PROGRESS REVIEW #7

INDIVIDUAL LAB REPORT [ILR06]

JANUARY 28, 2016

TUSHAR CHUGH

TEAM B-AUTOPIRATES

SHIYU DONG, BIKRAMJOT HANZRA, TAE-HYUNG KIM, WILLIAM SETO

PROGRESS REVIEW #7

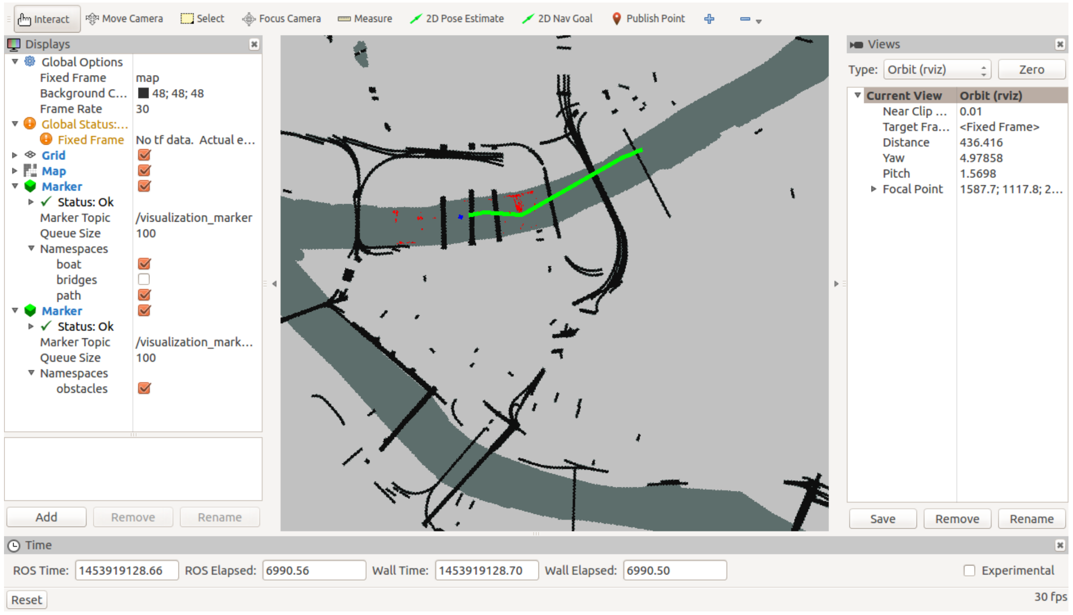
INDIVIDUAL PROGRESS

During this progress review we showcased some critical additions to the feature set of the project which also included the integration. Personally, I worked on making the systems code more robust and modular, integration of radar data to path planner and adding continuous re-planning option to the path planner.

Path Planning

a. *Making code more robust and modular*
   Our code base was getting bigger and harder to manage. Adding more features was taking a lot of time and finding bugs in the code was becoming difficult. So, during the winter break I restructured and rewrote the code from following object oriented pattern. This includes creating separate modules for environment, path planning, GUI (RViz Markers), interfacing with low level controllers and integrating with obstacles data. After decoupling these components, our turnaround time for adding new features has been improved.

b. *Integration with Radar Data*
   We wanted to start with integration as soon as possible because this is generally the phase which takes a lot of effort and time. This week, I was successfully able to complete the integration of radar data with our system. Now, environment is updated based of the obstacle data received from radar and the path planner can use the updated environment to output the new path. To receive radar data, the system subscribes to the ROS topics which are published after applying filters to the raw radar data. *Figure 1* showcases obstacles captured by radar with red color, boat with blue color and the path with green color.



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_marker
Queue Size	100
Namespaces	
boat	✓
bridges	☐
path	✓
Marker	✓
✓ Status: Ok	
Marker Topic	/visualization_mark...
Queue Size	100
Namespaces	
obstacles	✓

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Distance	436.416
Yaw	4.97858
Pitch	1.5698
Focal Point	1587.7; 1117.8; 2...

ROS Time: 1453919128.66 ROS Elapsed: 6990.56 Wall Time: 1453919128.70 Wall Elapsed: 6990.50 ☐ Experimental

30 fps

Figure 1: Integration of radar data with Path Planner

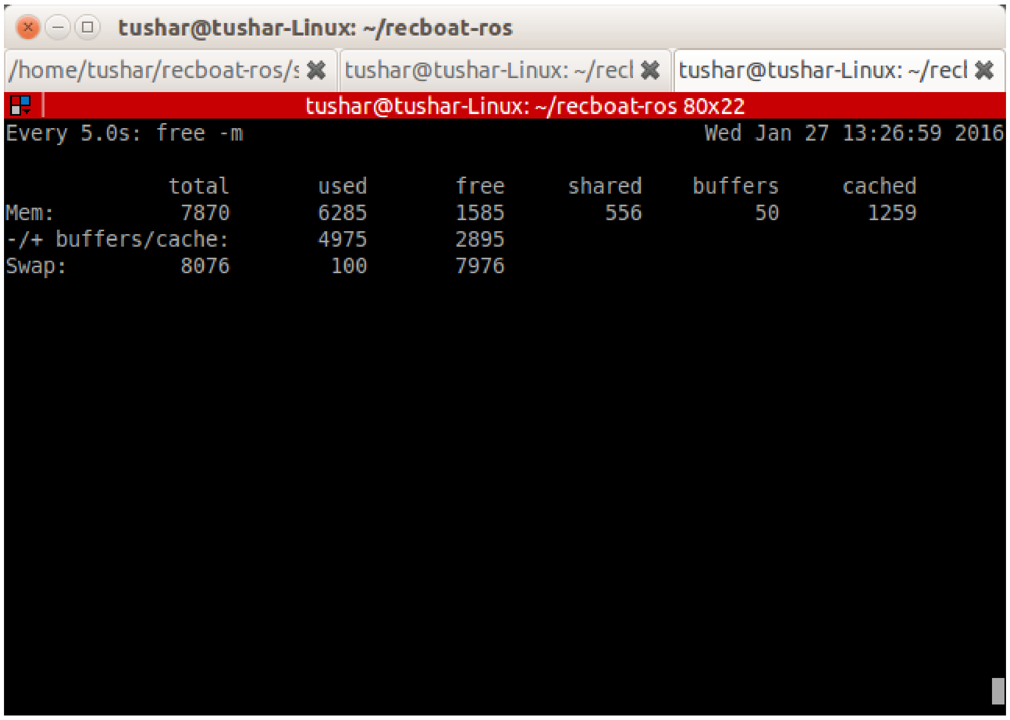
c. Continuous Re-planning

In all previous demos, we used to plan the path only once. But now, as we are getting regular updates of obstacles from radar, so we needed to continuously update the path of the boat. At present we are running the planner in the loop and it is taking around 2-3 seconds per iteration.

CHALLENGES

a. Issues with Memory

When I tried to run the planner again and again to re-plan the path, my computer started to crash. It used to take only 3-4 iterations for the planner to consume all 8GB available on my computer and hence landing the systems in the hanged state. *Figure 2* shows that the planner used more than 4GB of memory (6284 MB – 2000 MB for other processes) only in 3 iterations. [Running one more time would have taken my computer down]. Initially, I was thinking that there might be some bug in the code which is causing this issue. But after debugging for a couple of days, I was able to figure out that the memory used with our system was increasing every time we were re-planning. The reason for this is that SBPL saves the costs and heuristics every time we run the planner. Right now, I fixed this by deleting and reinitializing the objects of environment and planner after every run as there are no methods available to handle directly by the environment object of SBPL. Going forward, I would have to edit the SBPL library to handle this in a better and efficient way.


tushar@tushar-Linux: ~/recboat-ros
/home/tushar/recboat-ros/s
tushar@tushar-Linux: ~/recl
tushar@tushar-Linux: ~/recl
tushar@tushar-Linux: ~/recboat-ros 80x22

```
Every 5.0s: free -m                                    Wed Jan 27 13:26:59 2016

             total       used       free     shared    buffers     cached
Mem:          7870       6285       1585        556         50       1259
-/+ buffers/cache:       4975       2895
Swap:         8076        100       7976
```

Figure 2: Memory management and check

TEAM WORK

a. Shiyu Dong: To start our system we have to run a lot of modules separately from terminal. It's time consuming and mundane. In order to solve this issue, Shiyu created the launcher for launching all our required ROS packages at once.

b. Bikram Hanzra: Bikram created the module to fake obstacles using interactive_maker package of RViz. This feature has also been helpful to test the re-planning path planner.

c. Tae-Hyung Kim: Tae-Hyung explored independently on how to integrate robot localization package in ROS with navigation stack in simulation.

d. William Seto: William is setting ambitious goals for team so that we can complete the project before deadline. He is also helping other team members in achieving their respective goal. Other than helping me in resolving some of the memory related issue he also worked on adding the pylons to the map. In addition to this, he worked with Bikram to add 'interactive_markers' subscriber to the planner code base.

FUTURE WORK

a. Fixing bugs and improving performance

There are some of the very important components that I need to work on before the next field test. This includes:

- At present, we are clearing the list of waypoints and sending the list of new waypoints every time a new path is planned. This will cause the boat to stop and will cause the boat to jerks during this time interval. I would be solving this issue by only sending 2-3 waypoints to the boat (nearest) and then appending to the same list (rather than sending a clear command) when we re-plan.
- Right now, re-planning is done continuously. I need to see how to re-plan only when we get new obstacles in the path. Structure of this is already done. Some minor tweaks are required before the field test.

b. Rigorous testing during our next field test

There are a lot of features which still haven't been tested on the boat. I plan to test the following:

- Our new code structure and integration hasn't been test on the boat even once. We need to make sure that this works. This primary task would be to make sure that it works well.
- Interfacing with low-level controller has to be reliable. I also need to verify this.
- Motion Primitives: It is important for us to that the boat able to follow paths properly which are generated by the path planner. If I see the boat is not able to take some curves then I would need to update the motion primitives accordingly.