Group 5: Ashika Rohit, Karan Chauhan, Niraj Porecha Date: 12/11/2019

EE P 545: Final Project Report

Table of Contents

1. Objective

The objective of the final project is to navigate the robot successfully from Start Waypoint (Green X), hitting all five Good Waypoints (Blue X) while avoiding the Bad Waypoints (Red X) and obstacles as fast as possible.

The map below shows the positions of the Start Waypoint, Good Waypoints and Bad Waypoints.

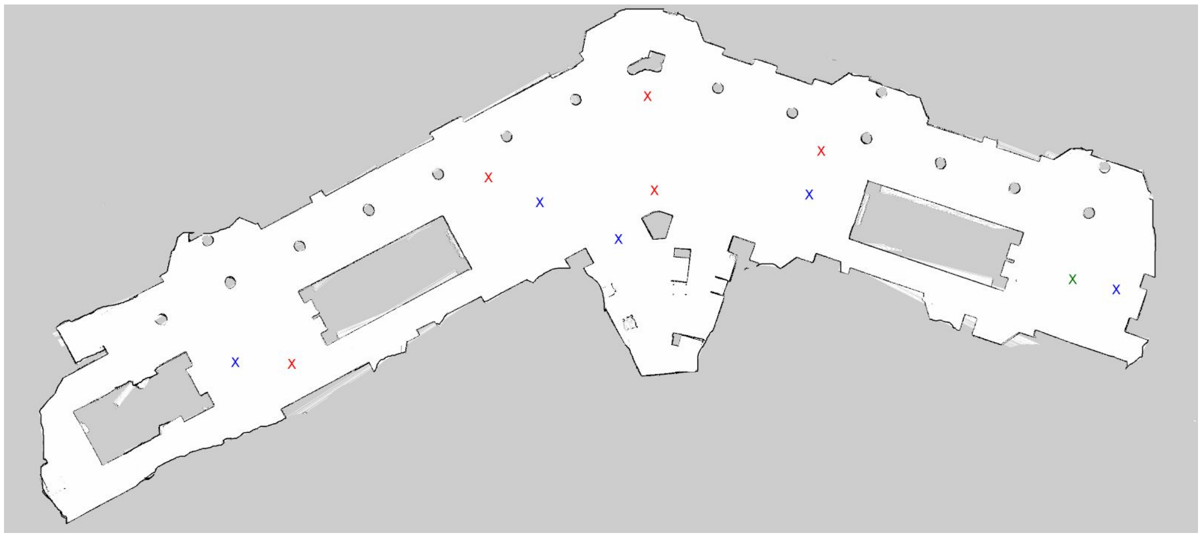

Fig1: Waypoints

2. Method Followed

We are doing this project demo in the ECE building basement and our main task is to navigate robot over the blue waypoints and to avoid the red waypoints which are already marked. To successfully navigate the robot from start to end point we used the knowledge of PID line follower from lab1, Particle Filter from lab2 and wrote a script to Plan & Publish the full path.

3. Design

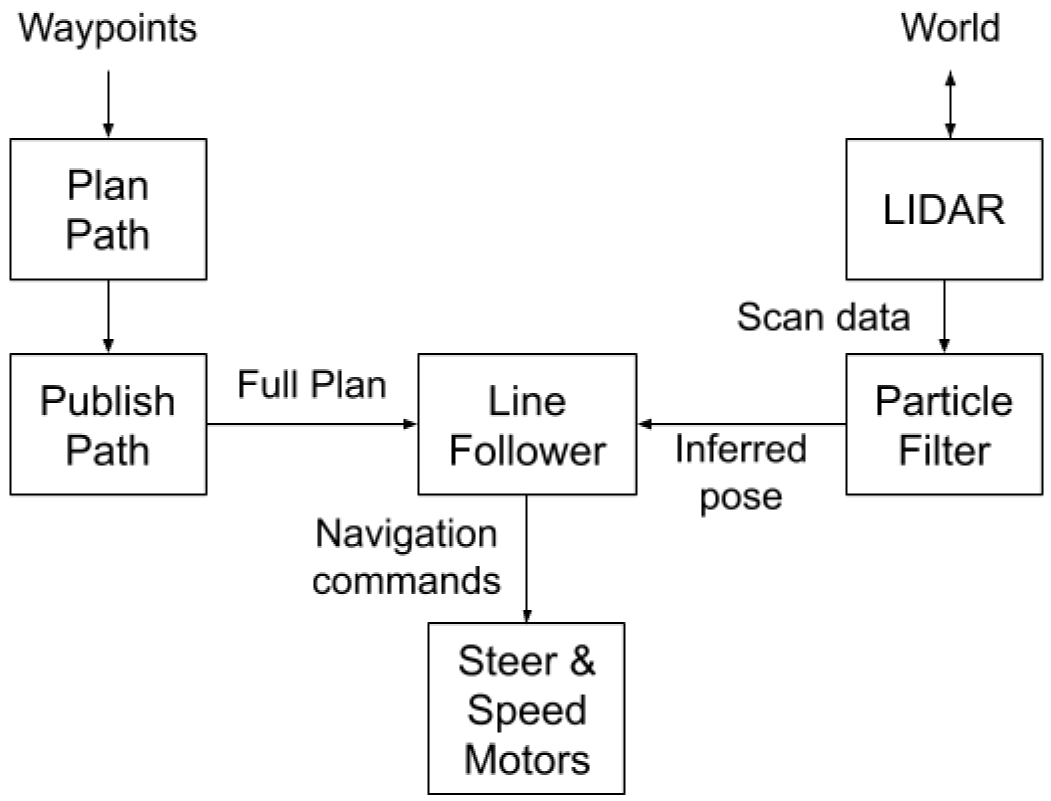

Fig2: System Architecture

As seen above, the whole system consists of three parts:

- **Particle Filter**
  The LIDAR on the robot scans the world data and sends laser scan data to the Particle Filter Script. This Data is used to find the position of the robot in the map. The Inferred pose generated by the script is published to the topic "*/pf/viz/inferred_pose*" which will then be used by the Line Follower.

- **Path Planner & Publisher**
  The Path Planner uses the numpy array of waypoints as an input. The 1st Start Waypoint is set Source and 1st Waypoint is set as Target / Goal. The Teleop script generates a Plan and publishes it to "*/planner_node/car_plan*". The path planner saves it in a full plan list. Then the 1st Waypoint is set as a source and 2nd Waypoint as Target. The generated plan is appended to the full plan list. This continues till the Last Waypoint has been set as a target. The generated plan is saved offline in a "**.npy*" file to be used in future.

The Publisher part of the script publishes the offline / new generated path (if no offline path found) continuously to the topic "*/planner_node/final_plan*". Which will then be used by the Line Follower.

- **Line Follower**
  The Line Follower gets the full plan message from the "*/planner_node/final_plan*" topic and stores it in a list. Also the script subscribes to the "*/pf/viz/inferred_pose*" topic published by the Particle Filter to get the approximate location of the robot.
  Further, the current inferred pose of the robot and the plan pose are compared and based on PID control, the steering command is given to the "*/vesc/high_level/ackermann_cmd_mux/input/nav_0*" topic keeping speed constant.

4. Project Steps

I. First of all, the robot (MIT racecar) was tuned and configured with the following parameters in "*vesc.yaml*" file:

● *speed_to_erpm_gain: 4614*	#Distance travelled
● *steering_angle_to_servo_gain: -0.8*	#Making Left and Right turn equal
● *steering_angle_to_servo_offset: 0.5880*	#Default position of servo
● *speed_min: -10000*	#Max. reverse speed
● *speed_max: 10000*	#Max. forward speed

The above parameters were found to be the best after multiple trial and error parameters.

II. A new package "*final*" was created in the "*racecar_ws/src*" for storing all the scripts and data. The Particle Filter files used were from lab2. The Line Follower files used were from lab1. The Path Planner & Publisher files were written.

III. Testing was performed on simulation before running the files on the real robot. The simulation only (Section 6) screen capture video "*SimOnly.mp4*" is submitted along with this report. Below is the Simulation path generated.

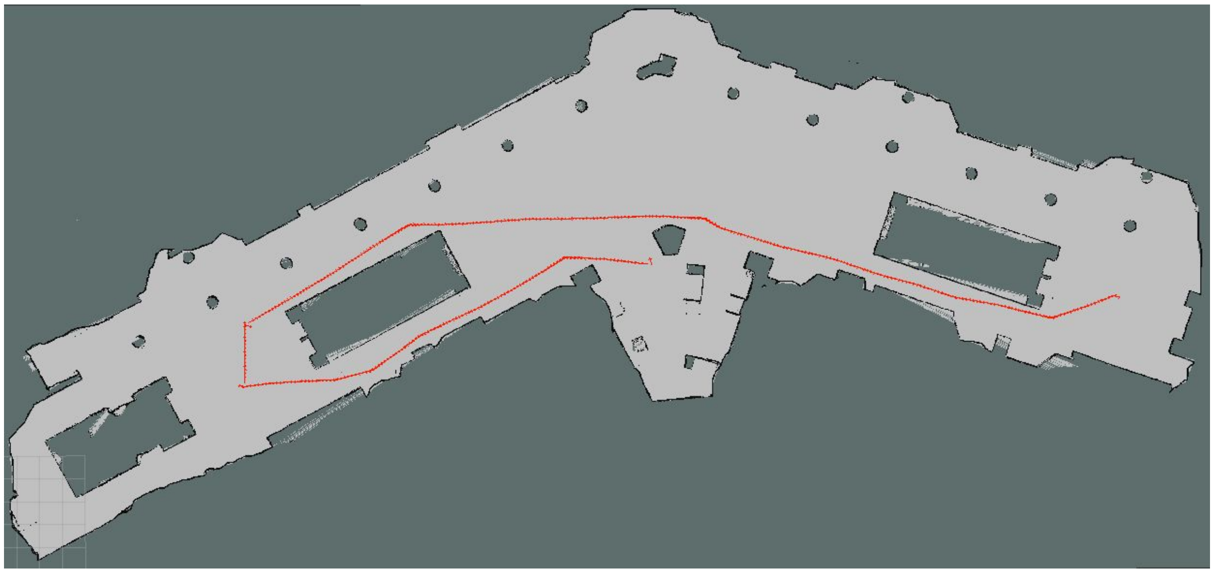

Fig3: Simulation Only Path

After performing Section 6 successfully, all the Waypoints were included in the Planner to generate a full path. However, we found that the path which was generated was not an optimal one. Thus, after multiple adjustments and additions of waypoints, below optimal path was generated by the Planner.

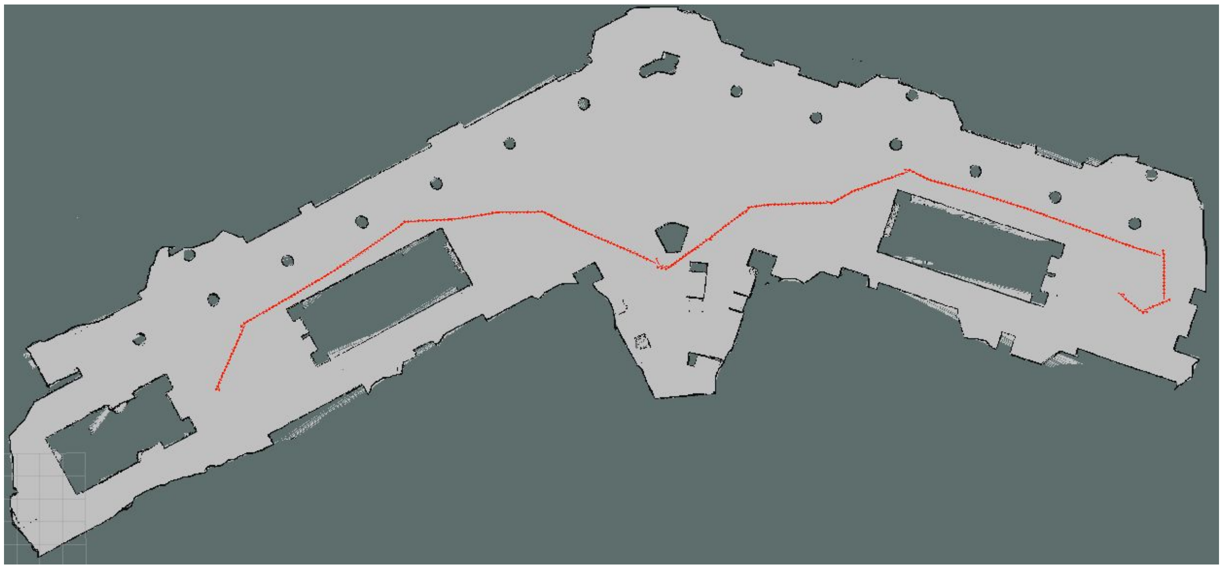

Fig4: Full Optimal Path

IV. After Simulation was performed successfully on the above path, all the files were transferred on the bot using SCP for real robot trials. The simulation screen capture video for the above path "*Simulation Optimal.mp4*" is submitted along with this report.

V. Multiple Runs were performed on robot each time adjusting the PID parameters and increasing the speed slightly in the Line follower launch file. It was found that the below parameters gave the best results.
   - *Kp = 0.8*
   - *Kd = 0.1*
   - *Ki = 0.1*
   - *Speed = 1.6*

   Further increasing the speed caused at least one good waypoint getting missed and hence these parameters were kept for the final demo. The final demo video "*Final Demo Group 5.mp4*" is attached along with this report.

5. Challenges

- Robot number 9 suddenly stopped steering and was moving only in Straight directions. After diagnosis along with TA, it was found that there was an issue with the servo motor since the commands from controller were being received by the Robot. So, we got another robot number 8.

- Doing development without skeleton code was a little challenging and we took some time to get the project up and running.

- Even though getting the robot to behave as expected in Simulation mode was more straightforward, getting the robot to work in the real world was more challenging. At first the robot wasn't moving at all. It took us about a day to figure out how to get the code running on the bot.

- Initial runs were unsuccessful and only the starting 3 waypoints were being hit. After the turn on the pillar, the particle filter was getting lost. Then, by setting the "*max_range_meters*" launch parameter of Particle Filter to "*16.0*" from "*11.0*", Particle Filter became accurate and all good waypoints were hit.

6. Results & Conclusion

Using the above method mentioned in the report, i.e. PID control on the line follower, we were able to achieve a time of *30.42* seconds to complete the task which was the fastest run in the whole class. It was amazing to see such a simple controller if tuned correctly can work wonderfully.

This method can be used to complete the final project without any problems if the Particle Filter is accurate. However, the shortcoming with this method is that the robot cannot detect any obstacles on the path generated and avoid them. Thus, laser data can also be used to avoid the obstacles. Once obstacle is passed, the robot will then again follow the path.

7. Timeline

● Project Start Date	: Nov 27th
● Design Complete	: Dec 4th
● Development Complete	: Dec 6th
● Simulation Testing Complete	: Dec 6th
● On-bot Testing	: Dec 6th-8th
● Project Demo	: Dec 9th
● Report Complete	: Dec 12th

8. Submission

● EE545 Final Report_Group 5.pdf	#Final Report
● final.zip	#complete package with scripts
● SimOnly.mp4	#Section 6 task video
● Simulation Optimal.mp4	#Final Project Simulation video
● Final Demo Group 5.mp4	#Final Project Demo video