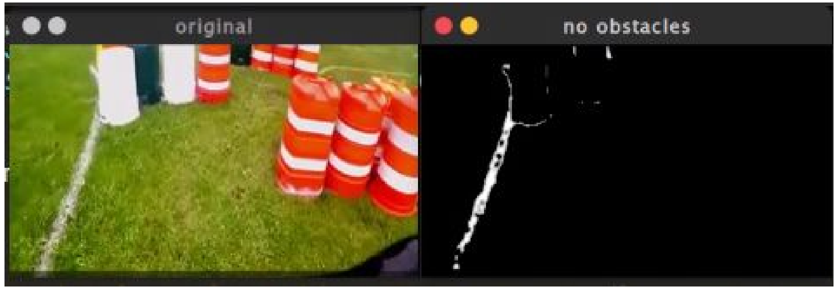


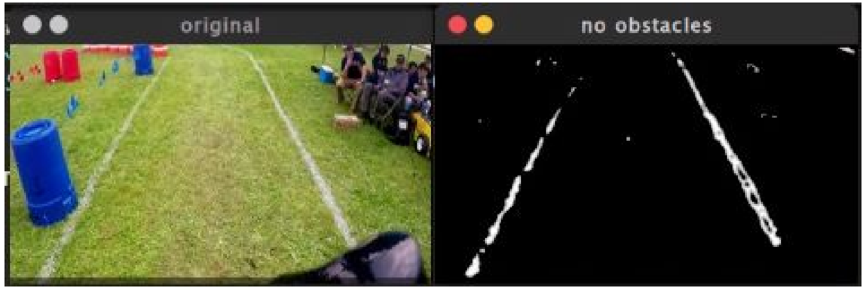


RGB Thresholding

Using **2 x Blue - Green** values to separate lanes

Used by previous teams





Spline

Cubic splines used to join points of interest

Inputs

Slope	*p1 p2*
Coordinates	*p1 p2*

Output

$$p(t) = (2t^3 - 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)hm_0 + (-2t^3 + 3t^2)p_1 + (t^3 - t^2)hm_1$$

Source: http://rvsn.csail.mit.edu/Pubs/phd_ashuang_2010feb_laneestimation.pdf

Spline

Select sample points from either half

Connect curves to a region 50px away

Choose the best curve (using *distance transform*)

Remove interest points traversed

Repeat

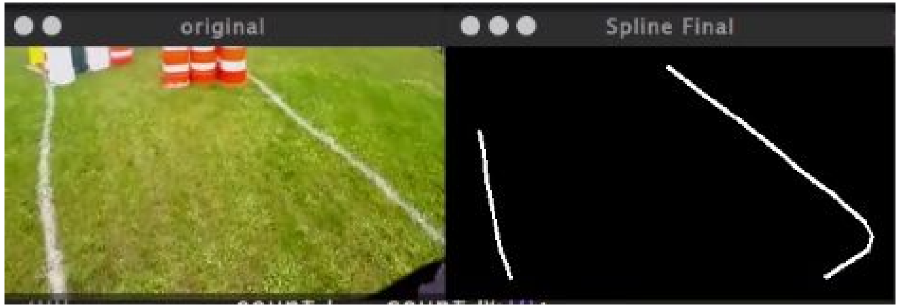
Spline++

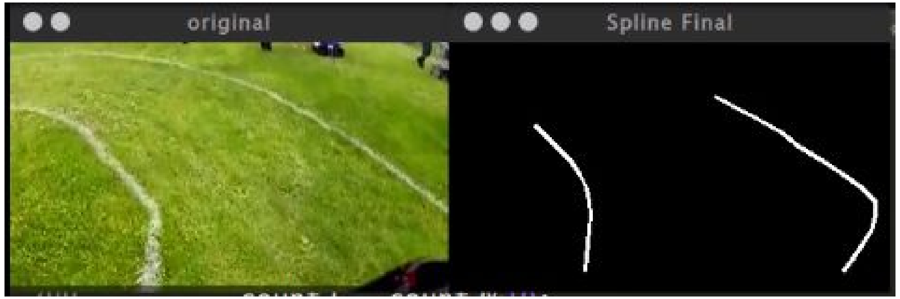
Already existing splines are *promoted*

“Leading” lane made first, allowing it to skip halfs

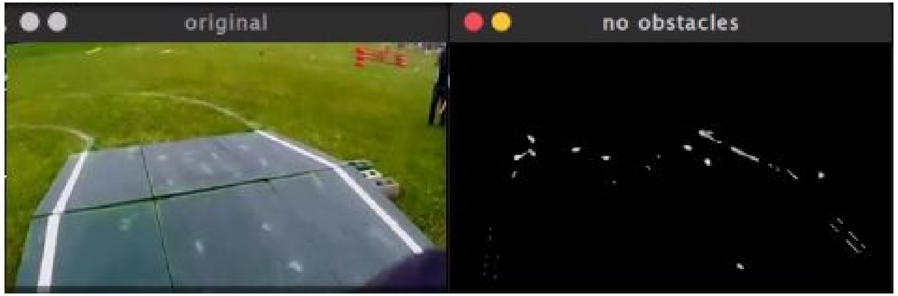
Data is *averaged* over 10 frames

More weight is given to *newer frames*`

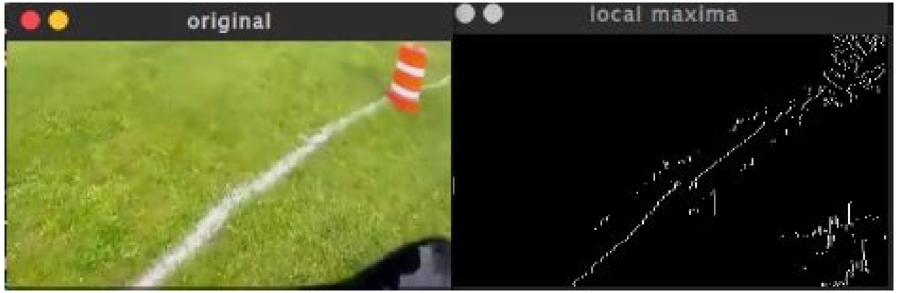




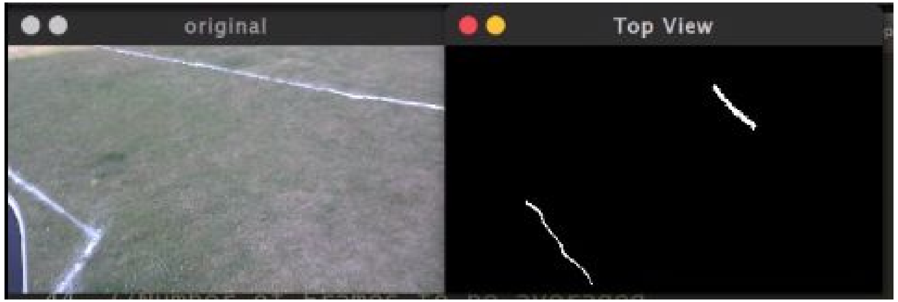
Current Issues



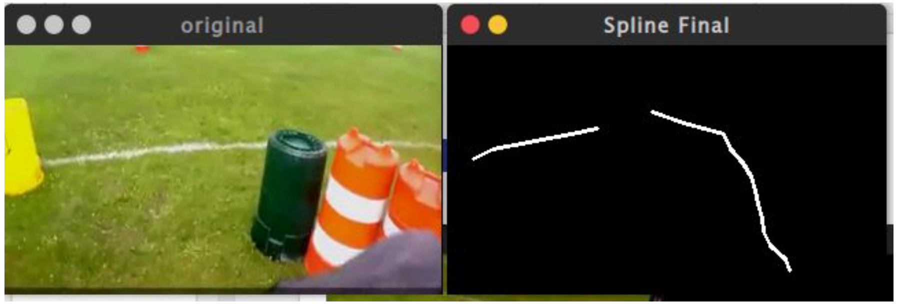
Painted lines identified as barrels



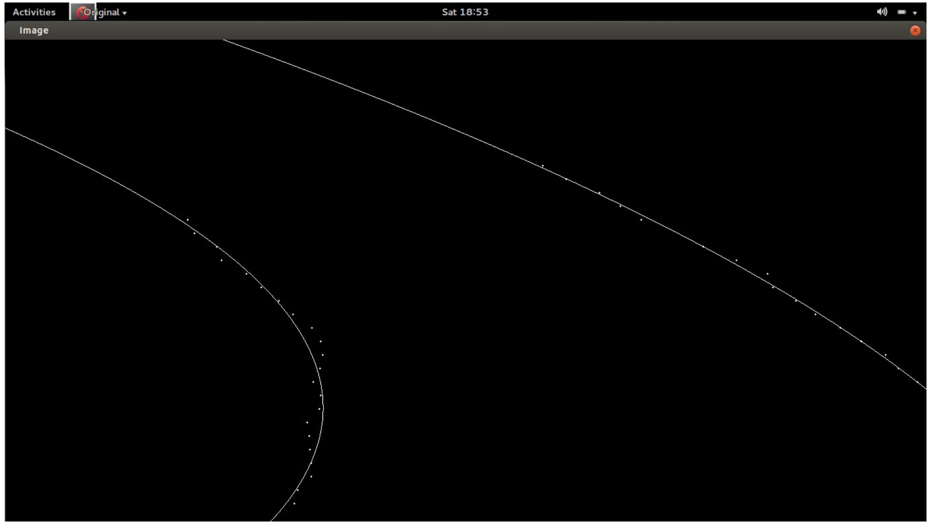
Lane too broad

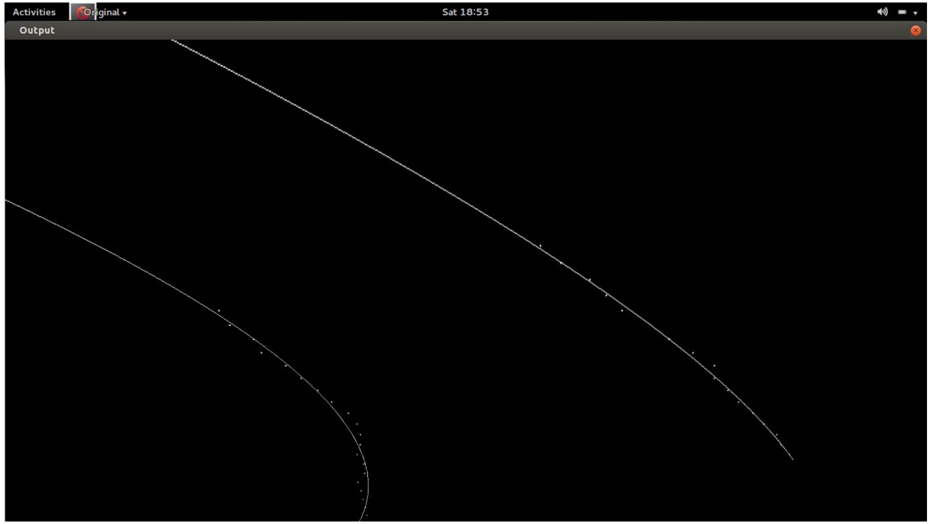


New thresholding for new data



Horizontal lane identified as separate lanes





Given 20 points most probable to be present on the lane, the function draws a 2-degree equation which would be a better approximation of the lane.

The Inverse Perspective Transform function (IPM) transforms the perspective view into the top view, the equation of which has to be sent to the mapping part.

Traversable region detection

After the lane data was obtained, the next problem was to **identify the region between the lanes**.

Algorithm

- Hough lines on detected lanes
- Classification of left and right lanes
- Average lines of clusters
- Remove region on lane outer side

Shortcomings:

- Wrong detection on line slopes close to horizontal.
- Obstacle edges generate error

Frame2

Camera Based Tracking Using Optical Flow

Classification of left and right lane lines was done using midpoint of image, which is a naive approach. A better approach would be to **track the lane lines and use it for lane classification**.

- Detected lane points superimposed on original image.
- Maximas in local neighbourhood designated as tracking features.
- Optical flow calculated.
- New points were generated if point count fell low.

Shortcomings

- Jerk produced errors
- Hard to differentiate between points on lane

Frame
University of New South Wales IGVC2015 Advanced Course Run

Grass Classification

The most recent approach we are working on is to detect grass in the image. Considering runtime speed as a factor, a very simple approach was used to try and **classify grass in the image**.

- Positive dataset of about 20000 50px X 50px images of grass.
- HSV conversion
- Global_mean and global_std._Deviation in H, S and V
- Video frame divided in 50px X 50px windows
- Classified as grass if window_mean is between global_mean +/- 2*global_std._deviation
- Identified grass images used for tuning global_mean and std_deviation

Shortcomings

- Not invariant to abrupt lighting changes

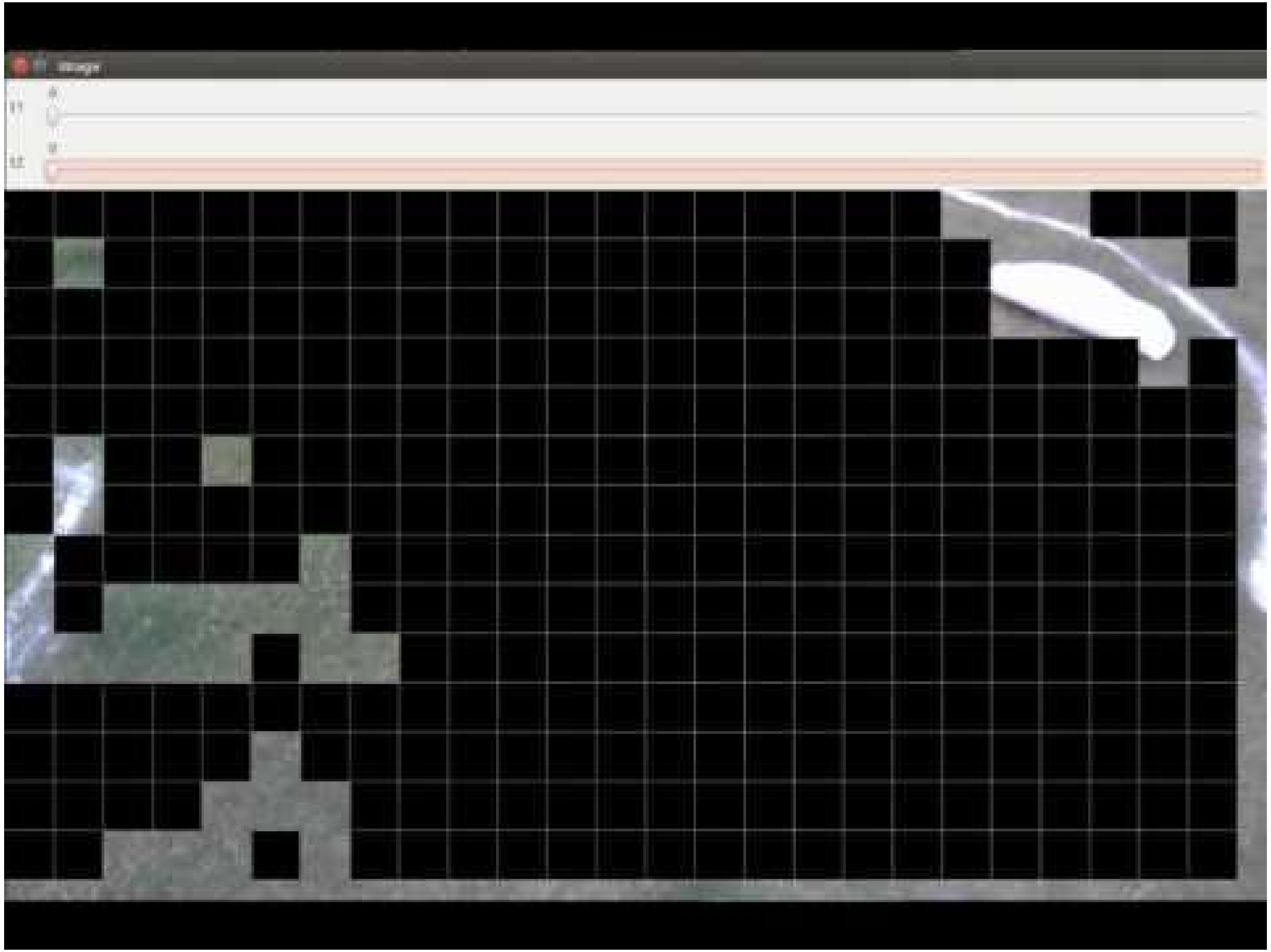


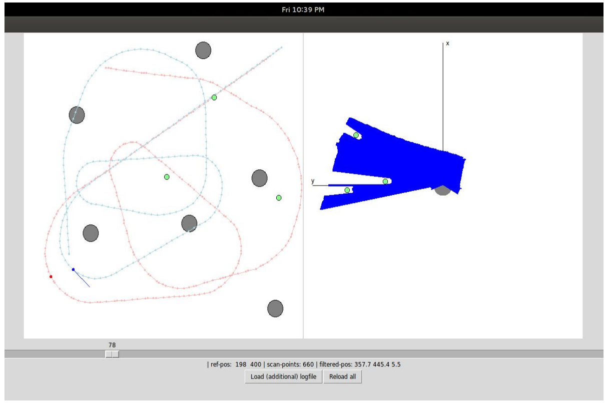
Image
t1
0
t2
0

Simultaneous Localisation and Mapping

Enlisted is an example of SLAM implemented by **Claus Brenner** using Encoders and 2-d lidar on a wheeled robot in an indoor arena . Till now the approach is non-probabilistic.

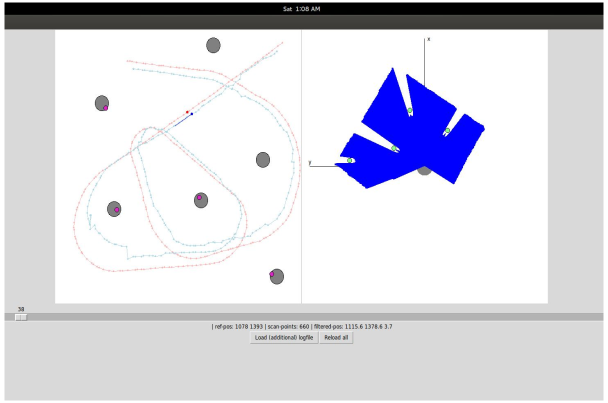
A. **Feature Based Localization :**
   a. Assignment of Landmarks (Spike Landmark extraction)
   b. Direct solution of similarity transform
   c. Correction of pose using transform
B. **Featureless Localization :**
   a. Assign the scan points to the wall
   b. ICP to find the optimal transformation

Odometry based motion model used .
Measurements are in form of scans from 2-D lidar mounted on robot and motor ticks given by encoders coupled with robot’s wheels.

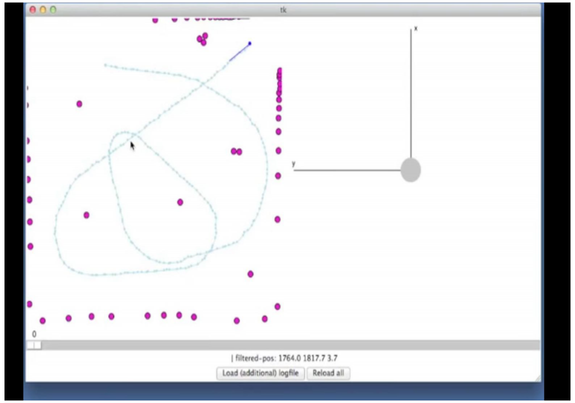


| ref-pos: 198 400 | scan-points: 660 | filtered-pos: 357.7 445.4 5.5

Trajectory using only odometry data.

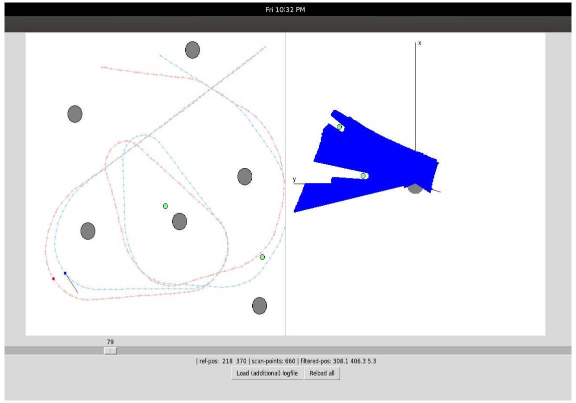


| ref-pos: 1078 1393 | scan-points: 660 | filtered-pos: 1115.6 1378.6 3.7



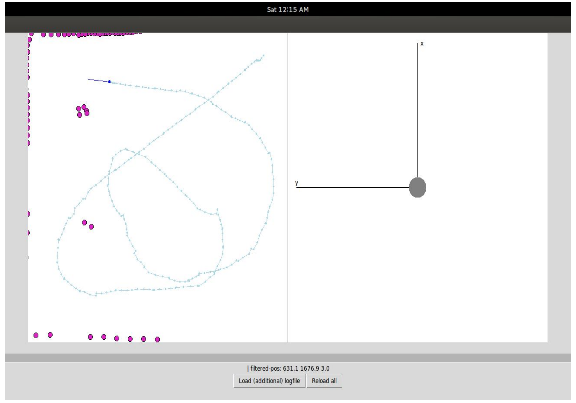
| filtered-pos: 1764.0 1817.7 3.7

Correcting pose using LiDAR data with feature based and featureless landmark extraction.



| ref-pos: 218 370 | scan-points: 660 | filtered-pos: 308.1 406.3 5.3

Odometry data with experimental Calibration

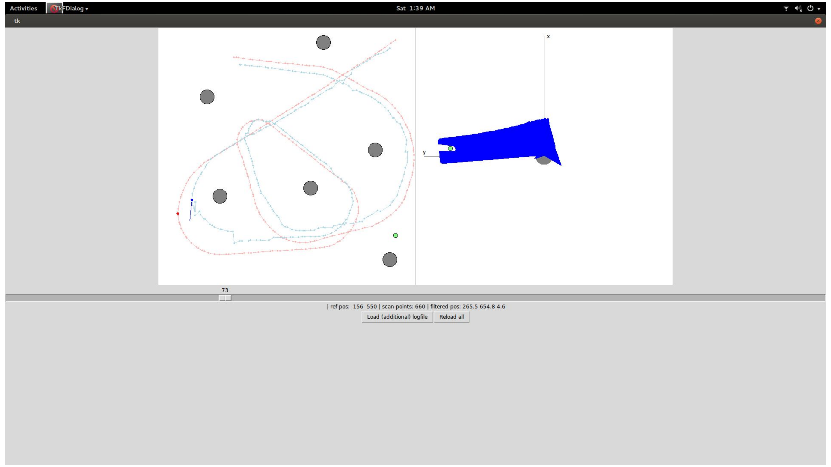


| filtered-pos: 631.1 1676.9 3.0

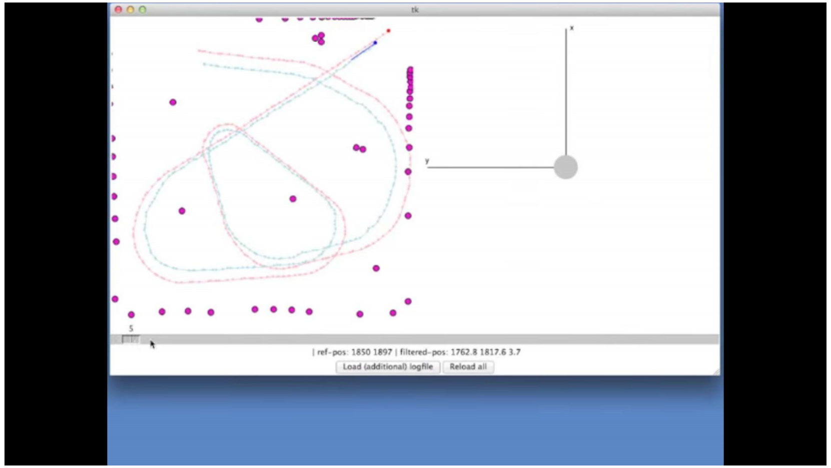
Presently the landmarks for most time are extracted using spike method (using sudden change in lidar distance values) which the robot matches with reference landmarks .

For parts lacking sufficient landmarks, featureless extraction is used with ICP (Iterative Close Point algorithm) to correct the pose.

This smoothens the trajectory further correcting the robot's pose .



| ref-pos: 156 550 | scan-points: 660 | filtered-pos: 265.5 654.8 4.6



| ref-pos: 1850 1897 | filtered-pos: 1762.8 1817.6 3.7

Future Approach

The present algorithm modified to SLAM can be implemented on Firebird robot for indoor tests on real-time testing its performance

Completing the lecture series of Claus Brenner and modifying the code in accordance with our problem statement.

Using already available packages in ROS for SLAM and determining the most efficient among them in our case. Some packages which are already being worked on :

Slam_gmapping
Uzliti_slam
Mrpt_icp_slam

Mapping

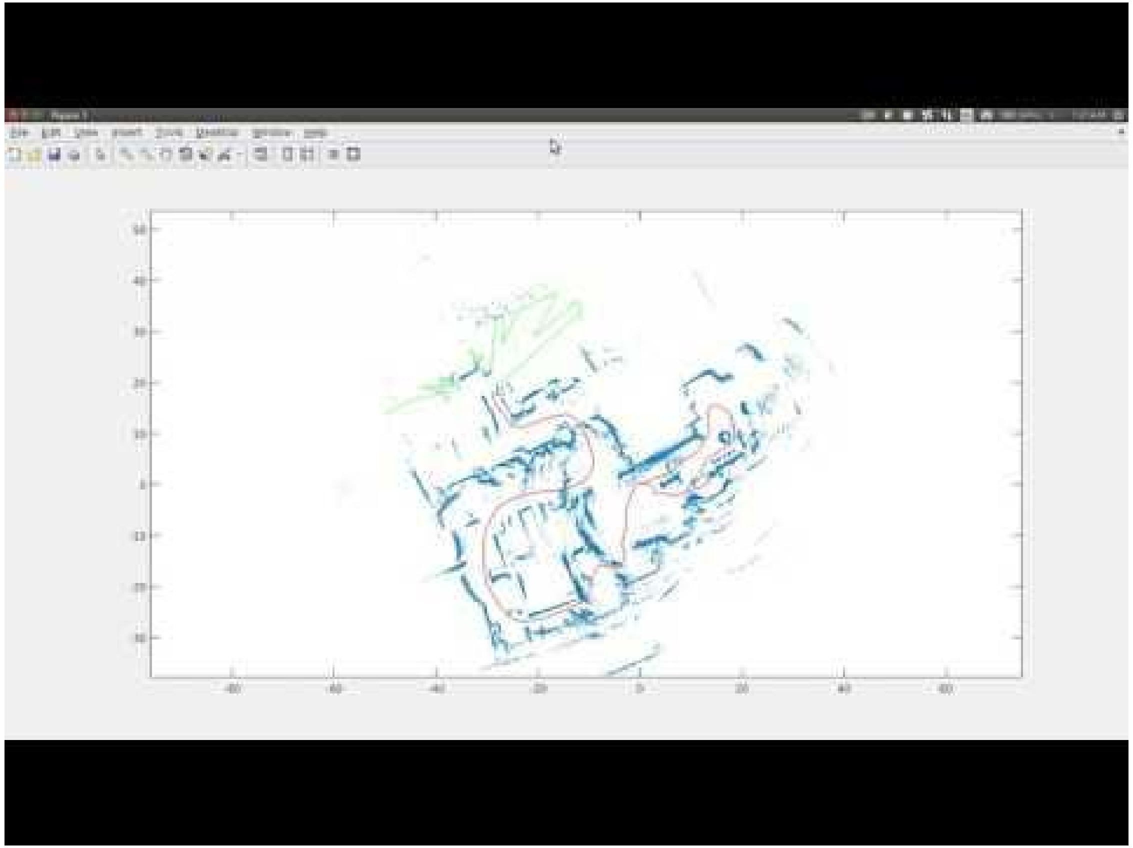
Global maps were generated based on various datasets available online by finding out the required transformations.

These datasets in general contained :

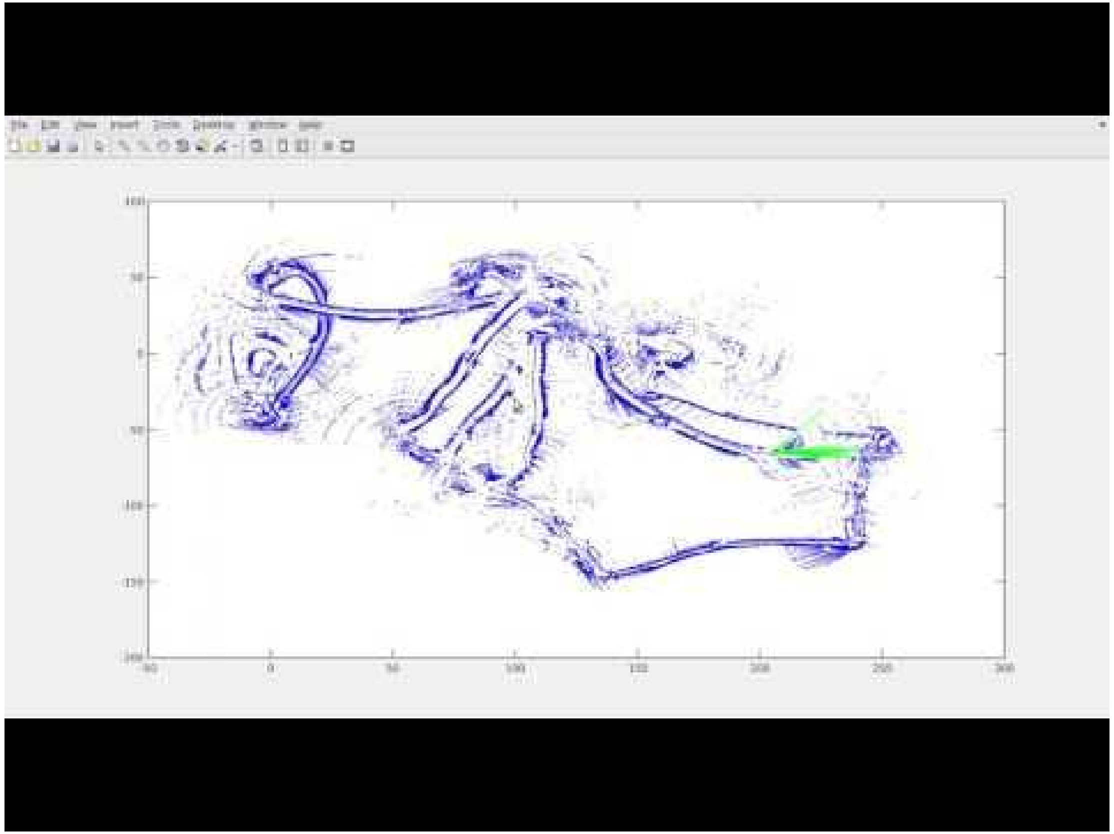
- Robot pose increments as measured by the odometry for every step.
- Laser scan at every step.

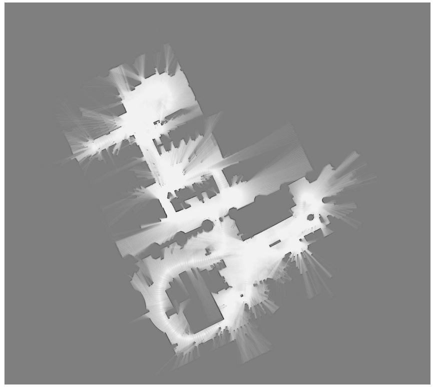
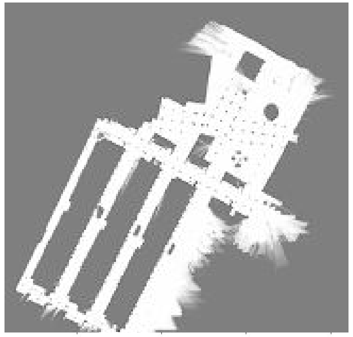
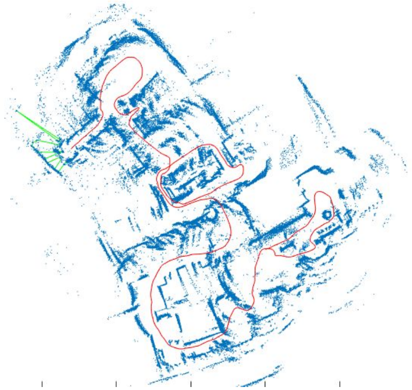
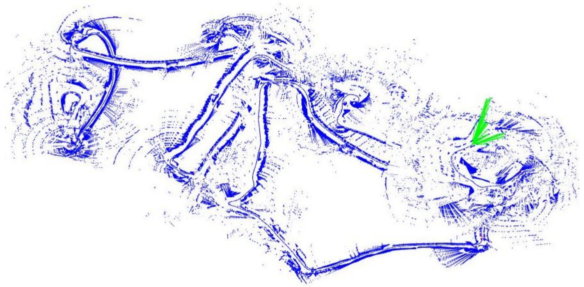
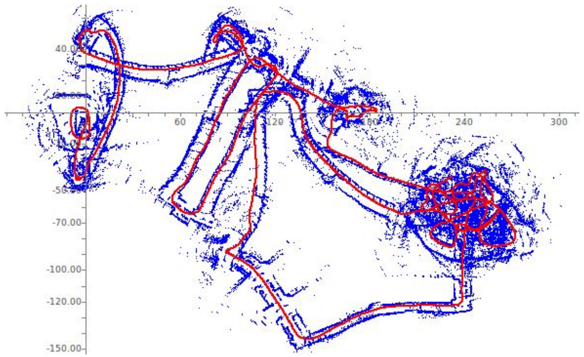
The problem we were troubled with most was with the maps getting distorted around turns.

Example Dataset : Edmonton



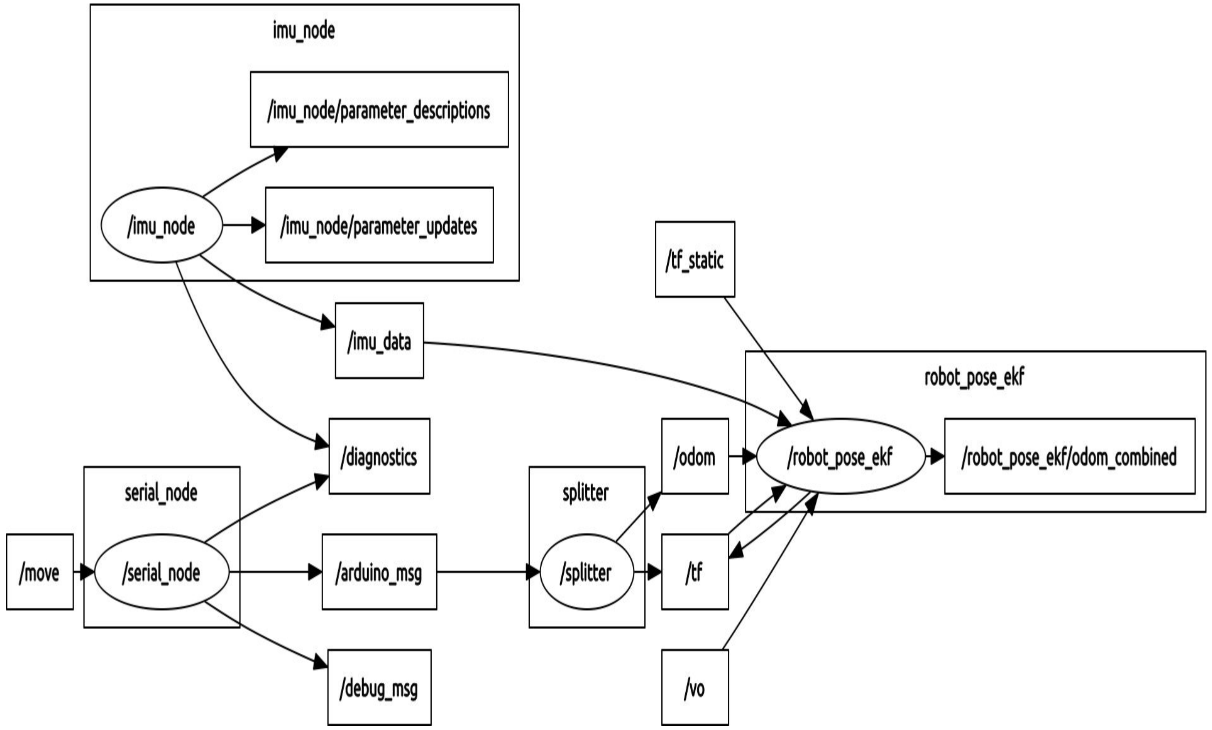
Example dataset : Malaga Campus



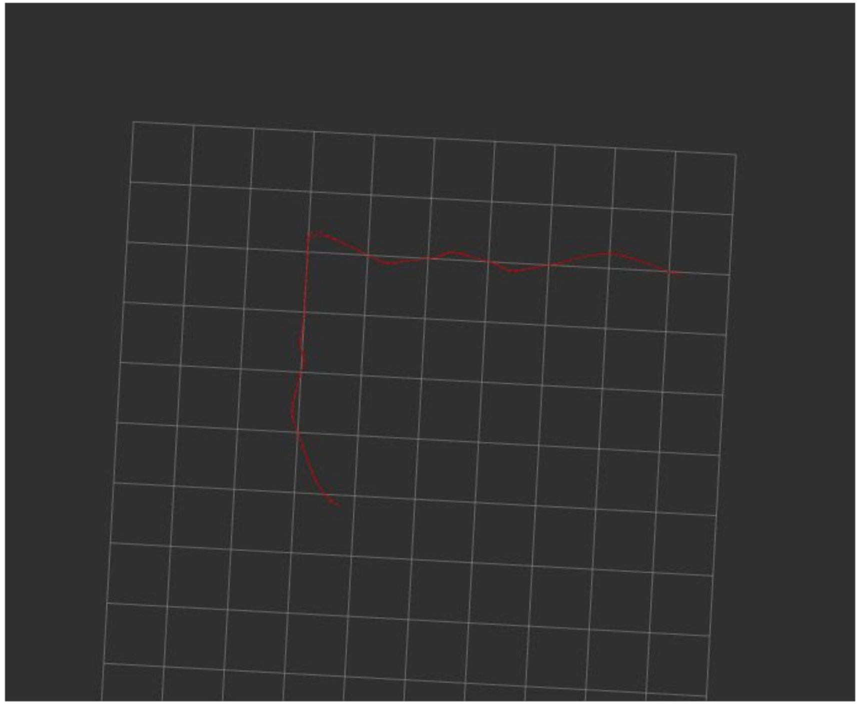
- The right images show the expected and the generated maps for the Edmonton dataset while the ones on the right are for Malaga campus dataset.
- Since the datasets contain only the encoder values and LiDAR data the generated map tends to distort when the robot takes sharp turns although this is more pronounced in the Malaga dataset.
- We tried using ICP for correcting this error but the computation times for it became very large.

Current approach



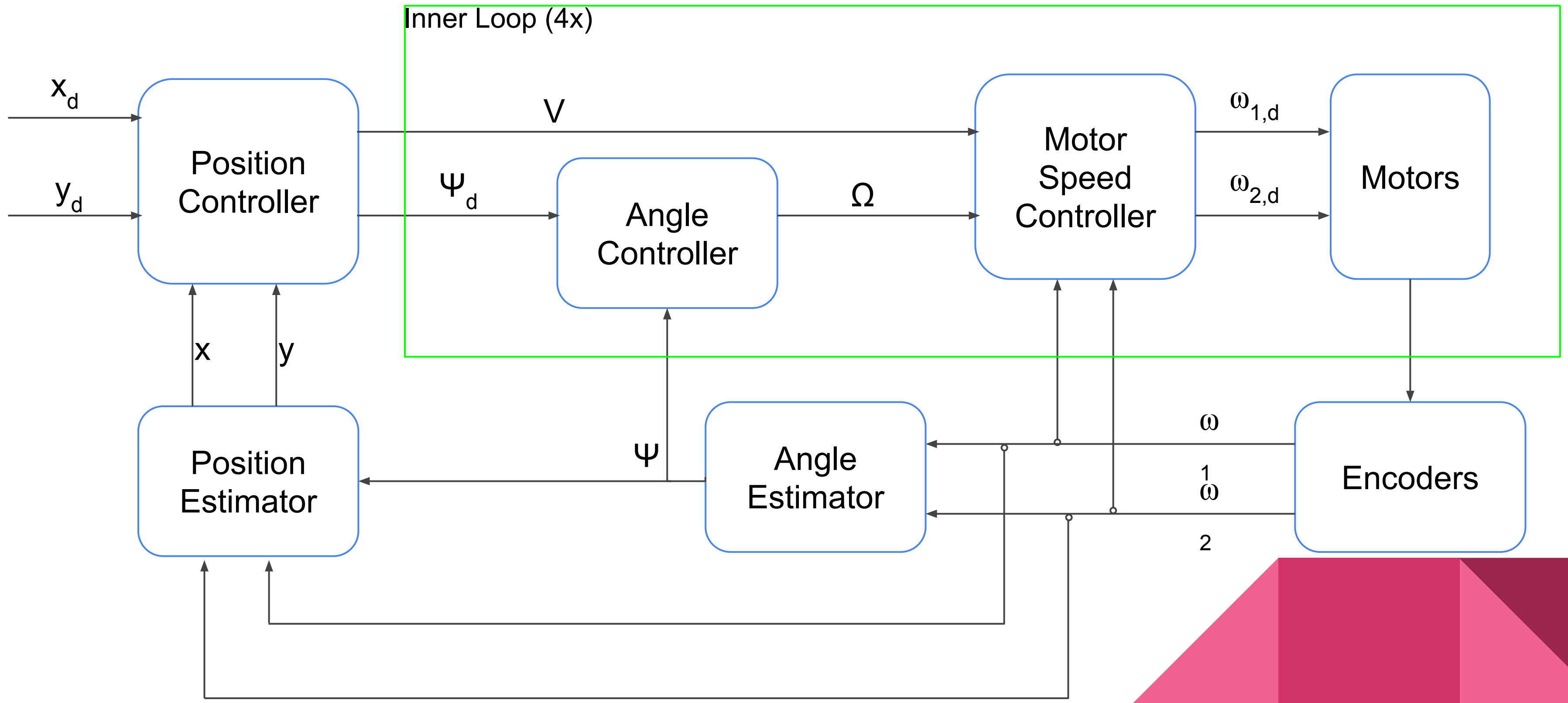
- The current plan as mentioned earlier is to use popular ros packages like slam_gmapping to create maps.
- We are able to get the transformations and the data from the IMU and odometry using encoders in the required format for slam_gmapping and need to combine LiDAR data next.

Odometry



- Odometry data is collected using encoder on both left and right wheels.
- The raw values obtained are then sent via custom ros message and transformed into the world frame and plotted.

FireBird VI Control Architecture (Differential Drive)



Feedback Control Law

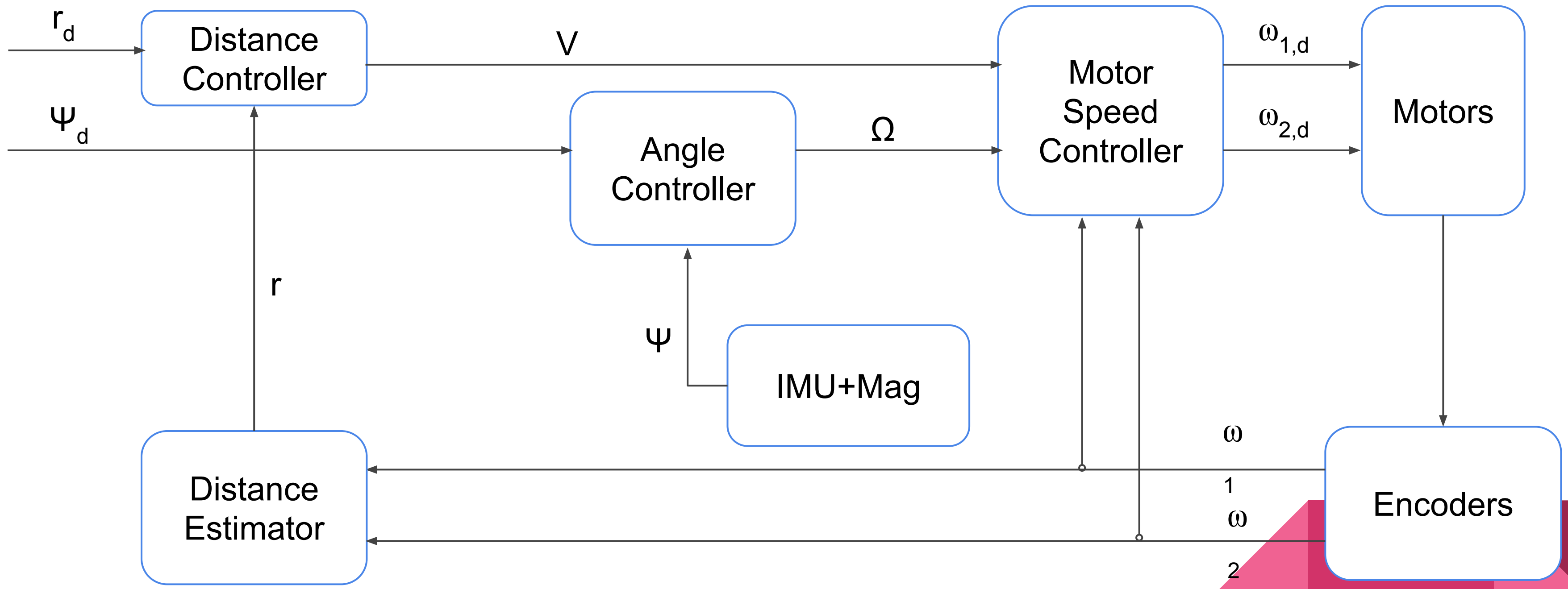
System Dynamics : $\dot{X}(t) = U$

Let r be the trajectory supplied, therefore we can get the error dynamics as follows: $\dot{e}(t) = \dot{X}(t) - \dot{r}(t) = U - \dot{r}(t)$

From first order stable error dynamics we have, $\dot{e}(t) = -K_p e(t)$ where K_p is the tuning parameter

Therefore, $U = -K_p e(t) + \dot{r}(t)$ will make the system exponentially stable.

Combat robotics Control Architecture (Differential Drive)



Motion Planning

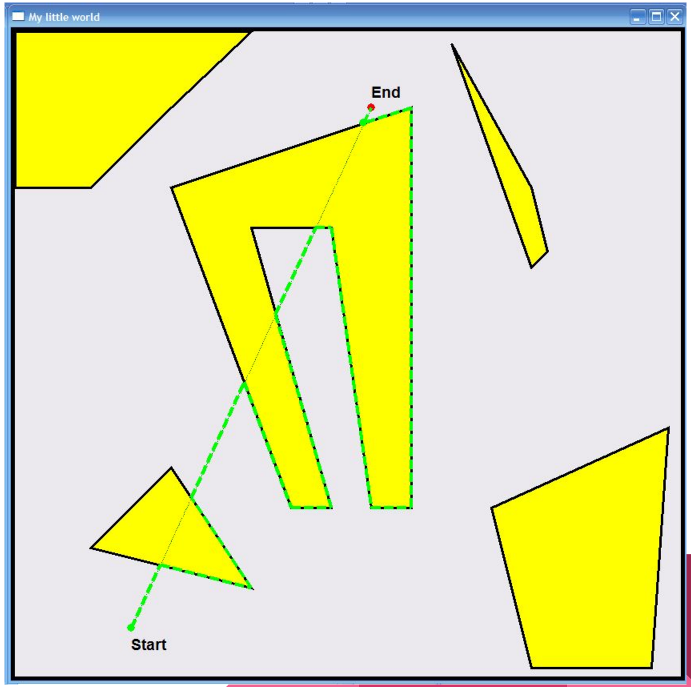
A wide range of algorithms were explored. It was decided that motion planning in 'lane' and that for 'waypoint navigation' will be dealt with separately.

Some of the algorithms tested/studied were:

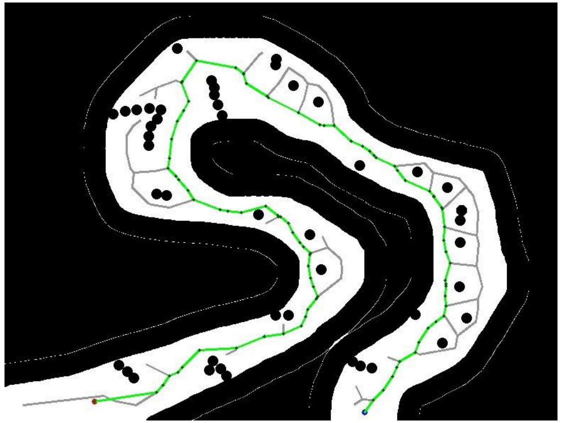
1. A*, Lifelong Planning A*
2. D* Lite
3. Bug
4. RRT
5. Potential field method
6. Voronoi Diagrams

Bug Algorithm

- This is the algorithm we tried to implement for waypoint navigation.
- It is a very basic algorithm, hence computationally inexpensive.
- What it does is that it creates a m_line between the start and the end coordinates and traverses along with it.
- When hit by an obstacle, it circumscribes it until it reaches back on the line and continues moving till it reaches the goal or it has circumscribed an obstacle and has not found the m_line.

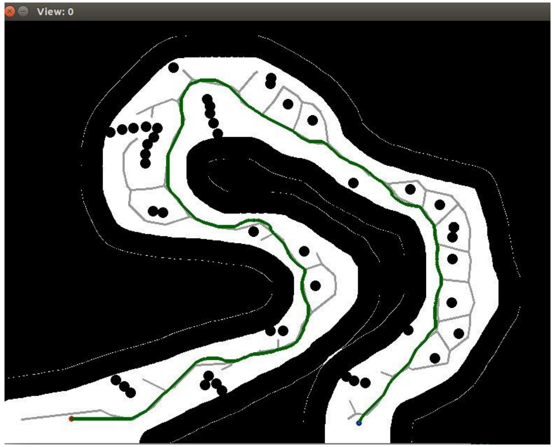


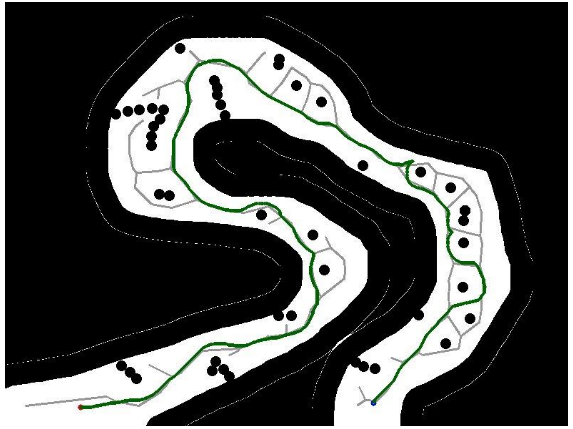
Voronoi Diagrams



A motion planned through voronoi diagrams.

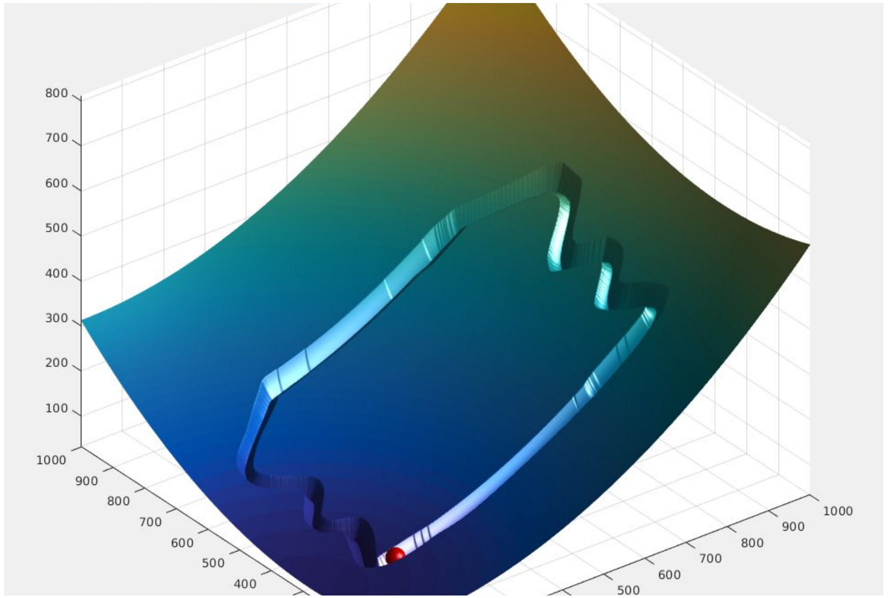
Improvements in voronoi diagram by incorporating potential fields along them . Gives best possible answer but computationally expensive



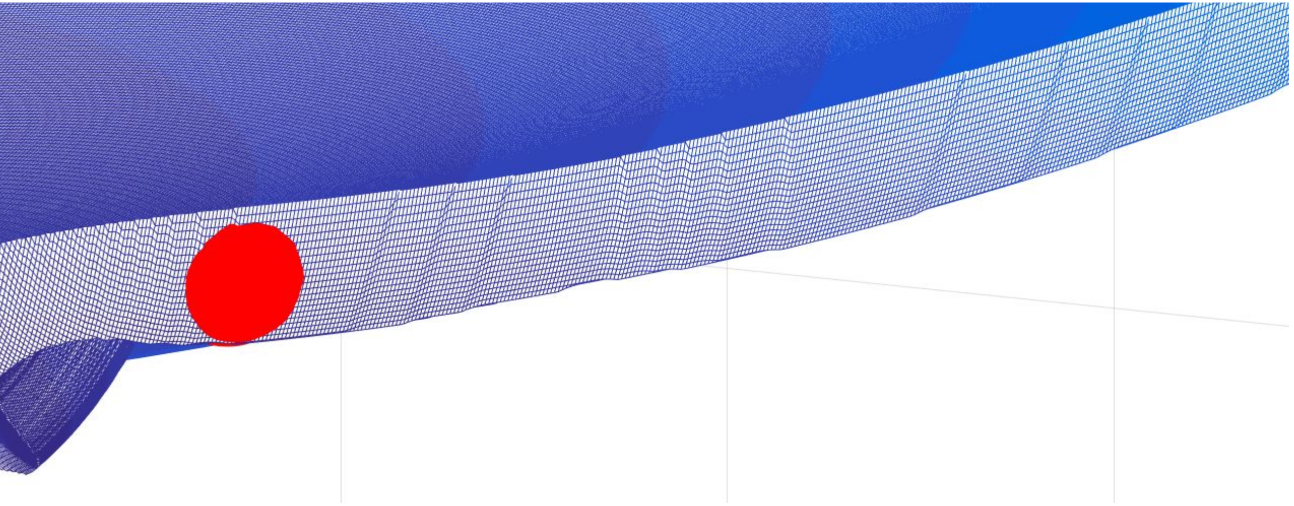


Smoothening the voronoi diagrams by giving more weightage in the x direction

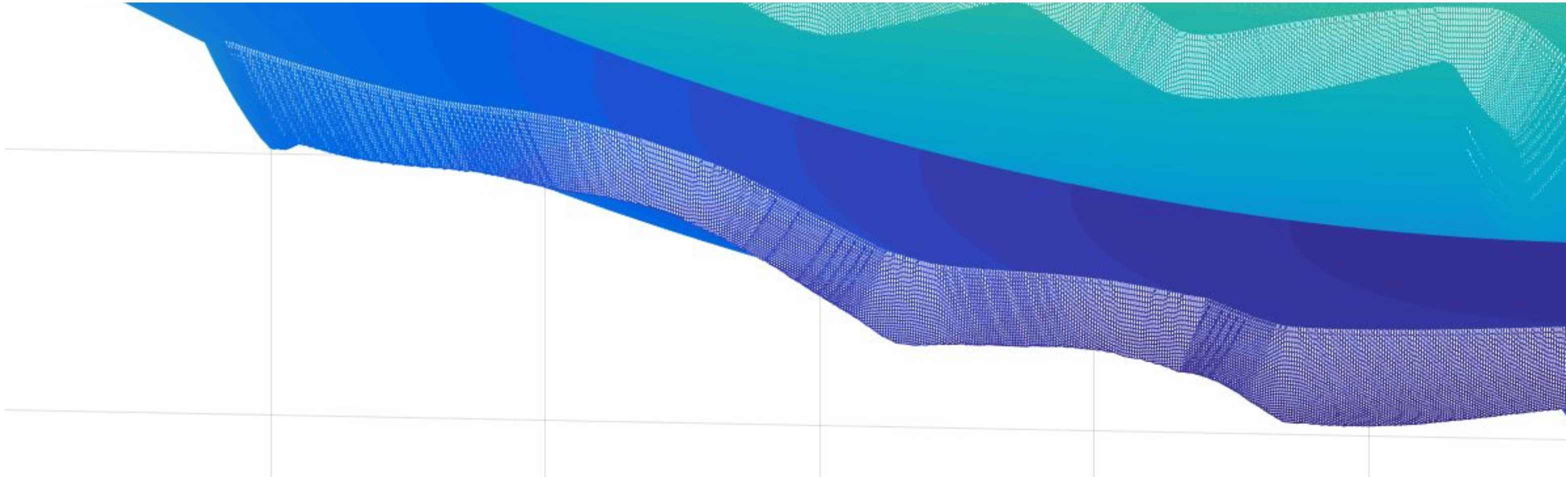
Potential Field Method - Drawbacks

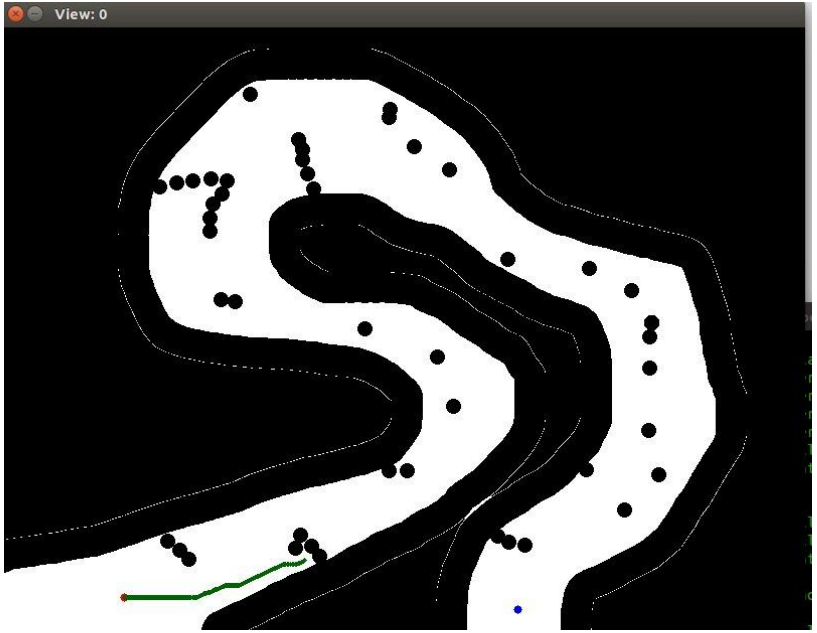


- Analogous to any other potential field
- **Attractive potential** towards goal - Quadratic function used.
- **Repulsive potential** from obstacles - Square of distance transform used
- Problems faced: Jerky potential obtained which leads to **local minimas along the lane**. The bot will get stuck here. Also, does not guarantee generating a path towards the 'right' local minima.



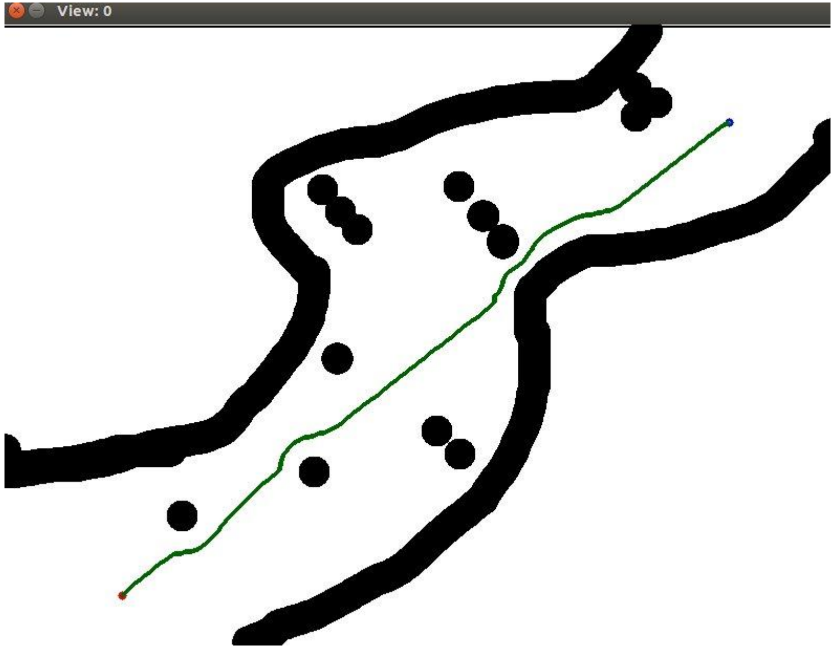
Snapshots of local minimas generated along the lane





An Example where mapping through potential field didn’t produce the expected result and got stuck in a local minima

A planned path through potential field method

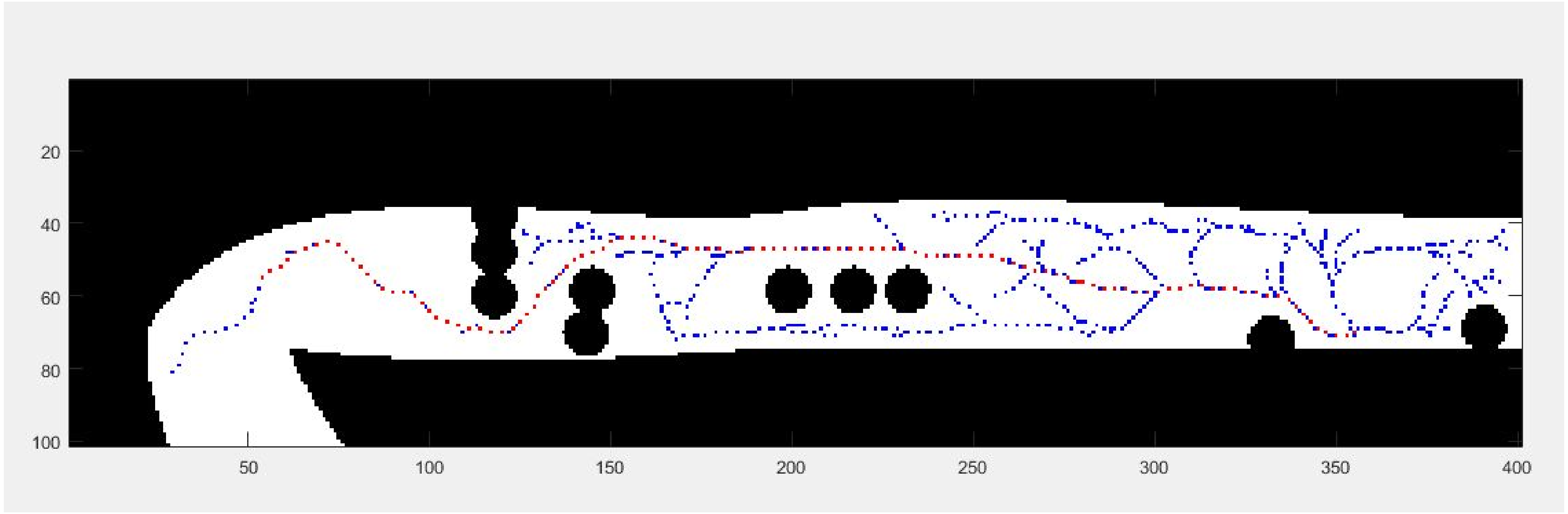


Motion Planning - Current Work

For lane, initially we had worked on **D* algorithm** with our own variations. The major reason this was not taken up further was because it could not account for **non-holononicity** of the bot.

Presently we are using RRT. The algorithm works as follows:

- A temporary goal is generated which is the **midpoint of the farthest end** of the lane visible.
- A tree is generated from the current position. Only a **fixed number of nodes** are generated irrespective of whether the goal is reached or not.
- The branch of the tree whose end lies closest to the goal is the one traversed.
- Nodes are generated are such that they lie in a **conical area of reach from the nearest node**, thus accounting for holonomicity.



A snapshot of the RRT algorithm implementation in Matlab.

The Blue nodes represent the entire generated tree. The red nodes make up the final path.
Present Problem: The path is jagged because of random generation of points. (is it a problem?)

Motion Planning - Future Work

- For waypoint navigation, we plan to use `global_planner` which is a package under the navigation stack of ROS
- A crude implementation of A* algorithm was done on ROS. Developing on the same lines, the RRT algorithm will be soon shifted to ROS.
- As and when faced with problems, the RRT algorithm will be improved upon. Some of the probable additions:
  1. Associating some probability function with the selection of random points. (is it needed?)
  2. Implement an efficient method for nearest neighbour search.

Waypoint Navigation

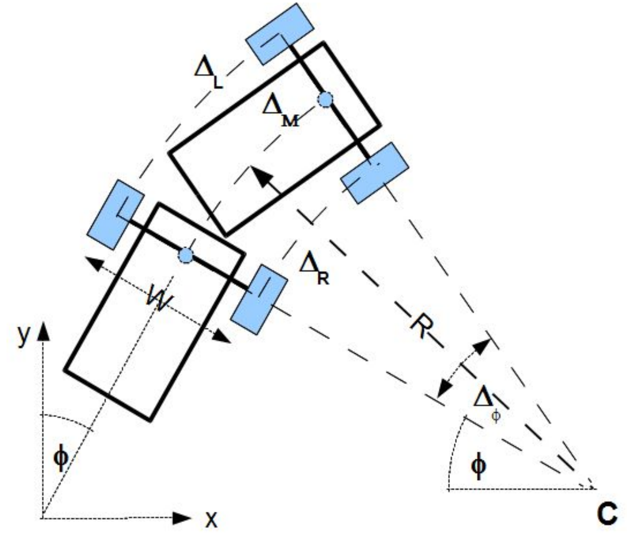
Problem statement : The robot to move through the terrain avoiding obstacles and reaching the destination coordinates.

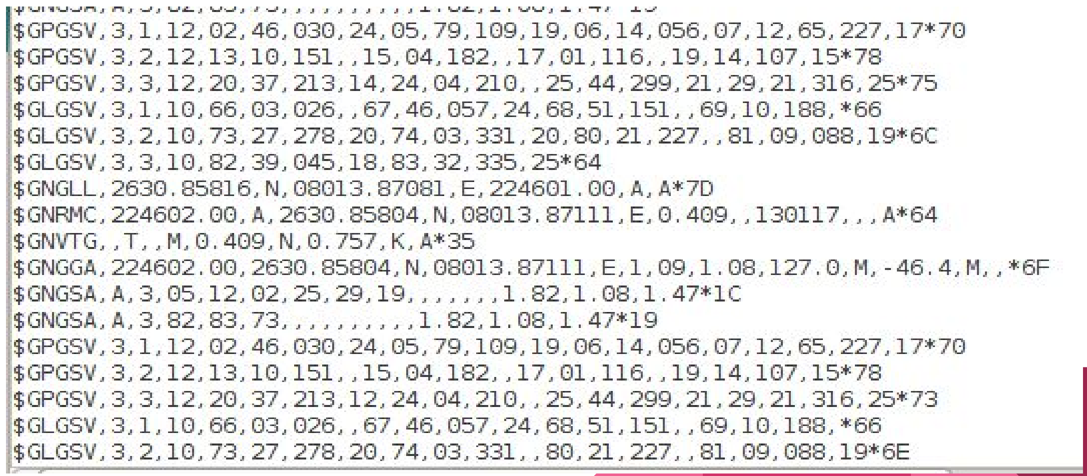
GPS :

- Extracted the GPS coordinates from raw NMEA GPS data.
- Calculated Desired Heading using present location by GPS and final coordinates.
- Converting the Latitude & Longitude to robot local frame.
- Published the information into a ROS Topic for fusing its data with other sensor data using ekf.
- Code written for waypoint navigation using GPS and Magnetometer (assuming magnetometer was calibrated and no obstacles are present).

Algorithm

- Parsing GPS data and converting it to usable form.
- Calculate heading using destination coordinates and present ones.
- True North as reference direction, fusing with IMU and finally Kalman Filter.
- Converting this data to a robot frame.





```
$GPGSV,3,1,12,02,46,030,24,05,79,109,19,06,14,056,07,12,65,227,17*70
$GPGSV,3,2,12,13,10,151,,15,04,182,,17,01,116,,19,14,107,15*78
$GPGSV,3,3,12,20,37,213,14,24,04,210,,25,44,299,21,29,21,316,25*75
$GLGSV,3,1,10,66,03,026,,67,46,057,24,68,51,151,,69,10,188,*66
$GLGSV,3,2,10,73,27,278,20,74,03,331,20,80,21,227,,81,09,088,19*6C
$GLGSV,3,3,10,82,39,045,18,83,32,335,25*64
$GNGLL,2630.85816,N,08013.87081,E,224601.00,A,A*7D
$GNRMC,224602.00,A,2630.85804,N,08013.87111,E,0.409,,130117,,,A*64
$GNVTG,,T,,M,0.409,N,0.757,K,A*35
$GNGGA,224602.00,2630.85804,N,08013.87111,E,1,09,1.08,127.0,M,-46.4,M,,*6F
$GNGSA,A,3,05,12,02,25,29,19,,,,,,,1.82,1.08,1.47*1C
$GNGSA,A,3,82,83,73,,,,,,,,,,1.82,1.08,1.47*19
$GPGSV,3,1,12,02,46,030,24,05,79,109,19,06,14,056,07,12,65,227,17*70
$GPGSV,3,2,12,13,10,151,,15,04,182,,17,01,116,,19,14,107,15*78
$GPGSV,3,3,12,20,37,213,12,24,04,210,,25,44,299,21,29,21,316,25*73
$GLGSV,3,1,10,66,03,026,,67,46,057,24,68,51,151,,69,10,188,*66
$GLGSV,3,2,10,73,27,278,20,74,03,331,,80,21,227,,81,09,088,19*6E
```

Issues in Waypoint Navigation

- Main issue was with magnetometer which gives erroneous errors .
- Error in GPS is not small to be ignored and affected by surroundings.
- GPS looses fix at certain times and robot loses track of its present position until fix is restored .

Future Work

- Obstacle detection and avoidance to be included.
- Magnetometer Calibration to be done.
- Better pose estimation to be done with Encoder , IMU fusion .

Thank You