Individual Lab Report 9

Progress Review 10

Pulkit Goyal

March 22, 2018

Team F - Falcon Eye
Pulkit Goyal
Pratibha Tripathi
Yuchi Wang
Rahul Ramkrishnan
Danendra singh

Individual Progress

I mainly contributed to the complete navigation stack of the husky which include localization, path planning, real-time obstacle avoidance and mapping. Rahul, pratibha and Danendra worked on this along with me. As it's a big problem statement, We referred to various stacks given on the internet and tried a lot of them before finalizing the Husky outdoor waypoint navigation.

Navigation of Husky Robot in Outdoor environment

We started off with ROS navigation package given on http://wiki.ros.org/navigation.
This page led us to different navigation tutorials available for Husky robot. After understanding the basic concepts of move base, transforms between different frames and various other sensor integrations we moved ahead with implementation of Husky outdoor navigation stack given on Husky's official web page.
http://www.clearpathrobotics.com/assets/guides/husky/HuskyGPSWaypointNav.html

To start with we modified the launch file of the default package given on the above link.
The complete package include following sensors and launch modules (as shown in Fig 1),

1. Husky bring up: Husky control, Publishes Husky odom data.
2. Laser: Launch file for the laser file (create map and register obstacles)
3. IMU
4. GPS
5. Localization: Localization using odometry, imu and navsat
6. Gmapping: creating the map
7. Move_base: A major component of navigation stack
8. Safety_node: Safety system of the robot.

We modified all the launch files in order to accommodate the sensors that were available with us. We had procured a separate IMU, GPS, and Laser sensor. We modified all the packages and launch files in order to replace them with the ones used with Husky default.

```
<launch>

    <!-- Run husky_bringup -->
    <include file="$(find outdoor_waypoint_nav)/launch/include/husky_bringup.launch" />

    <!-- launch lms1xx lidar -->
    <include file="$(find outdoor_waypoint_nav)/launch/include/laser.launch" />

    <!-- launch um6 imu -->
    <include file="$(find outdoor_waypoint_nav)/launch/include/imu.launch" />

    <!-- launch gps -->
    <include file="$(find outdoor_waypoint_nav)/launch/include/gps.launch" />

    <!-- Launch robot_localization and navsat -->
    <include file="$(find outdoor_waypoint_nav)/launch/include/localization_run.launch"/>

    <!-- Run gmapping to create map -->
    <include file="$(find husky_navigation)/launch/gmapping.launch" />

    <!-- Launch hector_slam for laser odometry and mapping -->
    <!-- include file="$(find hector_mapping_nav)/launch/scanmatch_nav.launch" /-->

    <!-- Run husky_navigation to enable move_base -->
    <include file="$(find husky_navigation)/launch/move_base_nav.launch" />

    <!-- launch safety node to allow the user to disable velocity commands to the robot -->
    <include file="$(find outdoor_waypoint_nav)/launch/include/safety_node.launch" />

  <!-- Run rviz -->
     <!-- Using standard configuration file -->
        <!--node name="rviz" pkg="rviz" type="rviz" args="-d $(find husky_viz)/rviz/model.rviz" /-->
     <!-- Using custom configuration file -->
        <!--node name="rviz" pkg="rviz" type="rviz" args="-d $(find outdoor_waypoint_nav)/rviz_config/model.rviz" /-->

</launch>
```

Fig.1: List of all the modules in the final package

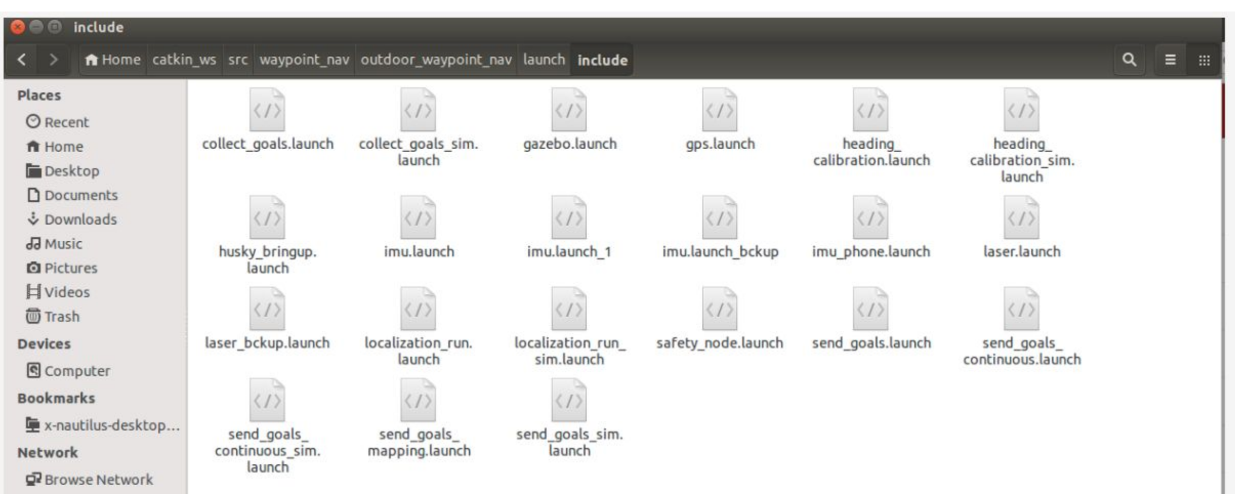

Fig.2: All the launch files being launched for running the complete package

We ran the module with the configuration we used used during FVE, we tried with the phone IMU(as shown in Fig.2). Once the things were working fine upto a certain extent we started using the sensors we worked upon independently until now. We faced a lot of integration issues, which will be highlighted in the challenges section. We actually spent a week, i.e. four people spent a full week understanding the different concepts of the different modules of outdoor way point navigation. There were a lot of integration issues with the module. But once those integration issues got solved the package worked pretty smoothly. Figure 3 shows all the topics and nodes once all the launch files and modules were integrated and started working perfectly fine.

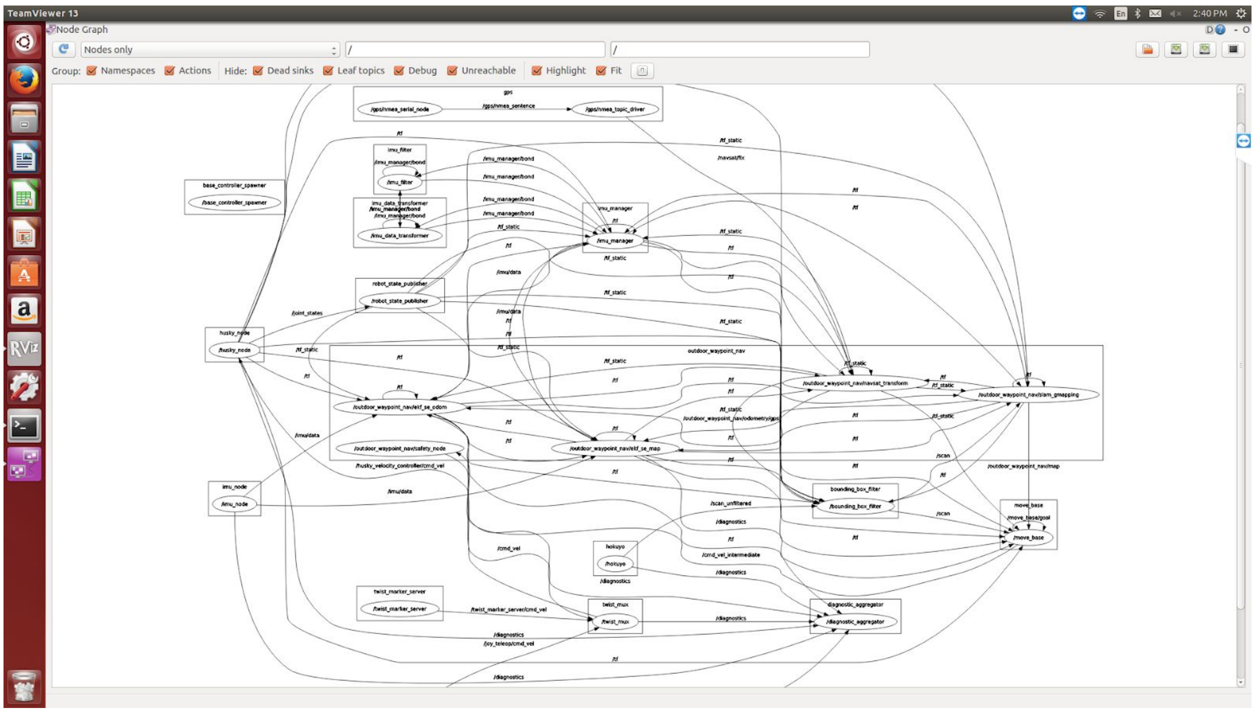

Fig.3: Nodes/Topics after all the modules are up and running

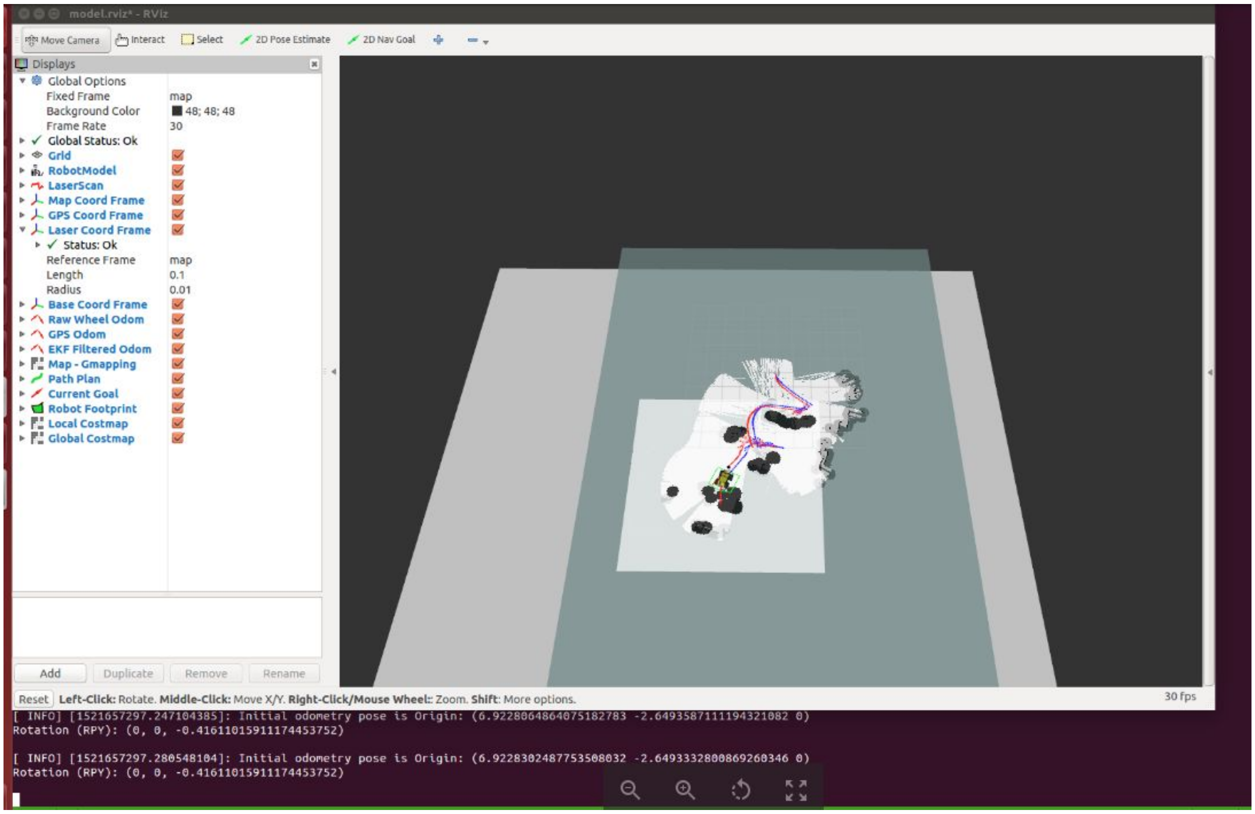

Fig.4: Husky moving towards the goal location after avoiding obstacles dynamically and creating a map

The integrated stack is pretty impressive. It has the capability of avoiding the obstacles and plan the path in real time as shown in Figure 4. The red markers is the feedback we are getting from the odometry and the blue marker is the EKF result after fusing all the state estimates.

The package has following features which we found really impressive:

1. Goal can be provided by clicking in front of the rviz window. The Husky will go towards the goal avoiding all the obstacles on the way.
2. As soon as any button on the controller is pressed, the control is given to the controller. It is a very good safety feature which helps us take manual control of the autonomous husky in case of any emergency situation.
3. Once a goal is placed in the rviz, it shows the planned path if you put an obstacle dynamically in its path it replans real time avoiding the obstacle.

Challenges

1. Understanding and putting the right transforms between different published sensor messages was quite challenging.
2. There were many Ubuntu / ROS related issues, we had a package installed on the Husky PC and the same package was also included in the waypoint navigation repository which we pulled from git in the catkin workspace. ROS is by default supposed to read the configuration and launch files from the catkin workspace in case the same package is present both in it's shared folder and workspace. But in our case things were working pretty normally in the first run but after that ros was somehow caching the file present in shared folder to run with the rest of the modules. We modified the package placed in catkin workspace but the one present in the shared folder was not modified. This led to running two EKF nodes and hence confusing the map and entire system in seeing robot positions at the same time. As a solution we had to make the same changes to the files present in the shared folder. So that even if ROS caches the packages placed in the shared folder things should work as expected.
3. Integration of GPS with the current EKF
    a. Right now there are problems with GPS transforms
    b. GPS values are continuously drifting
    c. GPS time clock synchronization issue
    d. Tuning the covariance matrix
4. Because of bad weather we could only test our completed software stack outside only a couple of times, as also pointed by Prof. John Dolan the 2D Hokuyo Lidar we are using may not work in outdoor settings.
5. Weather is definitely an issue, we need to tests all our sensors in the outdoor settings. GPS is also an important part of the solution of the problem.

Teamwork

1. Danendra worked on creating and debugging the cost map for the final module.
2. Yuchi worked on Bebop exploring the area for the possible april tags location.
3. I worked along with Pratibha, Rahul and Danendra in setting up the complete navigation stack for the Husky.

Future Plans

We specifically plan to work on the following tasks. Integrating the different modules to work together which involves the following:

1. Bebop sending the goal location to husky over the WiFi network.
2. Testing the Husky's control stack to move progressively on short term goals given by Bebop.
3. Working with the current sensor stack specially Hokuyo Lidar outside.
4. We also need to integrate the GPS with the other EKF and understand the transforms and synchronization of time which is required for the entire stack.
5. The Husky navigation stack requires a lots outdoor testing.
6. Integration of UAV and AGV subsystems.