Individual Lab Report #6

Pratik Chatrath

Team A / Team Avengers

Teammates: Tushar Agrawal, Sean Bryan

January 28, 2016

I. Individual Progress

Overall in the last semester, I worked on developing obstacle detection system for our UAV using array of 14 ultrasonic sensors. In Fall Validation Experiment, I demo the functionality of the sensor system. The sensor system met the performance requirements but wasn't consistent. At the end of fall semester, we as a team decided to shift to X8+ UAV platform.

As a part of this change, this semester we had to redo the trade study for which sensors to use. After conducting and analysing the trade study, I decided to go with LIDAR sensor. The next step was to develop obstacle avoidance algorithm and implement it. I found that the Robotic Operating System Navigation Stack is capable of taking sensor data and generating path while avoiding obstacles. In the past couple of weeks, I have been able to configure this Navigation Stack to take LIDAR data and generate path to a goal in simulation.

I would like to break down my previous week's work into following sections and then elaborate on each one of them:

1. High Level Plan for our system
2. Reason for shifting to LIDAR from array of Ultrasonic sensors
3. Implementation of Navigation Stack in Simulation

1. High Level Plan for our system

As shown below, the flowchart describes different tasks that the UAV will carry from take-off to landing back at the base station.

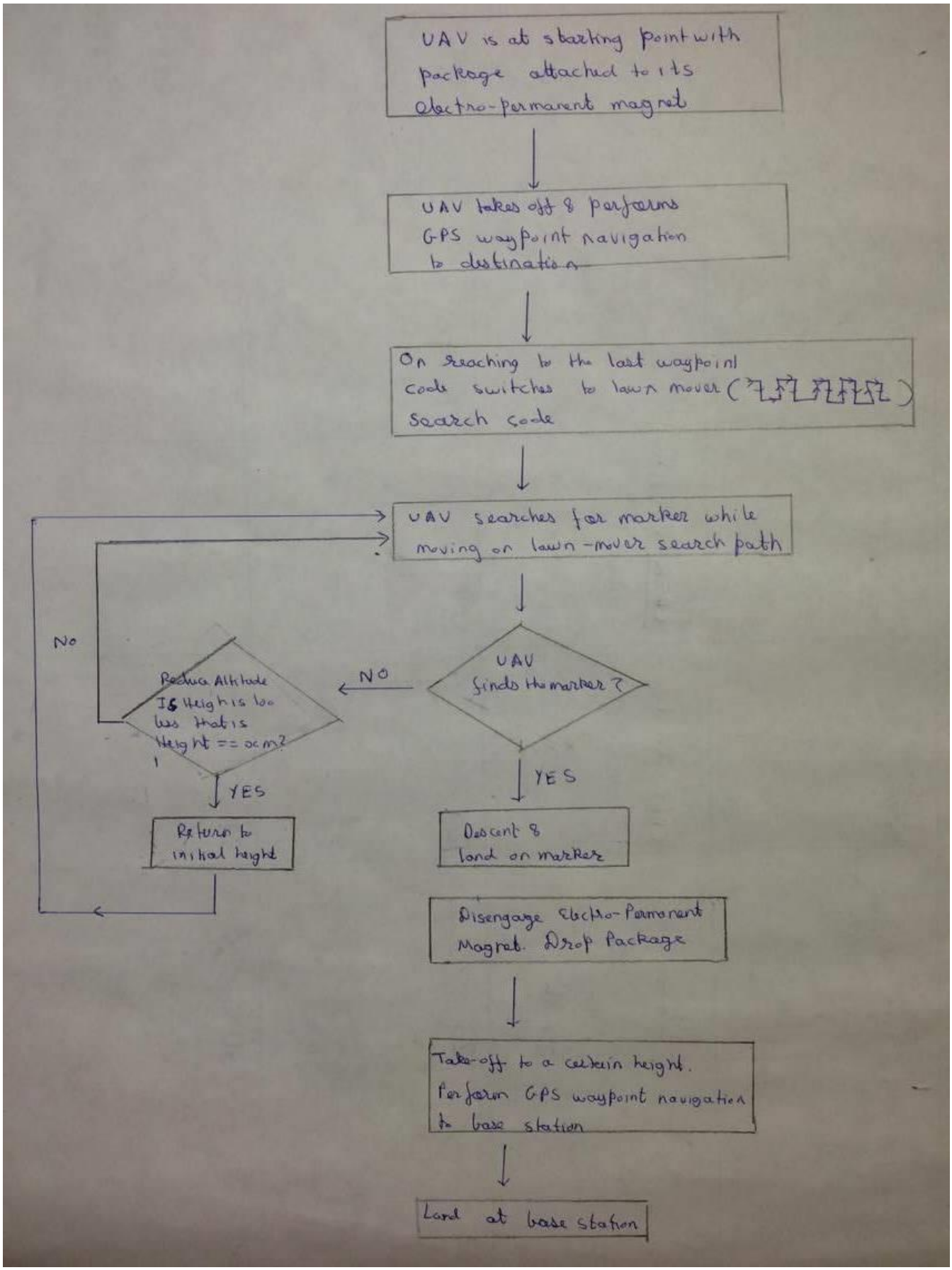

Figure 1 High level plan

2. Reason for shifting to LIDAR from array of Ultrasonic sensors

Below mentioned are the positives that led us to this decision:

- Since there are no aero-dynamic issues in mounting LIDAR on X8+ without altering our system requirements we can substitute one LIDAR for 14 Ultrasonic sensors
- Initially one of the reasons why we avoided using LIDAR was because we though LIDAR would require lot of processing power. However, we realise Odroid doesn't face any issue processing LIDAR data and we hence we can connect LIDAR to Odroid removing that constraint.
- Data from ultrasonic sensors is not consistent. Processing data from 14 sensors can be error prone and difficult to debug. LIDAR helps us get rid of this completely

3. Implementation of Navigation Stack in Simulation

ROS Navigation Stack takes information from odometry and sensor stream and outputs velocity commands to send the robot to its goal position. As a prerequisite navigation stack takes following things as its input:

- Coordinate Frame information using tf
- Sensor_msgs/LaserScan message from LIDAR
- Odometry information using tf and nav_msgs/Odometry messages

As an output the stack gives velocity commands on /cmd_vel topic.

Currently, I have generated dummy odometry data. However, when implementing navigation stack on our UAV we will receive odometry data from Pixhawk through Mav-ros global-position topic.

The below figure shows the coordinate frames. Here odom is the world frame. Base_link is the UAV frame and laser is the LIDAR frame.

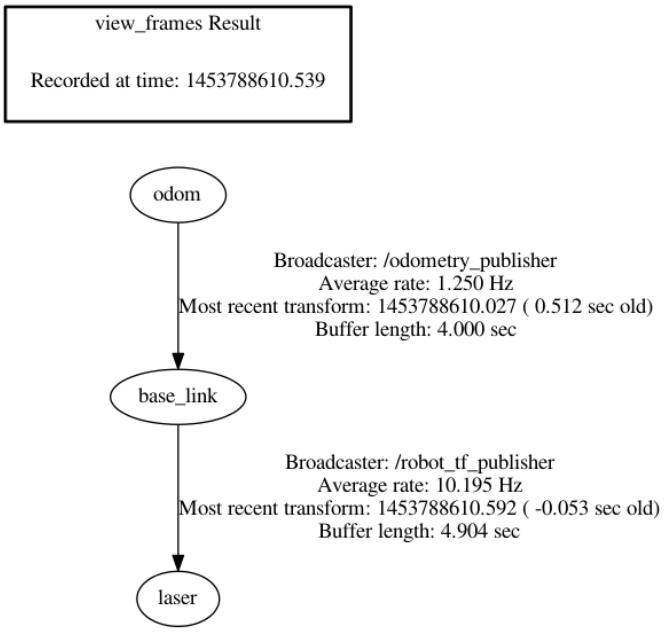

Figure 2 Transform tree

The navigation stack contains the following parts:

- Global costmap
- Global path planner – It uses the global costmap to compute paths ignoring the kinematic and dynamic vehicle constraints. It uses Dijkstra's algorithm to do this.
- Local costmap
- Local path planner – It accounts for the kinematic and dynamic vehicle constraints and generates feasible local trajectories in real time while avoiding obstacles
- Move_base – It implements the state machine

For the local and global costmap generation I have been able to configure the following parameters in the costmap_common_params.yaml, global_costmap_params.yaml & local_costmap_params.yaml files:

- Maximum range of sensor
- Minimum allowable distance between obstacle and uav
- Footprint of uav
- Inflation_radius

Other parameters, minimum and maximum velocities along with their associated accelerations are configured in the base_local_planner_params.yaml file.

The figure below shows the functioning of navigation stack in rviz. The black region are the inflated obstacles as seen by the LIDAR. Goal position is input by using the rviz GUI and the planner plans the path (the path is shown by a green colour line in the figure).

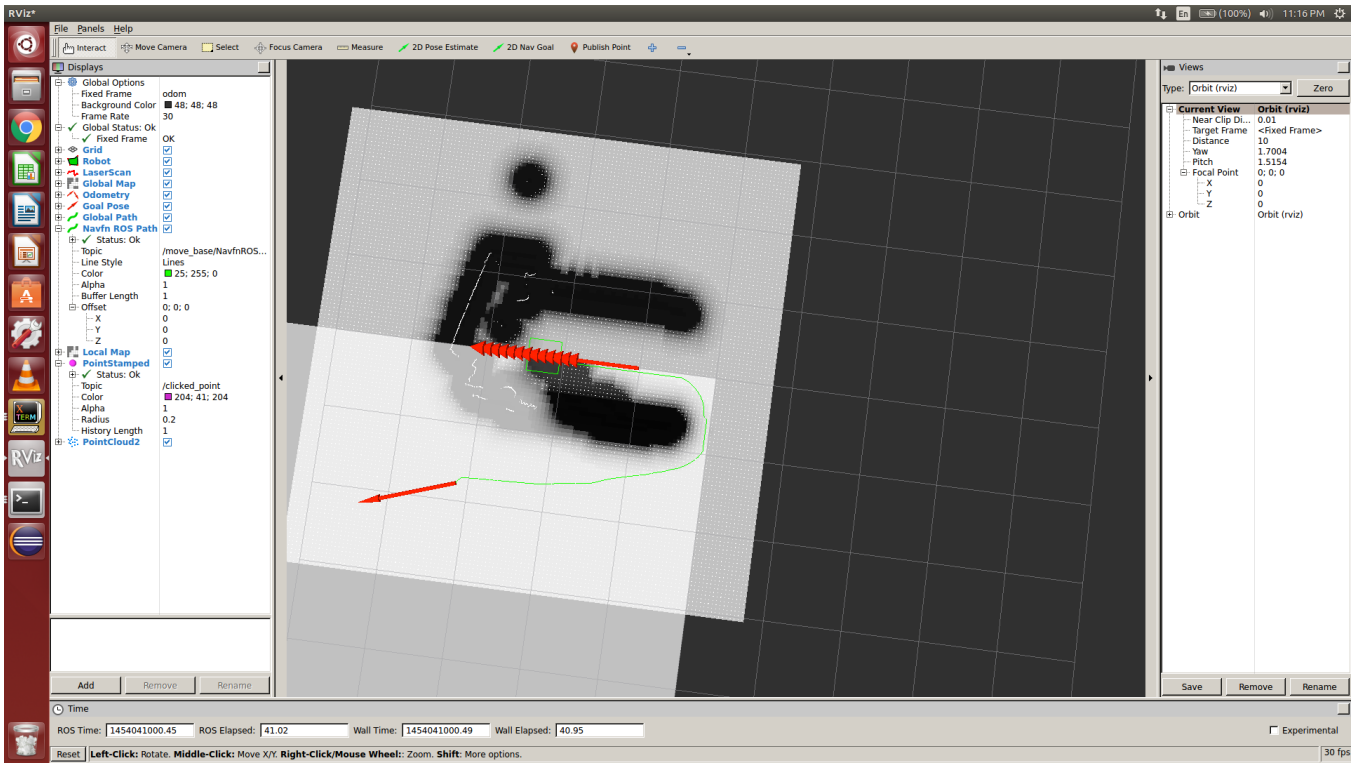

Figure 3 Rviz visualization.

II. Challenges

1. ROS has individual sections on how to generate odometry data and tf transforms. However, there is little information on how all the components including generating odometry, tf transform, laser scan, and other navigation stack specific nodes work together. It took me some time to understand how each component of navigation stack was related to each other.

2. Understanding the tf transform tree was challenging. I got errors because the tf tree wasn't correctly connected. I debugged each error one by one in order to solve this issue.

3. Currently I am trying to pass velocity output of navigation stack to the simulation robot and make it follow the generated path. However, while doing this I am facing error of "Empty header_id" in the Navfn ROS planner. I am taking help from the online ROS community to resolve this issue. However, I haven't figured out the solution yet.

III. Teamwork

Sean got the waypoint navigation working. He performed successful electrical functionality tests of electro-parmanet-magnet. He also fabricated underbelly design to mount camera, optical flow kit, and electro-permanent-magnet. Tushar established communication link between Odroid and Pixhawk.

As promised in the Lab PR, the figure below is a rough sketch of our control implementation. Odroid will have three behaviour codes running on it. "Fly to Destination code" will be the GPS waypoint following

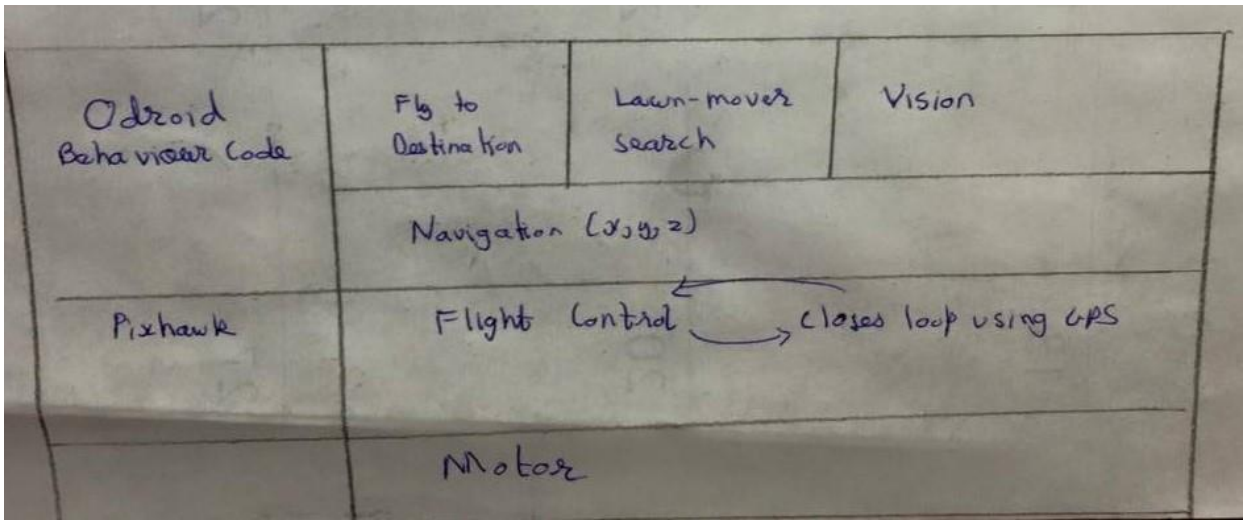

Figure 4 Control implementation

from base station to the vicinity of destination. All the three behaviour codes of Odroid will generate positions which will be passed to Pixhawk through Mavros. Pixhawk will implement the flight control using GPS to close loop.

IV. Future Work

Currently, I am able to generate a path in simulation. However, the UAV isn't following that path in simulation. In order to fix this, I will input the output velocity generated by the navigation stack to the UAV. After this I will simulate the complete lawn-mover search.