Aerial Refueling

Christian Sabile, Jimmy Chen, Xingyu Zhou

December 2020

1 Introduction

Aerial refueling is widely used by the military and navy for the purpose of achieving a longer operation time. The refueling plane is called the tanker, and be plane being refuelled is often a jet. The refueling is done using an extendable fuel pipe called the boom that extends from the tanker. Throughout the refueling process, the tanker will maintain a constant, linear trajectory, and the jet that needs to be refuelled will approach the tanker and match its trajectory and speed. Our project idea is inspired by this concept.

Our goal is to simulate aerial refueling action of a jet and the tanker. Our approach is to utilize aircraft line following algorithms: the tanker flies on a set straight path, while the jet approaches the tanker from a lower altitude. The approaching jet will fly faster than the tanker as it approaches the tanker. Once the tanker is in front of the jet and pointing its boom down towards the jet, the jet slows down and matches the velocity of the tanker for the remainder of the refueling mission. Our MAV is the jet.

2 Methods

2.1 Line Following Function

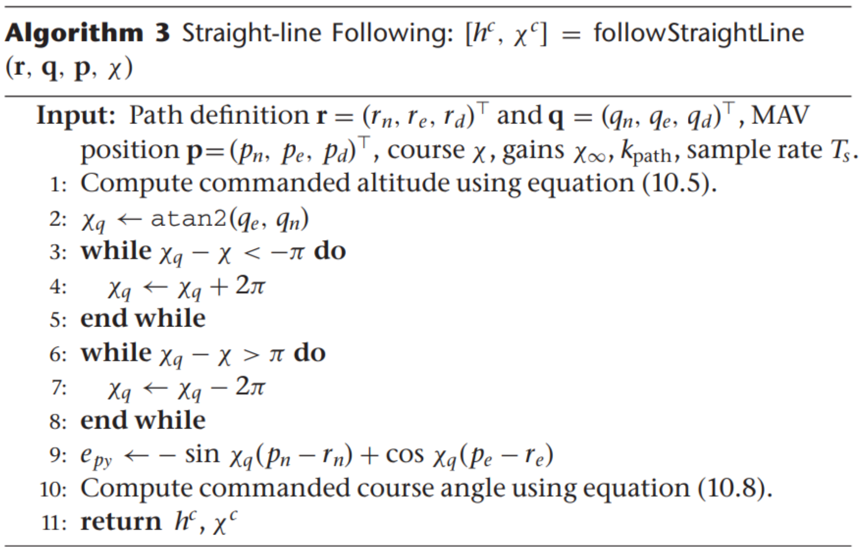
We implemented Algorithm 3 from Chapter 10 of our textbook for the line-following algorithm, as seen in Fig 1. The algorithm takes in the path starting point r, the path vector q, tanker position p, as well as course angle χ. The algorithm returns the commanded altitude and the command course angle, textbook equations 10.5 and 10.8 respectively. The variables used by the two equations are derived within Chapter 10, and we programmed these derivations before putting their results into equations 10.5 and 10.8

Command Altitude (10.5)

$$h_q(r,p,q) = -r_d + \sqrt{S_n^2 + S_e^2}\left(\frac{q_d}{\sqrt{q_n^2 + q_e^2}}\right)$$

Command Course (10.8)

$$\chi^c(t) = \chi_q - \chi^{\infty}\frac{2}{\pi}\arctan(k_{path}e_{py(t)})$$



Algorithm 3 Straight-line Following: $[h^c, \chi^c]$ = followStraightLine $(\mathbf{r}, \mathbf{q}, \mathbf{p}, \chi)$

```
Input: Path definition $\mathbf{r} = (r_n, r_e, r_d)^\top$ and $\mathbf{q} = (q_n, q_e, q_d)^\top$, MAV
    position $\mathbf{p} = (p_n, p_e, p_d)^\top$, course $\chi$, gains $\chi_\infty$, $k_{\text{path}}$, sample rate $T_s$.
1:  Compute commanded altitude using equation (10.5).
2:  $\chi_q \leftarrow \texttt{atan2}(q_e, q_n)$
3:  while $\chi_q - \chi < -\pi$ do
4:      $\chi_q \leftarrow \chi_q + 2\pi$
5:  end while
6:  while $\chi_q - \chi > \pi$ do
7:      $\chi_q \leftarrow \chi_q - 2\pi$
8:  end while
9:  $e_{py} \leftarrow -\sin\chi_q(p_n - r_n) + \cos\chi_q(p_e - r_e)$
10: Compute commanded course angle using equation (10.8).
11: return $h^c$, $\chi^c$
```

Figure 1: Algorithm 3

We programmed Algorithm 3 into a function called *straightFollow()* that is inside of a new python file. The function takes in r, q, p, and χ, and returns h^c and χ^c. We call this function inside Chapter6Simulate.py in order to access the real-time states of the jet.

2.2 Function Call

There are a couple of functions that we need to call for the algorithm to work properly. Just as mentioned above, we choose to modify Chapter6Simulate.py because we need to access the real-time states of the jet. More specifically, we are modifying the function *takeStep()* because our algorithm returns the calculated commandedAltitude and commandedCourse. *takeStep()* is the function which accesses the controller and we will be able to modify the command arguments to perform straight-line following.

2.3 Line and Path Drawing

In the simulator, we draw two arbitrary lines which represent our predetermined path, and we also use them to check if the algorithm is working as intended. To begin with, we modify *vehicleDisplay.py* and under the initialization function, we use *addArbtraryLine()* to generate two predetermined path we need. In order for the jet to follow the path, a path vector q is being manually set in the function *takeStep()* in Chapter6Simulate.py. Since our paths are predetermined, if you want to change them, you will need to modify both the displayed path and the path vector. Notice that, if the coordinates of the displayed path is set to [200,0,200], the path vector q should be set to [200,0,100] because our jet is starting at pd = 100.

2.4 Flying a Second Aircraft

Since we are simulating the process of aerial refuel, we need to generate a second aircraft representing the tanker. This is done by modifying the vehicleDisplay.py. We notice that from line 54 to line 81 is where the first plane is being rendered, so we copy-pasted all the codes and generate a second stationary plane, which is the tanker. We want tanker to fly higher and faster than the jet so we change both the X and Z direction of the tanker, using the function *getNewPoints()* on line 61 of vehicleDisplay.py.

2.5 Refuel Simulation

The simulation begins with the tanker flying on a straight path, with an altitude higher than the jet and will be flying at a constant speed. The jet will start at a lower altitude, following a predetermined ascending path towards the tanker. The jet will also be flying at a speed faster than the tanker. Once the path of the jet intersects the path of the tanker, the jet will switch the current path from ascending to following the tanker. The jet will then continue flying at a faster speed, and when the jet arrives at the refueling position, it will slow down and match the speed with the tanker. We let the jet to maintain its current position for a couple of seconds to simulate the refueling process. A new descending path is then generated when the jet is fueled up, allowing it to leave the refuel position to continue the mission. Figures in the last page show this process.

3 Guide to Running Our Code

NOTE: Please read the comment block at the top of Chapter6.py for additional information.This is just in case our final simulation deviates in some way from the process we describe here.

To run the simulation, use Chapter6.py, and you should see the scene described in the previous subsection. We modified the following files for this project: Chapter6Simulate.py, vehicleDisplay.py, RefuelPathFollowing.py.

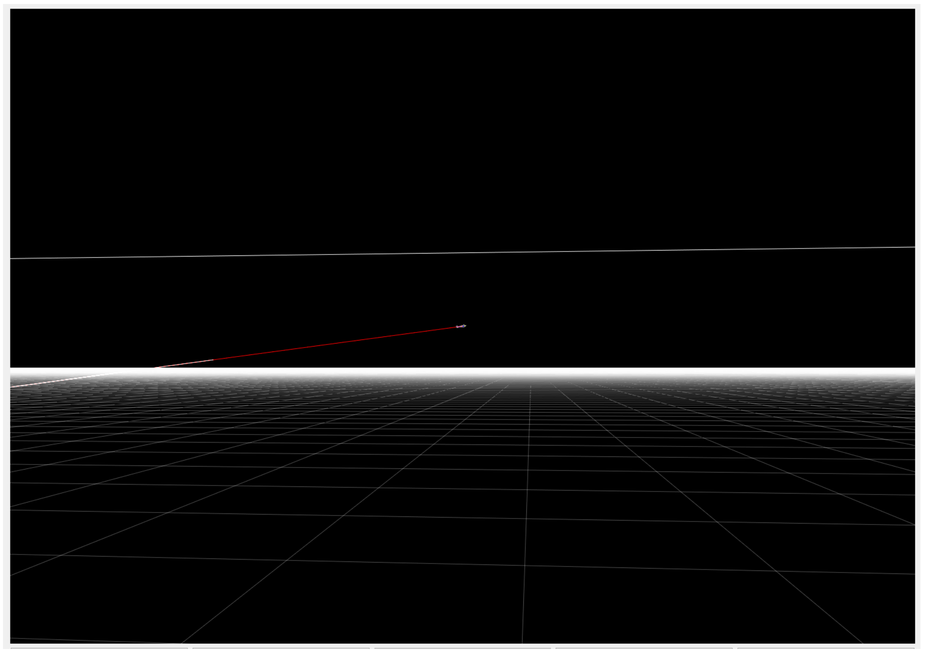


Figure 2: Jet flies towards the tanker flight path. The Jet follows an sloped path for ascending

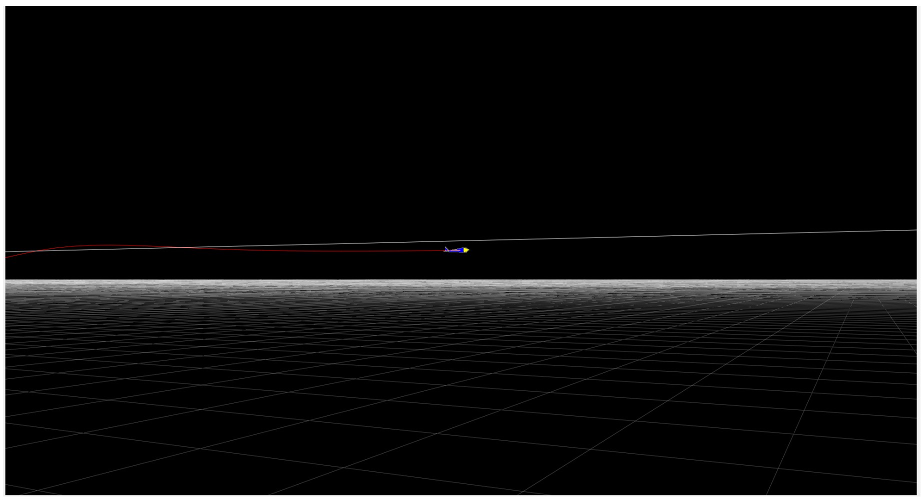


Figure 3: Jet changes its path tracker to follow the new horizontal path

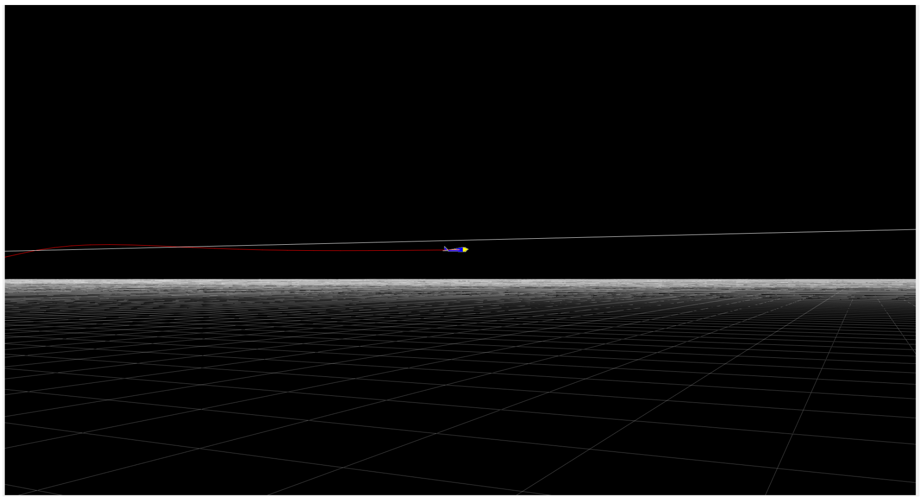


Figure 4: Jet stabilizes on the horizontal path

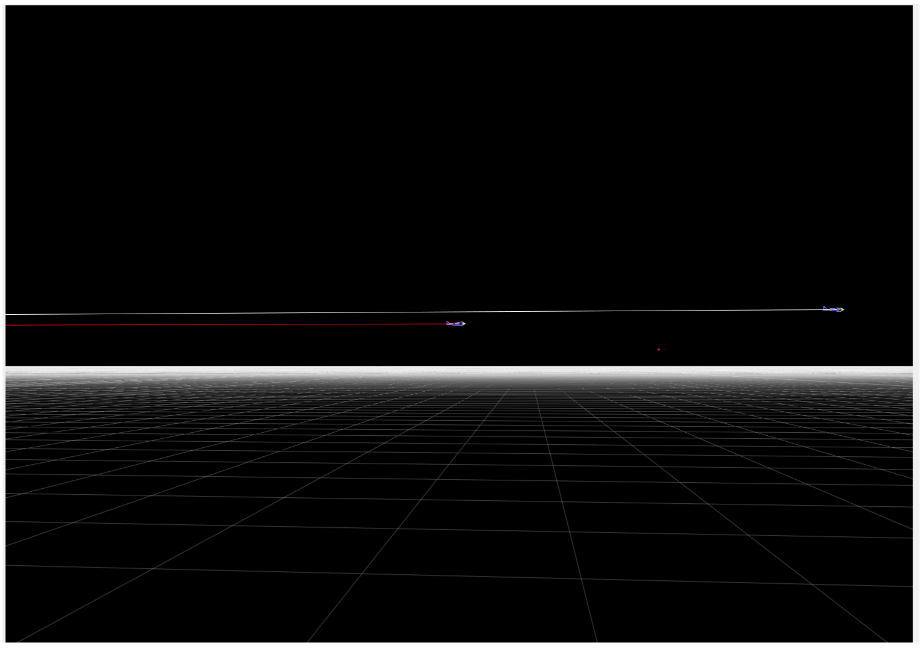


Figure 5: Jet approaches the tanker at a safe distance. It positions itself slightly below the tanker for refueling

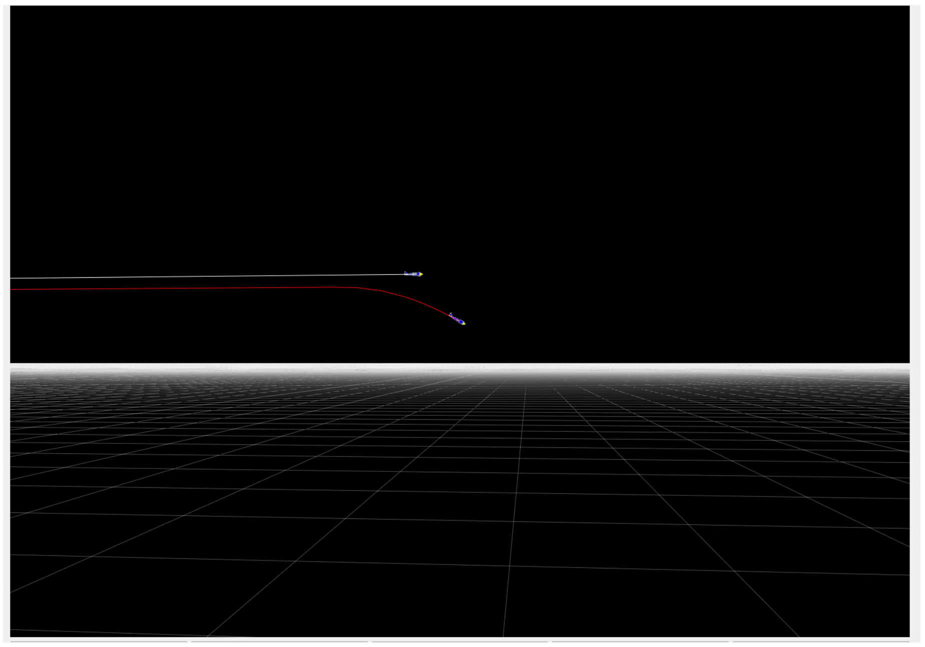


Figure 6: Jet sets a descending path and follows it once it is done refuelling, and departs

4 Team Contributions

The project was done on Discord meetings in real-time. Most portions of the project were done together rather than individually before meetings. Jimmy was in charge of planning and designing the project, and the report. Albert and Christian focused on the simulation and also contributed to the report. Overall, the roles were fluid. We regularly collaborated beyond our respective roles.