Fly Sense

Shivang Baveja

Team C: FlySense

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

ILR 03

October 27, 2017

Individual Progress

1. In the past week, I helped in defining our fall validation experiment a little better. We decided that we will demonstrate the functionality by two experiments. These are explained in Figure 1.
   a. Priority Objective: Simulate helicopter interface by sending data to the onboard computer at the same rate as it would be coming in the actual system. The onboard computer generates 2d map and warnings and sends it to AR controller to render.
   b. Secondary objective: The point cloud data from Velodyne VLP16 and pose estimate from GPS/INS is sent to the onboard computer, instead of flight data set. The sensors will be mounted on the chair with a person sitting on the chair with AR headset on. The person should be able to see the bird's eye view of surrounding obstacles in 20m region.

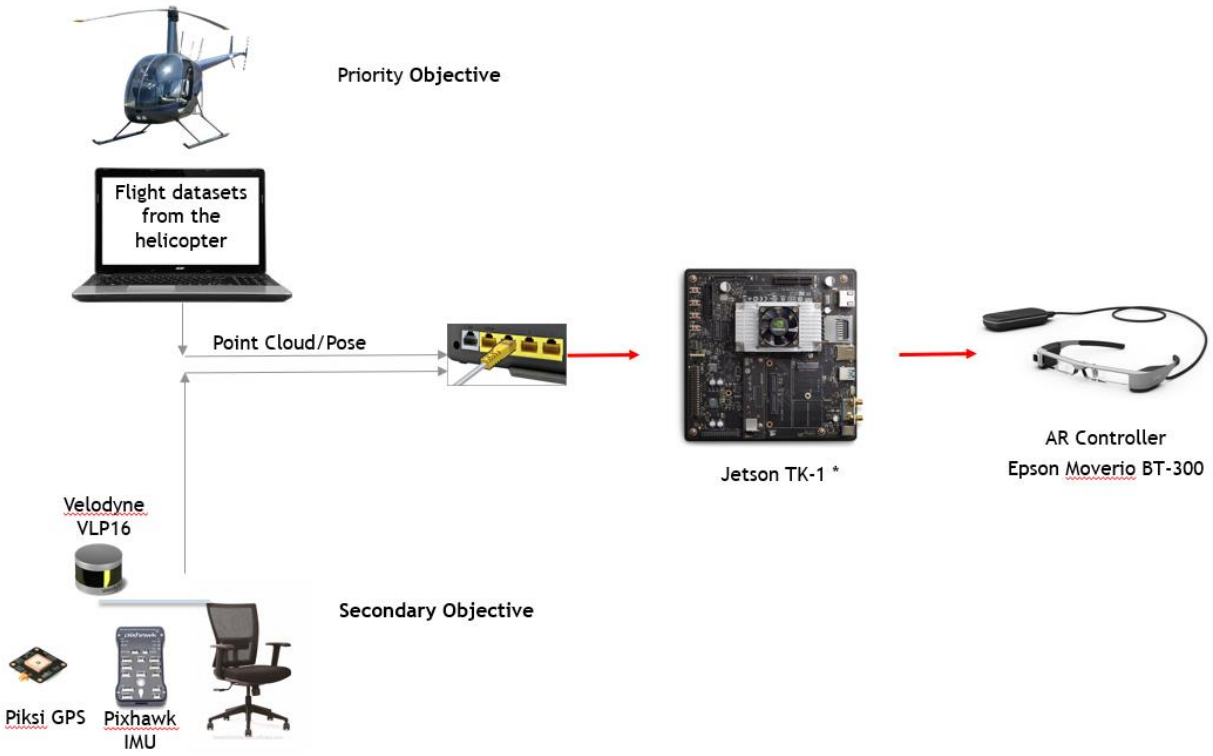

Figure 1

2. Completed setup of Jetson-TK1 with ROS and our current software for obstacle map generation:
   I had worked before with single board computers in the past, like Beaglebone, Raspberry Pi. So, I knew the general steps which are required to set up. I started with the wiki page for NVIDIA Jetson TK-1, which helped me understand all its hardware features. Here is the link for the same:
   https://elinux.org/Jetson_TK1

Then I found the “getting started” guide which detailed the steps to load a pre-built Ubuntu image, and a sample file system onto the system’s internal eMMC. The images and sample file system can be downloaded from this link:
https://developer.nvidia.com/embedded/linux-tegra

A detailed guide to set up the system can be found on this link:
https://developer.nvidia.com/embedded/linux-tegra

There is one issue with the guide on the above link when we do sudo apt-get install update, the graphics driver gets over-written. This prevents the unity desktop GUI to not get loaded. The solution for this is to issue the following command before sudo apt-get update.
sudo apt-mark hold xserver-xorg-core

Figure 2 shows Ubuntu unity desktop running on Jetson-TK1.

Figure 2 Right monitor is connected to Jetson

The next step was to set up a remote access and get internet access on Jetson-TK1. This can be done by following the guide:
https://elinux.org/Jetson/Remote_Access

The next step was to get ROS running for which I followed the following link:
http://wiki.ros.org/indigo/Installation/UbuntuARM
RVIZ gives segmentation fault with the default settings.
The GTK_IM_MODULE environment variable needs to be upset it in your ~/.bashrc to make it work.

I then fetched the octomap git repository to Jetson, built it on Jetson.
https://github.com/OctoMap/octomap_mapping

We were earlier using multiple commands to play the bag file with the point cloud data, start octomap, rviz nodes. So to simplify this process I wrote a simple launch file which takes in the name of the bag file as an argument and starts all the nodes pre-configured to generate and show the MarkedArray(3d map) and 2d map from the octomap.

Figure 3 shows octo-map running on Jetson.

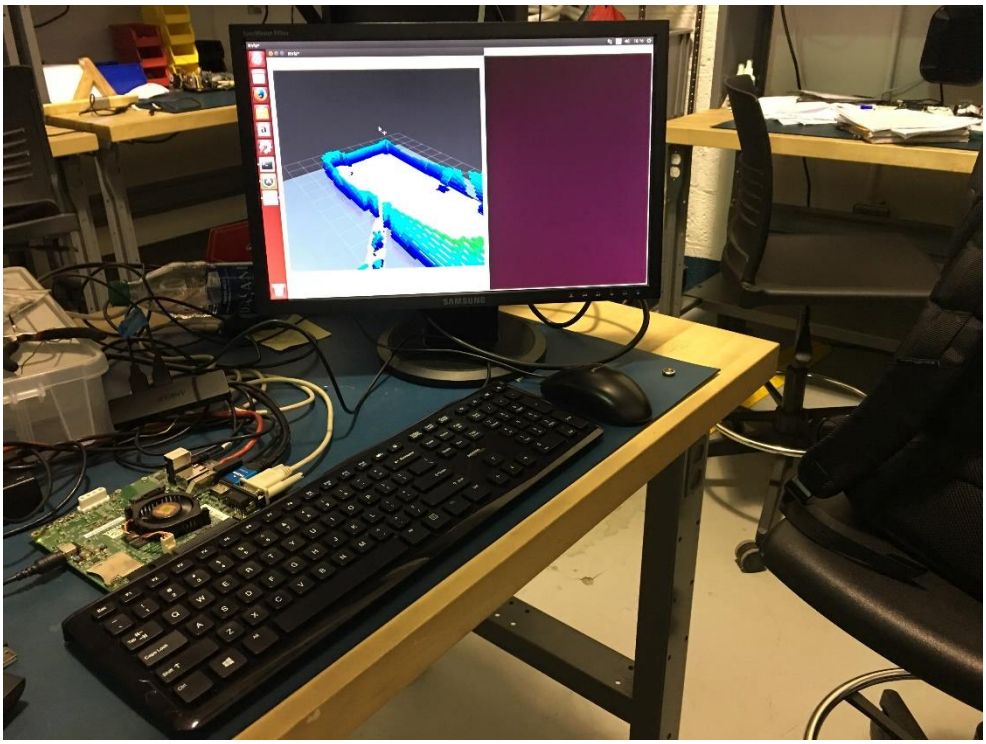

Figure 3

3. Communication between Epson Moverio BT-300 and Jetson TK-1 via USB.
   As Epson is an android device, I wanted to understand how we can setup ROS on it and communicate via USB. A little research showed that we can use ROS Java with the Android ADK. Also, we can setup the USB port on Epson as "Internet pass-through" which allows us to share computer's network with the Android device over USB. This allows ROS nodes running on the Android device like any other ROS.

Challenges faced

- I spent a couple of hours struggling with the bug on the NIVIDIA's Tegra-ubuntu image which was messing up the graphics drivers and was preventing the unity desktop from loading.

Teamwork

In the past week, we worked on finding the path of least resistance to our end goal of getting the system in Near Earth Autonomy helicopter. This was required to minimize dependencies among team members and on the sponsor (NEA).

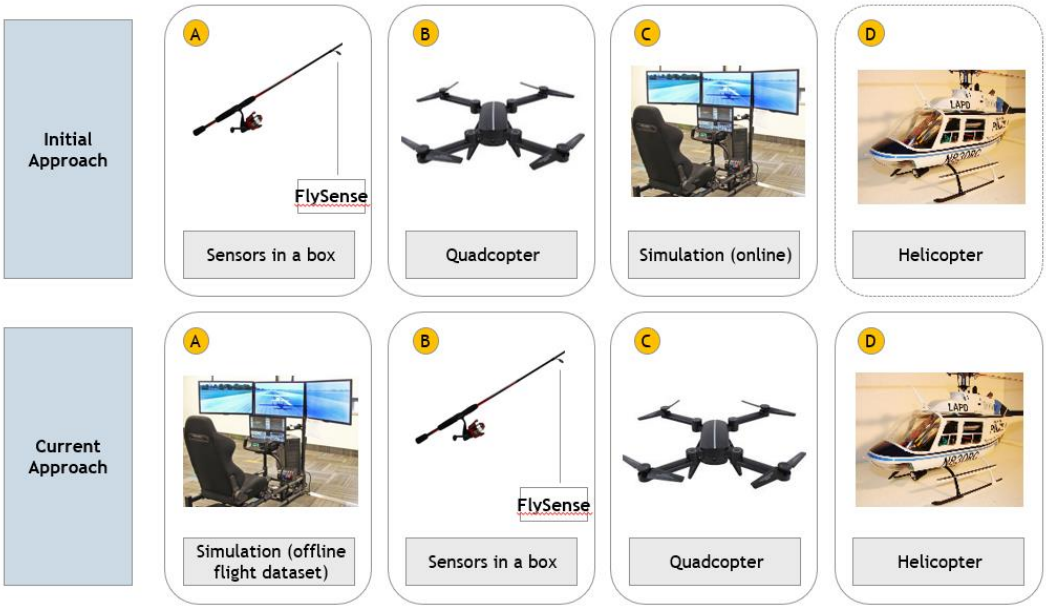

Figure 4

Figure 4 shows our current approach. We are aiming to simulation with flight data sets and "sensors in a box" completed in Fall. This directly translates to working with the quadcopter in the next semester.We developed work breakdown structure to identify all the primary work packages and developed a schedule for our fall semester with sufficient time for testing and buffer time.

These were the contributions from each of the team members:

Name	Contribution
Nihar Tadichetty	Epson-Jetson interface specification, Developed a basic android app for Epson
Joao Fonseca Reis	Epson-Jetson interface specification, Voice command recognition feasibility, audio generation library on android
Harikrishnan Suresh	Studied Octree Octomap research paper to understand the parameters, Fixed latency issues in map building with octo-map by fine-tuning the parameters, started working with the grid-map package.
Nicholas Crispie	Conceptual design and Schematic of the Power distribution board, Project management (work-packages, schedule)

Plans

Jetson TK-1 will be receiving point cloud data and the pose estimate, processing it to generate the bird's eye view of the obstacles surrounding the helicopter and send it to the AR-controller which will render it. I have divided it into following high-level work packages:

a. ROS interface between Epson and Jetson over USB
b. Implement a model for flight envelope of the helicopter (developed by Joao) and create a mask which can be used to filter the unnecessary lidar data.
c. Run 2d map generation algorithms (tested and tuned by Hari)
d. Label the grids into (Red, Yellow and Green) which have obstacles according to helicopter pose.
e. Convert the 2D map to the custom defined packet, which is sent to the AR controller.

For the next week,

1. Send IMU data to Epson over ROS for the heads-up display
2. Order stuff for quad-copter: IMU/GPS
3. flight envelope calculations v1