# AIRCRAFT COLLISION AVOIDANCE SYSTEM ON A 2-D PLANE USING MATLAB

VanderbiltUniversity
Chia-Cheng Chang
Email: chia-cheng.chang@vanderbilt.edu

## Abstract

Final project for class CS6376 at Vanderbilt University. The project uses matlab to simulate aircraft flying through a 2 dimensions plane. The two airplanes are allowed to communicate within a certain distance and have to avoid collision with each other. The model displays a workable solution to solve the requirements. The model and algorithm can expand to more complex systems and requirements.

## Introduction

The goal of the project is to develop an aircraft navigation system that guides the aircraft through a 2 dimension plane. The project requirements contain two phases. The first phase of the project requires the development of the simulation model of aircraft. The aircraft is required to reach the destination in the shortest route. Moreover, there are a total of two aircraft flying at the same time. The requirements and objective is to come up with a controller that takes in the input and produces an output that suits the best interest of the current aircraft. The specific requirements are as fellow

Phase 1: Aircraft model design

- Input: current location, and final destination location.
- Output: the direction of the aircraft should fly toward
- Requirements:
    - Design a simple controller that navigates the aircraft to its location in the shortest distance.
    - Design a complex controller that does the task of the simple controller but has to avoid collision with the other aircraft.
    - The aircraft should fly toward the direction that produces the shortest route to destination.
    - When near the other aircraft, the two aircraft can exchange messages about their location. The current aircraft should use the provided message to determine its next move.
    - The two aircraft can never collide at any moment.
    - Both aircraft should use the same controller, the controller should not have bias on either aircraft.

Phase 2: Aircraft model implementation

- Implement the model design in phase 1 to a simulation model
- Create a safety monitor to check for collision avoidance.

Since the project is not a group project, I did most of the work. I use the extended machine design from the CS6376 class and textbook to design the controller. I also use matlab to set up the aircraft controller model and simulate the aircraft. The safety check is also developed on matlab. For the model requirements, I have consulted from previous projects and added a timeout step input to make sure the simulator would not drain out the resources.

The model and design successfully meet the requirements of the project. The aircrafts are able to navigate through the 2 dimension plane without overlapping each other at the same time. I use Pythagorean theorem to estimate the shortest route to take from the current location and navigate the circraft toward the destination.

The collision avoidance controller is a bit more tricky. I wrote another algorithm to predict the possible locations of the other aircraft in the future and avoid them. The model runs smoothly and the airplane flies in the shortest distance possible. The project has proven the design of the system is solid and has met the project requirements.

**BackGround**

**Aircraft Model Design Approach- Simple Controller**

There are several ways to tackle the problem of developing the controller. The easiest way to design the system is to ground one of the aircraft and keep the aircraft on the ground before the other aircraft reaches the final destination. However, it's not ideal for the situation because it limits the liveliness of the system.

The second approach to the problem is to pre-calculate the route of the aircraft before take off. This approach gives the shortest route between the origin and destination but does not eliminate the thread of collision. People can argue that it's possible to calculate the collision area and avoid it in advance. However, since the project requirements state that the communication between two aircrafts only allowed within a certain distance, this approach is not ideal when the origins of the two planes are far apart.

Finally the approach I took was to calculate the shortest route on the fly. At each point, the aircraft would calculate which direction it should take that would result in the shortest route to the destination. The approach has two advantages and one minor downside. First, it still generates the shortest route between origin and destination. Secondly, it gives versatility to the aircraft. The aircraft can change direction based on its current condition. This advantage also gives me the opportunity to tackle the aircraft avoidance requirement. Lastly, the minor downside of this approach is the potential over time during calculation. Since the controller would have to calculate the shortest distance in every instance, if the data is too large, the system might not be able to handle the quantity in time. However, since the requirements of this project are fairly simple compared to the real word, this approach works perfectly. Moreover, since the system does not have to care about other aircraft most of the time, the model can expand from two aircraft to multiple aircraft without modifying much of the code on the controller.

**Aircraft Collision Design Approach- Complex Controller**

The requirement states that two aircraft could never collide and they are only allowed to exchange messages when nearby. One way to approach the problem is to calculate the route to avoid collision when receiving the messages from other aircraft, and allow the other aircraft to stay on it's current route. However, this approach can not make sure the two aircraft would never collide since we are not able to control the route of the other aircraft.

My approach to this problem is to pre-calculate all possible locations the other aircraft can take and avoid those locations. This method works like the shield around the aircraft. According to the project requirement (Figure 1), the collision avoidance zone has side with d= 1km, so we only need to calculate at most 9 points. Due to the short length of the collision avoidance zone, we do not need to calculate a large number of data on the fly. Therefore, this approach is practical for the scope of the project.

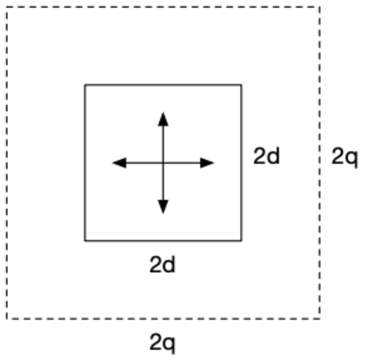

*Figure 1. Aircraft message exchange zone and collision avoidance zone*

**Implementation**

**Simple Controller Implementation**

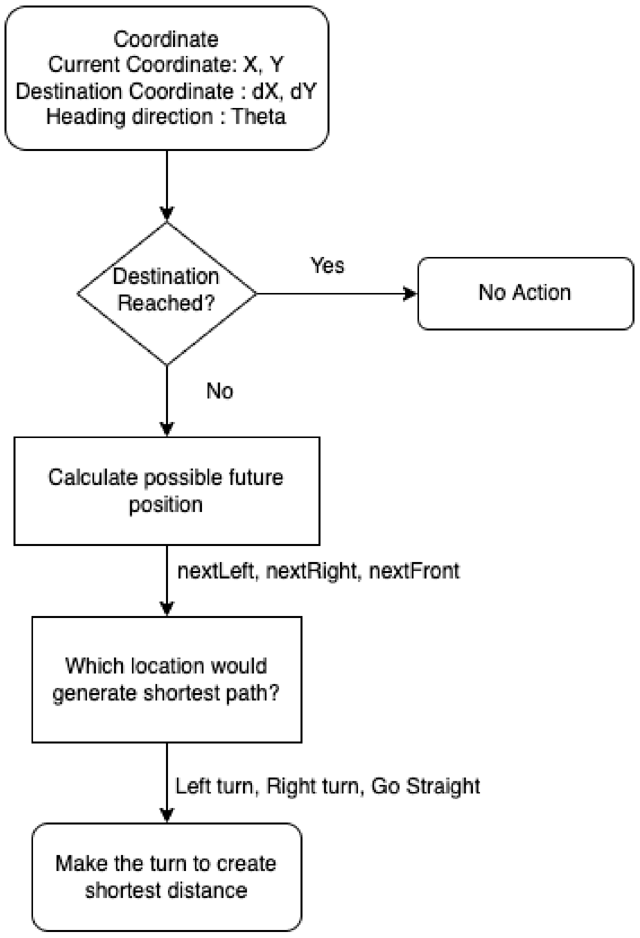
Coordinate
Current Coordinate: X, Y
Destination Coordinate : dX, dY
Heading direction : Theta
Destination Reached?
Yes
No Action
No
Calculate possible future position
nextLeft, nextRight, nextFront
Which location would generate shortest path?
Left turn, Right turn, Go Straight
Make the turn to create shortest distance

*Figure 2. Detailed flowchart for simple controller implementation*

The implementation of the simple controller is fairly straight forward. We have the coordinate of the current location, current direction theta and destination coordinate. Our first task is simply to compare the two coordinates to determine if we have already reached our destination by checking the current coordinate and destination. If we had reached the destination, no further action is needed.

If we are still not reached the destination yet, we use the current heading direction to generate the next possible coordinate. Since an airplane can only go left, right, and front, we would only need to calculate the 3 possible locations for the current direction, theta values. For example:

```
if(theta ==0)
        leftLocation = (curretX -1,  currentY);     frontLocation = (curretX +1,  currentY);
        rightLocation = (curretX +1,  currentY);
end
```

After we have the three possible values for our next move, we can calculate which turn would lead us to the shortest possible route. To calculate the shortest distance, we can use the simplified version of the Pythagorean theorem, we only calculate the difference of the between destination and current location. For example:

*distanceTurnLeft = abs(detinationX - leftLocation.X) + abs(detinationY - leftLocation.Y)*

After we have all three distances, we would just pick the one with the shortest distance and execute the turn accordinately. This method minimizes the data process calculation and has no long operation that takes time and resources.

**Complex Controller Implementation With Collision Avoidance**

As we can see from Figure 3, the complex controller is similar to the simple controller. I tried to make them alike so I can reuse most of the parts. Besides the common parts, the complex controller has an extra collision avoidance part to prevent the collision to other aircraft.

The collision avoidance system works as a shield of the aircraft. When the aircraft enters the message exchange zone (2q x 2q area in Figure 1), the aircraft would calculate all the possible locations the other aircraft would appear. In this particular project, the avoidance area has a length of 1km so there are 9 possible points at each moment. For example:

*array = [in.x in.y ;in.x-1 in.y+1; in.x in.y+1; in.x+1 in.y+1; in.x-1 in.y; in.x+1 in.y; in.x-1 in.y-1; in.x in.y-1; in.x+1 in.y-1];*

However, since we only need to consider the 3 possible turns the other aircraft could take, that makes the calculation more simple. We can use the heading direction theta from other aircraft to calculate the 3 possible turns.

Even though we would only encounter 4 possible theta in the scope of this project, I still include the calculation of all 9 possible locations for other aircraft in case there’s any complication to the system.

After we have all the possible locations for other aircraft, we tested the possible locations for other aircraft against our 3 possible locations we obtained from the simple controller. If our next turn would overlap their possible locations, we just choose the next turn with the shortest route.

By executing this algorithm in each step, we can gradually establish the shortest route to the destination while avoiding other aircraft’s encounters.

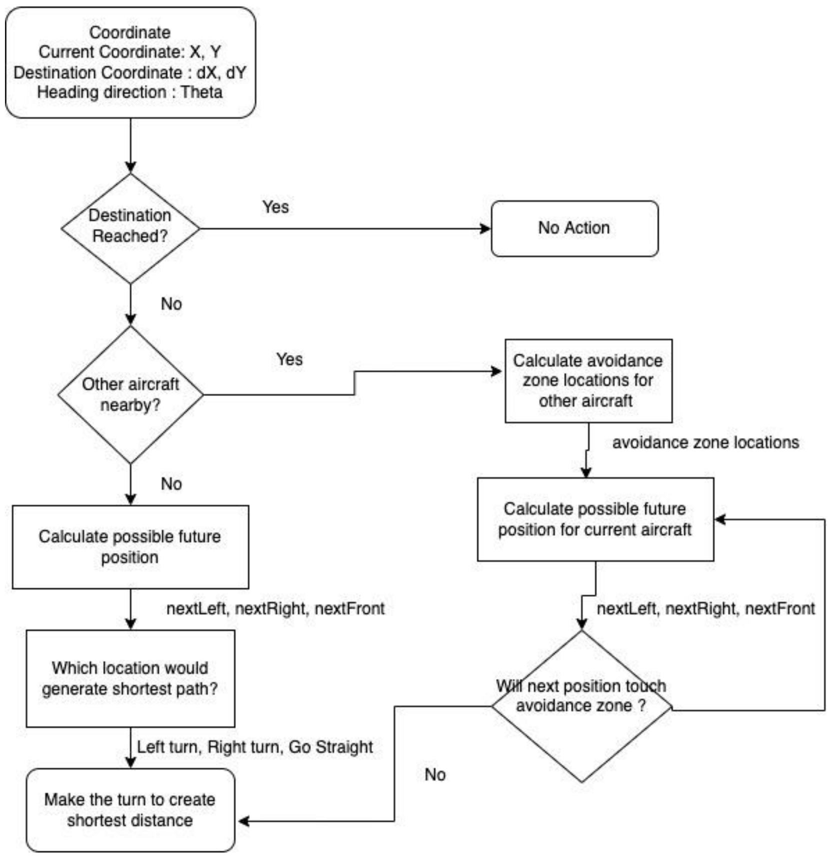
Coordinate
Current Coordinate: X, Y
Destination Coordinate : dX, dY
Heading direction : Theta
Destination Reached?
Yes
No Action
No
Other aircraft nearby?
Yes
Calculate avoidance zone locations for other aircraft
avoidance zone locations
Calculate possible future position for current aircraft
No
Calculate possible future position
nextLeft, nextRight, nextFront
Which location would generate shortest path?
nextLeft, nextRight, nextFront
Yes
Will next position touch avoidance zone ?
Left turn, Right turn, Go Straight
No
Make the turn to create shortest distance

*Figure 3. Detailed flowchart for complex controller implementation*

**Proof of Concept**
**Simple Controller**

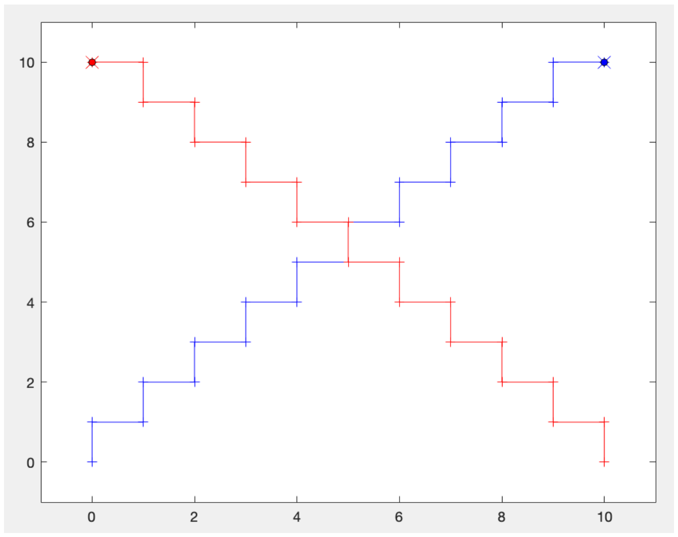

*Figure 4. Simple Controller route diagram*

To prove my simple controller concept is correct, I use this simple example for showcase. There are 2 aircrafts starting from the opposite side of the map. The blue aircraft starts from bottom left (0,0) and has destination coordination of (10,10). The red aircraft starts from the bottom right(10,0) and has a destination coordinate of (0,10). To calculate the shortest route from origin to the destination, we can just calculate the difference of the 2 points in the x-axis and y-axis. Therefore the ideal steps for this example is Math.abs(10-0) + Math.abs(0-10) = 20. As the diagram displays, each aircrafts takes 20 steps to reach their final destination.

Figure 4 displays the simple controller concept and algorithm works. However, the simple controller does not prevent collision of the 2 aircraft. As we can see from figure 4, the 2 aircrafts would overlap and collide at coordinate (5,5) and (5,6). Therefore, this simple controller does not satisfy our overall project requirement. Therefore, we need a more complex controller.

**Complex Controller**

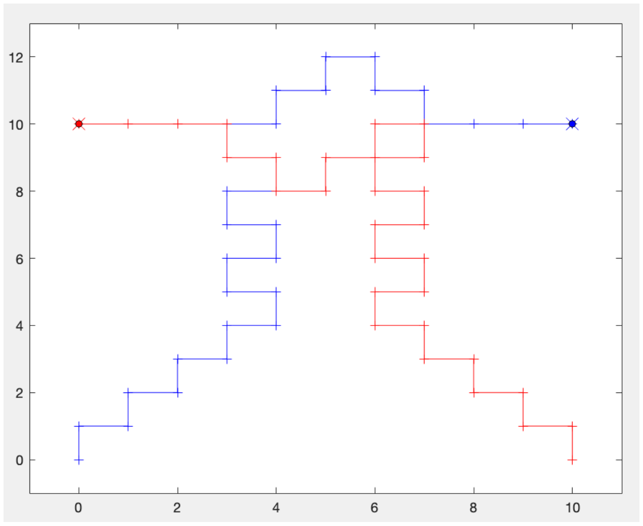

*Figure 5. Complex Controller route diagram*

To showcase the collision avoidance system in the complex controller, I use the example in Figure 4 to demonstrate. As we can see in Figure 5, both aircraft started at the same coordinate as the ones in Figure 4. They both headed toward their destination in the shortest route until the possible collision area near coordinate (5,5) and (5,6). Since both aircrafts had the same controller, they took the same action of avoiding other aircraft while moving toward the destination in the shortest route. Therefore, they both start moving up the Y-axis but maintain a minimum distance of 2d = 2km apart until they reach the destination Y-Axis of Y=10.

After two aircraft reach the height, they start to avoid other aircraft and move toward their destination. As we can see they both move in a similar route to reach their final destination. This similar route also proves that both aircrafts were given the same controller as stated in the project requirement. Figure 5 also shows that the aircrafts did not collide at any point on their path; both took 30 steps to reach their destination coordinates.

## Conclusion

The simple controller and complex controller successfully deliver the requirements for the project under the scope. Figure 4 and Figure 5 demonstrate the difference between two controllers and how they solve the problem as expected. There are also more examples shown in the appendix to prove the solidity of the algorithm.

Although the algorithm is proven to be correct, I believe there might still be room for improvement. For example, when the two aircraft meet, they can decide which one to take the longer route and make the other one stay on the shortest route. By doing so, the two airplanes could develop a system to generate the shortest overall route. However, the rule to determine which one to take the longer route is unknown and is not stated in the project scope, therefore, it's not implemented here.

For further extension of the project, an instance of 3 or more aircrafts are also interesting fields to explore, but I imagine the algorithm I developed can handle the task. Also, a 3 dimension scope of the field is also interesting, but would require deeper consideration of the theta value in 3 dimensions.

## References

Kharchenko, Volodymyr, et al. “*MODELING OF AIRBORNE COLLISION AVOIDANCE SYSTEM PERFORMANCE USING MATLAB*.” *Research Gate*, National Aviation University , Oct. 2013.

Mishra, Ankit, et al. “Project Phase 1 Report.” *Github*, May 2014, https://github.com/mishra14/AircraftController.

Alur, Rajeev. *Principles of Cyber-Physical Systems*. MIT Press, 2015.

## Appendix

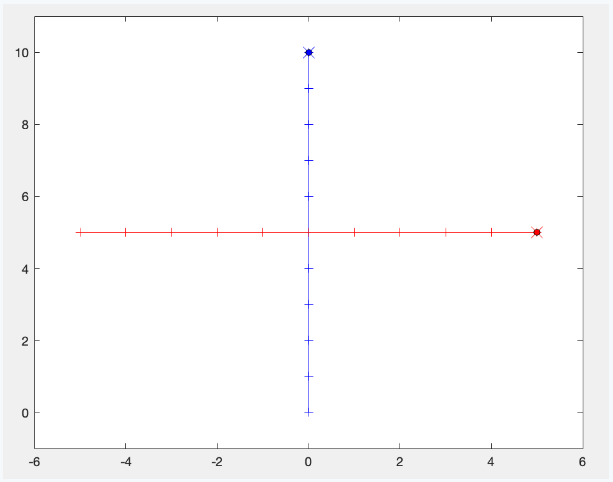

*Figure 6. Simple Controller Example*

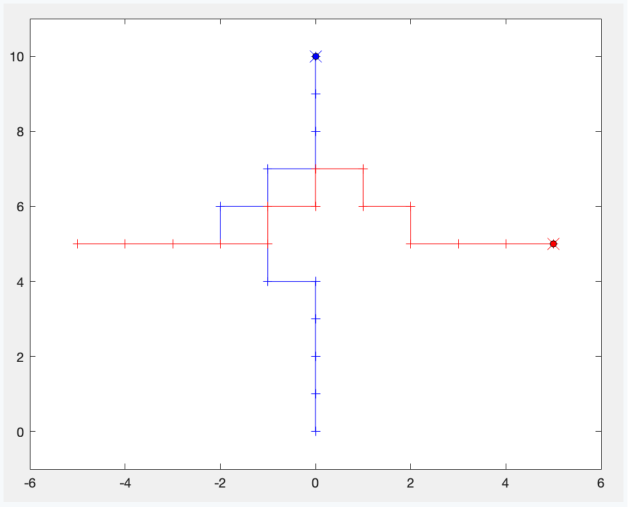

*Figure 7. Complex Controller Example*

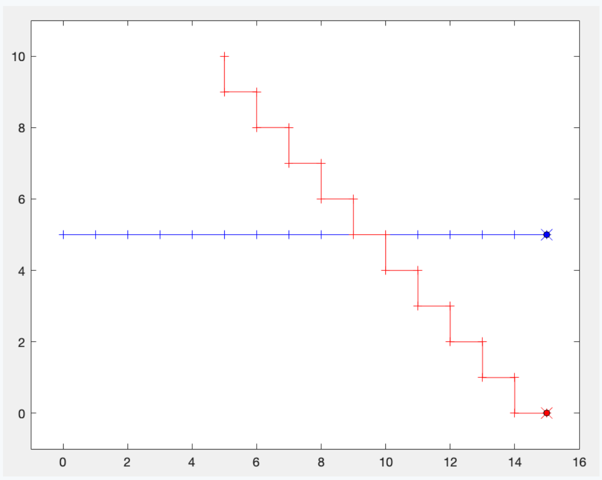

*Figure 8. Simple Controller Example*

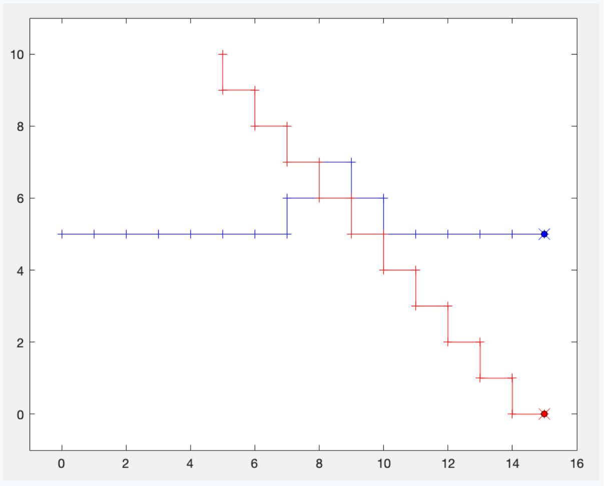

*Figure 9. Complex Controller Example*