An Collision Avoidance Algorithm for a Simplified Aircraft Navigation Problem

Chuilian Kong
Department of Electrical Engineering
and Computer Science
Vanderbilt University
Nashville, Tennessee 37235-1679
Email: chuilian.kong@vanderbilt.edu

Xenofon Koutsoukos
Department of Electrical Engineering
and Computer Science
Vanderbilt University
Nashville, Tennessee 37235-1679
Email: Xenofon.Koutsoukos@vanderbilt.edu

***Abstract*—This paper described a collision avoidance algorithm for planning a safe and relatively short path for a aircraft flying on a 2-D plane. The algorithm is firstly proposed by only satisfying basic liveness requirement of the system and then be further modified to achieve the safety requirement of the system. The algorithm has been tested on a data set of 22,174 sample cases and the result is promising.**

Keywords: collision avoidance, aircraft navigation, algorithm

I. Introduction

The study of automatic navigation for aircraft has a long history. Designing an effective and efficient algorithm for collision avoidance between agents is the core problem in this area. A qualified collision avoidance algorithm should satisfy the following requirements:

1. all aircrafts will eventually arrive their destination.

2. no collisions during the flight.

3*. the path that an aircraft went through is considered to be a shortest path given the information it gathered.

As you may notice, the third requirement is not technically a requirement for the algorithm, rather it is a requirement for the entire aircraft system. After a collision happens, there is a possibility that at least one of the aircraft will no longer be on its shortest path, so there should be no guarantee that all of the aircraft will go through a shortest path. However, what can be guaranteed is all aircrafts will eventually arrive their destination and no collisions during the flight and they will choose their flying path as short as possible.

In the real-world environment, there will be too much complicated noises and interference to focus on solving our core problem. Therefore, in this paper, we will consider a simplified version of the aircraft navigation problem, which basically models the problem to a 2-D plane and makes some of simplification assumptions about the flying velocity and directions.

In this paper, we will first describe our simplified aircraft navigation problem, then we will propose our idea for solving the problem and finally we will show you the algorithms we designed for the problem.

We followed two steps to solve the aircraft navigation problem: firstly, we designed a prototype algorithm which basically ensures the liveness requirement of the system, that is, the aircraft will eventually arrive their destination. And also make sure the path it went through should be the shortest path to the destination. Then we will modify the algorithm by adding some collision avoidance mechanisms to make sure there will be no collisions during the flight.

II. The Problem

The goal of this paper is to design an aircraft controller that navigates the aircraft from source to destination while ensuring that it does not collide with other aircraft in its path. The controller gets information about the current location and the target location of the aircraft. Further, it receives messages from aircraft in its vicinity and uses this information to navigate the aircraft and avoid collision with other aircraft.

In this paper, we will look at a simplified controller design problem as follows:

- The aircraft will fly in a 2-D plane. Its source and destination are integer-valued points in the plane.
- The aircraft flies with a constant velocity v=1km/minuate along either X-axis or Y-axis.
- The aircraft controller can update direction of flight every k=1 minutes. Note that, it can decide to either fly straight or rotate left or right by 90 degrees but not turn back.
- At the beginning of every k=1 minutes,
  - The controller can exchange messages with any aircraft in a square region with side length 2q=4km (communication zone) in the vicinity of the aircraft as shown in the figure 1.
  - Based on the messages received and the current state of the aircraft, the controller can update the direction of flight
- **Collision Avoidance**: Each aircraft has a square region with side length 2d=1km (danger zone) around it such that no aircraft should enter this region at any time during the flight as shown in figure 1.
- The source locations are distinct locations for different aircraft.
- Once an aircraft reaches its destination, it is no longer a threat and collision avoidance is not required for this aircraft. Also, the air craft stops sending messages to the other aircraft in its vicinity.

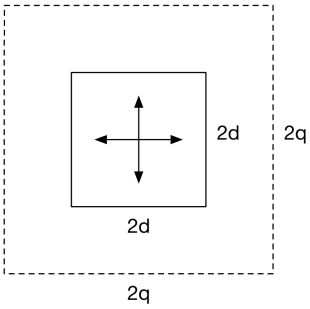


Fig. 1: Illustration of communication zone and danger zone.

Further, we assume that the airspace consists only of two aircraft that start at time t=0. The goal here is to design a controller (same algorithm for both aircraft) that ensures that each aircraft reaches its destination in as small time as possible while ensuring collision avoidance. In the problem, the sources and destinations for aircraft are provided as parameters and the designed controller should work with all such source-destination pairs.

III. The General Idea

There are one liveness requirement and one safety requirement for the system:

Liveness Requirement: both of the aircraft will eventually arrive their destination.

Safety Requirement: no collisions happen during the flight.

It is easier to come up with an idea that is able to design a controller that satisfy the minimum requirement of the problem, that is, to design a controller that navigates the aircraft to its destination. And it is litter bit harder to add a optimization request that each aircraft should go through one of its shortest path to the destination. And it is a lot more sophisticated to take the safety requirement into account, that is, there should be no collision happens during the flight.

I think it is a great idea to solve this problem step by step.Therefore, In next two sections, In order to solve the problem , we will first show you the design of a simple controller, which simply ensures the liveness requirement of the system while omits the safety requirement of the system. In addition, the simple controller will also ensure the paths that both aircraft go through should be the shortest path for them respectively. Then we will make a series of modification to the algorithm and thus enable its ability of collision avoidance.

IV. Design of A Simple Controller

In this section, we will show you the design of a simple controller, which basically ensures that the aircraft reach their destination on a shortest path, while this controller will not provide a mechanism of any sort of collision avoidance.

A. Problem Analysis

The core problem of the design of a simple controller is how to ensure the path an aircraft go through is on a shortest path to the destination. Since we simplified the problem by assuming

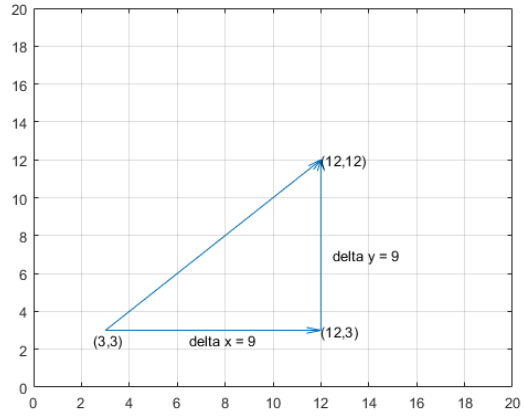


Fig. 2: Illustration of breaking down distance into x, y directions.

an aircraft flies in 2-D plane and it can only fly towards the direction either parallel with x-axis or y-axis, i.e. the direction of an aircraft must be a choice of {0, 90, 180, 270}, it could be easier to analyze the problem by breaking the distance to destination into 2 parts: remaining kilometers along x-axis $\mathbf{\Delta x}$ and along y-axis $\mathbf{\Delta y}$. We illustrate this kind of breaking down in Figure 2. In this way, we can easily see the solution to this shortest path problem: fly along the direction , for instance, if $\mathbf{\Delta x} < 0$, which means there are remaining kilometers on negative x direction, then we could let the aircraft fly at the direction of 180 to shorter the remaining distance. if $\mathbf{\Delta y} > 0$, which means there are remaining kilometers on positive y direction, then we could let the aircraft fly at the direction of 90 to shorter the remaining distance. if both $\mathbf{\Delta x} = 0$ and $\mathbf{\Delta y} = 0$, which indicates the aircraft has arrived the destination, then it should be stopped. If both $\mathbf{\Delta x} \neq 0$ and $\mathbf{\Delta y} \neq 0$, the controller may either set a priority to one of the axises or simply non-determinatively choose a direction to fly.

This solution for shortest path seems to be a good plan for designing the simple controller. However, It has one tiny issue: what if the initial direction is the opposite direction of the only best direction generated by the idea above. For instance, consider a aircraft with initial position $(x, y) = (10, 0)$ and the destination position $(x, y) = (0, 0)$ and the initial direction $/theta = 0$. We know the best choice of flying direction is along negative x, but we have the limitation that there are no turning back option for the controller. We can either turn right two times or turn left two times to achieve the same purpose of turning back. Therefore, for this situation, the controller may decide either turn right or turn left and let the controller of the next execution takes care of the rest.

B. Algorithm

Based on the analysis above, we can safely give the algorithm of a simple controller as follows:

Algorithm 1 A Simple Controller

$\Delta x = in.xd - in.x$
$\Delta y = in.yd - in.y$
if $\Delta x > 0$ **then**
  if $in.\theta = 0$ **then return** *go straight*
  if $in.\theta = 90$ **then return** *turn right*
  if $in.\theta = 270$ **then return** *turn left*
if $\Delta x < 0$ **then**
  if $in.\theta = 180$ **then return** *go straight*
  if $in.\theta = 90$ **then return** *turn left*
  if $in.\theta = 270$ **then return** *turn right*
if $\Delta y > 0$ **then**
  if $in.\theta = 90$ **then return** *go straight*
  if $in.\theta = 0$ **then return** *turn left*
  if $in.\theta = 180$ **then return** *turn right*
if $\Delta y < 0$ **then**
  if $in.\theta = 90$ **then return** *go straight*
  if $in.\theta = 0$ **then return** *turn right*
  if $in.\theta = 180$ **then return** *turn left*
  return *turn left*

C. Simulation

We run this algorithm on five test cases in Matlab and the result is promising. Part of the result shows on Figure 3. In each test case, the system will generate the actual path on which each aircraft fly to the destination. In all five cases, aircrafts eventually arrived the destination and each path an aircraft went through is justified as a shortest path for them respectively.

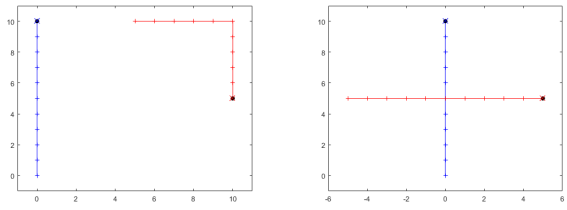
As you may notice, the test case 3 involved the tiny issue we discussed above: the initial direction is the opposite direction of the only best direction. The controller has to make two separate turns in order to achieve the same effect of turning back, and the path it went through is the shortest path under this limitation.

D. Conclusion

In this section, we showed the design of a simple controller which is capable of finding the shortest path to the destination and navigating the aircraft along this path thus eventually arriving the destination. This controller satisfies the basic liveness requirement for the system but is not consistent with the safety requirement of the system, i.e. this controller does not provide any mechanisms to avoid collision between aircrafts. We will further develop some mechanisms for collision avoidance and modify our algorithm to make it possible to both inherent the good features for finding shortest path and add some new features that enforce the safety requirement.

V. Design for a Complete Controller

In this section, we will show you the design of a complete controller, which not only ensures every aircraft in the system will eventually arrive their destinations, but also ensures there will be no collision during the flights. Additionally, the complete controller will ensure the path it goes through is considered to be the shortest to the best of its knowledge.



(a) test case 1 (b) test case 2

(c) test case 3 (d) test case 4

Fig. 3: Tests for simple controller

A. Problem Analysis

1) Definition of Collision and Design of Safety Monitor: In order to design an algorithm that avoid collision between two aircraft, firstly we have to formally define what exactly is a collision. As is described in the simplified version of the problem, the aircraft flies in a 2-D plane and each aircraft has a square region with side length 2d=1km that is considered to be the "danger zone" of the aircraft, any other aircrafts appear within the "danger zone" is consider to be a collision. For instance, as shown on the Figure 4 at the time t_0, the aircraft A is on the position $(5, 5)$, then the danger zone of this aircraft should be a square: $4.5 < x < 5.5, 4.5 < y < 5.5$. Since based on our simplification assumptions, the aircraft can only appear on the coordinate with integer x, y value, the only collision chance is, aircraft B appear on the same position $(5, 5)$ just as aircraft A does. In fact, based on our definition of "danger zone", collision happens only when two aircrafts located at exactly the same position, and we can thus write the algorithm for the safety monitor as follows:

Algorithm 2 Safety Monitor

if $in(1).x = in(2).x$ **and** $in(1).y = in(2).y$ **then return** *false*
else return *true*

2) Definition of Vicinity and Communication between Aircrafts: Similar to "danger zone", for each aircraft there is a "communication zone" in which the aircraft is able to communicate and share its information with each other. The sharing information including current location and destination and current direction of motion. The communication zone is much more larger than the "danger zone", it is a square that has

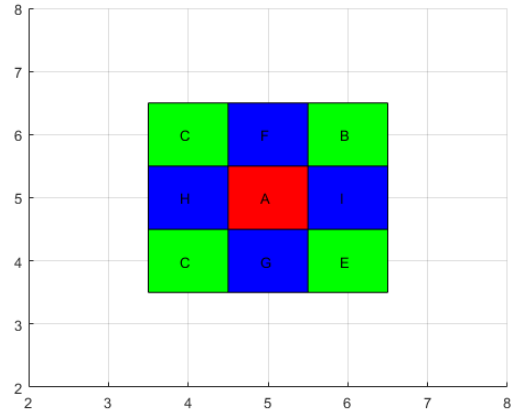


Fig. 4: Illustration of danger zone: collision happens only when two aircraft located at exactly the same position.

sides length of 4 km. Any aircrafts enter this zone will share their data will each other and the decision that the controlling algorithm makes is based on the shared data of both aircrafts.

B. Modification

Finally here, we will show you how to modify the algorithm in order to achieve collision avoidance among aircrafts.

The existing algorithm discussed above provides us a good solution for finding a shortest path to the destination, if no collision occurs, the controller should keep using the existing algorithm to ensure the aircraft is on the fastest route to the destination. Therefore, what we really need to modify is to add a procedure that takes care of upcoming collisions.

Here I will show you the detail of modification:

1) Collision detector: To avoid a collision, we need a procedure to detect it in the first place. So the first part of the modification is to add a collision detector onto the controller. Here is how it works, when a aircraft receives the message from the other in its vicinity, it will first calculate the next position of itself and the other aircraft, according to the existing procedure, then if it finds both of them have the same next position, a collision detected, the controller thus will not use the existing procedure to plan the path, instead it will jump to the procedures described in the following section.

2) Choices an Aircraft has: Before moving forward to further discuss the core procedure of avoiding collision, we need to stress the concept of "choices an aircraft has" when it making decisions to ensure keep flying on the fastest route. According to our analysis of the original shortest path algorithm, we know that the aircraft may have choices when there are more than one fastest route to the destination. for instance, consider an aircraft currently located at $(0, 0)$ and its destination is $(1, 1)$ and its current moving direction is 0. Both of $\mathbf{\Delta x} \neq 0$ and $\mathbf{\Delta y} \neq 0$, thus the aircraft can either choose to go straight along x-axis or turn left to be along y-axis, both of the choices will keep the aircraft on the fastest route to destination. As you may notice, based on basic logic, the choices that a aircraft may have is up to two and if there are two choices, one must be along x-axis and the other must be along y-axis. The choices that an aircraft has is an important concept when discussing the collision avoidance procedure, so we have to explain it here.

3) Collision Avoidance Procedures: After detected a upcoming collision, the controller needs to follow a procedure to avoid it. Let's break it down into two cases:

case 1: two aircrafts are going to collide face to face.

case 2: one aircraft is going to hit on the other's flank.

Let's first talk about the case 1, as shown in Figure 5a,if two aircrafts are going to collide face to face, there are two situations:

1. one of them has another choice (to keep fly on the fastest route), in this case, just take the other choice instead of the current one, since the other choice must lead to avoidance of upcoming collision. For instance, if two aircraft are about to collide along x-axis, and one of them has an alternative choice, the controller will simply let it choose the alternative choice (this choice must be along y-axis based on logic) and the other aircraft will just keep moving like nothing happened.

2. neither of them has a another choice (to keep fly on the fastest route), in this case, someone has to sacrifice. And we take their shared information to determine which one should sacrifice. For instance, In our design, we just simply choose the one with the higher current moving angle to modify its flying path and the other one will keep moving like nothing happened.

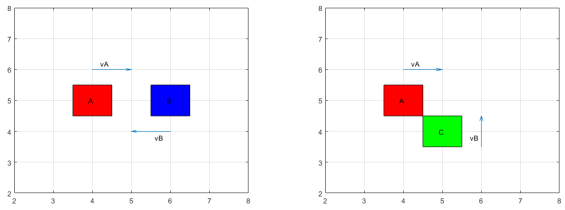
Then as shown in Figure 5b, if one aircraft is going to hit on the other's flank, there are also two similar situations but more complicated procedure to handle it:

1. one of them has anther choice (to keep fly on the fastest route) that is the opposite direction of the other aircraft, in this case, just take the other choice instead of the current one. As you may notice, the condition here is slightly different from "face to face" collision. It is because if the other choice is the same direction of the other aircraft, and we take it, this may lead to further collision of the same situation. For instance, if the aircraft A that is moving toward +x is going to hit the aircraft B that is moving toward +y, and aircraft A has an another choice of moving toward +y, which is the same direction of the aircraft B, if the aircraft A takes the +y to avoid collision at this time, it may want to moving toward +x the next time, and if the aircraft B is still moving toward +y, the same situation again, and the situation will repeatedly appear until one of them use up their Δy.

2. neither of them has a another choice (to keep fly on the fastest route) that is the opposite direction of the other aircraft, in this case, we will use the same technique to achieve collision avoidance, that is, we will choose one aircraft to sacrifice, and we make this decision by their shared information.

C. Examples

Let's illustrate our collision avoidance algorithm by examples. As shown in Figure 6, we give four examples to illustrate our collision avoidance algorithm.



(a) Collision: "Face to face" (b) Collision: "Hit on flank"

Fig. 5: Two basic situations of a collision

In the first example, as shown in Figure 6a, there is no collision, thus the controller just keep the controlling procedure like the simple controller does, navigating each aircraft to the fastest route toward destination.

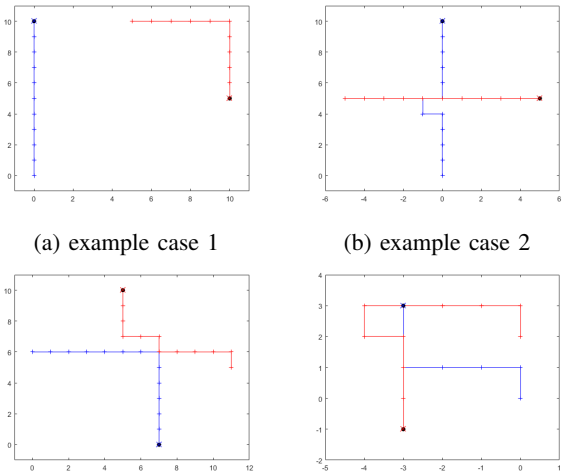
Then in the second example, as shown in Figure 6b, there is a "hitting by flank" collision detected by the controller when the red aircraft A is at $(0, 4)$ while the blue aircraft B is at $(-1, 5)$, the intended moving direction of A is 0, and the intended moving direction of B is 90. Since neither of them has a alternative choice to stay on the fastest route, the aircraft with the higher moving angle, which is B, have to make a sacrifice. It simply move one step towards the opposite direction of A. On the other hand, aircraft A just keep moving towards direction 0. After one step, the incoming collision has been avoided.

Then for the third example, as shown in Figure 6c, there is a "face to face" collision detected by the controller when the red aircraft A is at $(7, 6)$ while the blue aircraft B is at $(5, 6)$, the intended moving direction of A is 180, and the intended moving direction of B is 0. Luckily this time aircraft A has an alternative choice to keep on the fastest route, which is move toward +y instead of -x. Then according to our procedure, aircraft A makes a right turn towards +y and aircraft B go straight to keep on its fastest route. And a potential collision thus be avoided.

The last case as shown in Figure 6d, is also a example for "face to face" collision, but there is a slight difference against the former example, that is, the two aircraft are going to collide when they are making a turn. However, the procedure to handle this situation remains the same. A potential collision is detected by controller when the red aircraft A is at $(-3, 3)$ while the blue aircraft B is at $(-3, 1)$, and the intended moving direction of A is 270 and the intended moving direction of B is 90. Since neither of them has a alternative choice to stay on the fastest route, the aircraft with the higher moving angle, which is A, have to make a sacrifice. It simply move one step towards direction 180. On the other hand, aircraft B just keep moving towards direction 0. After one step, the incoming collision has been avoided.

VI. Experiment and Result

The collision avoidance algorithm we proposed in this paper has been tested on a data set that including 22,174 sample inputs, and our algorithm passed all of the test cases.



(a) example case 1 (b) example case 2

(c) example case 3 (d) example case 4

Fig. 6: Examples for complete controller

A. Conclusion

In this section, we proposed our collision avoidance algorithm based on the existing navigation algorithm. We equip it with procedures to handle potential collision situations. Basically, we divided collision situations into to cases, the first case is so-called "face to face" collision and the second case is hit by flank. In the first case, the controller will first see if either of them has another choice to stay on fastest route, if there is, take it (but not both), if there is not, just choose one to sacrifice. And the procedure to handle the second case is the similar except the alternative choice has to be the opposite direction of the other aircraft. And we run our algorithm on a large data set, all test cases are passed.

VII. Conclusion

In this paper, we first proposed a prototype algorithm to navigate aircrafts to keep flying on the fastest route towards their destination. Then we modified it by adding a series of collision avoidance procedure to make sure the controller is capable of navigating the aircraft to stay on the fastest route as much time as possible and reroute it only when a collision is going to happen. We tested our algorithm on a large data set and the result is promising.

Although our algorithm has theoretical explanation and experimental support, it's hard to apply this naive design into real-world application because there are too much simplification assumptions to this problem such as "the aircraft flies on a 2-D plane" and "the aircraft fies with a constant velocity. Further work should be devoted to loose those constrains and make multiple modifications to our algorithm in order to maintain its effectiveness in the new sophisticated model.

Acknowledgment

Thanks for the help from Professor Xenofon Koutsoukos and TA Xingyu Zhou during the entire semester!

REFERENCES

[1] Rajeev Alur, *Principles of Cyber-Physical Systems*, 1st ed. The MIT Press, 2005.