Aircraft Collision Avoidance

Bikram Kumar De
Institute for Software Integrated Systems
Vanderbilt University
Course: CS 6376 – Foundations of Hybrid and Embedded Systems

Abstract

In this project, I am designing a 2D aircraft controller that would navigate an aircraft in a 2D airspace to its destination along with avoiding collision with an incoming aircraft in its path. The controller is designed to follow the shortest route possible when traversing from the source to the destination. Both the controllers of the aircraft follow the same algorithm while traversing or avoiding a collision. Along with the controllers, a safety monitor is attached which verifies that the aircrafts have avoided a collision and reached the destination safely.

Introduction

Airplanes were invented in the year 1903 by the Wright Brothers. Over the century since then, aircraft has been improved, remodeled, developed, and made more efficient. Today, it is the fastest mode of transportation. With these advantages, there is a challenging issue of keeping an aircraft safe when it is flying thousands of feet above the surface. There are various issues in mid-air like thunderstorms, turbulence, jet streams, etc. Apart from this, the aircraft needs to make sure that it does not collide with any other aircraft while flying. Modern aviation technology uses a variety of instruments like GPS, radars, transponders to locate itself in space and any other aircraft in its vicinity. While in real life, the aircraft can move in 3 dimensions, in this project we have designed the aircraft to move in a 2D plane only. Also, in our project, we have assumed only possible collisions are head-on and perpendicular. However, in real life aircrafts with different speeds can collide from behind. Furthermore, the aircrafts in our work are assumed to have a constant velocity of 1 km/min. With these constraints, we have designed an aircraft control system that can navigate an aircraft from its source to destination and avoid colliding with another incoming aircraft in its path.

The problem

In this section, we describe the problem and discuss its requirements. We further describe the inputs and outputs of the controller.

Input output specifications

The inputs provided to the aircraft controller are as follows:

- *Source Coordinates:* The user provides the source or starting point of the aircraft as x-y coordinates on the 2D plane.

- *Destination Coordinates:* The user also provides the controller with the required destination of the aircraft as coordinates.
- *Current Location:* The controller continuously feeds itself with the current location of the aircraft which is x-y coordinates on the plane. Initially, the current location of the aircraft is set as the source coordinates.
- *Current Heading:* This is a single number indicating the direction of travel. First, it is initialized with a value depending on the source and destination coordinates to point towards the direction which will lead to the shortest path. It is updated by the controller as required.
- *Message:* One aircraft controller receives a message from any another aircraft controller containing the coordinates and heading of any other aircraft in its communication zone.

The controller generates the following outputs:

- *Updated Location:* The controller continuously generates the location of the aircraft and updates it after each unit of time.
- *Updated Heading:* The controller turns the aircraft either if it is required to navigate towards the destination or to avoid collision with another aircraft in a future collision course. In both these cases the heading of the aircraft is updated by the controller.

Requirements

The controller is expected to meet the following requirements.

- The controller should be able to navigate the aircraft towards the destination following the shortest path possible and make progress on each time unit. It is worthwhile to note that the aircraft is constrained to move in either x or y direction only, it can't move diagonally.
- Whenever an aircraft is detected in the communication zone, the controller should be able to calculate its future position based on its current position and heading. If the future position of both the aircraft coincides, then to prevent a collision, it should be able to plan a new route that results in minimum deviation from the original planned trajectory.
- The controller should be designed such that at every moment, there is at least one allowed move for both the aircrafts that are performing collision avoidance.
- After reaching the required destination, the controller should stop the aircraft from moving.
- The controllers should be universally designed such that the same algorithm applies for both the aircrafts.

My idea

First, I describe a simple controller which I have designed which formed the basis of complete controller designed later. The simple controller helped to identify possible flaws in navigation and test out various scenarios of navigation. Identifying all possible errors with a complete controller would have been challenging with all the functionalities. The simple controller essentially, breaks down the entire project to a step-by-step process.

Simple Controller

As a steppingstone towards developing a complete controller, we first designed a basic version of the controller. In this environment, the airspace contains a single aircraft, so there is no chance of collision. The job of the controller is to navigate the aircraft to the required destination in the shortest path possible. Since, the aircraft can move only in the direction of the axes, we have given a preference of the X axis over Y axis. The aircraft covers the displacement along the X axis and then makes a turn to cover the displacement along the Y axis. This is just a preference and the controller would work just fine if the other axis were preferred. Once, the aircraft reaches its destination and location coordinates of the aircraft matches with the destination coordinates, the controller stops the aircraft and issues a message indicating the same. The algorithm below in Figure 1 shows the functioning of a simple aircraft controller.

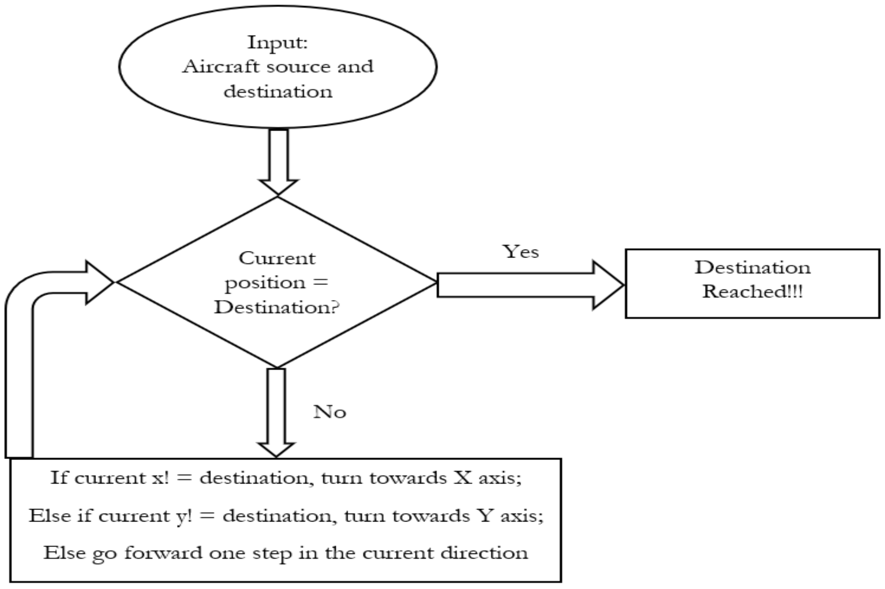
Input:
Aircraft source and destination
Current position = Destination?
Yes
Destination Reached!!!
No
If current x! = destination, turn towards X axis;
Else if current y! = destination, turn towards Y axis;
Else go forward one step in the current direction

Figure 1: Simple Controller

Complete Controller

After a successful test of the basic controller, the complete controller was designed. Here, in this scenario, there are two aircrafts in the airspace starting at the same time. The controller is required to navigate the aircraft from the source to destination without colliding with the other aircraft in the vicinity. Keeping in line with the design of the simple controller, the X direction is preferred over Y. When another aircraft is detected in a collision course, the controller turns the aircraft anticlockwise by 90°. The velocity of both aircrafts is equal and constant. Also, no parallel or rear collisions are allowed. Hence this method of turning works in every possible case. The remaining traversal of the aircraft is same as that of a simple controller. Along with the controller a safety monitor is attached which triggers an error on occurrence of a collision. The algorithm for the complete controller is shown below in Figure 2 as well as the safety monitor is described with Figure 3.

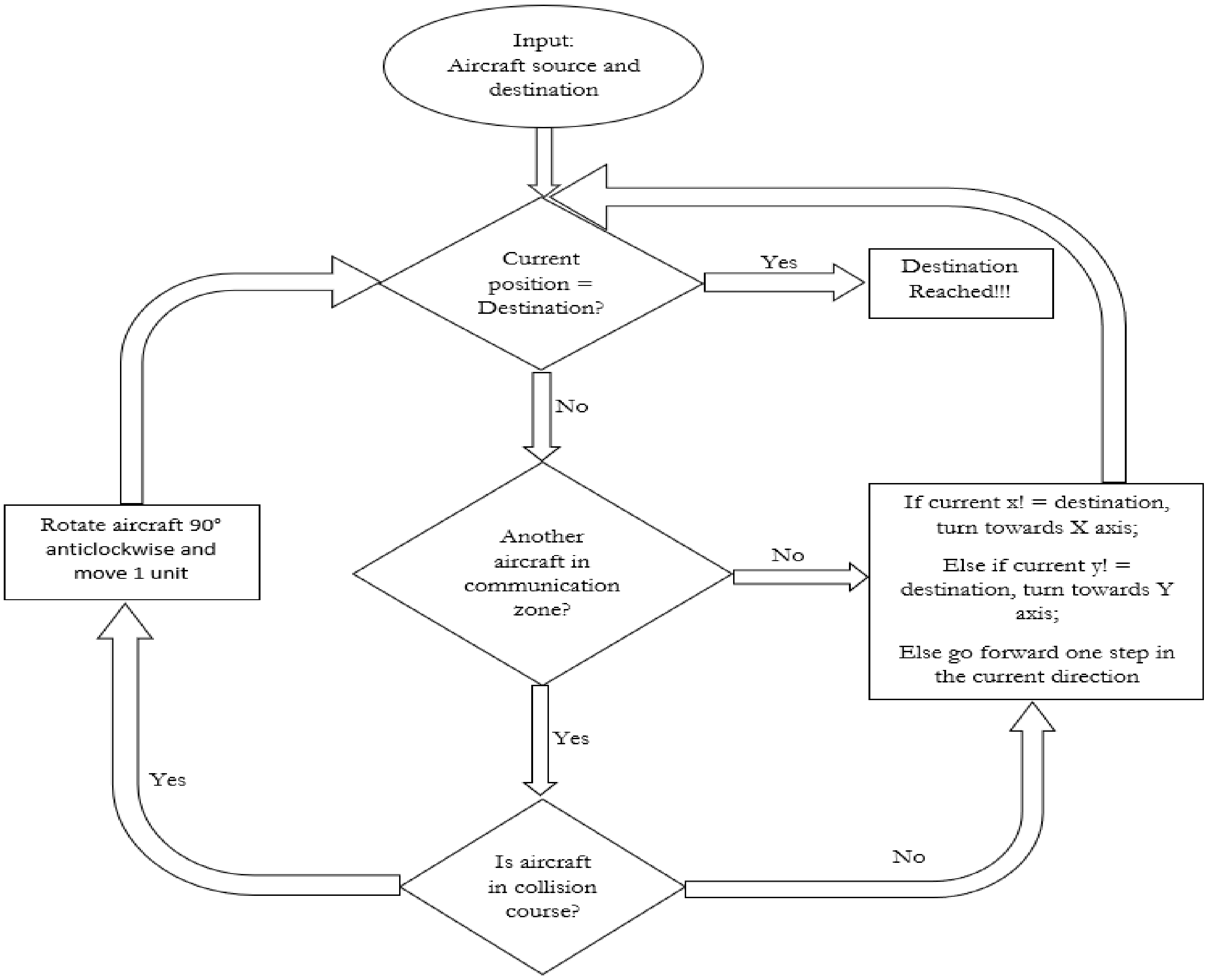
Input:
Aircraft source and destination
Current position = Destination?
Yes
Destination Reached!!!
No
Another aircraft in communication zone?
No
If current x! = destination, turn towards X axis;
Else if current y! = destination, turn towards Y axis;
Else go forward one step in the current direction
Yes
Rotate aircraft 90° anticlockwise and move 1 unit
Yes
Is aircraft in collision course?
No

Figure 2: Complete Controller

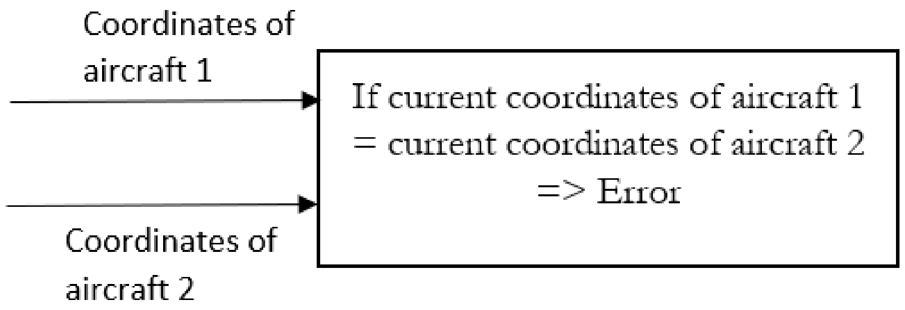
Coordinates of aircraft 1
Coordinates of aircraft 2
If current coordinates of aircraft 1 = current coordinates of aircraft 2 => Error

Figure 3: Safety Monitor

Proof of correctness

In the above sections, we have described the algorithm that we have devised for a controller to navigate itself through a possible collision. In this section, we provide an analytical proof that the algorithm we designed is correct and works for all possible combinations of input.

During any simulation of an aircraft from source to its destination, there are two possible cases for the controller:

i. It does not encounter any aircraft
ii. It encounters another aircraft

In the first case, the aircraft needs to cover a certain displacement in the X and Y direction. These distances travelled in any order would take the same amount of time and eventually the aircraft will reach its destination. For the sake of ease of implementation, we have given a preference to executing the motion in the X direction.

In the second scenario, the aircraft can be in a collision course with another aircraft. This is determined by computing the next position of both aircrafts. If they are in a collision course, the controller turns their respective aircrafts 90 degrees counterclockwise.

In this problem, collision can be of two types, *head on* and *perpendicular*. One aircraft travelling in positive X direction and another aircraft travelling in negative X direction with same Y coordinate may result in a head-on collision. In such a case, both the aircrafts turn 90° anticlockwise. The aircraft travelling in positive X direction will turn towards positive Y direction and the other aircraft would turn in negative Y direction. In this way a head on collision will be avoided.

Let's consider another example in which one aircraft travel along X axis from (-10,0) to (10, 0) and another aircraft travels along Y axis from (0, -10) to (0, 10). This is an example of perpendicular collision. These aircrafts are supposed to collide at origin (0,0). When the 1st aircraft is at (-1, 0) and 2nd aircraft is at (0, -1), the collision avoidance system will get triggered and both aircrafts will turn 90° anti clockwise. The 1st aircraft will move to (-1, 1) and 2nd one would move to (-1, -1). In this way they would avoid collision. Thus, we have shown that under any circumstances, these two aircrafts would not collide mid-air and our collision avoidance algorithm would function flawlessly.

Implementation Details

The collision avoidance system with two aircrafts is implemented in MATLAB. A 30x30 square grid is selected to be the airspace and is numbered [-15,15] and each unit represents 1 km. The grid limitations are selected for representation only and the controller works for any size of the airspace. Time is scaled to make 1 second equivalent to 1 minute. The two aircrafts are represented as *green* and *red solid* squares and their destinations are represented as *stars* of same color. A *hollow* square of same color represents their starting point. The '*Basic_Controller.m*' is a simulation of the simple controller with one aircraft in the airspace. The '*Final_Controller.m*' is the simulation of the complete controller. Another script '*Safety_Monitor_Verifier.m*' is kept which can be used to test the functioning of the safety monitor to ensure it detects collision correctly.

This script doesn't use any collision avoidance algorithm.

There are also three functions which are called in the scripts:

- '*grid_traverse.m*': This performs the navigation of the aircraft from the source to the destination.
- '*avoid_collision.m*': This function is used to steer the aircrafts safely when a probable collision is detected.
- '*safety_check.m*': This function is a safety monitor which will display an error message on occurrence of a collision.

Results

In this section, we show various simulations of the complete that we have designed. For the simple controller, the user needs to input the source and destination coordinates of one aircraft whereas for the complete controller, the user needs to provide details for both the aircraft. In the figures below we have shown the functioning of the complete controller for three different cases: *no collision*, *head on collision* and *perpendicular collision*.

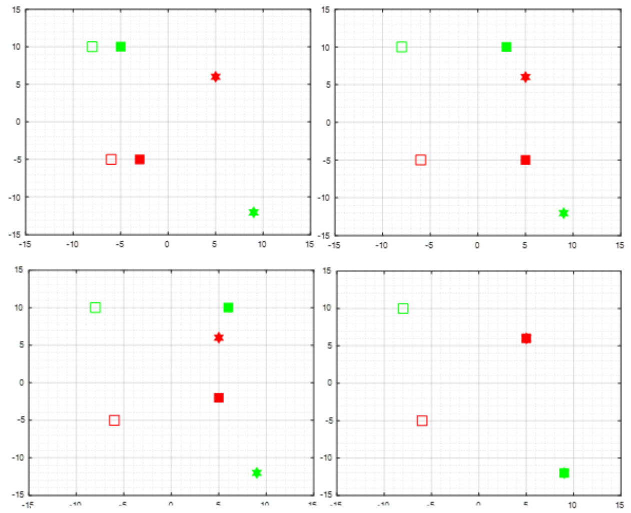
Figure 4: No collision

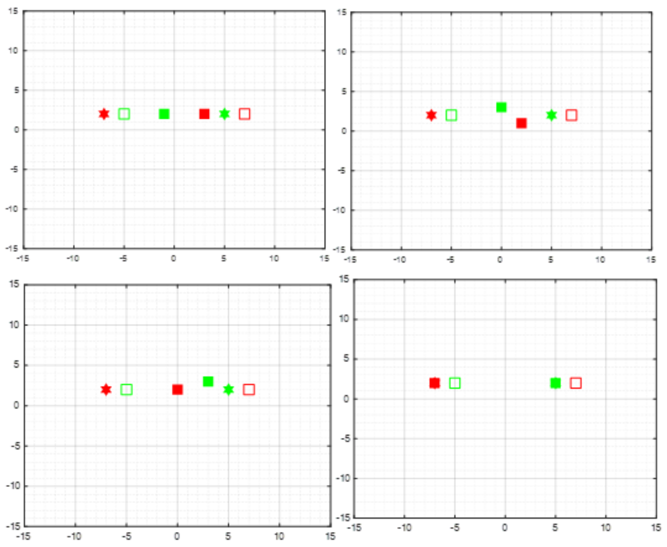
Figure 5: Head on Collision Avoidance

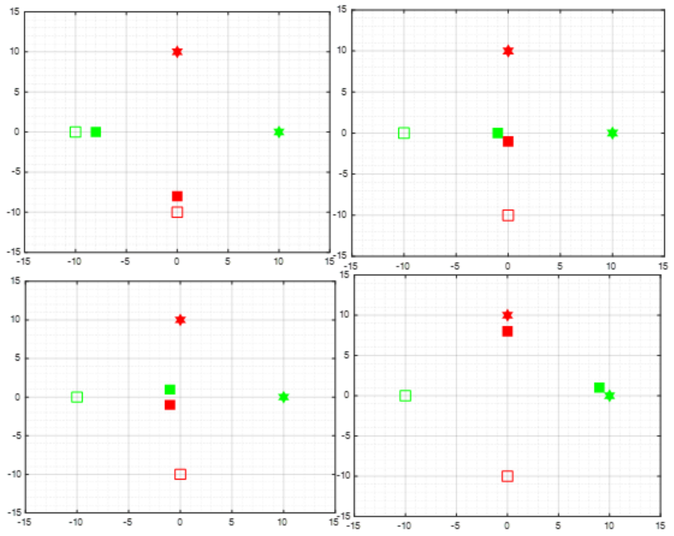
Figure 6: Perpendicular Collision Avoidance

Conclusion

In this project we have designed a simplified version of an aircraft controller. Though the airspace is designed for two controllers, the scripts can be modified with minor changes to accommodate multiple aircrafts and their controllers. This project also helps us understand the daily challenges that an air traffic controller faces in real life with so many aircrafts and increasing obstacles.