Progress Review 11

Individual lab report – 10 | March 30, 2016

TEAM DAEDALUS

Submitted By:

Mohak Bhardwaj

Team Members:
Dorothy Kirlew
Richa Varma
Shivam Gautam

Pranav Maheshwari

1. Individual Progress

The following responsibilities were taken up by me:

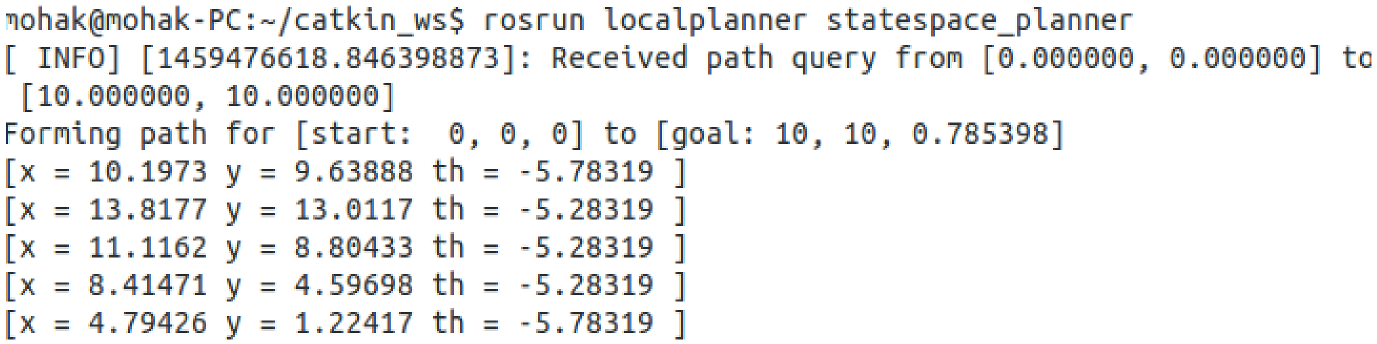
- Development of Local Planner
- Integration of Local Planner with Visualization tool.
- Navigation and testing in new parking lot.

LOCAL PLANNER

The local planner is responsible for returning an optimal path given a query of a start pos and end pose of the robot. The local planner was implemented in a basic form in python before the last progress review. But, because the local planner has to communicate with both the global planner and visualization, the rate at which paths are planned need to be very high to satisfy the large query volume.

The state-space planner was implemented using C++ in a ROS framework. In order to encapsulate pose and graph information, custom data structures had to be created and managed using functors in C++. The state-space lattice planner was implemented keeping in mind the different data structures for optimality in terms of speed and memory consumption. Also, special routines were defined for operations to be performed on poses and nodes which are not natively available in C++. Functors for maintaining STL sets and maps of custom data structures and classes had to be defined for keeping in track of the states that had been generated while traversing the state-space lattice. Along with this were a priority queue and linked list implementation for the actual path planning. Functions were also developed for the bicycle model based on certain specified motion primitives, environment loading, robust handling of queries and collision checking.

Since all the elements of the local planner were developed directly from scratch, a large amount of testing had to be done so that it performs well under different types of queries. The A* lattice planner successfully generates smooth trajectories with motion primitives of an ackermann drive.



```
mohak@mohak-PC:~/catkin_ws$ rosrun localplanner statespace_planner
[ INFO] [1459476618.846398873]: Received path query from [0.000000, 0.000000] to
 [10.000000, 10.000000]
Forming path for [start:  0, 0, 0] to [goal: 10, 10, 0.785398]
[x = 10.1973 y = 9.63888 th = -5.78319 ]
[x = 13.8177 y = 13.0117 th = -5.28319 ]
[x = 11.1162 y = 8.80433 th = -5.28319 ]
[x = 8.41471 y = 4.59698 th = -5.28319 ]
[x = 4.79426 y = 1.22417 th = -5.78319 ]
```

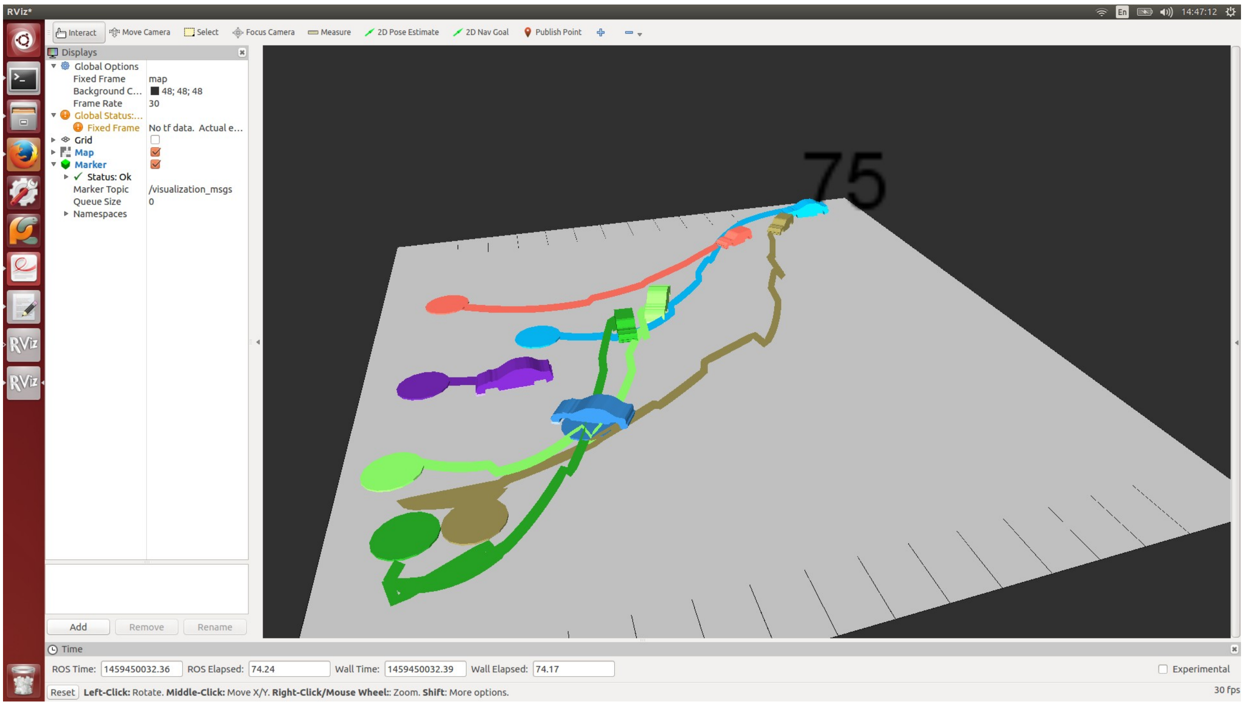
Figure 1: Output path of local planner for a specific query

INTEGRATION OF LOCAL PLANNER WITH VISUALIZATION TOOL

Once the local planner was functional, I developed the required components for integrating the local planner with visualization tool and global planner. I defined two different service files for ROS communication:

1. **optimPath**: This service takes care of communication between the local planner and visualization engine. The rendering engine sends a request to the local planner in the form of a geometry_msgs/PoseStamped ROS message containing a start and goal pose. Internally, the local planner performs conversion to the local data structure and returns a path in the the form of an array of gemoetry_msgs/PoseStamped.
2. **spotsTreadCost**: This service takes care of communication between local planner and global planner. The global planner again sends a query of a start and goal position but this time the local planner replies only with the cos of the optimal path the the goal location considering multiple heuristics such as distance travelled, state of lot, motion segments etc.

The local planner has been integrated with the visualization tool and figure 2 shows the paths visualized for random goal poses given as a query.



Global Options
Fixed Frame: map
Background C...: 48; 48; 48
Frame Rate: 30
Global Status:...
Fixed Frame: No tf data. Actual e...
Grid: □
Map: ☑
Marker: ☑
Status: Ok
Marker Topic: /visualization_msgs
Queue Size: 0
Namespaces

ROS Time: 1459450032.36 ROS Elapsed: 74.24 Wall Time: 1459450032.39 Wall Elapsed: 74.17 □ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options.

30 fps

Figure 2: Paths visualized for random goal positions

Figure 3 shows the detailed architecture of the simulation with different services that will be running. The figure also details he heuristics that will be taken into account by the global and local planners.

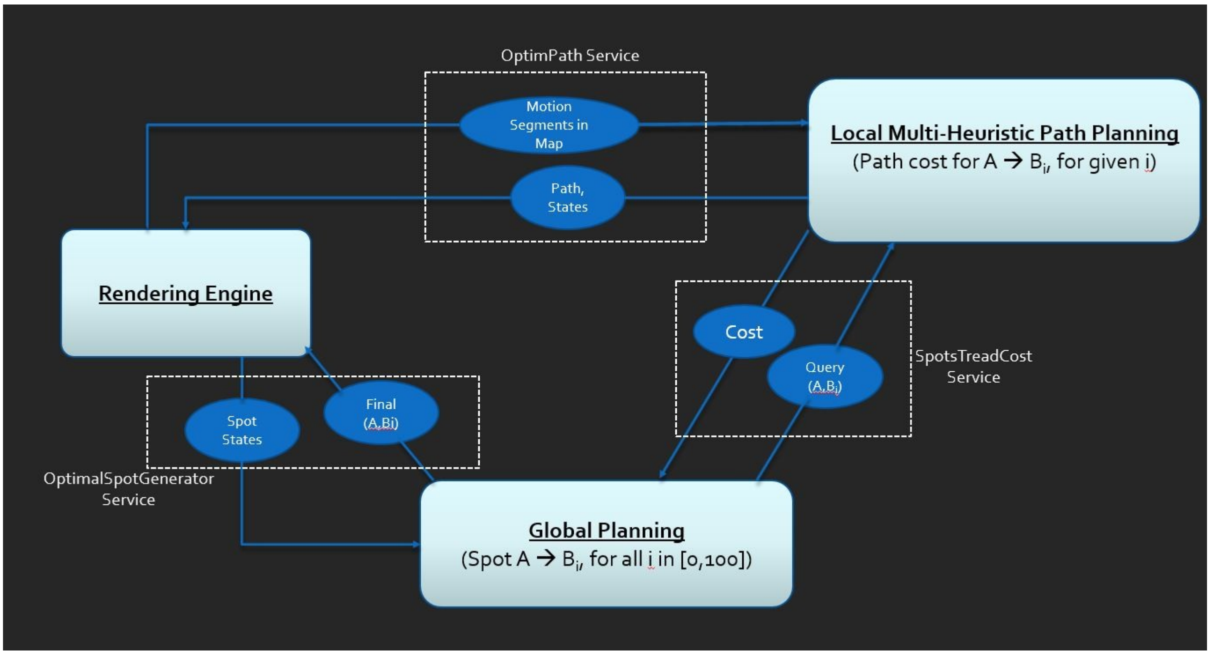


Figure 3: Detailed architecture of simulation environment

(Image created by Shivam)

NAVIGATION AND SYSTEM TESTING IN NEW PARKING LOT

Apart from the local planner, I also worked with Pranav and Shivam on testing and debugging of the navigation system in the new parking lot. This was followed by testing of the full pipeline from the app to navigation in the new lot. The map had been created with the partitions between the parking spots but during actual navigation, the partitions were folded because the platform was not able to find feasible paths in constrained areas. The platform was also having difficulties in localizing itself and finding paths that lead it into the parking spot. This was because the ROS DWA planner running in the background finds paths using a bicycle model and does not take into account the fact that our platform is actually a differential drive one. Hence, when it does not find a feasible path and quits, there actually is a path available. All these different problems had to be debugged and it was decided to make a separate routine specifically for entering into a parking spot.

2. Challenges

1. Implementing the planner in C++ was non-trivial and many different aspects had to be kept in mind so that everything works out together. Making the different data structures work in sync with each other required considerable amount of effort.
2. For integration of the local planner with ROS framework, appropriate service call definitions had to be maintained which increased the complexity of the code.
3. Many issues with localization and parking lot dimensions were faced while testing the platform and a lot of time was invested in resolving them.

3. Teamwork

The other members of my team were working on the following subsystems:

1. Shivam: He worked with Pranav on mapping of the new parking lot and testing of the platform and on developing a working version of the global planner with multiple heuristics.
2. Pranav: He worked on the rendering engine of the simulator and with me on integration of local planner. He also worked with Shivam on integration of global planner with visualizer and mapping of new parking lot. He also worked on the navigation aspects of the mobile platform.
3. Dorothy: She worked with Richa on setting up the hardware and software of the old platform and improving the visualization tool previously designed by Pranav. She also worked on the fabrication of the current parking lot.
4. Richa: She worked with Dorothy on setting up the hardware and software of the old platform and improving the visualization tool of the parking lot previously designed by Pranav.

4. Plans

Once the new platform arrives, we will be working on developing all necessary functionality of our SVE Demo 1. I will be working on improving the local planner by adding collision checking along many points on the generated path, multi-threading for query handling and adding multiple heuristics. This will be followed by dividing the code across multiple files for clarity and robustness. Pranav will be working on completing the functionality of the rendering engine and Shivam will integrate all the different facets of the global planner. Pranav, Shivam and I will also be working on ensuring the robustness of the physical platform and and full system integration. Richa and Dorothy will be working on the complete integration of the hardware and software of the old platform.