

**INTELLIGENT TERRAIN AVOIDANCE AGENT FOR GENERAL AVIATION  
FREE FLIGHT**

A Senior Honors Thesis

by

PAUL GESTING

Submitted to the Office of Honors Programs  
and Academic Scholarships  
Texas A&M University  
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE RESEARCH FELLOWS

April 2003

Engineering and Physics 3

**INTELLIGENT TERRAIN AVOIDANCE AGENT FOR GENERAL AVIATION**  
**FREE FLIGHT**

A Senior Honors Thesis

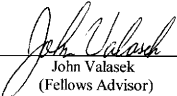
by

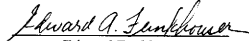
PAUL GESTING

Submitted to the Office of Honors Programs  
and Academic Scholarships  
Texas A&M University  
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE RESEARCH FELLOWS

Approved as to style and content by:

  
John Valasek  
(Fellows Advisor)

  
Edward Funkhouser  
(Executive Director)

April 2003

Engineering and Physics 3

## ABSTRACT

## Intelligent Terrain Avoidance Agent for General Aviation

Free Flight (April 2003)

Paul Gesting  
Department of Aerospace Engineering  
Texas A&M University

Fellows Advisor: John Valasek  
Department of Aerospace Engineering

In order to reduce the work load of Air Traffic Controllers, a new concept called Free Flight has emerged for General Aviation. This system takes the load off of the air traffic controller and puts the responsibility on the pilot. In order to help the pilot handle this responsibility, a hierarchical agent system is under development. This system will take information from traffic, weather, and terrain to determine a safe and efficient flight path. The terrain agent in this system must avoid Controlled Flight into Terrain. A simplistic conditional logic model was created and tested on a two-dimensional terrain slice. Then this algorithm was implemented with the dynamics of a Commander 700, a twin-engine general aviation aircraft. This algorithm was found to satisfy minimum altitude requirements and safely navigate the aircraft over the terrain conflicts for a simple terrain model. Further work, however, should be explored on adapting the algorithm for three-dimensions, testing on actual terrain, and implementing the terrain agent with the weather, traffic, and executive agents in this system.

**TABLE OF CONTENTS**

	Page
ABSTRACT .....	iii
TABLE OF CONTENTS .....	iv
LIST OF FIGURES .....	v
INTRODUCTION .....	1
RESEARCH ISSUES AND OBJECTIVES.....	3
RESEARCH APPROACH.....	4
CONTROLLED FLIGHT INTO TERRAIN .....	7
CONTROLLED FLIGHT INTO TERRAIN ALGORITHM AND EXAMPLE.....	8
RESULTS.....	14
CONCLUSIONS.....	21
FUTURE WORK.....	22
REFERENCES .....	24
APPENDIX .....	25

**LIST OF FIGURES**

	Page
Figure 1 – Agent System Architecture.....	2
Figure 2 – Simple Terrain Model.....	10
Figure 3 – Commander 700.....	12
Figure 4 – C700 Altitude Command and Hold Autopilot Block Diagram .....	13
Figure 5 – C700 Airspeed Command and Hold Autopilot Block Diagram .....	14
Figure 6 – CFIT Algorithm Commands.....	15
Figure 7 – C700 Altitude Autopilot Trajectory from CFIT Algorithm .....	16
Figure 8 – C700 Altitude and Airspeed Autopilot Trajectory with CFIT Algorithm.....	17
Figure 9 – Command and Autopilot Trajectory .....	18
Figure 10 – Second Terrain Model.....	19
Figure 11 – C700 Dynamic Trajectory for Second Terrain Model .....	21
Figure 12 – Commander 700 State Space Model .....	25

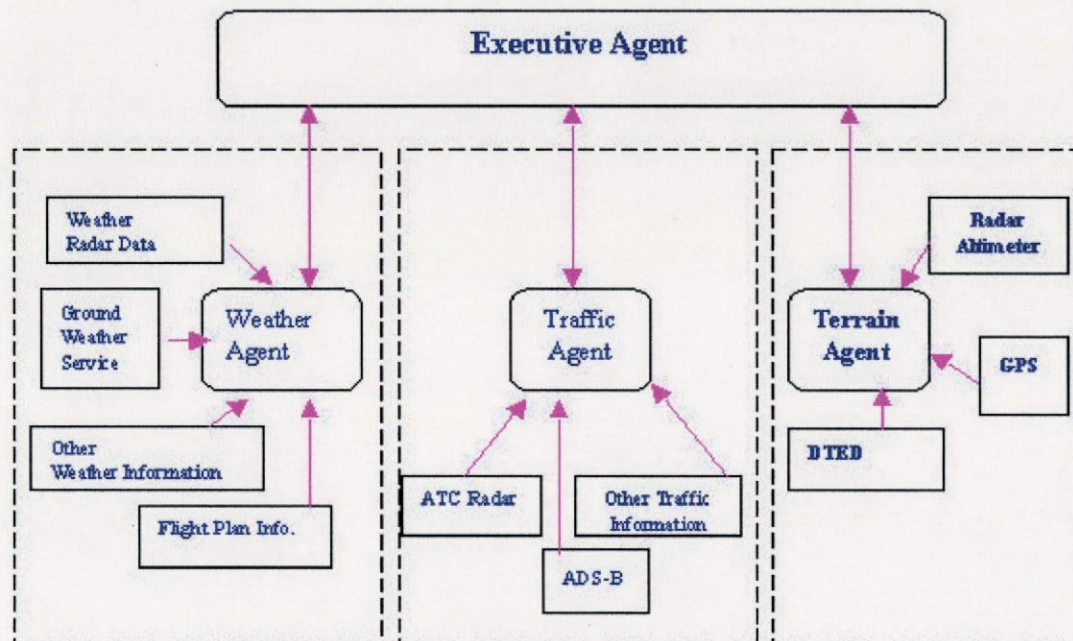
## INTRODUCTION

Currently, air traffic is supervised and controlled by Air Traffic Controllers (ATCs). This setup requires communication between the pilot and ATC, flight rules and regulations, ground tracking, and trajectory calculations. As air traffic increases, the workload of the ATC increases, and the possibility of an accident increases. If more of the workload could be taken from the ATC and put in the pilot's hands, the ATC would be responsible for less and able to oversee traffic in a more precise manner.

Free Flight, a new concept in air traffic management, can be seen by many as the future of air traffic management. In Free Flight, pilots operate under Instrument Flight Rules (IFR) and are able to choose their own flight path in real time. This puts enormous responsibility on the pilot at all times. An Intelligent Agent undertakes some of this responsibility. This agent is comprised of lower level agents in a hierarchical system that provides the best possible scenario within limited rules governing the decision maker. The intelligent agent is the decision maker, which determines appropriate actions for the data provided by lower level, independent agents such as a weather agent, a traffic agent, and a terrain agent. The final resolution provided by the intelligent agent is provided to the pilot for final authorization. The ATC would oversee and intervene only when a conflict arises which the pilot overlooks. Figure 1 shows this agent system.

---

This thesis follows the style and format of the American Institute of Aeronautics and Astronautics.



**Figure 1 – Agent System Architecture**

The intelligent agent will be fed information from lower level agents such as the traffic, terrain, and weather agents. These lower level agents will receive data from outside sources. For instance, the weather agent could receive weather information from satellites, on board radar, or ground weather observers. The traffic agent would receive traffic information from Automatic Dependent Surveillance- Broadcast (ADS-B). The ADS-B on an aircraft transmits pertinent flight data such as position, velocity, altitude, trajectory, and final destination, periodically. The lower level agents will be independent of each other, such that the resolution provided by the weather agent will not take into account traffic conflicts, and vice versa. Thus, the weather agent may propose a trajectory that violates conditions set by the traffic agent. It is up to the executive agent to arbitrate between the two. When the pilot approves the trajectory, it

will be broadcasted over the aircraft's ADS-B so that other aircraft and ground controllers are aware of the change. Most research up to this point has been dealing with only one conflict, not multiple conflicts such as terrain, traffic, and weather conflicts.

Separation distances of aircraft are regulated to 2.5 nautical miles horizontal and 1000 feet vertical. Given small aircraft and IFR regulations, the time interval for collision is two minutes for both vertical and horizontal distances. Separation in free flight would be accomplished through two separation zones, alert and protected zones. The alert zone would extend around the aircraft and would allow the aircraft to maneuver freely until its alert zone overlaps another aircraft's alert zone. The protected zone should never overlap another aircraft's protected zone. The size of these zones is determined by the aircraft's size and speed.

The intelligent executive agent will arbitrate a decision based on the information given by the lower level agents. This agent will employ fuzzy logic. Fuzzy logic breaks away from binary logic, where only "true" and "false" values are possible. Fuzzy logic allows such terms as "near" and "far" to be separated. It is a multi-valued logic that allows intermediate values to be defined and mathematically processed in a computer. This allows decision making to be more human-like.

### **RESEARCH ISSUES AND OBJECTIVES**

Research is in progress for the traffic, weather, and executive agents. Therefore the topic of this research will be the terrain agent. There are many issues that arise in the



development of a free flight terrain agent. Some of these issues are compiled in the following list.

- 1) How will the terrain agent get its Information?
- 2) How to build an effective and efficient conflict detection and resolution algorithm
- 3) Will the terrain agent be affordable?
- 4) How big and fast will the computer have to be and how much will the system cost?

A critical item to be determined is whether the terrain agent can be contained in a small computer for general aviation aircraft. Whether this computer could fit into a small aircraft will determine the ultimate validity. Current results imply that this is possible, but validation will be necessary. Along these same lines is the cost to the average General Aviation (GA) pilot. Will these pilots be willing to invest in such a system? Most pilots would not be willing to put a \$100,000 system on a \$50,000 airplane; therefore measures must be taken to ensure that the price of the terrain agent does not inflate beyond reach.

### **RESEARCH APPROACH**

The problem tackled in this research is the terrain agent. The terrain agent will prevent Controlled Flight into Terrain (CFIT). It will predict terrain conflicts and provide solutions to the executive agent. The executive agent will then compile the data

it receives from the traffic agent, the weather agent, and the terrain agent to decide the best course of action.

To answer Objective Number 1 from above, initially the plan is to obtain a database of known geographical information. This will be a very accurate way of determining known terrain and if a conflict exists. The problem with this is that a database with every known terrain obstacle could be quite a large file. Therefore, during the pre-flight planning phase, the pilot will download only the terrain in his planned route plus a safe amount for deviations. However, this information is only useful if you know where you are. Therefore, onboard GPS will determine the position of the aircraft very accurately. This will allow the computer to compare position with the database to see if a terrain conflict will occur. Terrain conflicts will basically occur when the altitude of the aircraft becomes 0 ft AGL (Above Ground Level). Each aircraft has an onboard pressure altimeter. This altimeter shows changes with density and must be calibrated before each flight and many times during the flight due to atmospheric pressure changes. Also, this altimeter only gives altitude relative to Mean Sea Level (MSL). This is not helpful in determining a possible terrain conflict with the ground. Therefore, another proposed idea is to use an onboard laser altimeter. This laser altimeter will provide very accurate altitude relative to the ground below (AGL). This will allow the aircraft to maintain required altitude requirements set by the FAA. This will provide a check on not only the pressure altimeter but also the altitude provided by the GPS.

This laser altimeter, however, only points straight down; it provides no information on what is in front of the aircraft. Therefore, theoretically you could fly straight into a wall, while still abiding by safe altitude requirements. Therefore research will have to be done to see if the database and the laser altimeter is enough to navigate the aircraft around terrain conflicts. For most terrain, hills and mountains change altitude gradually, allowing the laser altimeter and the terrain database enough information and time to correct for it. However, there exists steep sloping terrain that could not only provide terrain conflicts, but could quickly reduce the aircraft's altitude above ground, violating required safe altitudes. Sharp sloping terrain also causes bad wind shear for several thousand feet above the terrain. The terrain agent must take all this into account and provide a valid alternative to this conflict.

If the laser altimeter and terrain database are not enough, it will need to be found what other device will be required to prevent CFIT. The aircraft will assumedly have onboard weather radar; however this will not provide any information as to terrain in front. Another alternative is forward looking radar, which would find terrain conflicts in front of the aircraft, not just below it. However, this is highly expensive and of to date is only onboard military aircraft. The price of such a device might keep this agent out of reach for GA aircraft.

The many routes of a viable terrain agent will be searched and tested. It is the plan for the most viable and most efficient option, which addresses all the Research Objectives to be determined and evaluated.

## CONTROLLED FLIGHT INTO TERRAIN

Controlled Flight into Terrain (CFIT) continues to be a blight in the aviation industry. CFIT occurs from flying a perfectly functioning aircraft inadvertently into the ground or water. 40% of all accidents are CFIT and over half of all aviation fatalities occur from CFIT. Most (71%) of CFIT accidents occur in aircraft designed to carry 9 passengers or less.<sup>3</sup> There have been many attempts to slow this trend, most of which have occurred in commercial and military aircraft. The first such attempt was the Ground Proximity Warning System (GPWS) implemented by the FAA in the 1970s for commercial aircraft. The idea behind the GPWS was to provide adequate warning of a terrain conflict to the pilot. It provided a lookdown capability to take into account the rising slope of the terrain to provide an aural warning to the pilot. However due to the restriction of only the lookdown capability, the system generated a high number of false alarms. These create not only a nuisance, but also an apathetic response in time. Another drawback in the lookdown capability only is that in sharply rising terrain, the aircraft may not be able to pull up in time. Therefore in the 1990s the Enhanced Ground Proximity Warning System (EGPWS) came about. It has the same features as the GPWS with an added predictive component. The EGPWS incorporates a digital terrain database to predict terrain conflicts along the flight path, and also provide a visual representation of the terrain to the pilot. This could provide up to a 60 second warning to the pilot. These two systems, however, were designed with the commercial market in mind and are therefore too expensive for GA aircraft. Therefore an approach for GA aircraft is a GPS-based system. Some work has been done with incorporating GPS with

a digital terrain database.<sup>4</sup> This approach seems more likely to work and a version of this shall be looked at in this research, as was stated in the Research Objectives. However the aim and end of these past approaches has been solely to increase the situational awareness of the pilot, who would then make a decision on how to avoid a conflict. The end and aim of this research is to provide a CFIT algorithm for GA aircraft for use in a free flight environment. Therefore the information obtained about the aircraft and its surroundings will not only be fed to the pilot, but also to the terrain agent which, with the Executive Agent, will provide a safe and efficient course for the aircraft to proceed upon.

#### **CONTROLLED FLIGHT INTO TERRAIN ALGORITHM AND EXAMPLE**

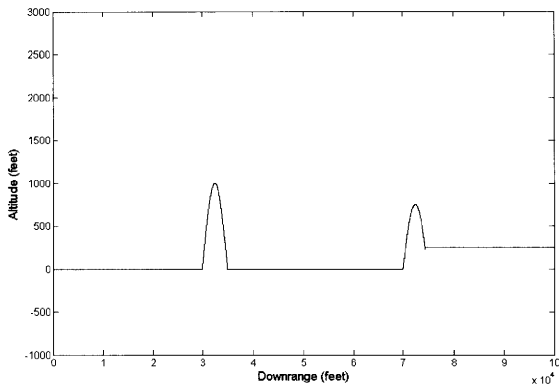
The first step in developing a CFIT avoidance algorithm was to decide on what the algorithm would need to do exactly. This information can be divided into four groups: Percepts, Actions, Goal, and Environment.

A percept is not only the data that shall be inputted to the system, but also information that is useful. The percepts of the terrain agent from the aircraft instruments will be speed and altitude MSL and AGL. The aircraft position will most likely be determined from GPS, and the course and destination will be provided from the flight-planning phase. Also, the agent will have access to a terrain database. Given the precepts above, the terrain agent will then need to decide if a terrain conflict exists. The agent must then decide on an action to make to avoid this conflict. This is termed conflict detection and resolution (CD&R). The minimum goals of the algorithm will be

set by FAA flight rules. Other goals, such as flight time, fuel, and passenger comfort could be set for utility. The environment the algorithm will be based in is real-time, onboard the aircraft.

Given this structure, the next step in developing a CFIT avoidance algorithm was to decide upon a simple case to be looked at. This was undertaken as a two-dimensional case where the only option for avoiding terrain is to climb over it, at a standard IRF climb rate of 500 ft/min. Also in this simulation, a completeness assumption was made that the CFIT avoidance agent had the capability of obtaining complete knowledge of the terrain. In this simplistic situation, a purely conditional logic model was deemed appropriate. Other options for this algorithm for a less simplistic situation will be discussed later.

A simple terrain model was designed for use in this case. Figure 2 shows this terrain.



**Figure 2 – Simple Terrain Model**

As can be seen from Figure 2, the scales on the axes are misleading. The horizontal distance is measured in tens of thousands of feet, whereas the altitude is measured in feet. This is due to the fact that the aircraft will be traveling much faster in horizontally than vertically.

Given a maximum climb rate of 500 ft/min, the aircraft must have a look-ahead capability far enough that it can climb over the terrain. Therefore if the terrain ahead will force the aircraft to climb 2000 ft, then the aircraft must look-ahead at least 4 minutes to have sufficient time to start climbing. The aircraft in this model will start at

an altitude of 550 feet. Therefore the maximum it will need to climb, as can be seen from Figure 2, is over a 1050 ft obstacle. Starting at 550 feet, and with a desired final altitude of 500 feet over the obstacle, this means the aircraft must climb 1000 feet, or look ahead 2 minutes. However, the slope of the terrain is somewhat gradual. Therefore the aircraft should sense a conflict and start climbing before it sees the peak of the “mountain” in Figure 1. This two-minute look-ahead value is used only in this example for looking at the algorithm's ability to react to changing terrain and is not a universal value that shall be always used.

A conditional logic algorithm was designed in Matlab to tackle this problem. This algorithm starts with a point-aircraft traveling at 100 knots at an altitude of 550 feet. The algorithm then looks ahead 2 minutes in its “terrain database” at 50-foot intervals of horizontal distance. The algorithm then looks to see if the terrain will come within the 500-foot minimum altitude requirement. If so, then the algorithm stores this terrain altitude and distance from the aircraft into an array. Therefore if multiple terrain conflicts exist, the algorithm will store all of them. Then the algorithm looks at each of these terrain conflicts. It calculates the time required to climb to 500 feet above each one, the horizontal starting point that it must start climbing at in order to reach 500 feet above that point, and also the time until the aircraft reaches that starting point. These values are calculated in order to determine urgency, which will be discussed later. However, for this simplistic model, as soon as the aircraft discovers a terrain conflict, it issues a command to climb. Once it reaches 500 feet above the highest peak it in the array, and thus the highest peak it can see, it issues a command to stop climbing and



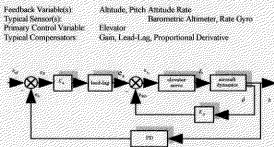
level off. Then as the aircraft flies over the terrain conflict, it is still looking ahead at all the terrain. Then if there are no terrain conflicts in the 2-minute window ahead and the aircraft is more than 500 feet above the terrain directly below it, it issues a command to descend until it is either 500 feet above the terrain below it, or it reaches a desired altitude commanded by the pilot, which in this example was 550 feet. The algorithm was run and the trajectory calculated by the algorithm was determined.

The next step in the algorithm development was to test this trajectory on an actual aircraft dynamic model. Previous research had been performed in the development of a Commander 700 (see Figure 3) linear flight model<sup>5</sup>. This was done in preparation for the Commander 700 model in the Engineering Flight Simulator at Texas A&M. Given that only a two-dimensional problem was proposed, only the longitudinal dynamic models are required. The state-space representation of the longitudinal dynamic model for the Commander 700 for steady, level, cruise flight is shown in the Appendix<sup>5</sup>.



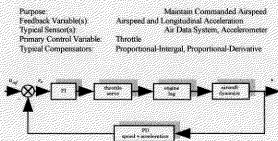
**Figure 3 – Commander 700**

From this linear model, an altitude command and hold autopilot was developed, and is shown in Figure 4<sup>5</sup>.



**Figure 4 – C700 Altitude Command and Hold Autopilot Block Diagram**

The Matlab CFIT algorithm was then implemented in Simulink with the altitude command and hold autopilot as a dynamic model. However in implementing this autopilot, a critical assumption was made. If an aircraft climbs and no adjustment is made to the throttle, its speed shall decrease. However it was assumed that the aircraft's speed shall remain constant. Therefore, an airspeed command and hold autopilot was developed to ensure that as the aircraft climbs, it will not slow down. This autopilot is shown in Figure 5<sup>5</sup>.



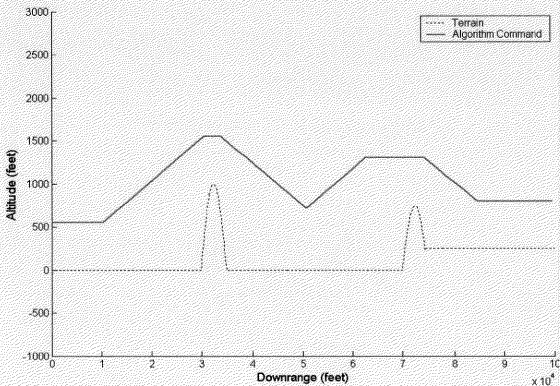
**Figure 5 – C700 Airspeed Command and Hold Autopilot Block Diagram**

This autopilot was hoped to improve the performance of the altitude command and hold autopilot. These two autopilots were therefore run in tandem with the CFIT algorithm to produce a final result.

## RESULTS

At first the algorithm only stored the highest altitude terrain conflict. However, this might cause the aircraft to only avoid the highest peak in the 2-minute window, and therefore crash into a smaller peak, but one that still violates the altitude requirements. Therefore the algorithm was changed to look at all the terrain conflicts within the 2-minute window ahead of the aircraft.

The trajectory determined by the algorithm is shown in Figure 6.

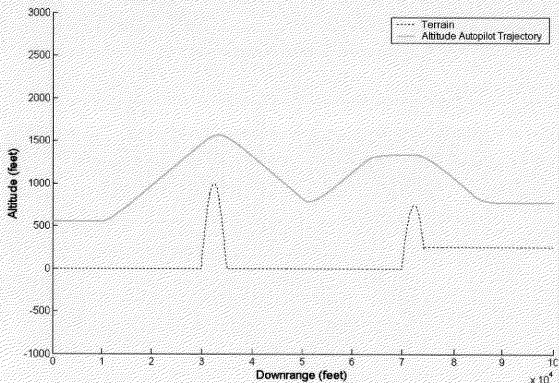


**Figure 6 – CFIT Algorithm Commands**

As can be seen from Figure 6, the aircraft does not see the first mountain until it gets within about 20,000 feet of it. Since the aircraft is traveling at 100 knots, or 168 feet/sec, a 2 minute look-ahead would amount to 20,160 feet. Therefore the aircraft will be given a command to climb. When the aircraft gets over the first mountain, the only terrain it can see is the flat ground ahead, so it gives a command to descend. When it gets to about 50,000 feet downstream, it is within 20,000 feet of the second mountain, it determines there will be a conflict and issues the command to climb again. Once it gets to 500 feet above the highest point it can see (the peak of the second mountain), it levels

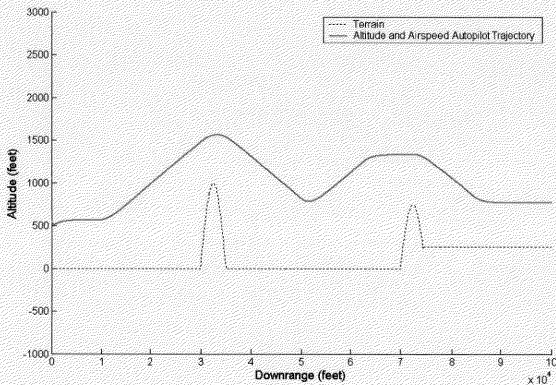
off. Once the aircraft passes the highest point, it sees the flat ground at 250 feet of elevation. Therefore it issues a command to descend. It wants to get back to its original desired altitude of 550 feet; however that would put it less than 500 feet above the ground, and therefore the algorithm issues a command to level off at 750 feet.

As can be seen from Figure 6, the trajectory that the algorithm determined requires abrupt changes in the flight path. An aircraft cannot make these sudden changes and therefore the dynamics of the aircraft must be determined. The resulting dynamic trajectory of the altitude autopilot in Figure 4 can be seen in Figure 7.



**Figure 7 – C700 Altitude Autopilot Trajectory from CFIT Algorithm**

In order to determine true aircraft dynamics, the airspeed command and hold autopilot was run in tandem with the altitude autopilot to ensure that the aircraft's speed did not decrease as it climbed. The resulting dynamics from this system can be seen in Figure 8.

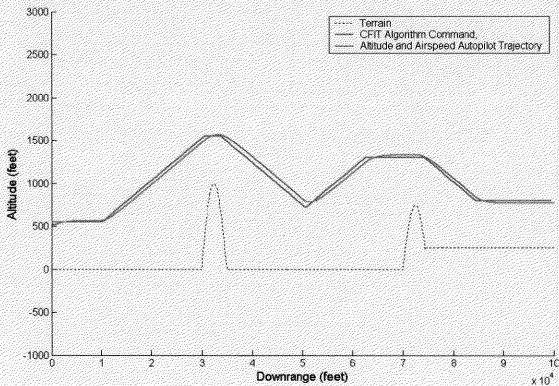


**Figure 8 – C700 Altitude and Airspeed Autopilot Trajectory with CFIT Algorithm**

As can be seen from Figures 7 and 8, the dynamics are almost identical. This is a verification that the airspeed autopilot used in Figure 8 works, since the dynamics in Figure 7 resulted by forcing the speed to stay constant. The only difference is the drop

in altitude in the airspeed command and hold in Figure 8 at the beginning of flight. This is resulting from a non-minimum phase zero in the airspeed autopilot, which means that the command is so far out of phase with the output that it actually results in the opposite of the command at first.

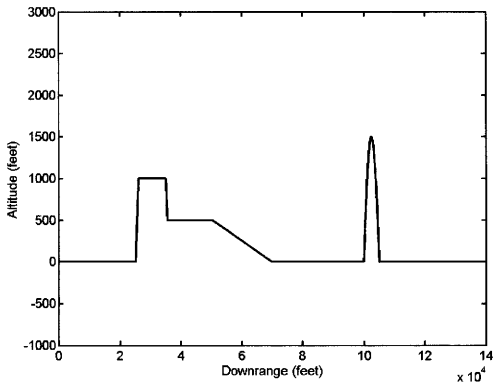
In order to demonstrate how the algorithm command and the actual aircraft autopilot trajectory correlate, the two were superimposed and are shown in Figure 9.



**Figure 9 – Command and Autopilot Trajectory**

The main difference between the algorithm trajectory and the aircraft dynamics is that the aircraft takes time once a command is given to respond. This is true of any aircraft, and the lag is very minor. The lag is greatest when the algorithm commands a change from climb to descend, or descend to climb without leveling off first.

The algorithm was also run for a second terrain, shown in Figure 10.

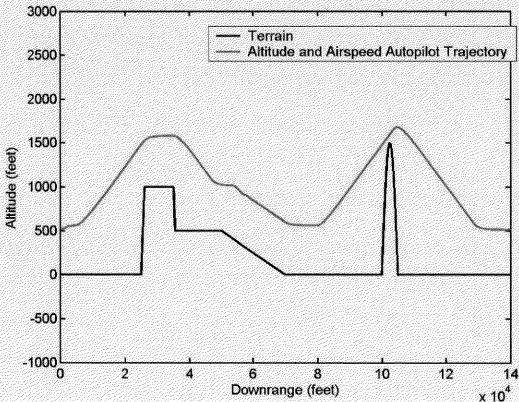


**Figure 10 – Second Terrain Model**

As can be seen from Figure 10, a few new terrain features were implemented to test aspects of the algorithm. First, there is an almost vertical wall that must be



overcome. Then there is a plateau, followed by a vertical drop, another plateau, and finally a gradually descending terrain. This model was aimed to find how the algorithm would respond to these different terrain features. As can be seen at the end of Figure 10, there is a very tall terrain conflict. As was stated earlier, it was previously known that the algorithm would fail if required to climb more than 1500 feet, due to the two minute look-ahead chosen for this algorithm. Therefore it was desired to see just how the algorithm would fail. This terrain was run in the CFIT algorithm, and then the commanded trajectory run through the C700 autopilot and airspeed command and hold autopilots. The results are shown in Figure 11.



**Figure 11 – C700 Dynamic Trajectory for Second Terrain Model**

As can be seen from Figure 11, the aircraft followed the initial terrain conflicts very closely and maintained required altitude limits at all times. However, it did fail as expected for the tall peak at the end of the terrain model, failing to reach a 500 foot minimum above the peak.

## CONCLUSIONS

A conditional logic algorithm was constructed in Matlab to avoid CFIT. This algorithm was tested on a simple terrain model. An altitude command and hold

autopilot for the Commander 700 was constructed to take the commands from the algorithm and execute them into a trajectory. An airspeed command and hold autopilot was also constructed to ensure a constant airspeed. These autopilots were then integrated into the CFIT algorithm in Simulink and the trajectory of each determined.

Based on the results from this system, the following conclusions are made:

- 1) The trajectory that the algorithm determined successfully avoided the terrain and also stayed within minimum altitude requirements for this terrain case.
- 2) The aircraft dynamics under the control of the CFIT algorithm were able to closely follow the algorithm commands, with only minor differences allowing for smooth transitions.
- 3) The conditional logic approach was found to be sufficient for this simple two-dimensional case.

### **FUTURE WORK**

Though the CFIT algorithm developed in this research was found to be of good quality, there is much work that can be done to improve it, most of which shall be undertaken by the author in graduate school. The first step that shall be undertaken is to use an actual two-dimensional terrain slice. A database of digital terrain elevation has been obtained from the United States Geological Survey (USGS).

The next step will be to alter the CFIT algorithm for three-dimensions. For in two-dimensions, the only avoidance maneuver possible is a change of altitude. However, in three dimensions, it may be more advantageous to go around the terrain.

To maximize utility, a different approach must be used than a conditional logic model. There is more than one way to solve this problem. The most likely choice would be a heuristic search function. This search function could then take into account the utility of the flight path.

To test the validity of the traffic agent, the free flight controls will be simulated on a 6 degree-of-freedom flight simulator with the three-dimensional terrain that has been obtained.

The next step would be to implement to terrain agent with the other hierarchical agents already under development. This would require the terrain agent to provide not only a viable flight path, but also the urgency of the terrain conflict. The executive agent might receive 3 different flight plans from the traffic, weather, and terrain agents. Therefore it must decide which is the most critical. Fuzzy Logic could be implemented for this task. This would have to be correlated with the urgency of the weather and traffic agents so that an equally vital conflict would get equal reaction from the Executive Agent.

## REFERENCES

1. Rong, Jie. "Intelligent Executive Guidance Agent for Free Flight." AIAA-2002-0015, 40<sup>th</sup> AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 14-17, 2002
2. Shandy, S. and Valasek, J. "Intelligent Agent for Aircraft Collision Avoidance." AIAA-2001-4055, AIAA Guidance, Navigation, and Control Conference & Exhibit, Montreal, Quebec, Canada, August 6-9, 2001
3. Moroze, Michael and Snow, Michael. "Causes and Remedies of Controlled Flight into Terrain in Military and Civil Aviation." Proceedings of the Tenth International Symposium on Aviation Psychology, Columbus, Ohio, May 3-6, 1999.
4. Baldwin, Jonathan and Cassell, Rick. "GPS Based Terrain Avoidance Systems – A Solution for General Aviation Controlled Flight into Terrain." Institute of Navigation, Anaheim, California, 1995.
5. Valasek, John. Commander 700 Linear Flight Models. Class Notes, Aerospace Engineering 625, Fall 2002.

## APPENDIX

## AUTOPILOTS

Commander 700, cruise, longitudinal

$$U_1 = 206.21 \text{ feet/sec}$$

$$H_1 = 8,500 \text{ feet}$$

$$\alpha_1 = 5.25^\circ$$

$$\bar{q} = 37.7 \text{ psf}$$

$$\delta_c = 0.1^\circ$$



$$\begin{bmatrix} \dot{u} \\ \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.0246 & 6.847 & 0 & -32.17 \\ -0.0012 & -1 & 1 & 0 \\ 0 & -3.535 & -2.245 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ \alpha \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \\ -0.0915 \\ -8.574 \\ 0 \end{bmatrix} \delta_c$$

$$\begin{aligned} \lambda_{1,2} &= -1.63 \pm 1.77j \\ \omega_{sp} &= 2.41 \text{ rad/sec} \\ \zeta_{sp} &= 0.68 \end{aligned}$$

$$\begin{aligned} \lambda_{3,4} &= -0.00722 \pm 0.153j \\ \omega_p &= 0.154 \text{ rad/sec} \\ \zeta_p &= 0.047 \end{aligned}$$

1 angular quantities in radians

Figure 12 – Commander 700 State Space Model