Mapping of Remotely Received Data from Small Unmanned Aerial Vehicles

Justin Gray, Aerospace Engineering
Mentor: Dr. Subodh Bhandari
Kellogg Honors College Capstone Project

Introduction/Background

Unmanned Aerial Vehicles (UAV) have become exceedingly popular and easy to use in recent years. A few examples, which can be seen in Figure 1, of some hobbyist UAV platforms are small battery powered fixed wing aircraft and quad/hex/octorotor copters. In order to monitor and control UAVs autonomously, open source programs have been made to provide a simple and easy to use ground control station (GCS) for anyone who may want to explore UAVs and what they have to offer. One of the most popular programs that people tend to use is Mission Planner, an open source GCS program that can run on Windows, Apple OS and Linux. Mission Planner makes setting up a UAV very simple and provides an easy to use interface. As an open source program, Mission Planner allows for individuals to edit the and modify it so that it may better suite their needs. There are two ways which one can customize Mission Planner which are through modifying the source code and creating a script in the Python computing language. By changing the source code the user is creating a unique version of Mission Planner because they have changed what the program actually compiles whereas python scripts can be used on any version of Mission Planner that allows for the use of those scripts.

Source: http://www.aerofly.com/afpd/afpro-telemaster-hodges-02.jpg

Source: https://southernerabroad.files.wordpress.com/2013/04/20130401_075446.jpg

Figure 1: Hobbyist UAV platforms

Methods of Detecting

A serious concern in the field of UAVs is the possibility of collisions that may occur while flying. In order to prevent these disasters a lot of research has been put into creating algorithms for detecting, sensing and avoiding other aircraft in the air. In terms of operator concern, it is difficult for someone using a UAV to really know what is in the airspace that they are flying in. Other aircraft or birds could be flying in the vicinity of the UAV and the operator could have no knowledge of this which creates a safety concern. In order to fully grasp an understanding of the airspace a UAV is flying in, it would require real time data to be outputted onto the GCS which in this case would be Mission Planner. There are multiple methods of sensing a flying object on board a UAV such as wireless communication, LIDAR, and computer vision. Wireless communication is the optimum choice for picking up other aircrafts' signals because the aircraft would be broadcasting out their position and velocity to the owners ground station and this signal could also be picked up by other aircraft. LIDAR uses lasers and imaging to measure the distance objects are from the sensor which could be used to pick up objects that may not be broadcasting a signal, for example birds. Computer vision works similar to LIDAR where it can pick up object not broadcasting signals however it uses cameras to take images of the airspace then analyses those images to determine a depth field. Each one of these methods can be used to approximate the location of an object in the air and when that has been established, return that coordinate back down to the user's GCS. In Figure 2, an example of what an output of a local object in the air would look like on the Mission Planner graphical user interface can be seen. In the image the green points with the yellow trace between them represents the flight path of the UAV and the red point would represent an object that was detected within the airspace the aircraft was flying.

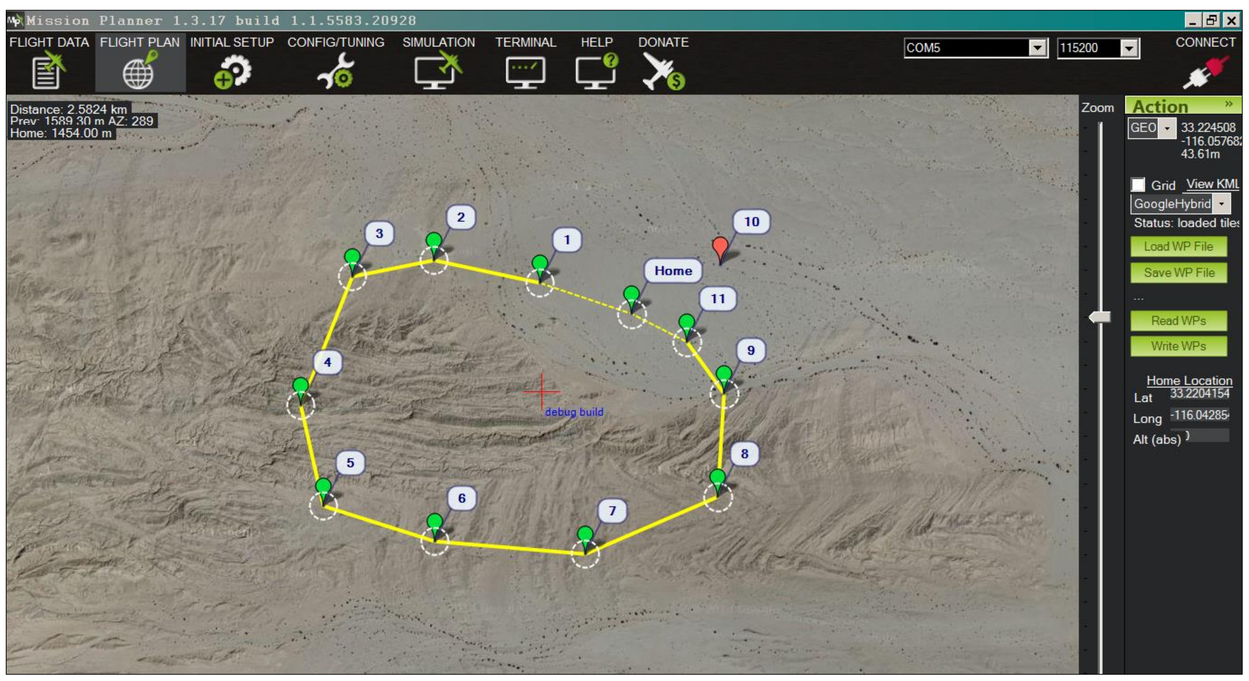

Figure 2: Mission Planner GUI with outputted position of detected object

```
print 'Start Script'
for chan in range(1,9):
        Script.SendRC(chan,1500,False)
        Script.SendRC(3,Script.GetParam('RC3_MIN'),True)
        Script.Sleep(5000)

while cs.lat == 0:
        print 'Waiting for GPS'
        Script.Sleep(1000)
        print 'Got GPS'
        jo = 10 * 13
        print jo
        Script.SendRC(3,1000,False)
        Script.SendRC(4,2000,True)
        cs.messages.Clear()
        Script.WaitFor('ARMING MOTORS',30000)
        Script.SendRC(4,1500,True)
        print 'Motors Armed!'
        Script.SendRC(3,1700,True)

while cs.alt < 50:
        Script.Sleep(50)
        Script.SendRC(5,2000,True) # acro
        Script.SendRC(1,2000,False) # roll
        Script.SendRC(3,1370,True) # throttle

while cs.roll > -45: # top half 0 - 180
        Script.Sleep(5)
```

Source: http://plane.ardupilot.com/wiki/common-other-mission-planner-features/common-using-python-scripts-in-mission-planner/

Figure 3: Example of a python script

Process of Data Output

To actually output the display point onto the Mission Planner graphical user interface, a python script is utilized. The source code for Mission Planner is written in the C# programming language however allows for the python computing language to be interfaced with the program. Python is a relatively simple programming language to learn which makes it ideal for people who are just beginning to touch the surface when it comes to UAVs and want to immerse themselves deeper. The format for a python script can be seen in Figure 3. There are also resources online to help the user understand the variables within the script so they can accurately describe what they want their UAV to do. Some examples of what the python scripts are able to do are create new waypoints, execute maneuvers such as a flip, and many other applications. To output an object on the screen the python script would essentially create a region of interest which in Mission Planner does not add a waypoint to the UAV's flight plan but instead creates a placeholder on the map. In order to do this there will need to be a communication protocol which is already implemented onto the autopilot software that is loaded into the UAV. The communication protocol that Mission Planner uses is called MAVLink which enables the user to create new missions for their UAV and upload them midflight. It also allows the UAV to communicate to the user its approximate location, velocity and heading. Using MAVLink the UAV could then communicate down the approximate location of the object it detect which then the python script will have an input for. Then the script can create a new point of interest which will show up on the map as a red marker.

Other Applications

This process however can be expanded to meet a number of applications that can be useful to both civilians and the military. For example using mapping protocol, the user can implement a tracking protocol which can be used to follow an object whether it be in the air or on the ground. This can also be used in firefighting where cameras could be used to map the spread of a wildfire which could then be analyzed to find the optimum spots to stop the fire from spreading. This could also be used in search and rescue mission where the cameras could be used to analyze an area to detect anything abnormal and then relay those coordinates back to the rescue operatives.

However in order to do those types of applications, it would require more sensor technology which not be readily available to a hobbyist but the principle is the same.

References

http://dev.ardupilot.com/wiki/building-the-code/buildin-mission-planner/#editing_and_debugging_mission_planner_and_other_tips
https://diydrones.com/m/blogpost?id=705844%3ABlogPost%3A1469666&maxDate=2014-12-02T13%3A28%3A33.675Z
http://plane.ardupilot.com/wiki/common-other-mission-planner-features/common-using-python-scripts-in-mission-planner/
https://github.com/diydrones/MissionPlanner