Progress Review 12
Project Pegasus

Tushar Agrawal

Team A – Avengers Ultron
Teammates: Pratik Chatrath, Sean Bryan

ILR #11
April 13, 2016

1. Individual Progress

After the last Progress Review, the UAV was capable of autonomously delivering packages starting at a point and landing at the marker. Also, the UAV could navigate using command velocities from the navigation stack, but the process had many issues.

A. Obstacle-less Package Delivery Process

Improved Landing process
Until now, landing was done by progressively reducing z position in the setpoints to the UAV. This was sometimes ineffective as the initial Z values were not very robust, as different sensors were used at different altitudes and hence the offset would change between landing. I switched to using the autonomous landing mode provided. It would maintain the UAVs X and Y coordinates while descending on the point. Also, I discovered that the pixhawk itself would detect whether or not the UAV is landed using multiple sensors (like changes in position and velocities, lidar lite, etc). I used this as the appropriate transition condition into the next state in the behaviour. The results were much more robust. The UAV lands appropriately and touches down on the ground before moving on to further actions. This can be seen in the demo of the entire system (Video 1: https://www.youtube.com/watch?v=-XWTWJ8rYi4)

Return back to truck and land
Based on the application, after delivery, the UAV should takeoff again and to back to the predefined position of the delivery truck and again find and land on the marker. This functionality was added using the same basic states used to send the UAV to the house for delivery. Video 1 (https://www.youtube.com/watch?v=-XWTWJ8rYi4) shows the delivery process, when the house coordinates were used as the final truck coordinates (as a proof of concept).

Speed up AprilTag Detection
Based on the work in the previous semester, the AprilTag detection was made robust and fast using Lucas Kanade tracking. Unfortunately, when a ros wrapper was built around the code, the detection speeds reduced to 40% of earlier. The marker detection ran at 0.5 fps on the Odroid. This seemed surprising, but we continued working with it as there were more pressing issues. Finally, we have started seeing issues where the marker is not detected sometimes even at low speeds after an entire sweep of the area. Hence, I came back to this problem and tried to determine what caused the reduction in the speed. Based on my tests, it seems that the catkin build environment was in some way adding extra dependencies that were slowing down the system. I updated the AprilTag CmakeLists.txt with roscpp and related required packages of ros and built it without the catkin workspace. Now, the Speeds have come back to what they orginally were (30fps) even when the apriltag pose is continuously published over ROS. This has been mainly tested offboard as of now, but will soon be incorporated into the UAV and an improved video will be shot.

Miscellaneous
I added a new “Spiral Out” pattern to search for the marker. This pattern is depicted in Image 1. It might be more robust as it would prefer searching closer to the house before searching further from it. Initial tests seem promising, but more tests need to be

conducted before they could be proven to be definitely better.

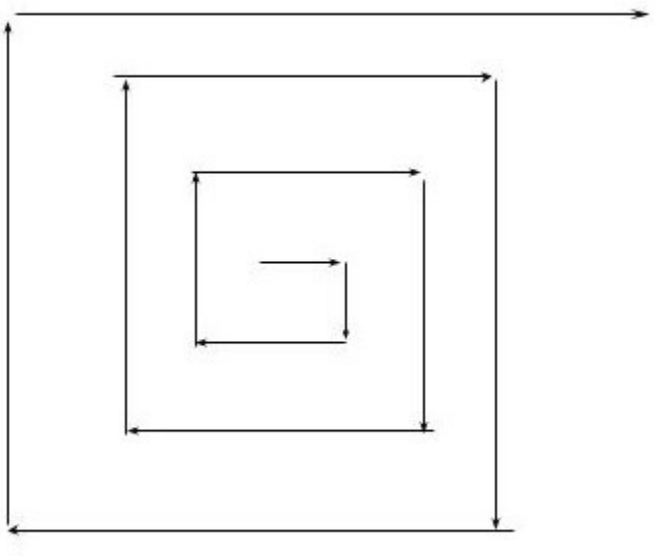

Image 1: Spiral Pattern. Center to Outwards. House is at the center

I also added the exception case states to the behavior flow. As illustrated in image 2, in the case where no marker is found, the UAV shall decide on an appropriate safe location, travel to it and land (even without marker). If the marker is not found when delivery is to be made, the UAV assumes the customer is not ready to receive the package and travels to the new position of the truck and lands there. In case where the marker is not found at the final truck location, the UAV goes to a predefined “safe” location and lands. The framework has been made to update this logic iteratively during testing.

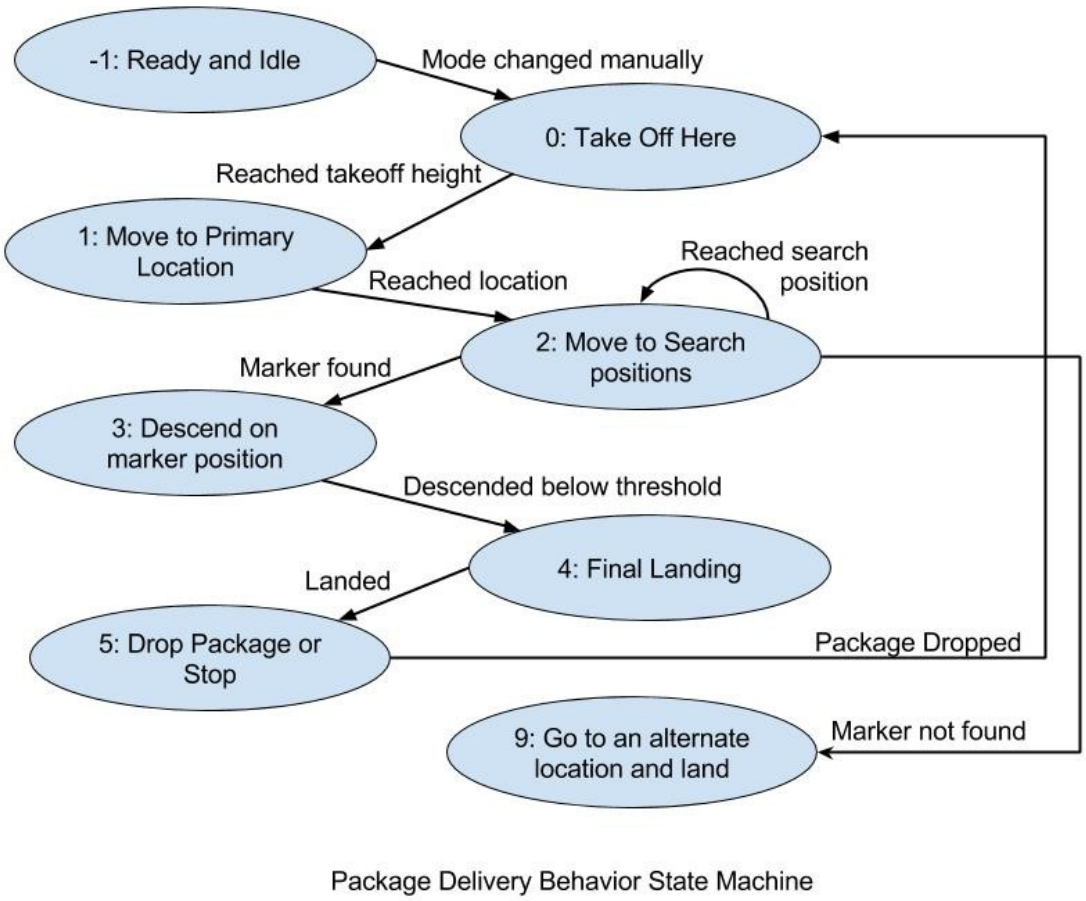

Image 2: New state machine for package delivery. Note: state 9 has been added as the exception state when marker is not found

B. Package Delivery while Avoiding Obstacles

Pratik and I found multiple issues while working on obstacle detection and avoidance in the last few weeks.

Hokuyo URG-04LX-UG01 is meant for Indoor use

Probably the biggest issue we discovered was with the Lidar sensor that we were using. Based on our recoded data, we discovered that the LIDAR gave very noisy data in the day during our tests (Image 3a). On researching further, we discovered that this Lidar was meant for indoor use only. Since then, we have been trying to find an appropriate outdoor Lidar to replace this and trying to set up appropriate filters which may help us in reducing the noise in sunlight. We set up the Median and Range Filters which cleaned up the data from the Lidar to a good extent (Image 3b). Further tests were not possible as the Lidar was refusing to respond in too much sunlight. A few hours before submitting the ILR, we were finally able to source the UTM-30LX Lidar and set it up on the UAV. Daylight testing would be done tomorrow.

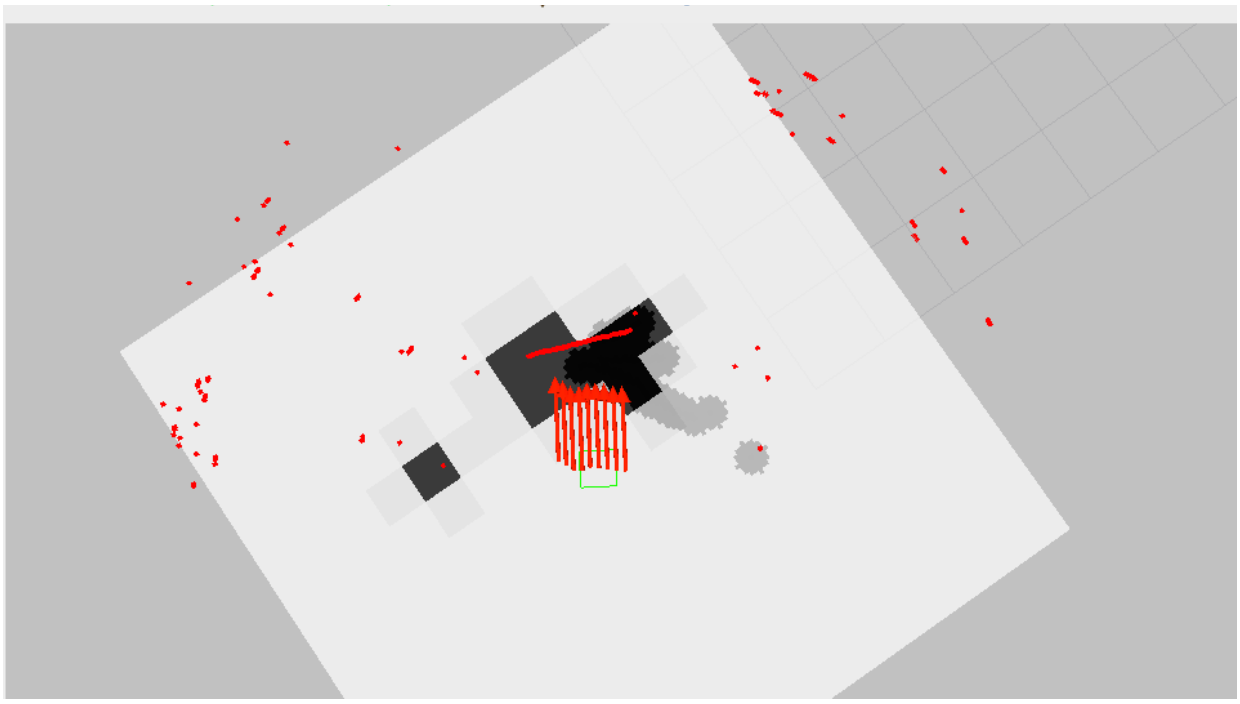

Image 3a: Noisy data obtained from the Lidar in the sunlight

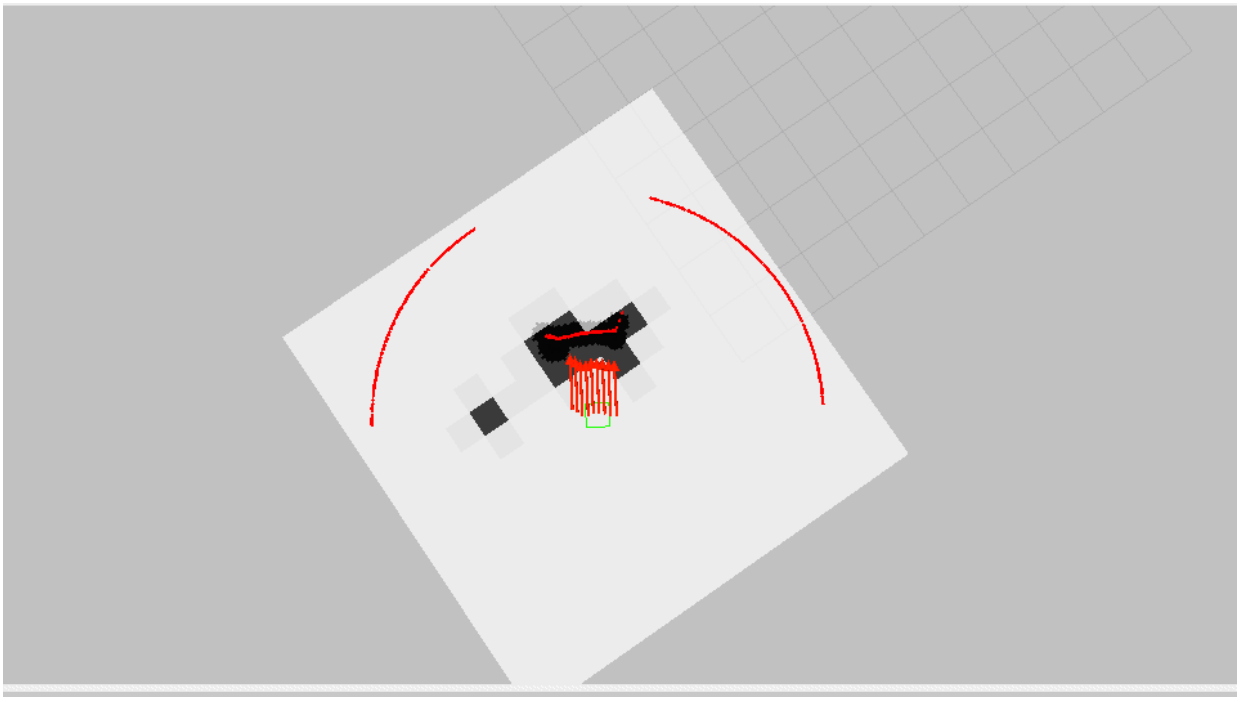

Image 3b: Same frame as Image 3a after filtering noise using median and range filters. The semi circle is explained in the next sention

Navigation Stack related Filter Tweaks

Another issue we saw while testing was due to the memoization of obstacles by the navigation stack. When the navigation stack would see an obstacle in front of it it would add it to the costmap for planning. But, this obstacle would not be cleared even if the

next scan of the same space came back clear. Pratik uncovered that the “raytrace” parameter would help us in clearing obstacles, but it did not work as it was supposed to. After further testing, we realised that if the laser scan came back as infinite (no obstacle until maximum range of Lidar), the obstacles would not be cleared. The obstacles were being cleared only till the current laser obstacle. Using that, I made a workaround with the range filter, to convert all infinites to a fixed value a little less than the maximum (4 meters in our original Lidar). As seen in Image 4 this gives a circular scan until which all non present obstacles are cleared.

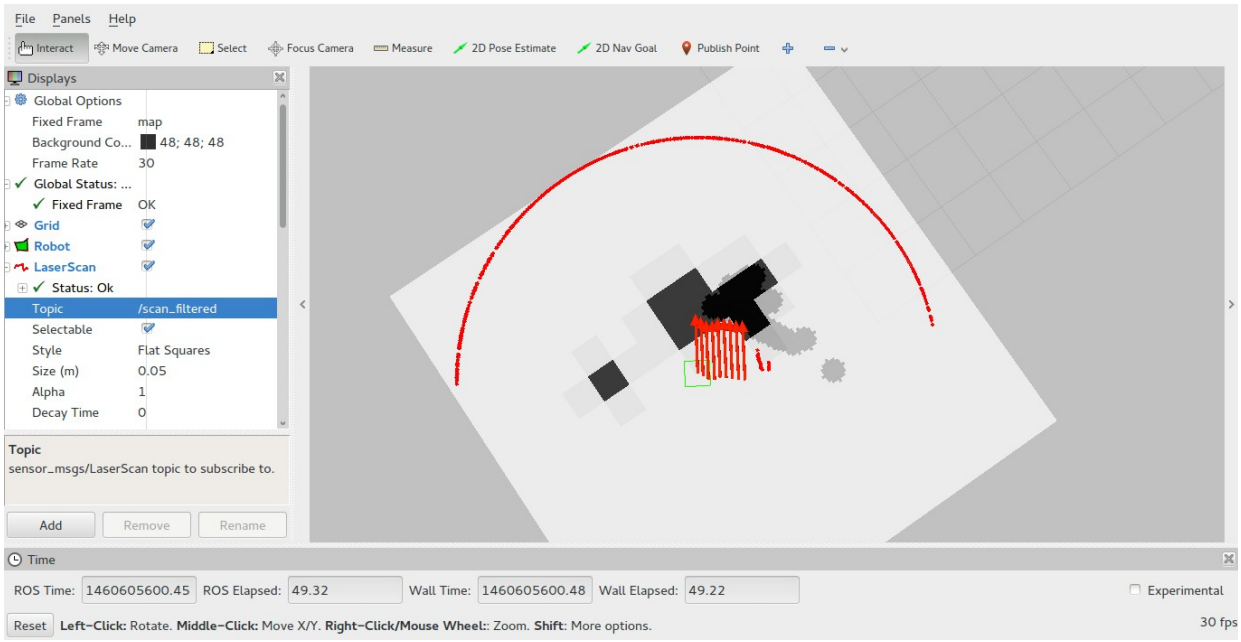

Image 4: Circular max range workaround for clearing costmap

Odometry 3D to 2D conversion

As navigation stack is used only in 2D to plan paths, the odometry had to be corrected for that. Orignially the 3D data was passed as 2D to the navigation stack assuming it would use only those parts, but we realised that the UAV was actually making transforms in 3D and updating the costmaps only when the obstacles position was at z=0. I made a quick ros node to convert force the z, roll and pitch of the UAV's odometry to zero. This resulted in much better path planning and updation of obstacles from the laser.

Height Control PID

As discussed before, while flyng the UAV in velocity control state, providing only x, y velocities results in loss of control in the Z axis, which leads to the UAV descending unnecessarily and ruining tests. I made a makeshift PID control loop which would try to hold the z position of the UAV while the UAV moved using command velocities (refer Video 3 https://youtu.be/-WRN0iiLfrw). Sean worked on this further to clean up and tune.

Behavior code

A version of the behavior control code was written as the exact state machine in the obstacle-less package delivery case, but for controlling of command velocities and goal

positions to the Navigation stack.

C. Operations

I made a few tweaks in the current workflow to help speed up testing and debugging. We set up a recording framework using rosbag. This helped in recording data during flights and replaying later. I also set up “tmux” scripts which enabled me to open required terminal windows automatically for fast setup during testing. Image 5 shows an example tmux window. I also backed up the latest image of the Odroid onto the backup odroid to protect from failure cases.

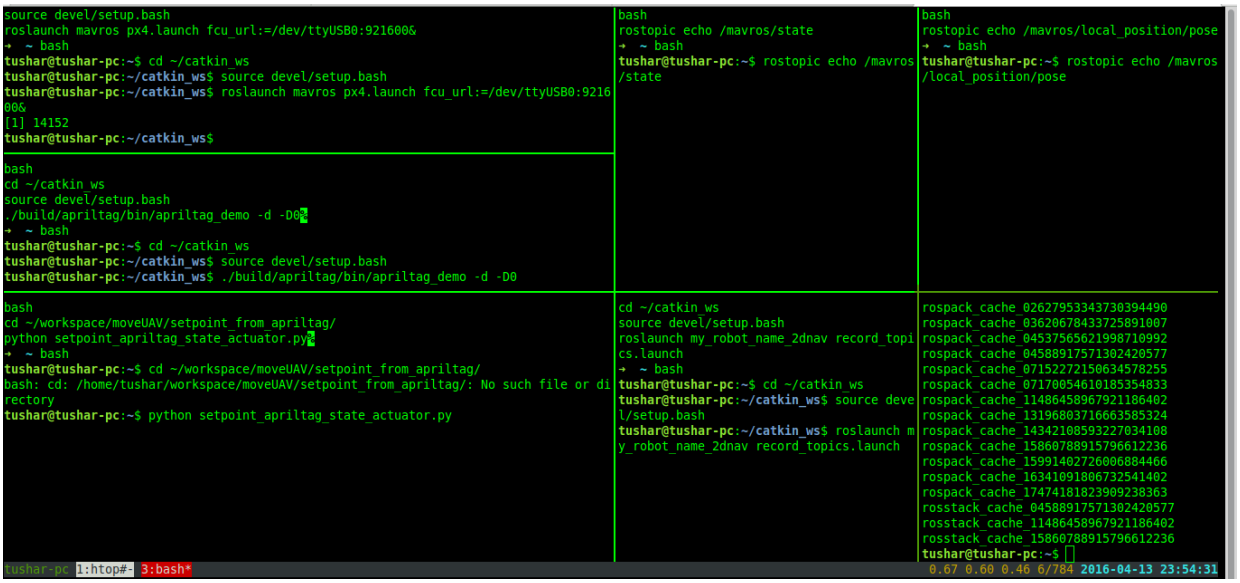

Image 5: Tmux window opened using a tmux sxript for obstacle-less package delivery process. This has panes for running the process on the left and monitoring state data on the right.

2. Challenges

Many challenges were faced during this process:

- Most importantly, the Lidar used earlier was discovered to be only meant for indoor use. We tried using multiple filters and borrowing appropriate outdoor Lidar to resolve this.
- Marker detection was slow and not robust. I worked on improving the speed. Field tests are still pending.
- Height control challenges were dealt with by using a simple PID controller on the altitude of the UAV.

Current challenges include the setup of the new Lidar (UTM-30LX) and finishing up obstacle avoidance using that.

3. Teamwork

As we are approaching the final demos, Pratik and I are worked on solving the obstacle avoidance issues in the system. This included researching filters, implementing and debugging them. Setting up appropriate tests and analysing them later.

Sean set up his PX4 simulator and started working on the initial z control PID to fine tune it.

As a group, we did multiple tests together and tried to determine the extent of our issues and best ways to fix them. Sean was also proactive in contacting many faculty and student bodies requesting for a spare Lidar.

4. Future Work

The most important priority is to set up the obstacle detection and avoidance system using the correct Lidar as soon as possible and testing the full pipeline multiple times for ironing out any small possible issues.

We will also work on setting up an appropriate test environment for testing and final demonstration to show the working package delivery system.