A Collision Avoidance Simulation for a UAV Landing at a Non-Towered Airport

Abstract

As the operation of unmanned aerial vehicles (UAVs) expands, the day may soon come when UAV traffic is integrated with manned vehicle traffic. While many studies focus on separating UAVs from manned vehicle airspace, little to no research exists focusing on the integration of UAVs and general aviation traffic. This study creates a simulation of a UAV entering the landing traffic pattern of an uncontrolled (non-towered) airport. A collision avoidance (CA) and wind correction algorithm is also presented, including various maneuvers to mimic a pilot's movements. Finally, this study simulates a UAV in a 3D environment, presents an algorithm for wind-correction and collision avoidance, and successfully lands the vehicle without collisions.

1 Introduction

1.1 Context and General Introduction

The development of unmanned aerial vehicles (UAVs) has rapidly increased since their inception. From military aircraft, search and rescue vehicles, to photography drones, UAVs span multiple fields with a wide array of applications. When one examines UAVs in the context of airspace where manned aircraft may operate, collision avoidance systems become imperative. While there are many studies that develop UAV collision avoidance systems within commercial airspace, little to no research has been done on UAV integration with general aviation traffic.

1.2 Description of Problem

Currently, the FAA prohibits UAVs from flying in airspace reserved for manned aircraft. [**?**] However, as UAV technology advances, it is worth considering that manned traffic may one day be integrated with UAV traffic.

This raises new levels of complication, because many small airports lack a control tower and rely solely on "see-and-avoid" anti-collision strategies. This is a Federal Aviation Association (FAA) mandate that all pilots should vigilantly search for other aircraft in flight and actively avoid them. However, the FAA also cites limitations to this method, such as: weather conditions, blind spots, concentration on flight, and the physical limitations of the human eye.

While several UAV collision avoidance systems have been developed using a central controller, none have been explored within an uncontrolled airspace where manned aircraft rely on "see and avoid" tactics. Additionally, for manned aircraft, nearly 50 % of all collisions occur in or near the traffic pattern during the landing and take-off phases of flight. Therefore, as the day approaches for UAV and manned aircraft integration, collision avoidance systems for UAVs near uncontrolled airports, particularly the landing pattern, is an area of research that requires more attention.

Modeled after a high-traffic, untowered airport in the southeastern United States with a high volume of student pilot traffic, this paper presents a collision avoidance system of a UAV entering a traffic pattern at an uncontrolled airport populated primarily by general aviation traffic. The goal of this project is to create a

3D simulation of a UAV landing at an uncontrolled (non-towered) airport. Four (4) Traffic paths are created based on the two (2) runways at the airport, and a 3D flight simulation is conducted based on Cessna 150 flight performance. The UAV flies the correct path given a runway, avoids collision with other aircraft, and adjusts for wind displacement in-flight, performing typical maneuvers for avoiding manned aircraft in the pattern. Ultimately, this research will open a door to safely integrating UAV and MAV traffic at an uncontrolled airport.

2 Literature Review

2.1 Risk Area Modeling and Autonomous Landing

Two studies model the risks of collision with a UAV. The first, conducted by Nanyang Technological University, examines UAV traffic near a high-traffic, towered, commercial airport and created "alert zones" for air traffic control to alert pilots when the risk of a UAV collision is high. [6] The second study created cluster maps of the same airspace and found a corridor of safe UAV operation. [2]. While these studies both examine UAV operation near busy, restricted airspace, neither factor in the environment, as wind direction and velocity may certainly play a substantial role in UAV trajectories. Furthermore, these studies only examine a towered airport, failing to account for non-towered, general aviation operations.

A study at UC Berkeley created a simulation of an autonomous UAV approaching landing, detecting an obstacle on a landing strip, aborting landing, and circling back to land. While this study successfully models an autonomous landing using hybrid system theory, no system isgiven for mid-air collision avoidance or flight path adjustment. [3]

2.2 Collision Avoidance Approaches

One paper provides a survey of various collision avoidance methods within four major categories: geometric, force-field, optimization, and "sense and avoid". The geometric-based method uses the distances and velocities of UAVs to calculate the time of a collision and performs a maneuver to avoid impact. The force-field technique implements the idea of charged particles, repelling the UAV from obstacles it detects. Optimization-based approaches rely on geographical information to calculate the most efficient path to reach a destination. Lastly, sense and avoid is process is simplified to a single drone detecting and avoiding obstacles using various sensor methods. [7]

A 2017 study developed a sampling-based path planner for collision avoidance with commercial aircraft and moving obstacles. Each path was added to a graph, and the shortest path to the goal was selected. The algorithm generated collision-free paths in real time for UAVs among moving obstacles of different numbers, speeds, and approaching angles. [4] While this research is relevant for collision-avoidance algorithms, in an airport traffic pattern it is not directly applicable to airport traffic patterns.

A NASA researcher explores a geometric optimization approach using position and velocity vectors. For collision detection, the velocities, headings, and positions of the planes are calculated. The algorithm was successful, but solutions based on a pre-planned path were not considered. So while a certain maneuver could be performed for CA in general, it may not be best for the safety of the other vehicles within a traffic pattern. [1]

3 Body

3.1 Approach

First, it is necessary to detail assumptions made when approaching this project.

- Each aircraft is equipped with ADS-B out

- The UAV is equipped with ADS-B in
- Manned aircraft are not practicing "see and avoid"
- At any time, position coordinates and velocity of UAV are known
- UAV's performance capabilities similar to that of a student pilot in a Cessna 172

A Cessna 172 was chosen for the UAV model because it is a popular training plane for student pilots and general aviation pilots. Two other "planes" are to be simulated in the traffic pattern to test the collision avoidance algorithm. A Cessna 150, an aircraft slightly slower than the 172, will enter the pattern, followed by the "UAV" Cessna 172, followed by another Cessna 172 flying slightly faster than the UAV. It is important to note that pilots may fly the traffic pattern at the same speed, which is why a faster aircraft was chosen to follow the UAV. This provides a realistic range of slow and fast aircraft, allowing the UAV to adjust for both speeds.

The airport which provides the basis of this study was chosen because it is a non-towered airport, hosts a high volume of student pilot traffic, and boasts approximately 300 operations per day as of December 2021 [5]. Additionally, it has two runways, which expands the scope of this project to be more applicable to other airports with multiple runways.

3.2 Creating the Traffic Pattern

The first obstacle tackled was orienting the runways and traffic patterns on a graph. At many uncontrolled airports, it is common to find multiple student pilots "flying the pattern." This is the process of performing multiple take-offs and landings for the sake of practice in the standard rectangular pattern. It is therefore essential to understand how airport traffic patterns function. Most runways have what is known as "left traffic," where the pilot flies a rectangular pattern with the runway off their left wing. The pilot enters the pattern at a 45° angle to midfield. This leg is known as the "45." Next, the pilot turns slightly right to fly parallel to the runway on what is known as the "downwind" leg. This is flown until the end of the runway is at a 45° angle behind the pilot. Then, the pilot makes a 90° left turn to start the "base" leg. Finally, the pilot makes another 90° turn when they are lined up with the runway and begins what is known as the "final." After landing, the pilot departs on the "upwind," turns left to start the "crosswind," and may continue the pattern on the downwind.

Runway numbers represent headings and may be considered degrees of rotation in space by simply adding a 0 onto the end of each runway number. For example, runway 18 is actually heading 180, or south. A diagram of the runways can be seen in Figure 1. Runways 36/18 and 29/11 are positioned on a graph. Vector $< 0, 4000 >$ represents runway 36/18, and can be rotated 250° counterclockwise to find runway 29/11 using a standard rotation matrix.

$$R\mathbf{v} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} xcos\theta - ysin\theta \\ xsin\theta + ycos\theta \end{bmatrix} \text{ where } \theta = 250^\circ$$

This results in the vector $< -1368.081, -3758.77 >$. Lining this runway up with the end of runway 36 as shown in Figure 1, simply requires adding (5264, 4000), resulting in (3895.9, 241.2). Thus, runway 11/29 is the line segment from (5264, 4000) to (3895.9, 241.2) and runway 18/36 is the line segment from (0,4000) to (5264, 4000), as shown in Figure 2.

After adding runways, it is necessary to calculate the path for the plane to fly. First, the downwind leg: Since the FAA recommends flying this leg $0.5 - 1$ nautical mile from the runway, it will be flown 3500 feet (approximately 0.58 nautical miles) from the runway. Since this project focuses on general aviation and simulates a Cessna 172, a tighter pattern is realistic. The base point and crosswind point are calculated by extending the length of the runway 3500 feet in each direction. This is because those legs are typically begun once the runway is 45° behind the pilot. Those two points translated 3500 feet parallel to the runway become the downwind leg. The downwind leg was translated according to this logic, where (x_1, y_1) and (x_2, y_2) represents the extended line segment that will become the downwind leg once translated 3500 feet:

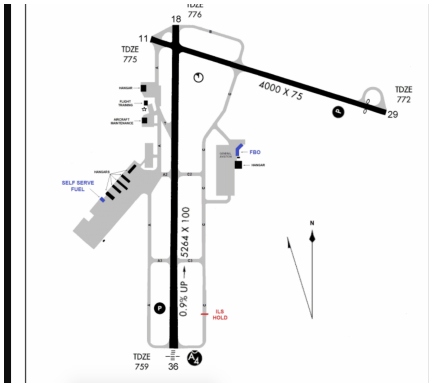

Figure 1: A high-traffic, untowered airport in the southeastern United States [5]

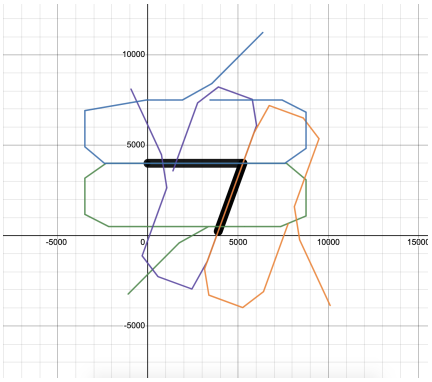

Figure 2: Graph of Runways and Traffic Patterns

$$r = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$\delta x = \frac{r}{3500}(y_1 - y_2) \text{ and } \delta y = \frac{r}{3500}(x_2 - x_1)$$

Translated line segment: $(x_1 + \delta x, y_1 + \delta y)(x_2 + \delta x, y_2 + \delta y)$

Once the downwind leg is established, every turning point in the pattern is calculated. Simply connecting the downwind points to the crosswind and base points completes the rectangle runway pattern. The final leg to construct is the 45 approach line. True to its name, this line is a 45° counterclockwise rotation from the downwind leg and connects to the midpoint of the downwind. We use the same rotation vector detailed above, but $\theta = 45°$ and $< x, y >$ is the vector from the midpoint of the downwind to the start of the downwind. The length of this vecotr is one mile. The process for finding a new point a certain distance along a line segment is as follows:

$$V = (X_2, Y_2) - (X_1, Y_1) \text{ and } u = \frac{V}{||v||}$$

Thus, the new point is: $((distance * u) + X_1, (distance * u) + Y_1)$. This establishes the runways and sets up the dimensions of each traffic pattern as modeled in figure 2.

3.3 Classes Used

Node.java : This node class forms the building blocks of the Queue class. It constructs nodes and forms a linked list that will be used to create a queue.

Queue.java : This is a First In, First Out (FIFO) algorithm that is made from a circular linked list using the Node class. It includes methods such as isEmpty, enqueue, dequeue, clone, and front.

Plane.java : This is a class to store attributes of each aircraft in the pattern. A plane is constructed with a name, x position, y position, heading, and V speeds specified by the manufacturer of each plane. Assuming the plane starts on the 45, height is initialized to 1000 feet.

Runway.java : This is a class to store calculations for each essential point in the landing pattern. An x and y variable is declared for every turning point in the pattern. Though this project references a particular airport, the runway class presented in this algorithm applies to any airport and may easily be changed to accommodate other airports.

3.4 Algorithm Design

What follows is an outline of the algorithm approach to the airport client file:

- A Queue is constructed to store legs of flight
- An array of plane classes to store each plane
- A runway object is constructed based on user input for heading and wind
- For each clock pulse, update position of each plane
- If time to turn a leg, dequeue the Leg Queue and update position according to new leg
- Position data is output to a file containing the runway name

Three planes are initialized. A slow Cessna 150, the Cessna 172 UAV, and a faster Cessna 172. This order ensures a collision without proper avoidance strategies, allowing full testing of the algorithm's effectiveness.

changeLeg() is the method for switching legs: After checking if collision avoidance is necessary, the program checks to see if the plane has passed a turn point by calling the passed() method. Next, the program calculates the distance to turn, updates the plane heading, dequeues the leg queue, and calls the increasePos() method to increase the position.

increasePos() is a method to update plane position based on current leg: It performs collision avoidance maneuvers if deemed necessary, updates the plane's velocity based on its acceleration, finds the new point to move to using the newPoint() method, and corrects for wind displacement.

newPoint() is a method to determine a new point a certain distance along a vector. This method calculated the projected path of the UAV based on the current position on the leg and then endpoint of the leg, specified in the runway class. The new point is then calculated, where d is the distance to go until the end of the leg, v is the vector of the current leg, and (x, y) are the starting coordinates of the current leg:

$$\text{new point before wind displacement} = (d * \left(\frac{v}{||v||}\right)) + (x, y)$$

The new point is then used to calculated the new heading of the UAV. If wind is present, a change in the x and y position is also adjusted, to simulate the UAV blowing off course.

Change in the x direction: Wind speed * *cos* (Wind Direction in radians) + start position

Change in the y direction: Wind speed * *sin* (Wind Direction in radians) + start position

Where hv is the unit heading vector, wv is the unit wind vector, and (x, y) is the starting position on the leg, the new position with wind displacement becomes:

$$\text{new position} = (hv * \text{velocity}) + (wv * \text{wind speed}) + \text{start position}$$

accel() is a method that calculates the acceleration needed based on current position of plane, end of current leg position, and velocity desired at end of leg based of the plane's V speeds specified in the plane class. First, the distance formula is used to calculate distance to end of vector. Next, a physics is used to solve for acceleration: ${V_f}^2 = {V_0}^2 + 2 * \text{acceleration} * \text{distance}$. Given the initial velocity, final velocity and distance to end of vector, the method returns the necessary acceleration.

With this algorithm, the program successfully simulates three flying a standard traffic pattern with the correct dimensions, speeds, and acceleration. Figure 5 and figure 6 show the final flight path before any collision avoidance or wind displacement. Note that the touchdown point is purposefully kept small, as ground roll is negligible when performing touch and go's.

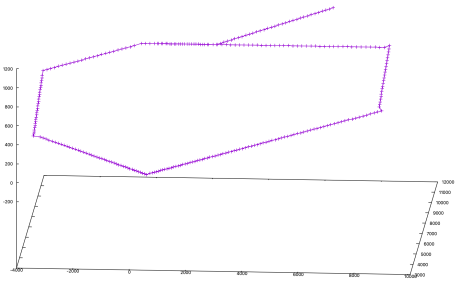

Figure 3: Runway 36 traffic pattern view from side

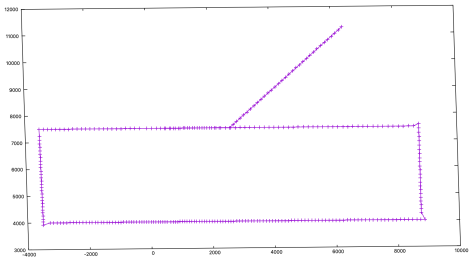

Figure 4: Runway 36 traffic pattern view from top

3.5 Wind Correction

The majority of research on UAV flight paths often neglects to factor in one key element: wind. While small levels of wind velocity and direction may not have drastic impact on commercial aircraft, it makes quite a difference for smaller, general aviation planes. Since this project focuses on such recreational traffic, it is essential to consider wind direction and velocity shifts when constructing the simulation. In the program, the user is prompted to enter a wind direction and velocity. Figure 5 and Figure 6 show runway 36 with a 15 knot crosswind blowing at heading 270, a direct crosswind from the east. While the UAV still hits its aim points, as shown by the accuracy of the corners and the touchdown and abeam points, there is concerning curvature in the path.

This study takes a geometric approach to wind correction, allowing the UAV to determine the wind magnitude based on it's displacement from its projected path and correct its trajectory accordingly each time it calculated the next point to move to. This is most realistic to how pilots typically fly in windy conditions by "crabbing," or turning into the wind to avoid being blown too far off course.

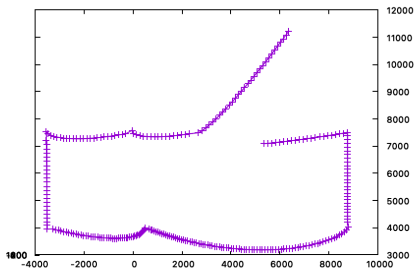

Figure 5: The flight pattern for runway 36 adjusted for wind displacement, side view

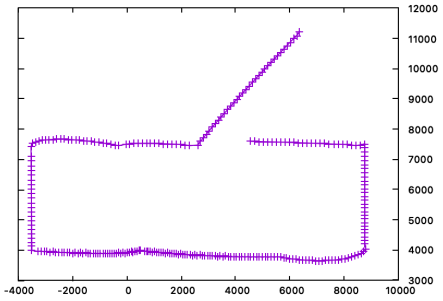

Figure 6: The flight pattern for runway 36 adjusted for wind displacement, top view

First, the difference in the plane's heading and the actual and expected heading is calculated. The previous point is then calculated depending on the user's selected wind speed and direction. Next, according to how great the difference in heading is, a force in the opposite direction of wind is applied to the plane for wind correction. The ratio depends on the difference in heading. If the difference is great, then a large ratio value is used to correct. If the difference is small, a small value is used to correct. The wind-corrected-position is then compared to the previous point to calculate the new heading.

3.6 Collision Avoidance

After successfully programming the UAV to fly the pattern and correcting for wind displacement, the next step to elevate the simulation is to incorporate collision avoidance. The goal of this project was not to be a simple physics simulation, but to present an algorithm that could one day be incorporated in UAV flight and step toward UAV and MAV integration. Thus, in order to realistically and safely fly a traffic pattern, the UAV must avoid other planes in the pattern. While there are numerous studies that examine collision avoidance with UAVs, most either find the best path around an object, or seek to avoid said object. However, these strategies do not apply in collision avoidance along a path. First, a plane should never cut in front of another aircraft in the landing traffic pattern. Second, the UAV is seeking landing, and would not aim to simply desert the pattern when at risk of colliding with another aircraft. Thus, this paper proposes three collision strategies for the UAV to perform that align with typical pilot maneuvers:

1. Accelerate
2. Make a 360 degree turn
3. Ascend to avoid immediate impact

While pilots are usually concerned with aircraft in front of them, this study assumes that no aircraft see the UAV. Therefore, this proposed method applies to aircraft both in front and behind the UAV. In order to store these maneuvers, a String called "collision method" was added to the plane class. Before advancing to the next point, the program checks this string to verify the collision avoidance path intent of the UAV.

1. Accelerate First, the program measures the rate at which the UAV is overtaking or being overtaken by another plane. This is done by comparing the current distance between each plane and the previous distance between each plane. If this rate is less than 1 but greater than .9, the UAV will accelerate accordingly using the accel() method detailed in section 3.4. The current point is the UAV's position, and the end point is the other plane's previous coordinates and previous velocity. Thus, the UAV slows down for the slower aircraft in front and speeds up for the aircraft behind, while still hitting its target velocities at each essential point in the pattern. While this method is effective for keeping distance between aircraft as long as possible, it is not a conclusive solution for collision avoidance.

2. Make a 360 Degree Turn This maneuver was chosen for collision avoidance because it is a typical maneuver performed in a traffic pattern to buy more time for slower aircraft ahead. Not only does this maneuver create space for aircraft ahead of the UAV, but also allows fast aircraft behind to safely pass the UAV without a collision. This maneuver is performed if the distance between the UAV and any plane falls between 700 and 1700 feet. However, it will not occur within 200 feet of the ground, however, as performing a 360 that low may be dangerous.

The maneuver is broken into two parts: "start360" and "360" to prevent the UAV from spiraling continuously. "Start360" establishes the center of the circle as a counterclockwise rotation 270 degrees in the $< y, -x >$ direction and scales the radius to 500, making the circle have a diameter of 1000 feet. This direction was chosen to keep the circle on the outer edge of the pattern. Once the plane has completed the 360, the collision method string is set to "none" and the UAV breaks out of the 360.

3. Ascend This maneuver is occurs when the distance between the UAV and another aircraft falls below 700 feet. This means that an immediate collision is likely, so the UAV ascends to avoid a collision. When ascending, a plane pitches to Vx, its best angle of climb speed, allowing for the greatest altitude gain.

To perform this maneuver, the distance to the end of the current leg is calculated and then divided by the UAV's Vx speed, yielding the time taken to reach the end of the leg at speed Vx. This time is multiplied by the UAV's climb rate, which yields the end height the UAV can climb to at Vx in given a certain distance. Then, a new point is calculated using the newpoint() method and the plane ascends. Once the distance between the UAV and other plane is over 700 feet, the collision method string is switched to "none" and the plane descends to enter back into the standard traffic pattern.

As further research develops, maneuvers such as S turns may be added to this list to expand the CAS. However, these three options are successful in avoiding collisions, and were chosen not only because of their realism and success, but their potential to serve as a launching point for future research.

3.7 GUI Implementation

To visualize the path of the UAV, the 3D graphical user interface, Gnuplot, a Graphical User Interface (GUI) for Java is utilized. This program allows for 2D and 3D graph rendering of functions and data points.

4 Results

Displayed below are figures that exemplify the CAS system without wind displacement for visual ease. In these figures, the first plane (Cessna 150) enters the pattern at 0 seconds, the UAV enters at 15 seconds, and the last plane (the faster Cessna 172) enters at 23 seconds. Figure 8 shows the UAV begin to perform a 360 on the 45 but break out as it detects the possibility of a close collision. 7 shows the UAV ascending to avoid immediate collision, performing a 360 to put further distance between it and the fast plane, and descending to continue the traffic pattern.

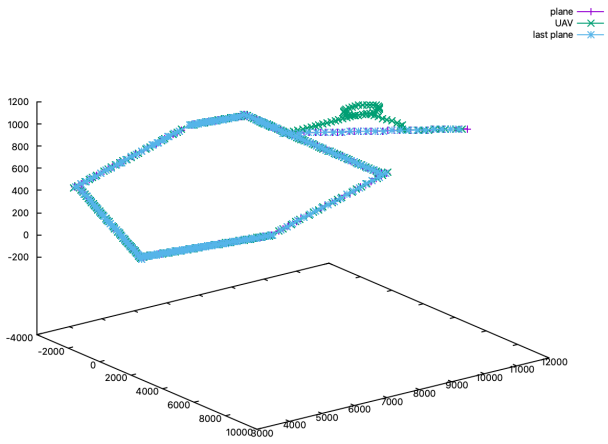

Figure 7: Collision Avoidance for Runway 36, no wind

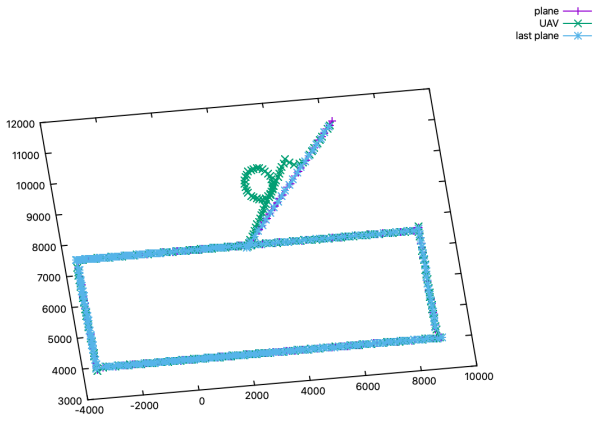

Figure 8: Collision avoidance for Runway 36, no wind

Figures 10 and 9 show the CAS for runway 29. For this example, the first plane (the Cessna 150) enters the pattern at 0 seconds, the UAV starts at 40, and the fast plane (the faster Cessna 172) starts at 120 seconds.

The next figures show the CAS when overlapping runways are in use by different planes. While this is a danger that rarely happens, at uncontrolled airports stubborn pilots may refuse to switch runways. The UAV starts at 3 seconds and the faster plane starts at 40 seconds. In figure 11, one can see how the UAV starts a 360 but then ascends on the downwind leg to avoid the downwind leg of runway 29. Figure 12 displays the multiple 360s the plane performs to not only avoid the other aircraft on runway 36, but to also avoid the plane on runway 29. It is necessary to note that in this scenario, the CAS is not as refined as a single runway. While it does function realistically and successfully avoids collisions, sometimes the UAV will behave in sporadic ways that many pilots would not perform. The scenario of multiple runways is certainly an opportunity for further refined research, but this system serves as a launching point for such research.

Finally, the next figures display the flight without wind correction and the flight path with wind correction for runway 36 where the winds are 270 at 15 knots. In Figure 13, it is clear that the planes are being blown drastically off course. In Figure 14, This wind displacement has been accounted for and corrected. It is important to note that the path is still not perfectly straight, and this is due to heading adjustments in flight on each leg. Each heading adjustment thus results in wobbling and curvature in the path. This can be seen in the Cessna 150 path, which slightly over-corrects on the upwind leg.

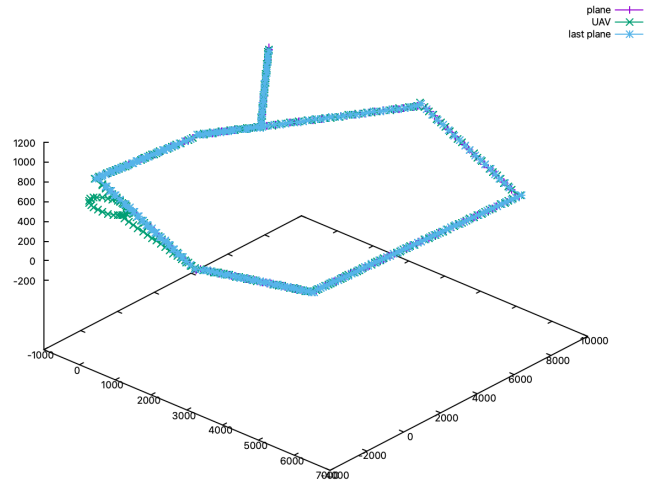

Figure 9: Collision Avoidance for Runway 29, no wind

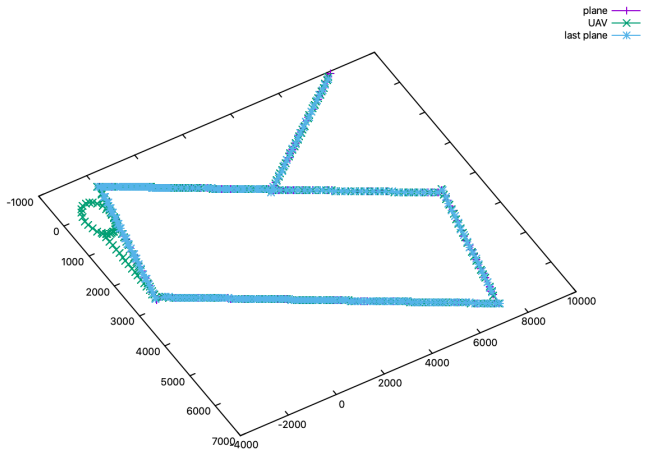

Figure 10: Collision avoidance for Runway 29, no wind

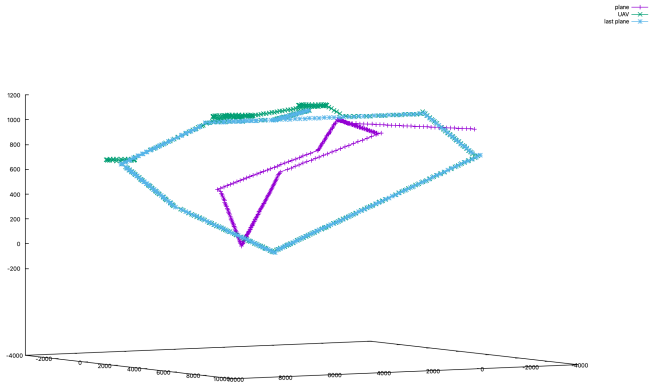

Figure 11: Runways 18 and 29 Collision Avoidance, no wind

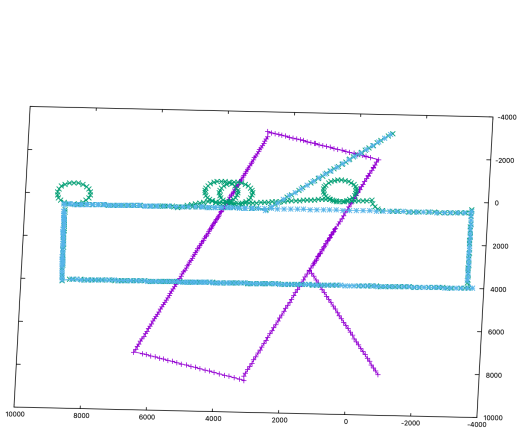

Figure 12: Runways 18 and 29, no wind

5 Conclusion

This paper proposes a successful Collision Avoidance System for UAVs landing at an uncontrolled airport. While it is acknowledged that current FAA restrictions require UAVs to stay "well clear" of MAVs, the day may come when UAVs and MAVs will fly in the same airspace. Future research may expand the scope of this paper by focusing on different types of aircraft, exploring simulations where the runway must be switched halfway through a pattern, and even creating simulations where other manned aircraft are not following a standard traffic pattern.

Modeled after a high-traffic, untowered airport in the southeastern United States, 4 traffic patterns according to standard general aviation practice are correctly flown by this simulation. Two other planes of varying speeds are implemented to test the collision avoidance system. The simulation also handles collision avoidance when multiple runways are in use at once, though this needs to be refined in future research. The UAV performs typical avoidance maneuvers such as accelerating to create distance between aircraft, performing a 360 degree turn, and ascending to avoid immediate impact. The UAV safely circles the pattern, lands, avoids other aircraft, and corrects for wind displacement.

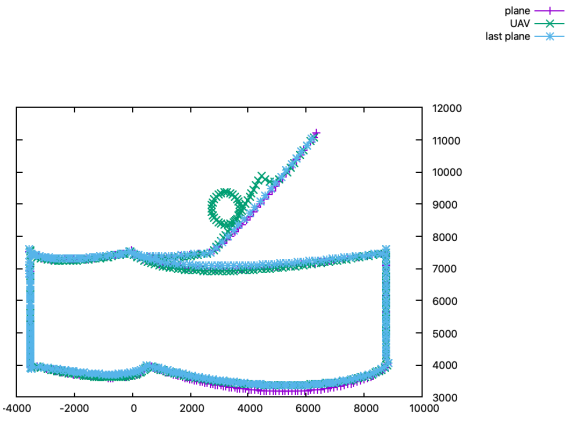

Figure 13: Runway 36 with no wind correction

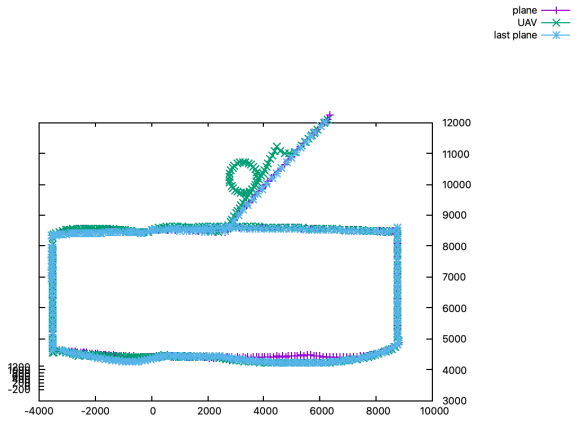

Figure 14: Runway 36 with wind correction

References

[1] Karl Bilimoria. *A geometric optimization approach to aircraft conflict resolution.*

[2] Wei Dai, Bizhao Pang, and Kin Huat Low. Accessibility analysis of unmanned aerial vehicles near airports with a four-dimensional airspace management concept. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–9. IEEE, 2020.

[3] Takkuen Koo and Shankar Sastry. Hybrid control of unmanned aerial vehicles for autonomous landing. In *2nd AIAA " Unmanned Unlimited" Conf. and Workshop & Exhibit*, page 6541, 2003.

[4] Yucong Lin and Srikanth Saripalli. Sampling-based path planning for uav collision avoidance. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3179–3192, 2017.

[5] AirNav LLC. Auburn University Regional Airport, 2022.

[6] CH John Wang, Shi Kun Tan, and Kin Huat Low. Three-dimensional (3d) monte-carlo modeling for uas collision risk management in restricted airport airspace. *Aerospace Science and Technology*, 105:105964, 2020.

[7] Jawad N. Yasin, Sherif A. S. Mohamed, Mohammad-Hashem Haghbayan, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches. *IEEE Access*, 8:105139–105155, 2020.