Individual Lab Report 10
Progress Review 11

Shivam Gautam
sgautam@andrew.cmu.edu

Team Daedalus
Mohak Bhardwaj, Pranav Maheshwari, Shivam Gautam, Richa Varma, Dorothy Kirlew

March 31st, 2016

1 Individual Progress

I primarily worked on coding the Global Motion Planner and testing the Oculus Prime in the parking lot. This involved coding the following sections-

1. Global Planner
2. Mapping of the Parking Lot
3. Testing Navigation of the Oculus Prime Platform

1.1 Global Planner

In addition to refining the code for the assigning of spots to a car, I worked on adding ROS services for the global planner to communicate with the local planner as well as the rendering engine. All the services declared are isolated from the planning aspects in a separate class that deals with handling the ROS interface. The code is written in a way that allows for integration of any planning algorithm to be integrated with the planner as long as it conforms to the coding conventions.

For implementing the communication between the rendering engine and the local planner, I needed to create services that conformed to the architecture in figure 1. The architecture depicts all the three services running between the three nodes of the simulation environment. The services that the global planner advertises and subscribes to are explained below.

1.1.1 Optimal Spot Generator

This service is advertised by the global planner and its client is the rendering engine. The goal of this service is to tell the rendering engine where to send the car next in the queue.

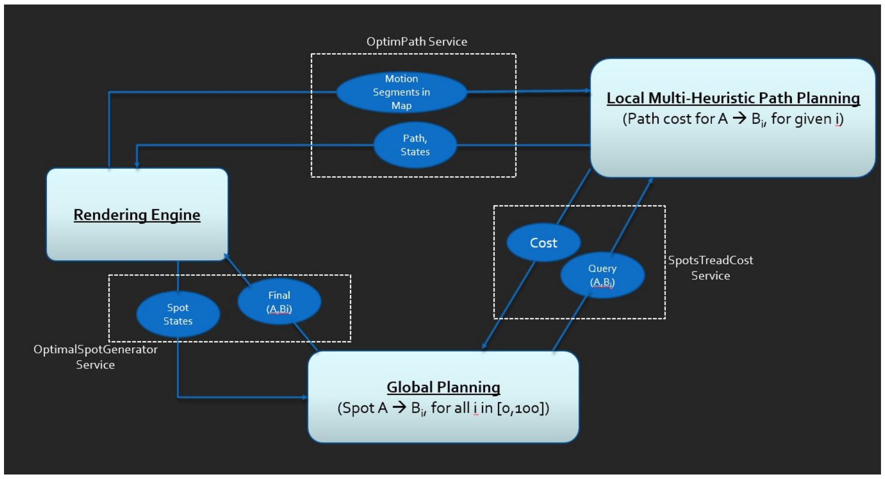

Figure 1: Architecture Depicting ROS Services between nodes

The rendering engine queries the global planner with a request, asking the global planner for the current-best spot for the car in the queue. The rendering engine shares additional information in this service request like the number of vehicles in the queue and the state of the parking lot.

The Global planner service responds with the pose of the car that the rendering engine should allocate as the destination to the car. The pose message is of geometrymsg/PoseStamped type, a common ROS message type used to send time-stamped poses.

1.1.2 Spots Tread Cost

This service is advertised by the local planner and the global planner is its client. This service essential tells the global planner what the cost would be to move to a certain spot in the parking lot, based on the heuristics of the local planner.

The global planner sends a request to the local planner, with the starting pose of a car and a request flag. The local planner replies to this service after computing a path from this current pose to all the spots in the parking lot. The local planner then returns the cost of sending a car from the current pose to the end pose, taking the motion primitives of the car into account.

The global planner uses this cost information and combines it with the other heuristics it considers when assigning a car a spot.

1.2 Mapping of the Parking lot

Making improvements from the previous progress review, we decided to test the mapping on the parking lot fabricated from wood. The parking lot that I mapped is depicted in 2.

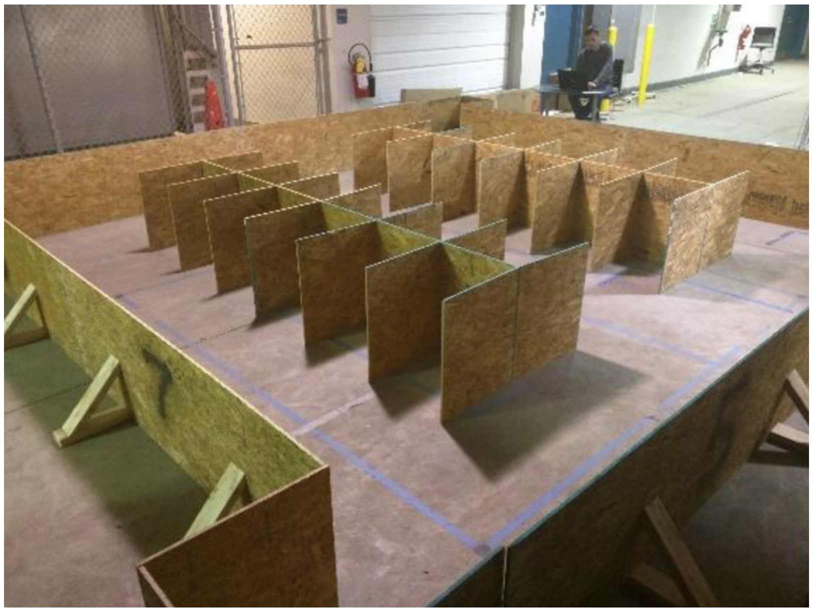

Figure 2: Parking lot used to create the map (Picture Taken By Pranav)

The lot was of significantly better quality as disassembling and reassembling did not alter the localization within the map. The map created for the lot is depicted in 3.

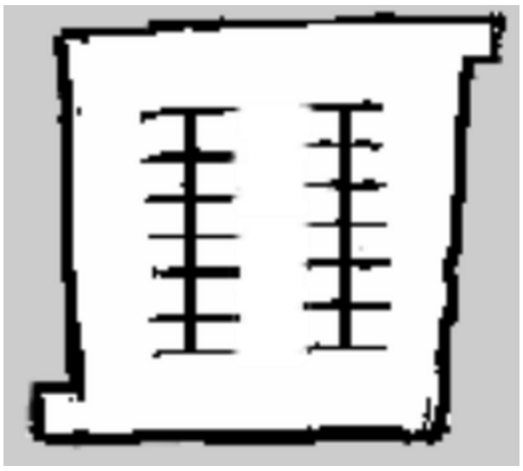

Figure 3: Map created of the parking lot (Joint work of Shivam and Pranav)

The map was made by running the GMapping package and taking 3D depth data from the Asus Xtion. The mapping, however, took more time than the previous instances as now the robot was not just mapping an empty space with boundaries. The robot now had to map a more feature-dense environment and the process was more time consuming. We decided to use this map for localization after refining it in bitmap editor, which was done by Pranav.

1.3 Testing Navigation of the Oculus Prime Platform

We tested the navigation of the platform in the new parking lot. Additional waypoints were added to the map to include all the parking spots as waypoints. On testing, the platform did localize itself in the parking lot but could not navigate robustly due to the less clearance between walls.

The ROS DWA (Dynamic Window Approach) planner tries to find a control to navigate within the constrained parking lot but failed to do so in our parking lot. This could be attributed to the fact that the DWA planner 'shoots' feasible trajectories towards a local goal considering the robot to have the same motion primitives as a bicycle, even though our platform can turn on its axis. Upon removing the walls between the spots, the platform navigated to the designated parking spots. We also faced certain issues in starting the navigation stack with the XBee connected which we debugged.

2 Challenges

One of the challenges I faced was at the time of navigation testing of the Oculus Prime platform in the parking lot. Initially, the parking lot included wooden walls in between spots. The platform could plan a path into the spot but could not find a suitable exit manoeuvre out of the lot. The ROS navigation stack would throw the error 'DWA Planner Failed to produce path.'. The DWA (Dynamic Window Approach) planner essentially performs a forward simulation from the robot's current state to predict what would happen if a sampled velocity were applied for some period of time. In our case, the DWA planner could not find a valid trajectory due to the proximity of the walls to the platform. Hence, we decided to remove the walls in between spots and the problem was solved.

Mapping the new parking lot was also a challenge to an extent due to the tight corners in the parking lot and tight clearances between parking spots. We were, however, able to create a map which resembled the lot to a certain extent and could be refined later. I also worked on debugging minor hardware issues with the platforms.

While writing code for the global planner, I faced certain issues with the linker. I isolated the class definitions and declarations in separate ".h" and ".cpp" files to separate the interface from the implementation.I sorted these issues out during the debugging phase. Implementing the ROS interface as a separate class is extra work but definitely increases the comprehensibility of the code.

3 Teamwork

I collaborated with Mohak and Pranav for the simulation environment. This involved deciding the data for the services required and testing the srv files that were created. Mohak and Pranav worked with me to test the navigation of the Oculus Prime. I also worked with Pranav for mapping the parking lot. Mohak also created a working version of the local planner with the functionality of ROS services. Dorothy created the new parking lot from wood. Richa collaborated with Dorothy on fixing the old platform and debugging the hardware issues with it. Further, they also made additions to the visualization tool designed by Pranav.

4 Plans

Our next goal is to test the navigation in the parking lot and ensure its robustness. We are currently waiting for the new platform to arrive but there is a high likelihood that it might not arrive before the next progress review. To mitigate this, we would try and fix the issues with the old platform to have two working platforms. Having done that, we would be able to test our system that we are going to present for the Spring Validation Experiment. Pranav and Mohak would be working with me to test our SVE. Also, I would be working on integrating the planning part of the global planner with the ROS services of the planner. Pranav, Mohak and I would be integrating and testing the simulation environment. Dorothy and Richa would be working on getting the old platform up and running.

5 References

1. DWA Planner - http://wiki.ros.org/dwa_local_planner