Aircraft Automated Collision Avoidance

Ronald Picard
CS 6376 Hybrid and Embedded Systems
Vanderbilt University
Nashville, TN, USA
ronald.s.picard@vanderbilt.edu

Aircraft collision avoidance is a pervasive need in this modern age of flight. We present a 2-D automated collision avoidance flight controller for automated collision avoidance amongst 2 aircraft. This controller is designed to satisfy safety and liveness requirements, implemented using Simulink and Stateflow, and verified using safety and liveness monitors.

Keywords— Aircraft automated Collision Avoidance, Simulink, Simulation, Analysis, Safety Requirements, Liveness Requirements, Verification, Validation, Controls

I. INTRODUCTION

Aircraft collision avoidance algorithms are important to the safe flight in this modern age. If multiple aircraft are in the air at the same time, catastrophic problems can occur if one aircraft were to run into another. To avoid this aircraft controllers must be able to automatically sense and avoid nearby aircraft by altering their trajectory before such an issue can occur. The focus of this paper will be on the development and analysis of a 2-D flight controller with built-in automated collision avoidance.

II. BASIC CONCEPTS

A. 2-D Flight Concepts

For the purposes of automated collision avoidance flight may be simplified to three primary concepts; orientation, direction, and motion. The orientation of an aircraft in 3-D space is usually described with the terms roll, pitch, and yaw shown in Figure 1. These refer to the angular rotations about the x, y, and z axes. However, for the 2-D case we can drop roll and pitch, leaving only yaw about the y axis as illustrated in Figure 2. This restricts orientation to a single degree of freedom about the z axis and restricts the aircraft position to the x-y plane as illustrated in Figure 3.

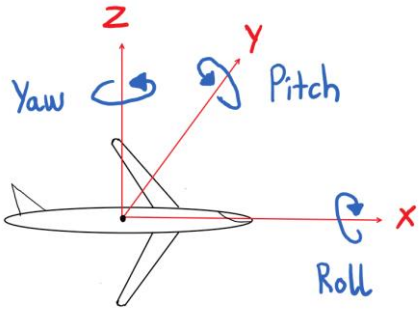

Figure 1: Orientation of Aircraft 3-D

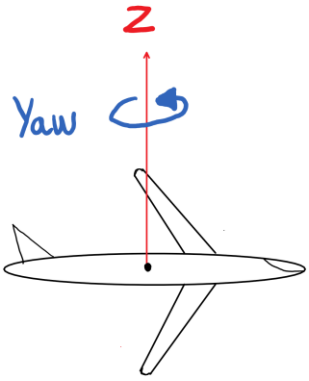

Figure 2: Orientation of Aircraft in 2-D

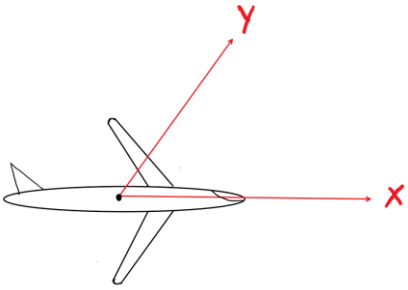

Figure 3: x-y plane

The orientation state-space may be reduced further by restricting the direction of flight to only along the x and y axes, restricting orientation angles to 0, 90, 180, and 270 degrees as illustrated in Figure 4.

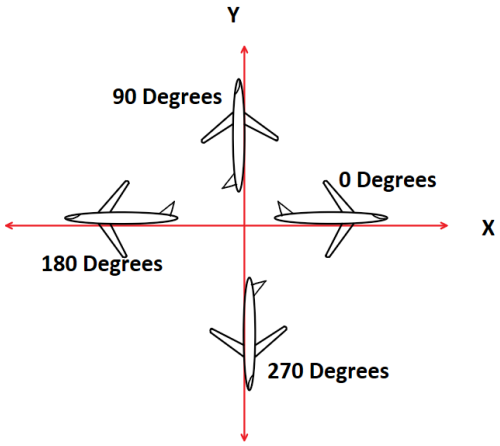

Figure 4: Restricted X and Y Axes Direction and Orientation

To simplify the motion of an aircraft a constant velocity may be assumed.

B. Simulink

Simulink is a model-based design and simulation tool that provides a graphical framework for developing and simulating control software. The tool allows for the model-based creations of signal flow diagrams in which results can be displayed to a graph. The tool allows simulation time steps to be fixed-step. [1] The following are list of Simulink Components used for software associated with this paper.

- Constant
- Scope
- Subsystem
- Delay

C. Stateflow

Stateflow is a Simulink integrated tool that allows the creation of state-machines diagrams. The diagrams can interface with the signal flows at each timestep in a simulation. Stateflow supports extended-state machine notation. [2] The following are list of Stateflow Components used for software associated with this paper.

- Chart
- Transition
- Junction

D. Synchronous Model

Synchronous components receive inputs and produce outputs (execute) based on a sequence of rounds. [3] The fixed-time step simulation capability of Simulink allows the synchronous model to be utilized when designing components.

E. Continuous-Time Model

Physical components receive inputs and produce outputs based on a continuous-time model. The continuous-time model is like the synchronous model in that components execute in a sequence of rounds; however, it differs in that the time steps are infinitely small because time is continuous. [3] Therefore, continuous-time components react by approximating the solution to a differential equation.

F. Hybrid Systems

Hybrid Systems involved the integration of multiple reactive component models. [3] For this Aircraft Collision Avoidance Controller simulation the two integrated component models are the Synchronous and continuous-time models.

G. Saftey

Safety requirements are a formal method of describing the correctness of the system. Safety requirements are written as Boolean-valued statements (known as invariants) that must not be violated by any reachable state. The intuition of safety requirements is to ensure that nothing bad every happens (i.e. two aircraft run into each other). [3]

H. Liveness

Liveness requirements are a formal method of describing the behavior of the system over time. Safety requirements are written using Linear Temporal Logic (LTL). The intuition of liveness requirements is to ensure that something good eventually happens. [3]

III. Problem

A. Requirements

An indexed list of a natural language requirements for this 2-D controller design (provided by the class assignment []) are provided below.

1. The aircraft can fly in a 2-D plane. Its source and destination are integer-valued points in. [4]
2. The aircraft flies with a constant velocity v = 1 km/minute along either X-axis or Y-axis. [4]
3. 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. [4]
4. At the beginning of every k = 1 minutes,
    1. The controller can exchange messages with any aircraft in a square region with side length 2q = 6 km (communication zone) in the vicinity of the aircraft as shown in figure below. [4] Refer to Figure 5 for a communication zone and danger zone diagram. (*Note: This has been adjusted from*

Identify applicable funding agency here. If none, delete this text box.

*the referenced requirements from 4 km to 6 km to avoid a failing edge case. See **VIII**, **B** for details.*)

2. Based on the messages received and the current state of the aircraft, the controller can update the direction of flight. [4]

5. Collision Avoidance: Each aircraft has a square region with side length 2d = 1 km (danger zone) around it such that no aircraft should enter this region at any time during the flight as shown in figure below. Refer to Figure 5 for a communication zone and danger zone diagram. [4]
6. The source locations are distinct locations for different aircraft. [4]
7. Once an aircraft reaches its destination, it is no longer a threat and collision avoidance is not required for this aircraft. Also, the aircraft stops sending messages to the other aircraft in its vicinity. [4]

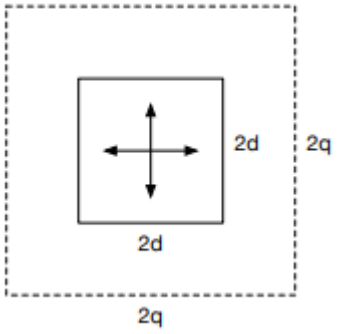

Figure 5 Communication Zone and Danger Zone [4]

8. 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. [4]

B. Formalized Saftey Requirements

The primary safety requirement of this flight controller this that no aircraft should ever enter 1000 m danger zone of another aircraft. This invariant can be formally written:

- ((X_State_Aircraft – Danger_Zone <= X_Incoming_Aircraft) Λ (X_Incoming_Aircraft <= X_State_Aircraft + Danger_Zone)) ¬ = True
- ((Y_State_Aircraft – Danger_Zone <= Y_Incoming_Aircraft) Λ (Y_Incoming_Aircraft <= Y_State_Aircraft + Danger_Zone)) ¬ = True

C. Formalized Liveness Requirements

The primary liveness requirement of this flight controller is that each aircraft must eventually (◊) reach their destination. This liveness requirement may be formally written as:

- ◊(X_State_Aircraft = Aircraft_X_Dest Λ Y_State_Aircraft = Aircraft_Y_Dest)

IV. AIRCRAFT COLLISION AVOIDENCE SOLUTION

To satisfy the requirements above we utilize Simulink and Stateflow to build up two aircraft subsystems and a communications environment subsystem inside of the Synchronous (fixed-step) Simulink model. The aircraft subsystems are duplicates of each other. Inside of each aircraft subsystem is a Stateflow chart containing the aircraft collision avoidance controller logic. The controller logic ensures that the aircraft meets the requirements above while traveling from and initial state to a final state. This includes formalized safety and liveness requirements. Inside of the communications subsystem are two identical Stateflow charts that contain the communication environment logic. The communication environment logic simulates the physical message environment by marking whether a message is within the communication zone of the aircraft at a given step in the simulation.

V. INPUT, OUTPUT, STATE COMPONENT SPECIFICATION DETAILS

A. Aircraft Subsystem Specification

The designed specification for the aircraft subsystem is bulleted below.

- Inputs
  - **Constant Integer {0, 90, 180, 270}** Initial Orientation (degrees) of Aircraft
  - **Constant Integer** Initial X Coordinate (m) Aircraft
  - **Constant Integer** Initial Y Coordinate (m) Aircraft
  - **Constant Integer** X Destination Coordinate (m) Aircraft
  - **Constant Integer** Y Destination Coordinate (m) Aircraft
  - **Integer** X Message In (m)
  - **Integer** Y Message In (m)
  - **Integer {0, 90, 180, 270}** Orientation Message In
  - **Integer {0 1}** Message Received In
  - **Integer** Rounds to Avoid Upon Communication
  - **Constant Integer** Danger Zone In (m) = 500 (m)
  - **Constant Integer** Turn Time In = 1000 (m)
- Outputs
  - **Integer** X Out (m)
  - **Integer** Y Out (m)
  - **Integer {0, 90, 180, 270}** Orientation Out (m)
  - **Integer {0 1}** Danger Zone Breached out
  - **Integer** X Message Out
  - **Integer** Y Message Out
  - **Integer {0, 90, 180, 270}** Orientation Message Out

B. Aircraft Collision Avoidence Controller Specification

The designed specification for the aircraft collision avoidance controller is bulleted below.

- Inputs
  - **Constant Integer {0, 90, 180, 270}** Initial Orientation (degrees) of Aircraft
  - **Constant Integer** Initial X Coordinate (m) Aircraft
  - **Constant Integer** Initial Y Coordinate (m) Aircraft
  - **Constant Integer** X Destination Coordinate (m) Aircraft
  - **Constant Integer** Y Destination Coordinate (m) Aircraft
  - **Integer** X Message In (m)
  - **Integer** Y Message In (m)
  - **Integer {0, 90, 180, 270}** Orientation Message In
  - **Integer {0 1}** Message Received In
  - **Integer** Rounds to Avoid Upon Communication
  - **Constant Integer** Danger Zone In (m) = 500 (m)
  - **Constant Integer** Turn Time In = 1000 (m)
- Outputs
  - **Integer** X Out (m)
  - **Integer** Y Out (m)
  - **Integer {0, 90, 180, 270}** Orientation Out (m)
  - **Integer {0 1}** Danger Zone Breached out
- States

- **Integer** Y_State
- **Integer** X_State
- **Integer {0, 90, 180, 270}** Orientation_State
- **Integer** Rounds_State
- **Integer** Turning State

C. *Communication Environement Subsystem Specification*

The designed specification for the communication environment subsystem is bulleted below.

- Inputs
  - **Constant Integer** Comm Zone In = 3000 m
  - **Integer** X In Aircraft 1
  - **Integer** Y In Aircraft 1
  - **Integer** X Message In Aircraft 2
  - **Integer** Y Message In Aircraft 2
  - **Integer {0, 90, 180, 270}** Orientation Message In Aircraft 2
  - **Integer** X In Aircraft 2
  - **Integer** Y In Aircraft 2
  - **Integer** X Message In Aircraft 1
  - **Integer** Y Message In Aircraft 1
  - **Integer {0, 90, 180, 270}** Orientation Message In Aircraft 1
- Outputs
  - **Integer** X Message Out From Aircraft 2
  - **Integer** Y Message Out From Aircraft 2
  - **Integer {0, 90, 180, 270}** Orientation Out From Aircraft 2
  - **Integer {0 1}** Send Message Out From Aircraft 2
  - **Integer** X Message Out From Aircraft 1
  - **Integer** Y Message Out From Aircraft 1
  - **Integer {0, 90, 180, 270}** Orientation Out From Aircraft 1
  - **Integer {0 1}** Send Message Out From Aircraft 1

D. *Communication Environement Specification*

The designed specification for the communication environment is bulleted below.

- Inputs
  - **Constant Integer** Comm Zone In = 3000 m
  - **Integer** X In Aircraft
  - **Integer** Y In Aircraft
  - **Integer** X Message In Aircraft
  - **Integer** Y Message In Aircraft
  - **Integer {0, 90, 180, 270}** Orientation Message In Aircraft
- Outputs
  - **Integer** X Message Out From Aircraft
  - **Integer** Y Message Out From Aircraft
  - **Integer {0, 90, 180, 270}** Orientation Out From Aircraft
  - **Integer {0 1}** Send Message Out From Aircraft

VI. HYBRID SYSTEM MODEL DETAILS

A. *Synchronous Simulation Model*

The simulation model follows a synchronous (fixed-step) model in which each simulation time step is equivalent to $1/1000^{th}$ of a minute of physical system time.

B. *Synchronous Controller Model*

The reactive controller algorithm runs as a synchronous model where each time step is equivalent to 1 simulation time step. The aircraft travels at a constant velocity of 1 km/min, and therefore moves 1 m per 1 simulation time step. Therefore, the controller may update its internal states every 1 simulation time step, or equivalently 1 m travelled.

C. *Synchronous Enviroment Model*

The communications environment model follows a synchronous model in which the time step is equivalent to the simulation time-step. A real communications environment follows a complex continuous-time model, however it is modeled as a synchronous model here for simplicity, since it does not affect the results of the aircraft controllers.

D. *Continous-Time Physical Model*

The velocity of the aircraft follows a continuous-time model. However, because the velocity (dx/dt) is a constant 1 km/min (i.e. 1/1000 m/s), and the synchronous controller runs off of a $1/1000^{th}$ of a second timestep, the controller can calculate the position of the aircraft at any given point and likewise adjust it by incrementing or decrement its x or y position according to its current orientation.

VII. ALORITHMS DETAILS

A. *AircraftCollisionAvoidanceController (Top-Level)*

The *AircraftCollisionAvoidanceController* is built using hierarchical extended-state machines. At the top-level, there are two modes; a C*ontroller* mode containing the flight control and collision avoidance logic, and a *Final* mode in which the either the destination has been reached or the danger zone around the aircraft has been breached. Refer to Figure 6 for the *AircraftCollisionAvoidanceController* Top-Level structure Stateflow implementation.

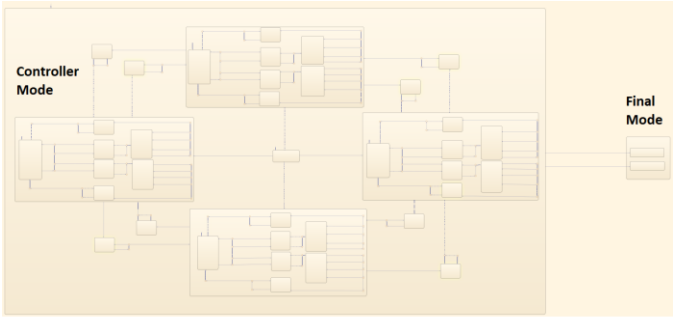

Figure 6: *AircraftCollisionAvoidanceController (Top-Level)*

B. *Flight Control Alorithm*

The flight control algorithm logic is contained inside of the top-level C*ontroller* mode. One level down (second level) in the C*ontroller* mode, we initialize the controller in the *Initialize* mode. Refer to Figure 7 for the Stateflow implementation.

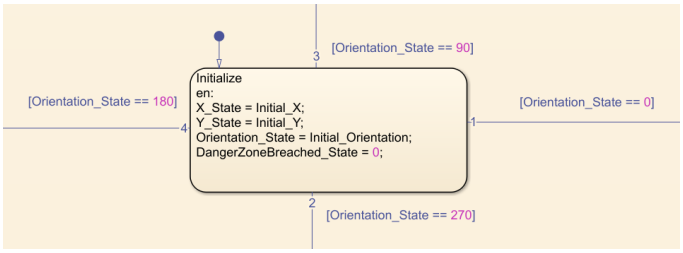

Figure 7: *Initalize* Mode For Flight Controller

After initialization, the state transitions to one of the 4 surrounding second level *CollisionAvoidanceMonitor* modes depending on the initial flight orientation of the aircraft (0, 90, 180, 270 degrees). Refer to Figure 8 for the Stateflow implementation *CollisionAvoidanceMonitor* for the 0 degree orientation.

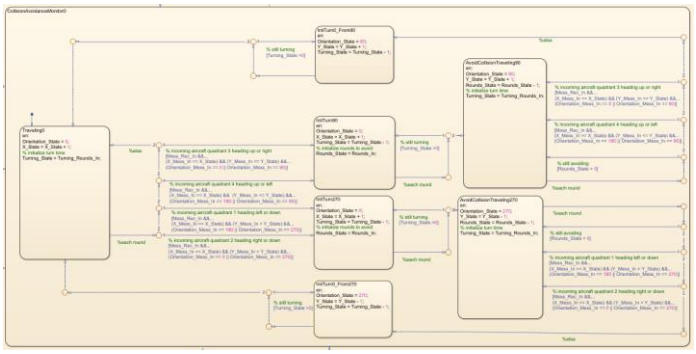

Figure 8: *CollisionAvoidanceMonitor* Mode

Inside the a *CollisionAvoidanceMonitor* mode the state switches to a *Traveling* mode (3 level down in the *Controller* mode) in which the mode increments or decrements the position of the plane 1 meter along either the x axis or y axis, respective to the orientation, every 1/1000th of a minute (or simulation step). Refer to Figure 9 for at the Stateflow implementation of the *Traveling* mode.

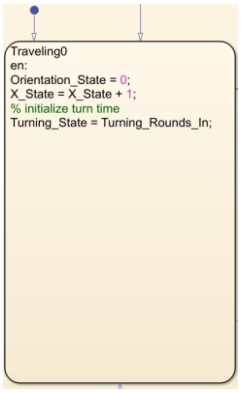

Figure 9: *Traveling* Mode

While inside of a *CollisionAvoidanceMonitor* and traveling along an axis there are 3 possible transitions out of the *CollisionAvoidanceMonitor* that may occur due to the aircraft x and y state position relative to the x and y destination positon. The possible transitions for the *CollisionAvoidanceMonitor0* for the 0 degree orientation are bulleted below.

1. If [(X_In<=X_State) && (Y_In=>Y_State)], then transition to *CollisionAvoidanceMonitor90*
2. If [(X_In<=X_State) && (Y_In<Y_State)], then transition to *CollisionAvoidanceMonitor270*
3. If [(Y_In==Y_State) && (X_In==X_State)], then transition to *Final*

Since the turn requires 1000 meters to complete, the turn modes wait 1000 meters to complete to transition to the next modes and adjust their orientations. Similar logic applies the other *CollisionAvoidanceMonitor* modes. This enables the aircraft to always traverse to the final destination in shortest path possible, given the orientation constraints.

C. Collision Avoidence Control Alorithm

The collision avoidance algorithm is contained in and tailored to the 4 *CollisionAvoidanceMonitor* modes. To describe the logic we exam the *CollisionAvoidanceMonitor0* mode from Figure 8. Inside of this mode we have 7 modes. The initial mode is Traveling0, which maintains a transition into itself allowing causing it to increment the x position state of the aircraft 1 meter every simulation time step.

However, if a message from the second aircraft is received and its orientation is facing the current aircraft in the x or y direction, then a transition fires to a turning mode. There are two turning modes; a *Turn90* mode, which turns the aircraft to a 90 degree orientation, and a *Turn270*, which turns the aircraft to a 270 degree orientation. Since the turn requires 1000 meters to complete, the turn modes wait 1000 meters to complete to transition to the next modes and adjust their orientations. Refer to Figure 10 for the Stateflow implementation of the collision avoidance turn transitions.

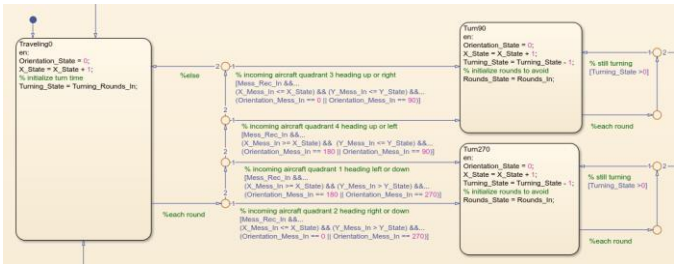

Figure 10: Collision Avoidence Turn Transitions

After the turn modes have waited 1000 meters, the state transitions to the AvoidCollisionTraveling90 and AvoidCollisionTraveling270 modes, respectively. In these modes, the aircraft increments or decrements the y position, respectively, for as long as the message is still received, and the position and orientation still pose a threat to the danger zone. (Note: The implementation allows for an optional parameter to be set that to specifies the minimum number of rounds that the aircraft must remain in an avoidance mode before it can test if a message is still received and exit. When this parameter is set to zero you know you have the aircraft will avoid to the minimum distance needed to leave the communication zone of the other aircraft, or until the position and orientation of the other aircraft no longer poses a threat to the danger zone.) Refer to Figure 11 for the Stateflow implementation of the collision avoidance modes.

Figure 11: Collision Avoidence Modes

Once the message is no longer received, or the position no longer poses a threat to the danger zone, the controller initiates a turn back to the 0 orientation. After the 1000 meter turn is complete, the state transitions back to the 0 orientation and returns to incrementing the x direction position. Refer to Figure 12 for the Stateflow implementation of the return to *Traveling* mode logic.

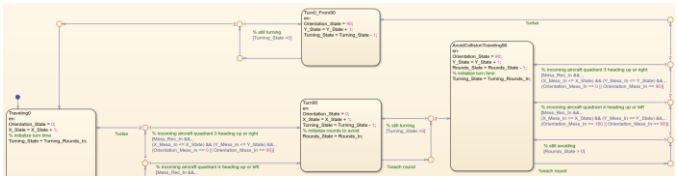

Figure 12: Return To *Traveling* Mode Logic

A similar collision avoidance algorithm has been written for the other 3 *CollisionAvoidanceMonitor* modes.

D. Communication Enviroment Modeling

The communication environment for each aircraft is combinational component and is modeled using an extended-state machine with one mode and 3 cyclic transitions. Each round the environment outputs the x position message and y position message of the other aircraft and a flag that says whether or not the message is within the 3000 meter range of the aircraft so that it may be received it. If the other aircraft is within the 3000 meter range, then the aircraft controller will receive the message. If the other aircraft is outside of the 3000 meter range, then the aircraft controller will not receive the message. Refer to Figure 13 for the Stateflow implementation of the communications environment for an aircraft.

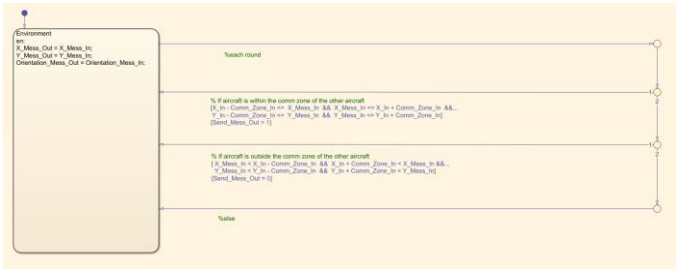

Figure 13: Aircraft Communications Environment

E. Saftey Monitor Error State

A transition from the Top-level C*ontroller* mode to the *ErrorDangerZoneBreached* mode in the To-Level *Final* mode is used to monitor our primary safety requirements regarding the danger zone. If at any point before *DestinationReached* mode in the *Final* mode is reached, a x position and y position message is received from the other aircraft that it has is within the danger zone, the transition fires and switches the state to the *ErrorDangerZoneBreached* mode. Therefore, there are only two outcomes for which the Final mode may be reached; either the aircraft has arrived at its destination, or the aircraft's danger zone was breached, and the automatic collision avoidance controller failed. Equivalently, either our safety requirement is satisfied, or it is not satisfied.

VIII. Verification & Validation Details

A. Verification & Validation Results

To verify the successful implementation of our aircraft controller algorithm we look at 4 Simulink scopes; two for each aircraft. One will display the current x, y, and orientation information of the aircraft, and one will display the results of *DangerZoneBreached* flag. We set up a worst-case scenario, in which the two aircraft are heading directly towards each other on the same axis, to test our collision avoidance controllers. The specifications of this test are indexed below.

1. Aircraft 1:
    a. {0, 90, 180, 270} Initial Orientation (degrees) of Aircraft 1 = 0
    b. Initial X Coordinate (m) Aircraft 1 = 0
    c. Initial Y Coordinate (m) Aircraft 1 = 0
    d. X Destination Coordinate (m) Aircraft 1 = 4000
    e. Y Destination Coordinate (m) Aircraft 1 = 0
2. Aircraft 2:
    a. {0, 90, 180, 270} Initial Orientation (degrees) of Aircraft 1 = 180
    b. Initial X Coordinate (m) Aircraft 1 = 4000
    c. Initial Y Coordinate (m) Aircraft 1 = 0
    d. X Destination Coordinate (m) Aircraft 1 = 0
    e. Y Destination Coordinate (m) Aircraft 1 = 0
3. Simulation Parameters:
    a. Turn Time = 1000
    b. Comm Zone = 3000
    c. Rounds To Avoid Upon Communication (m) = 0
    d. Danger Zone (m) = 500
    e. Simulink: Fixed-Step Discrete (Auto)
    f. Simulink Simulation Start Time: 0
    g. Simulink Simulation End Time: 10000

There are 4 primary requirements we need to reason about per aircraft; two safety requirements (as specified above), and one liveness requirement (as specified above). To verify that our safety requirements we monitor the Danger Zone Monitor scope to ensure that it always take on the value 0 (false), and never takes on the value 1 (true). To verify that our liveness

requirements we monitor the *X, Y, Orientation* scope to ensure that the final destination is eventually reached. The X, Y, Orientation scope shows the path taken by the aircraft to avoid the collision.

Refer to Figure 14 for the Danger Zone Monitor scope simulation results. The simulation results of the safety monitors reveal no breaches of the danger zone and prove that our safety requirements are satisfied in the context of this simulation.

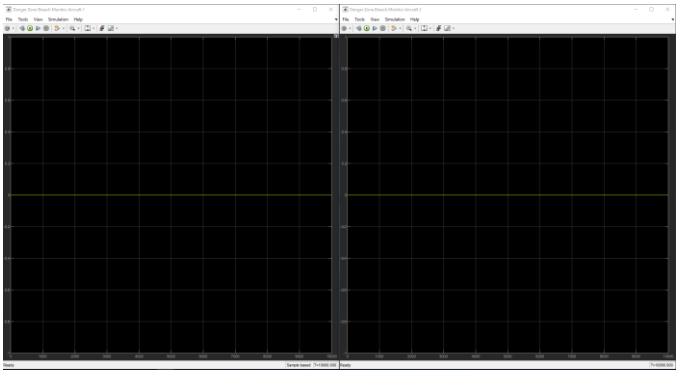

Figure 14: X, Y, Danger Zone Monitor Scope Simulation Results (Aircraft 1 is on the left, Aircraft 2 is on the right)

Refer to Figure 15 for the Simulink X, Y, Orientation scope simulation results. As shown, Aircraft 1 begins by moving along the x positive direction (orientation of 0), and Aircraft 2 begins by move along the negative x direction (orientation of 180). When one aircraft come within communication range of each other (approx. at x = 500 meters for Aircraft 1, and x = 3500 meters for Aircraft 2), they initiate a turn (which takes 1000 meters to become effective). When the turn becomes effective (approx. at x = 1500 m for Aircraft 1, and x = 2500 meters for Aircraft 2) Aircraft 1 switches it's orientation to 90 and travels along the y axis in the positive direction until Aircraft 2's orientation is no longer heading towards Aircraft 1. Likewise, Aircraft 2 switches it's orientation to 270 and travels along the y axis in the negative direction until Aircraft 1's orientation is no longer heading towards Aircraft 2. Because they are traveling opposite of each other immediately, they both detect that the orientation of the other aircraft is no longer poses an issue and they immediately initiate a turn back to their original orientation (which takes 1000 meters to become effective). Therefore, they are approximately 2000 meters away from each other when the turn is finished. Because the communication range is 3000 meters and they are only 2000 meters away at this point, they both immediately initiate a collision avoiding again in the same manner because their orientations are facing each other again. After the process repeats a second time they are a total of 4000 meters away from each other, outside of the communication range, and there for, continue to their x destinations directly. When they are 1000 meters away from their x destinations they initiate a turn (which takes 1000 meters to become effective) so that they arrive at their x destination and turn at the appropriate point. Aircraft 1 turns to an orientation of 270 degrees and decrements back the y destination of 0 meters, and Aircraft 2 turns to an orientation of 90 degrees and increments back the y destination of 0 meters. Since the destinations of both aircraft are eventually reached, the liveness requirements are satisfied.

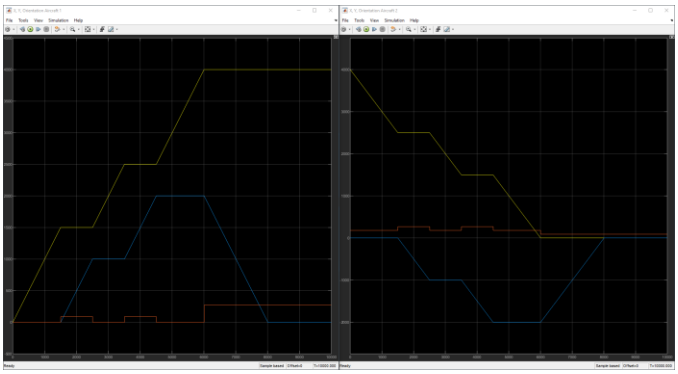

Figure 15: X, Y, Orientation Scope Simulation Results (Yellow = X Position, Blue = Y Position, Orange = Orientation) (Aircraft 1 is on the left, Aircraft 2 is on the right)

B. Force a Failure for Illustration Purposes

To illustrate what a failure would look like in this aircraft collision avoidance controller we must adjust our required communication zone of 3000 meters down 2000 meters and force a failure. We set this simulation to illustrate what a failure due to this requirement change would look like. This simulation will be equivalent to our verification simulation above, only the communication zone has decreased. The specifications of this test are indexed below.

1. Aircraft 1:
    a. {0, 90, 180, 270} Initial Orientation (degrees) of Aircraft 1 = 0
    b. Initial X Coordinate (m) Aircraft 1 = 0
    c. Initial Y Coordinate (m) Aircraft 1 = 0
    d. X Destination Coordinate (m) Aircraft 1 = 4000
    e. Y Destination Coordinate (m) Aircraft 1 = 0
2. Aircraft 2:
    a. {0, 90, 180, 270} Initial Orientation (degrees) of Aircraft 1 = 180
    b. Initial X Coordinate (m) Aircraft 1 = 4000
    c. Initial Y Coordinate (m) Aircraft 1 = 0
    d. X Destination Coordinate (m) Aircraft 1 = 0
    e. Y Destination Coordinate (m) Aircraft 1 = 0
3. Simulation Parameters:
    a. Turn Time = 1000
    b. Comm Zone = 2000 (Adjusted)
    c. Rounds To Avoid Upon Communication (m) = 0
    d. Danger Zone (m) = 500
    e. Simulink: Fixed-Step Discrete (Auto)
    f. Simulink Simulation Start Time: 0
    g. Simulink Simulation End Time: 10000

Refer to Figure 16 for the Danger Zone Monitor scope simulation results. As illustrated our safety requirements are not satisfied in this forced failure case as the danger zone monitors show that both aircraft had their danger zones breached approximately 1800 steps into the simulation.

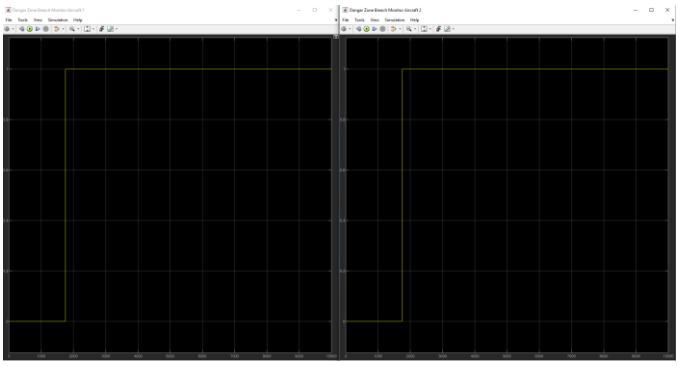

Figure 16: X, Y, Danger Zone Monitor Scope Simulation Results (Aircraft 1 is on the left, Aircraft 2 is on the right)

Refer to Figure 17 for the Simulink X, Y, Orientation scope simulation results. To illustrated why our safety requirements were violated we can look at X, Y, Orientation scope simulation results. As shown both aircraft head directly into each other and violate their danger zone safety requirements. This happens because the aircraft are heading towards one another and their velocities are compounded. Because the communication zone (in this forced failure case) is 2000 meters, and it takes 1000 meters to turn from the first notification of an oncoming aircraft, neither of them would be able to turn until they are directly on top of one another, since their relative velocities are compounded. Therefore, though both aircraft initiate turns, they are not able to complete the turns before violating their danger zone safety requirements. As illustrated when Aircraft 1 passes approximately 1750 meters on the x axis and Aircraft 2 drops below approximately 2250 meters on the x axis (i.e. below a 500 meter distance from each other), one of our danger zone safety requirements is breached.

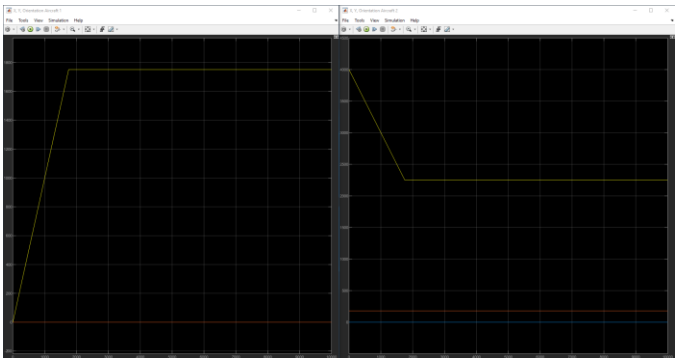

Figure 17: X, Y, Orientation Scope Simulation Results (Yellow = X Position, Blue = Y Position, Orange = Orientation) (Aircraft 1 is on the left, Aircraft 2 is on the right)

IX. Conclusion

We have illustrated the design, implementation, and verification of a 2-D automated collision avoidance controller. We converted the requirements into formal safety and liveness requirements and designed component specifications. The controller was implemented into a Simulink and Stateflow simulation environment, and verified using safety and liveness requirement monitors.

References

[1] mathworks.com, 'Simulink', 2018. [Online]. Available: https://www.mathworks.com/products/simulink.html. [Accessed: 11-Dec- 2018].

[2] mathworks.com, 'Stateflow', 2018. [Online]. Available: https://www.mathworks.com/products/stateflow.html. [Accessed: 11-Dec- 2018].

[3] R. Alur, Principles of *Cyber-Physical Systems*, 1st ed. Camebridge, Massachusetts: The MIT Press, 2015.

[4] brightspace.vanderbilt.edu, 'CS 6376: Foundations of Hybrid and Embedded Systems Fall 2018 Project: Aircraft Collision Avoidance', 2018. [Online]. Not Publically Available: https://brightspace.vanderbilt.edu/d2l/le/content/85900/viewContent/873623/View [Accessed: 11-Dec- 2018].