FlySense

Shivang Baveja

Team C: FlySense

Teammates: Nihar Tadichetty, Joao Fonseca, Harikrishnan Suresh, Nicholas Crispie

ILR 10

April 5, 2018

Individual Progress

For this progress review, I was responsible for developing an interface to override pilot commands for obstacle avoidance and setup flight simulation to test this feature. I also worked on improving the bird's eye view image generation and improve the performance of this code. Apart from this, I assisted Nick, Hari and Nihar in software integration. Following sections describe these in detail:

DJI Pilot Override Interface:

I implemented a flight control interface as a separate flight mode. The goal of this flight mode is to allow us to test the obstacle avoidance functionality.

So far we have been flying the quadcopter in position control mode (P-mode) using the remote controller. In this mode the pilot stick commands are converted to speed targets and if there's no stick command aircraft holds the position. DJI allows custom functionality on Function mode (F-mode) which can be selected via the DJI remote controller. Our custom flight mode gets activated when F-mode is selected. We are mimicking DJI Position control mode in a way that the stick commands from the remote controllers are linearly converted to velocity commands.

This code was setup in hardware-in-loop simulation using DJI flight simulation software suite. Figure 1 shows DJI flight simulation software with the DJI-M100.

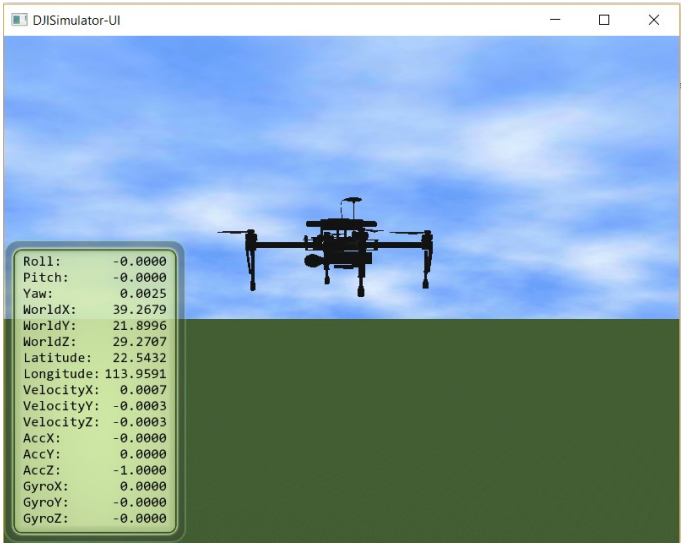

Figure 1: DJI Flight Simulation Software

Obstacle avoidance algorithm:
Johnny wrote the algorithm for Obstacle avoidance and developed a matlab code for it. I implemented this algorithm in our onboard software and tested it with DJI flight simulation software. There were a few frame convention inconsistencies wich had to be resolved to make the code work.

A fixed obstacle was hard-coded to be able to test the algorithm. It was observed that when we move the quad towards the obstacle, the velocity commands generated cause the quad to brake but also cause oscillations. The reason for these oscillations was found to be the dynamics model parameters. Our dynamics model is based on real aircraft which has more drag compared what is simulated in the simulation software. To make the code work, we collected a small bag file with dji flight simulation working and ran regression to tune the parameters for the dynamics model. We haven't been able to test enough after changing the parameters but we are planning to do it soon.

Gazebo environment simulation:
I helped Nick to setup the Gazebo simulation environment in our onboard computer Jetson-TX2. We are planning to be able to run full simulation with flight dynamics simulated in DJI flight simulator and velodyne, camera and obstacles simulated in Gazebo. We are almost done with this integration and haven't been able to test it yet.

Coloring code integration and improvement:
I helped Hari in implementing the coloring node which generates the colored bird's eye view. We faced a few issues related to numerical instability when using exponential function. After fixing those we were able to get the code working but there were some other issues. One of the biggest issues was the rate at which image was getting generated. We were earlier publishing an image whenever we were receiving point cloud frame, which comes at 1 Hz. The image generation code was decoupled from the point cloud callback function to ensure we publish at a fixed rate of 10Hz.

There was stray points seen in the image due to noise in point cloud data. I configured a statistical outlier removal filter from PCL library to reduce noise. This filter uses k-means to remove stray points. The value of k was tuned using the collected flight data and was set to 4.

I also did some other modifications to make the objects appear more dense. This was done by treating each point int the filtered point cloud as 10 pixels. This makes it easier to see them when using the epson AR headset. Figure 2 shows screenshot of the final bird's eye view image with arrow showing the quadcopter and is drawn upto scale.

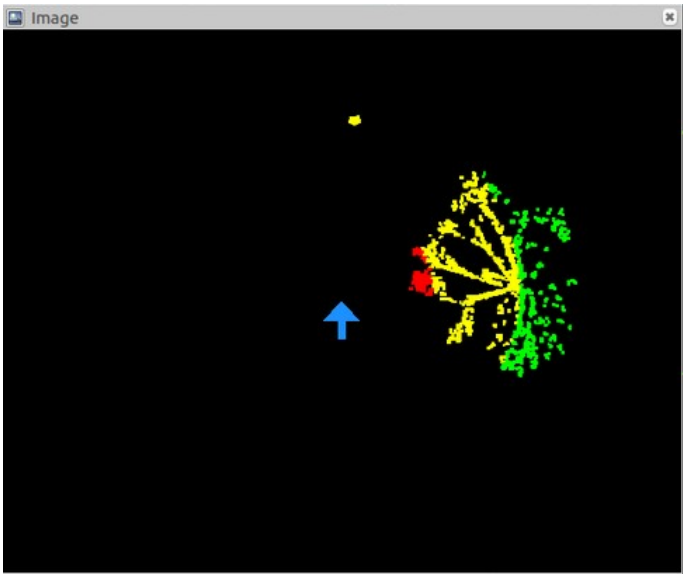

Figure 2: Colored Bird's eye view with blue arrow showing aircraft

Software Integration and Flight Test

We integrated all the software and tested it with Epson BT300 AR headset in lab. The system was then flight tested at Flagstaff hill. In the first couple of flights I was piloting the aircraft without the AR headset and Nick was responsible to make sure that the video is robust enough to be able to fly with it. In the third flight I flew the system with just the AR headset without directly looking at the aircraft. This was quite challenging as I have never flown an RC aircraft in First Person View (FPV) mode. I was not confident enough with my skills to take the aircraft closer to the obstacles this time. I hope to do it next time. I am planning to practice flying FPV in simulation before flying our system.

Figure 3 shows FPV view from the quad.

Figure 3: FPV view with colored bird's eye view

Challenges faced

- The biggest challenge was to develop a robust flight control mode which can be used for obstacle avoidance. Testing the code in simulation helped alot in this regard.
- I realised today how hard flying FPV can be for the first time. It was especially difficult as aircraft's altitude was oscillating due to wind.

Teamwork

Name	Contribution
Nihar Tadichetty	• Improve FPV+ Bird's eye view merging code • Improve Sound warning code in Epson • Set-up android tablet as a backup option to fly with our user interface code.
Joao Fonseca Reis	• Help Hari and me in integrating alogorithms related to sound warnings, bird's eye view and obstacle avoidance.
Harikrishnan Suresh	• Implement coloring node • Implement sound warning code
Nicholas Crispie	• Setting up Gazebo environment to simulate velodyne sensor on a quadcopter flying near obstacles.

	• Project management and procurement • System integration and flight testing

Plans

Goals for Next Progress review:

- Complete the emergency braking for obstacle avoidance and test it in simulation and in flight
- Improve and test sound warning code in flight
- Conduct more flight tests to improve the algorithms
- Complete enhancements to the Bird’s Eye view with blinking of obstacles
- Complete Gazebo simulator and integrate it with the DJI flight simulator

My tasks:

- More flight testing to get used to flying in FPV mode and collect more flight data.
- Complete the emergency braking for obstacle avoidance and test it in simulation and in flight
- Complete Gazebo simulator and integrate it with the DJI flight simulator