Harikrishnan Suresh

Team C: Fly Sense
Teammates: Shivang Baveja, Nicholas Crispie, Joao Fonseca, Sai Nihar Tadichetty
ILR09
March 22, 2018

1. Individual Progress

1.1. Obstacle Coloring and Sound Warnings

My major contribution was the ROS implementation of the obstacle coloring in Bird's Eye View. Based on the algorithm developed by Joao in MATLAB and ported by me to C++, it is now integrated with the Bird's Eye View. The obstacles now appear in Red, Yellow and Green instead of light blue color that was present in our FVE version. The basic idea behind the algorithm is to predict the addressable flight envelope based on the current dynamics and maximum possible pilot input within the decided time limits. The point cloud is then segmented into 3 levels of time to impact and colored accordingly. Through our preprocessing step, we are essentially eliminating a major portion of the waste point cloud, and only the valid ones within the dynamic window of our quadcopter are included in this coloring function. Our initial tests have shown this algorithm to work real time, which essentially generates a smooth rendering of the Bird's Eye View.

It was seen that a lot of ground points were appearing on the Bird's Eye View image. It makes the image very cluttered and filtering out these ground points might be required. The '*passthrough filter*' in PCL library will be used to remove the ground points based on Eqn.1. Since the Z value is w.r.t. to the Lidar origin, the minimum value must be changed real time using the dynamic reconfigure tool in ROS.

$$if\ altitude \leq 1m,\ \ Zmin = 0.3m$$
$$else\ Zmin = \ -(altitude) + 0.3m$$

However, these ground points give the pilot a sense of their surroundings. Due to this confusion, images of Bird's Eye View with and without the ground points will be presented tomorrow during our meeting at NEA and the final decision will be taken based on the pilot's feedback.

The code for sound warnings is also implemented in ROS with a new set of equations based on the quadcopter. Here, the time to impact is calculated based on what the pilot is actually inputting at the moment thereby eliminating unnecessary alarms. Also, the plan is to generate only few relevant points from the filtered point cloud using '*velodyne height map*' and use for the time to impact calculation. I am also planning to compare this with a clustering algorithm for the point cloud, and finally choose the algorithm that works better. While the Bird's Eye View requires dense representation of point cloud to render beautiful visuals, only the critical points are required for calculation of sound warnings. Due to a few challenges (highlighted in section 2), we were unable to integrate sound warnings code with the software stack.

1.2. Modified Bird's Eye View

Based on the pilot workshop at NEA, I worked on a few changes to the Bird's Eye View. The objective is to present a few designs to the pilot during tomorrow's meeting and finalize the Bird's Eye View. A major feedback we received was to change the icon for the quadcopter in the Bird's

Eye View to another shape that conveys the heading of the vehicle. The new design options for the same in shown in Figure 2 and 3.

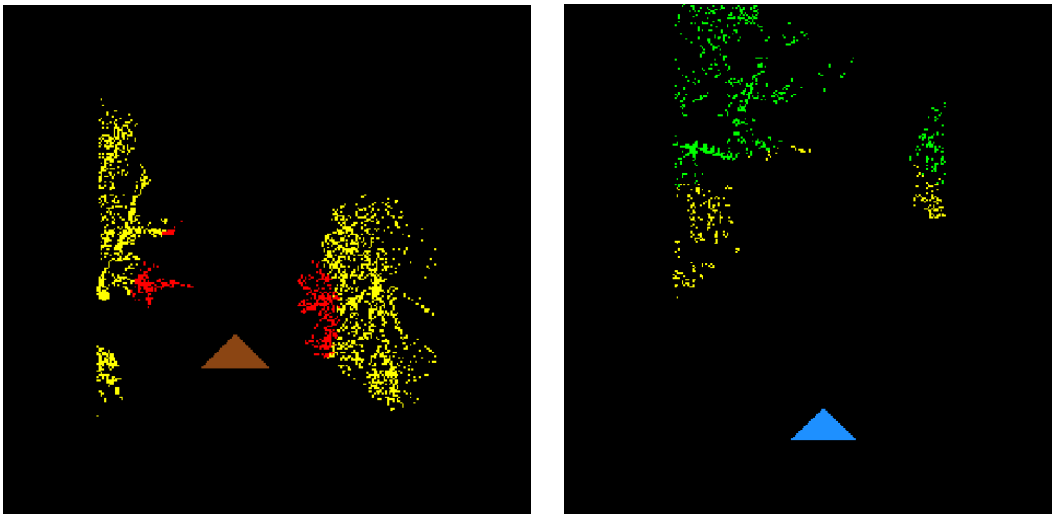

Figure 1: Color options for vehicle icon (blue or brown)

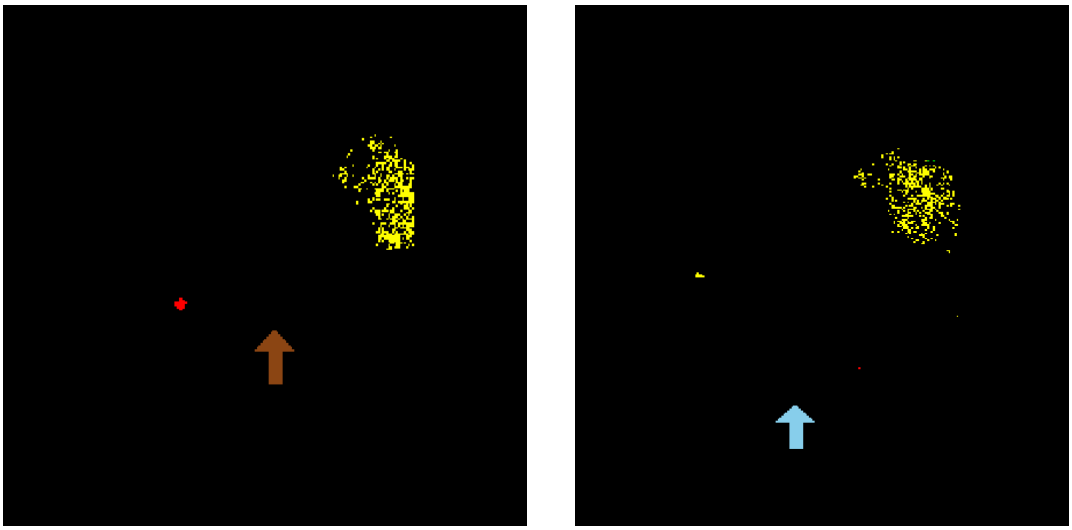

Figure 2: Alternative shape for vehicle icon (triangle or arrow)

1.3. Test with First Person View Camera

The hardware setup for the quadcopter is now complete with the addition of First Person View (FPV) camera. The new interface for the Epson with FPV and Bird's Eye View combined is now ready. The next step is to include the HUD as well. We did our first test at Schenley Park with the FPV camera and a snapshot of the new visual is shown in Figure 3.

Figure 3: Version 1 of new AR interface

2. Challenges faced

- Another laptop damage issue happened in our team, and this time I was the unlucky one. I had to spend an entire day before the progress review to fresh install Ubuntu and set up the code again. This was a major reason for not being able to integrate the sound warnings code.
- While developing the obstacle avoidance algorithms, Joao found a few mistakes in his sound warnings code and had to do several rounds of correction. This also delayed the integration of the sound warnings code.
- I was really stuck at one point where a code that was generating colored obstacles in my laptop produced blank images on the Jetson. I was unable to rectify the problem, until Shivang discovered that the C++ 'float' variable in the Jetson does not have enough bit value and had to be replaced by "double".

3. Team work

Team Member	Contribution
Shivang Baveja	• Lead the first flight test with FPV Camera • Looked into DJI flight controller SDK to implement the obstacle avoidance algorithm. Facing a setback due to version upgrade of the SDK

	• Helped resolve the code failure issue in Jetson, and also fine tune the algorithm to get 3 colors of obstacles
Nick Crispie	• Completed the hardware setup including integration of FPV camera • First version of Gazebo simulation to test obstacle avoidance algorithm, including generation of point clouds for virtual obstacles
Joao Fonseca	• Assisted in completing ROS implementation of obstacle coloring algorithm • Modified the sound warnings algorithm • Completed obstacle avoidance algorithm in MATLAB
Nihar Tadichetty	• Completed code to capture the FPV video • Finished the first version of the new AR interface with FPV and Bird's Eye View • Assisted in FPV flight test

4. Tasks

Team goals

- Complete integration of sound warnings code with the software stack
- Gather feedback about the Bird's Eye View options and finalize the designs
- Conduct more flight tests to improve the algorithms
- Test the obstacle avoidance code in simulation

My goals

- Complete obstacle clustering and integrate with Bird's Eye View
- Compare the performance of clustering with '*velodyne height map*' and integrate with sound warnings code
- Complete integration of sound warnings code with the software stack
- Finalize Bird's Eye View design