# Analysis and Comparison of Clothoid and Dubins Algorithms for UAV Trajectory Generation 

Mohanad AI Nuaimi

Follow this and additional works at: https://researchrepository.wvu.edu/etd

## Recommended Citation

Al Nuaimi, Mohanad, "Analysis and Comparison of Clothoid and Dubins Algorithms for UAV Trajectory Generation" (2014). Graduate Theses, Dissertations, and Problem Reports. 7059.
https://researchrepository.wvu.edu/etd/7059

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

# Analysis and Comparison of Clothoid and Dubins Algorithms for UAV Trajectory Generation 

## Mohanad Al Nuaimi

Thesis submitted to the
Benjamin M. Statler College of Engineering \& Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of
Master of Science in
Aerospace Engineering

Mario Perhinschi, Ph.D., Chair

Patrick Browning, Ph.D.

Jennifer Wilburn,Ph.D.

Department of Mechanical and Aerospace Engineering

Morgantown, West Virginia
2014

Keywords: Path Planning,; Dubins; Clothoid; Unmanned Aerial Vehicles; Flight Control Laws; Abnormal Conditions.

# ABSTRACT <br> Analysis and Comparison of Clothoid and Dubins Algorithms for UAV Trajectory <br> Generation 

## Mohanad Al Nuaimi

The differences between two types of pose based UAV path generation methods clothoid and Dubins are analyzed in this thesis. The Dubins path is a combination of circular arcs and straight line segments; therefore its curvature will exhibit sudden jumps between constant values. The resulting path will have a minimum length if turns are performed at the minimum possible turn radius. The clothoid path consists of a similar combination of arcs and segments but the difference is that the clothoid arcs have a linearly variable curvature and are generated based on Fresnel integrals. Geometrically, the generation of the clothoid arc starts with a large curvature that decreases to zero. The clothoid path results are longer than the Dubins path between the same two poses and for the same minimum turn radius. These two algorithms are the focus of this research because of their geometrical simplicity, flexibility, and low computational requirements.

The comparison between clothoid and Dubins algorithms relies on extensive simulation results collected using an ad-hoc developed automated data acquisition tool within the WVU UAV simulation environment. The model of a small jet engine UAV has been used for this purpose. The experimental design considers several primary factors, such as different trajectory tracking control laws, normal and abnormal flight conditions, relative configuration of poses, and wind and turbulence. A total of five different controllers have been considered, three conventional with fixed parameters and two adaptive. The abnormal flight conditions include locked or damaged actuators (stabilator, aileron, or rudder) and sensor bias affecting roll, pitch, or yaw rate gyros that are used in the feedback control loop. The relative configuration of consecutive poses is considered in terms of heading (required turn angle) and relative location of start and end points (position quadrant). Wind and turbulence effects were analyzed for different wind speed and direction and several levels of turbulence severity. The evaluation and comparison of the two path generation algorithms are performed based on generated and actual path length and tracking performance assessed in terms of tracking errors and control activity.

Although continuous position and velocity are ensured, the Dubins path yields discontinuous changes in path curvature and hence in commanded lateral accelerations at the transition points between the circular arcs and straight segments. The simulation results show that this generally leads to increased trajectory tracking errors, longer actual paths, and more intense control surface activity. The gradual (linear) change in clothoid curvature yields a continuous change in commanded lateral accelerations with general positive effects on the overall UAV performance based on the metrics considered. The simulation results show general similar trends for all factors considered. As a result, it may be concluded that, due to the continuous change in commanded lateral acceleration, the clothoid path generation algorithm provides overall better performance than the Dubins algorithm, at both normal and abnormal flight conditions, if the UAV mission involves significant maneuvers requiring intense lateral acceleration commands.

To My Dad:
I want you to know, you're with me in spirit wherever I go.....

## ACKNOWLEDGEMENTS

I would like to sincerely thank my parents for supporting me throughout my life and showing the value of hard work and that any goal can be accomplished.

I would like to thank my family and friends for their love and unforgettable support, especially my best friend and brother Saad Oribi Al Nuaimi and his family. I also would like to thank my friend Christy Woodward for her kind assistance.

Most importantly, I thank my advisor Dr. Mario Perhinschi for his boundless patience, invaluable advice, wisdom, guidance, support, and teachings throughout my education. Also I thank my other committee members Dr. Patrick Browning and Dr. Jennifer Wilburn for their guidance and assistance.

Finally, I would like to thank WVU Writing Center consultant Andrea Bebell and Literacy Volunteers Kayla Kreuger and Nathaniel Collins for helping me to organize my thesis.

## Table of Contents

Chapter ..... Page
I. INTRODUCTION ..... 1
1.1. Background ..... 1
1.2. Objective ..... 2
1.3. Thesis Layout ..... 2
II. LITERATURE REVIEW ..... 3
2.1. Road Map Methods ..... 3
2.1.1. Visibility Graph ..... 3
2.1.2. Voronoi Diagram ..... 4
2.2. Probabilistic Methods ..... 5
2.2.1. Dijkstra’s Algorithm ..... 5
2.2.2. Rapidly Exploring Random Trees ..... 5
2.3. Stigmergic Approaches ..... 5
2.3.1. Pheromone Based Approach ..... 5
2.3.2. Physics Based Approach ..... 6
2.4. Soft Computing Technologies ..... 7
2.4.1. Genetic Algorithm ..... 7
2.4.2. Fuzzy Logic ..... 7
2.4.3. Neural Network ..... 8
2.5. Pose Based Methods ..... 9
2.5.1. Dubins Algorithm ..... 9
2.5.2. Clothoid Algorithm ..... 9
2.5.3. Pythagorean Hodograph ..... 10
III. SIMULATION ENVIRONMENT ..... 11
3.1. Overview ..... 11
3.2. Graphical User Interface (GUI) ..... 15
3.2.1. Number of Vehicles GUI ..... 15
3.2.2. General GUI ..... 16
3.2.3. Visualization ..... 16
3.2.4. Failure Options ..... 19
3.3. Simulation Setup Using the Main Simulink Model ..... 21
3.3.1. Switch between Path Planning Algorithms ..... 22
3.3.2. Switch between Trajectory Tracking Algorithms (Controllers) ..... 23
3.3.3. Setting up a Failure Scenario ..... 24
3.3.4. Other Simulink Blocks and Parameters ..... 24
IV. PATH GENERATION ALGORITHMS ..... 25
4.1. Dubins Algorithm ..... 25
4.1.1. Dubins Path Planning ..... 25
4.1.2. Dubins Trajectory Generation ..... 26
A. Computing the Straight Tangent Solutions ..... 29
B. Computing the Cross Tangent Solutions ..... 31
4.2. Clothoid Algorithm ..... 32
4.2.1. Clothoid Path Planning ..... 32
4.2.2. Clothoid Trajectory Generation ..... 33
A. Define Poses ..... 34
B. Coordinate Axes and Notation ..... 35
C. Numerical Solution of the Fresnel Integrals ..... 35
D. Generating the Clothoid ..... 36
E. Conversion of Clothoid to Earth Coordinate System ..... 37
4.3. Definition of Solution Space Quadrants ..... 37
V. EXPERIMENTAL DESIGN ..... 40
5.1. Performance Metrics ..... 40
5.2. General Experimental Design ..... 42
5.3. Graphical Distributions of Poses ..... 43
5.4. Trajectory Tracking Control Laws ..... 47
5.4.1. Fixed parameter control laws ..... 47
A. Position PID Control Laws ..... 47
B. Outer Loop NLDI Control Laws ..... 48
C. Extended NLDI Control Laws ..... 48
5.4.2. Adaptive Control Laws ..... 50
A. Adaptive \#1 ..... 50
B. Adaptive \#2 ..... 51
5.5. Automated Data Acquisition Tool ..... 52
VI. ANALYSIS AND COMPARISON OF PATH GENERATION ALGORITHMS ..... 54
6.1. Variation of Bank Angle and Lateral Acceleration ..... 54
6.2. Path Length Analysis ..... 56
6.3. Performance Indices Analysis ..... 61
VII. CONCLUSION AND RECOMMENDATIONS ..... 68
VIII. References ..... 69
IX. Appendices ..... 74
Appendix A ..... 74
Comparison of the Generated Path Length ..... 74
Appendix B ..... 80
Comparison of Trajectory Tracking Performance Indices ..... 80
Appendix C ..... 86
Save Function Output Data ..... 86
LIST OF FIGURES
Figure 1. General Voronoi Diagram [13] ..... 4
Figure 2. Enhanced Potential Field Corresponding to Physical Assets [19]. ..... 6
Figure 3. Comparison of a Dubins Path with a Pythagorean Hodograph Path [32]. ..... 10
Figure 4. General Architecture of the WVU Simulation Environment. ..... 12
Figure 5. Path Planning, Trajectory Generation, and Tracking Data Transfer. ..... 13
Figure 6. Flowchart of the UAV Simulation Scenario Setup [39] ..... 14
Figure 7. User Interface with the UAV Simulation Environment ..... 15
Figure 8. Menu of Number of Vehicles Selection. ..... 15
Figure 9. GUI for the Main Selections without the Navigation and Control Options. ..... 17
Figure 10. Conventional Controller Selection GUI ..... 17
Figure 11. Adaptive Controller Selection GUI. ..... 18
Figure 12. UAV Dashboard ..... 18
Figure 13. FlightGear Visualization Software with WVU YF-22 Model. ..... 19
Figure 14. Locked Control Surface Failure GUI. ..... 19
Figure 15. Missing Surface Failure. ..... 20
Figure 16. Sensor Failure GUI. ..... 20
Figure 17. WVU YF-22 Simulink Model ..... 21
Figure 18. Changing the Path Generation Algorithm within the Simulink Model. ..... 22
Figure 19. Conventional Controllers. ..... 23
Figure 20. Adaptive Controllers. ..... 23
Figure 21. The Trajectories of Two Combinations CLC and CCC. ..... 26
Figure 22. Tangent Lines between Two Circles ..... 28
Figure 23. Relevant Path Tangents [42]. ..... 28
Figure 24. Straight Tangent Construction Geometry. ..... 29
Figure 25. Cross Tangent Construction Geometry. ..... 31
Figure 26. Dubins Path with Curvature Profile. ..... 33
Figure 27. Clothoid Path with Curvature Profile. ..... 33
Figure 28. Clothoid Arc Profile with Maximum Curvature Held Constant and Sweep Angle
Increasing. ..... 34
Figure 29. Coordinate Systems. ..... 35
Figure 30. End Point in Quadrant I with $\phi_{\text {total }} \geq 0$ ..... 38
Figure 31. End Point in Quadrant II with $\phi_{\text {total }} \geq 0$. ..... 38
Figure 32. End Point in Quadrant III with $\phi_{\text {total }} \geq 0$ ..... 38
Figure 33. End Point in Quadrant IV with $\phi_{\text {total }} \geq 0$ ..... 38
Figure 34. End Point in Quadrant I with $\phi_{\text {total }} \leq 0$ ..... 39
Figure 35. End Point in Quadrant II with $\phi_{\text {total }} \leq 0$. ..... 39
Figure 36. End Point in Quadrant III with $\phi_{\text {total }} \leq 0$. ..... 39
Figure 37. End Point in Quadrant IV with $\phi_{\text {total }} \leq 0$. ..... 39
Figure 38. Experimental Design Summary ..... 43
Figure 39. Trajectory Shape for the First Seven Factors. ..... 44
Figure 40. Zero Angle Trajectory. ..... 45
Figure $41.45^{\circ}$ Trajectory ..... 45
Figure 42. $90^{\circ}$ Trajectory ..... 45
Figure 43. $180^{\circ}$ Trajectory ..... 45
Figure $44.360^{\circ}$ Trajectory. ..... 45
Figure 45. First Quadrant Trajectory and Positive Total Sweep Angle ..... 46
Figure 46. Second Quadrant Trajectory and Positive Total Sweep Angle. ..... 46
Figure 47. Third Quadrant Trajectory and Positive Total Sweep Angle. ..... 46
Figure 48. Fourth Quadrant Trajectory and Positive Total Sweep Angle. ..... 46
Figure 49. Geometry of Trajectory Tracking Error [44] ..... 47
Figure 50. Position PID Controller. ..... 47
Figure 51. Two Phase Dynamic Inversion Inner Loop NLDI Controller. ..... 49
Figure 52. Biological Immune System Feedback Response Diagram ..... 50
Figure 53. NLDI Inner Loop Based Control Laws with L1 Adaptive Augmentation. ..... 51
Figure 54. Error Message for Unsaved Scenario. ..... 52
Figure 55. Numbering Code Inputs to Save Each Test Result. ..... 53
Figure 56. Flow Chart of the Automated Data Acquisition Tool. ..... 53
Figure 57. Variation of Bank Angle with Curvature Changes ..... 55
Figure 58. Variation of Lateral Acceleration with Curvature Changes, ..... 55
Figure 59. Difference between Commanded Curvature for Clothoid and Dubins Algorithms. ..... 56
Figure 60. Clothoid at Nominal Conditions L1+PPID Controller ..... 57
Figure 61. Dubins at Nominal Conditions L1+PPID Controller. ..... 57
Figure 62. Clothoid and Dubins Distances at Nominal Conditions L1+ PPID Controller. ..... 57
Figure 63. Distance with Nominal Conditions in (m) ..... 58
Figure 64. Distance with Sensor Failure in (m). ..... 58
Figure 65. Different Actual Distances With The Same Trajectory Errors. ..... 59
Figure 66. Wind Direction Effect on the UAV Trajectory. ..... 59
Figure 67. PI with Nominal Conditions ..... 61
Figure 68. Left Actuator Locked. ..... 62
Figure 69. 3D Plot for Commanded and Actual Trajectories. ..... 62
Figure 70. Distance with Nominal Conditions in (m). ..... 74
Figure 71. Distance with Sensor Failure in (m). ..... 74
Figure 72. Distance with Left Actuator Locked in (m). ..... 75
Figure 73. Distance with Right Actuator Locked in (m). ..... 75
Figure 74. Distance with Left Actuator Missed in (m). ..... 76
Figure 75. Distance Right Actuator Missed in (m) ..... 76
Figure 76. Distance with Different Wind Directions in (m). ..... 77
Figure 77. Distance with Different Wind Magnitudes in (m). ..... 77
Figure 78. Distance with Turbulence in (m). ..... 78
Figure 79. Distance with Different Turn Angles in (m). ..... 78
Figure 80. Distance with Different Quadrants in (m). ..... 79
Figure 81. PI with Nominal Conditions. ..... 80
Figure 82. PI with Sensor Failure. ..... 80
Figure 83. PI with Left Actuator Locked ..... 81
Figure 84. PI with Right Actuator Locked ..... 81
Figure 85. PI with Left Actuator Missed. ..... 82
Figure 86. PI with Right Actuator Missed. ..... 82
Figure 87. PI with Different Wind Angles ..... 83
Figure 88. PI with Different Wind Magnitudes. ..... 83
Figure 89. PI with Turbulence ..... 84
Figure 90. PI with Different Turn Angles ..... 84
Figure 91. PI with Different Quadrants. ..... 85
LIST OF TABELS
Table 1. Direction Choices Based Upon Quadrant and Sign of Total Sweep Angle. ..... 37
Table 2. Summary of Path Length Analysis ..... 60
Table 3. Summary of Trajectory Tracking PI Results. ..... 65
Table 4. Summary of Control Activity PI Results. ..... 66
Table 5. Summary of Total PI Results ..... 67

## CHAPTER I

## INTRODUCTION

### 1.1. Background

The use of unmanned aerial vehicles (UAVs) started as early as the 1930s with the Queen Bee being the first UAVs flown in the UK in 1935 [1]. UAVs are used today for a variety of purposes including reconnaissance, combat, surveillance, and payload delivery. UAVs are very attractive because they are inexpensive, unmanned, light weight, versatile, and capable of long endurance. The high demand for UAVs encourages researchers to develop design methods that increase UAVs efficiency via trajectory planning, which is expected to optimize a variety of metrics such as range, stability, energy usage, safety, or path tracking errors. In the context of integrating UAVs within the national airspace [2], safety becomes a major concern and objective. The UAV is expected to perform safely not only under normal conditions but also when one or more sub systems fail or experience abnormal operational conditions. Path planning and trajectory tracking algorithms that can mitigate the effects of aircraft subsystem failures can play a significant role in increasing both performance and safety.

Planning a path for UAVs is challenging due to the dynamic constraints that the UAVs are subject to, such as, the minimum turn radius. The UAVs are considered a type of nonholonomic mechanical system because they are subject to nonholonomic constraints. A nonholonomic constraint contains time derivatives of generalized coordinates of the system and is not capable of being integrated [3]. The use of a specific path planner method is driven by the purpose of the mission. Very often, the best choice for the path is associated with the nature of the task. For example, in military maneuver tasks, accuracy and stable performance are critical, while for reconnaissance missions that can sometimes exceed flight duration of 24 hours, lower energy usage and the shortest distance may be the most important parameters. Path planning may have a very important part in producing the desired outcomes of the UAV missions. Very often it is necessary or beneficial that the UAV trajectory be updated in real time as needed using computationally efficient software that run on airborne processors [4].

### 1.2. Objective

The main objective of this thesis is to analyze and compare through simulation two path generation algorithms for UAVs: clothoid and Dubins. The experimental design is expected to address several factors and levels such as different trajectory tracking control laws, normal and abnormal flight conditions, relative configuration of poses, wind, and turbulence. This thesis also includes an ad-hoc developed automated data acquisition tool within the WVU UAV simulation environment, which is the framework used for collecting and analyzing data. Special consideration is given to the evaluation and comparison of metrics, which include commanded and actual path length, trajectory tracking, and control activity.

### 1.3. Thesis Layout

The thesis is organized as follows: Chapter II is a literature review that presents previous work and methods that are used for UAVs path planning and trajectory generation. Chapter III describes in detail the graphical user interface (GUI) within the simulation environment and its operation, including procedures to switch between different simulation scenarios and features, such as path generation algorithms, trajectory tracking control laws, and normal or abnormal flight conditions. Chapter IV describes the path generation for clothoid and Dubins algorithms, including path planning and trajectory generation with the steps to produce a flyable and smooth path and introduction for the definition of solution space quadrants. Chapter V discusses the experimental design factors and levels, the performance metrics that evaluate the trajectory tracking error and the control activities, and graphical distribution of poses. It also introduces the trajectory tracking control laws within fixed parameters and adaptive control laws, and the automated data acquisition tool that saves and organizes the data outputs. The results of all test level analysis and comparison studies among the path algorithms and controllers are presented in Chapter VI. Finally, Chapter VII draws conclusions from the persistent effort exerted while carrying out these comparisons and analysis studies and discusses potential for future improvements.

## CHAPTER II

## LITERATURE REVIEW

The wide use of path planning in robotics and unmanned aerial vehicles makes it an important topic that researchers always try to improve in order to come up with the most efficient technique for the desired mission. Most approaches that are used for UAV path planning originate from the approaches that are used for mobile robots [5] however, path planning for unmanned aerial vehicles is more complicated because of the UAV's kinematic and dynamic constraints. This chapter outlines some of the major approaches and classifies them based upon their general properties.

### 2.1. Road Map Methods

The road map method is usually applied for shortest collision free path between two points. It relies on a two dimensional environment, containing the start and the final points connected by a network of straight lines that does not intersect with any obstacle. The robot is typically considered a material point, while the work space, which represents all of the points the robot can reach, may become very large. This method consists of selecting a set of straight segments to ensure the shortest distance travelled. Search algorithms must be used for generating the shortest path, such as A* [5]. The start and the end points for each straight segment in the shortest path are called "way points" through which the vehicle is expected to travel.

### 2.1.1. Visibility Graph

The visibility graph consists of a route connecting the initial and the goal points avoiding polygonal regions, which represent obstacles. The path is allowed to touch these regions without intersecting, which is producing a semi free path, and resulting in a connectivity graph network composed of straight lines that represent the obstacles’ vertices. The route is found using a graph search algorithm. The visibility graph was used in the late sixties for navigating SHAKEY, an early robot vehicle [6]. In the late seventies it was extended to more general collision avoidance problems [7]. A study done by Sholer et.al finds the shortest path in a bounded 3-dimensional

Euclidean space without limiting the number of geometric obstacles. This method is based on building a visibility graph for pairs of subsequent way points. An approximation to the optimal path can be found by using an existing graph search algorithm [8].

### 2.1.2. Voronoi Diagram

The Voronoi diagram consists of a network of straight lines, where each line is set between two obstacles at equal distance (Figure (1)). These lines result to be perpendicular to the invisible lines connecting the obstacle centers and form polygons. The minimum set of vertices belonging to the polygons will represent the shortest collision free path. A Voronoi diagram, or Dirichlet tessellation, was studied by René Descartes in 1644 and then by Dirichlet in 1850, who did their studies on the positive quadratic formulation [9]. Later in 1907, Voronoi was the first to consider the dual of this structure, where any two point sites are connected and whose regions have a boundary in common [10]. A recent study was conducted by creating the radar threat field based on the Voronoi diagram. In this study, the Dijkstra algorithm was enhanced, and utilized for path planning in a dynamic environment [11]. The Voronoi diagram method was used in [12] to produce a more predictable path grid with reduced computational overhead and by constructing the external path segments as tangent lines encircling the outer most threat zones in the environment.


Figure 1. General Voronoi Diagram [13]

### 2.2. Probabilistic Methods

Probabilistic methods consist of a uniformly sampled space in the form of a network that represent the probable solutions. The desired points that meet some metric such as the shortest path can be selected randomly. The probabilistic methods for the path planning problem can be treated as a search problem.

### 2.2.1. Dijkstra’s Algorithm

Dijkstra's Algorithm and its extensions, known as A* algorithm, are an optimal search method with a significant computational efficiency. This algorithm was applied to the path planning for a mobile robot in 1994 by Stentz [14].

### 2.2.2. Rapidly Exploring Random Trees

Rapidly Exploring Random Trees is an intuitive method for randomly exploring a set of points to connect to the closest part of the path tree. This method was used by Kothari et al. to implement multi UAV path planning [15].

### 2.3. Stigmergic Approaches

Stigmergy is an idea associated with biological sciences that considers the environmental effects of the past behavior [16]. Pierre Paul Grasse described stigmergy in the 1950s, within the context of communications and social studies associated with insect societies [17]. The brief definition is as follows: "The stimulation of the workers by the very performances they have achieved is a significant one inducing accurate and adaptable response, and has been named stigmergy" [17]. One of the most common examples of stigmergic approach is the process of ants in path planning to find food.

### 2.3.1. Pheromone Based Approach

In this approach, as described in [18], the target is the food source and the searching area is divided into an equally spaced grids which represent the enemy defense region. The ant will
move to the target node through the grid nodes. As they carry the food back to the nest, they mark the path with scent markers called pheromones. These scent markers dissipate over time. The simplicity of the path planning for an individual ant translates to a wider view of the ant colony as a whole for food gathering. Evaluation function considers the weighted sum of the threat intensity on the path, the distance to the target, and the maximum yaw angle. The amount of pheromones in the path is updated upon the evaluation of the function values. The probability of a UAV to choose a path is increased with the amount of the pheromone on this path.

### 2.3.2. Physics Based Approach

One of the vastly applied physics based approaches is the potential field approach. This approach was influenced by the field of electrostatics. The electrostatic force, according to Coulomb's law, is determined by the physical distribution of the charge. In the potential field, the target is treated as an attractive point, while the threats are treated as repulsive points and the vehicle as a point of mass. One of the applications of this method is presented in [19]; the UAVs can be guided through the battlespace by using the potential field to destroy the enemy defense and avoid the threat areas (Figure (2)). The "charges" are placed at different locations to represent targets (attraction) or threats and obstacles (repulsion) and the resulting UAV velocity vector is computed within the electrostatic field.


Physical Assets


Field Representation

Figure 2. Enhanced Potential Field Corresponding to Physical Assets [19].

### 2.4. Soft Computing Technologies

The attributes of soft computing allow, unlike conventional computing, for handling ambiguity, inexactness, approximation, and uncertainty. In addition, the soft computing is flexible, robust, and a relatively inexpensive solution compared to conventional computing.

### 2.4.1. Genetic Algorithm

The genetic algorithm is a very popular met heuristic technique, inspired by Darwin's theory on the evolution of species which is based on the survival of the fittest individuals as a result of natural selection. The first book on genetic algorithms as problem solvers was published in 1975 [20]. Genetic algorithms are widely used in path planning for optimization purposes. The starting step in building a genetic algorithm is to select the initial population or set of potential solutions through a random process. The fitness function or the optimization criterion of the problem is then used to calculate the fitness value of each solution. The fittest solutions are selected to produce the next generation, then the genetic operator such as crossover and mutation are applied to the selected solutions to generate a new population. The algorithm is repeated until the maximum number of iterations or another stopping criterion is reached. One of the most common applications of genetic algorithm in path planning is to determine the shortest collision free path. In [21] the genetic algorithm was used in a dynamic environment to calculate the shortest path planning in optimal time. The size of the obstacles were variable. Here, the genetic algorithm was applied at a point in the problem space, which is an equally spaced grid. In [22] the flyable path of multi UAVs was constructed using genetic algorithm. First, a feasible path was calculated by using a genetic algorithm, and then the path is smoothed by using Bezier curves to ensure that it is flyable. More details on Bezier curves can be found in [23].

### 2.4.2. Fuzzy Logic

Fuzzy logic is an alternative logic that uses continuous truth values between 0 and 1 as opposed to classical logic, which only accepts binary alternatives. Control system methodologies based on fuzzy logic are equivalent to real time expert systems relying on the experience and knowledge of a human operator. In 1965 Lotfi Zadeh, a professor at the University of California
at Berkley, published his first paper on fuzzy logic entitled "Fuzzy Sets" which was the beginning of numerous applications of the fuzzy logic concept. [24]. In 1973 he published a paper on the analysis of complex systems and decision processes, and then in 1979 he reported (1981 paper) on possibility theory and soft data analysis [25].Within a fuzzy logic based controller, inputs are converted into outputs in three important steps: fuzzification, decision making logic, and defuzzification. The methodology for two dimensional motion planning of a UAV using fuzzy logic is presented in [26].In this paper, the fuzzy inference system takes information in real time about the target location and obstacles within the sensing range of the sensors. The outputs consist of changes in heading angle and speed. A fuzzy logic approach was also examined in [27] for path tracking and obstacle avoidance. The obstacles considered in both situations were still or moving and appeared along the preplanned path instantaneously. The capabilities of a fuzzy logic based scheme for UAV navigation were demonstrated to be better as compared to a potential field controller in [28].

### 2.4.3. Neural Network

One of the interesting sources of inspiration for soft computing techniques is the way the human brain works. The human brain functionality relies on the complex interactions of a large number of specialized cells, the neurons, within a highly inter connected system called the neural network. The artificial neural network is a learning based computational model depending on information processing inspired by the biological nervous system. It attempts to mimic the way information is processed by the brain. The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pits [29]. A method based on neural computing that implements real time path planning of a mobile robot is presented in [30]. The method created a neural network model for a robot workspace capable of path adjustments in the presence of dynamic obstacles. The robot movement was controlled in the two dimensional space with random linear motion of planar obstacles. The back propagation neural network model was used to predict the movements of the dynamic obstacles.

### 2.5. Pose Based Methods

Pose based methods depend on creating a set of poses to define the commanded motion of the vehicle. A pose consists of associating a direction angle for 2-D path planning, or 2 direction angles for 3-D path planning to each way point. Therefore, the pose specifies the velocity vector direction at each way point. The aircraft reaches the poses in the order of their creation. The ability to allow the human operator to specify the waypoints makes it a convenient method that has a wide range of flexibility and is sometimes used for obstacle avoidance.

### 2.5.1. Dubins Algorithm

Dubins path consists of straight lines combined with constant curvature arcs, which produce the shortest path between two poses. Dubins path was introduced by Lester Dubins, a famous mathematician and statistician, in a paper published in 1957 [31]. To find the shortest path between two positions, the starting position with its orientation and the finishing position with its orientation should be defined as the starting pose and the finishing pose. According to Dubins, to achieve the minimum distance between two poses in a plane, the path could be either a CCC or a CLC or subset of them, where $C$ is a circular arc and $L$ is a straight line [32]. The work with Dubins path extended from two dimensional to three dimensional which makes it more applicable for aerial vehicles. In [33] a path planning algorithm based on three dimensional Dubins path algorithm was presented for UAVs avoidance of static and moving obstacles.

### 2.5.2. Clothoid Algorithm

Clothoid is a curve with a continuous curvature, which varies linearly over the path. The concept of clothoid path is the same as Dubins path, but the circular arcs are replaced with nonconstant curvature arcs defined by Fresnel integrals. Clothoid was probably first studied by Johann Bernoulli (1667-1748) around 1696 [34] .The other name of clothoid is Euler's spiral named after Leonhard Euler (1707-1783), or Cornu named after the physicist Marie Alfred Cornu (18411902). Ernesto Cesàro (1859-1906) named it a clothoid, which comes from the Greek "кок $\lambda$ ó $\theta \varepsilon v$ " meaning "to twist by spinning." The clothoid path was used as a geometric continuous curvature path planning for an automatic parallel parking feature in a motor vehicle in [35]. The strategy was
to create a simple geometric path for parallel parking in one or more maneuvers. It would be formed by circular arcs and then transformed to a continuous curvature path with the use of clothoids. The vehicle can park by following the generated control input for the steering angle and the longitudinal velocity. The used method is independent of the initial position and of the orientation of the vehicle. A combination of clothoid curvature and straight lines that form the shortest path was implemented in [36] using a quadrant based scheme that relies upon the relative position and angle of the poses as well as a numerical solution of the nonlinear vector equation.

### 2.5.3. Pythagorean Hodograph

Pythagorean Hodograph is a pose-based path planning method consisting of a continuous curvature path constructed from polynomial functions similar to the B-spline curve [32]. This method was first introduced in 1990 by Farouki and Sakkalis [37]. Figure (3) shows a comparison of a Dubins path with a Pythagorean hodograph path. The Dubins path is the shortest path but it lacks the curvature continuity. The Pythagorean hodgraph path has continuity, but is longer for the same curvature bound.


Figure 3. Comparison of a Dubins Path with a Pythagorean Hodograph Path [32].

## CHAPTER III

## SIMULATION ENVIRONMENT

### 3.1. Overview

The comparison and analysis of the two path generation approaches, which are the focus of this study, were performed using the UAV simulation environment [38] developed at West Virginia University (WVU). The WVU UAV simulation environment was designed to support the development, testing, and analysis of fault tolerant trajectory generation and tracking algorithms for UAVs. It includes several aircraft models and allows for the simulation of a variety of scenarios such as multiple vehicles, different types of trajectory tracking algorithms (controllers), different types of path generation algorithms, subsystem failures affecting actuators and sensors, including GPS malfunctions, different wind patterns and turbulence severity, and user imposed waypoints and obstacles/threat zones on 3-D map. The WVU UAV simulation environment significantly facilitates the processes of analysis and design of different trajectory planning and tracking algorithm for UAVs under normal and abnormal conditions.

The simulation environment consists of a set of MATLAB scripts, functions, graphical user interfaces (GUI) and Simulink blocks, which interact simultaneously with the FlightGear simulator [38] and the UAV Dashboard. FlightGear is a free open source simulation package for visualization that creates a sophisticated and open flight simulator framework. UAV Dashboard was generated using C\#. It can visualize and customize the obstacles, way points, poses, the flight initial point, the flight goal point, and the heading angles. The WVU UAV simulation can be run in real time or accelerated time, and has the advantage of wide flexibility for upgrading and modifying the software. Figure (4) explains the relationships between the different segments of the WVU UAV simulation environment.

Because there are general differences between aircraft, such as aircraft flight envelope and maneuverability properties, each aircraft has a specific aerodynamic model. Consequently, five aircrafts’ aerodynamic models were simulated and represented as Simulink block models in the simulation environment. Each model was based on the non-linear vehicle equations of motion and
lookup tables for aerodynamic and propulsion data. The inputs for each model consist of general control commands and external effects. The control commands include: elevator, aileron, rudder, and throttle signals. The external effects include, but are not limited to: wind, gusts, or turbulence.

The updates of the visualization segment consists of FlightGear and UAV Dashboard, which receive their inputs as an updated set of 41 variables at each integral step, thus producing a high fidelity animated scene. Another important segment represents the sensor feedback model, which is responsible for transforming the variables into factual sensor readouts. The sensor signal is used as feedback to control the flight path by trajectory tracking algorithms. One of the most important advantages of using the simulation environment is the flexibility to upgrade or replace the path planning or the trajectory tracking very easily without affecting other simulation segments.


Figure 4. General Architecture of the WVU Simulation Environment.

Planned paths are typically represented by way points that are geometrically and statically connecting start and end points. The trajectory generating algorithms included in the process information on the commanded velocity of the vehicle thus associating each path point to the moment in time when it should be reached.

The path planning and trajectory generation algorithms implemented within the WVU UAV simulation environment are of two types. One consists of first producing a geometrically computed path, which is then fed to the controller at each time step. The second consists of generating of the next desired way point at each time step of the location. The trajectory is stored as a matrix and each row of this matrix contains the position vectors in Cartesian coordinates at each time step. Stored trajectories can be uploaded to repeat similar simulation scenarios. The path planning and trajectory generation algorithms inputs can be defined by the user as: initial aircraft position with the heading angle, target and waypoint location(s), and the location(s) and radius of the threat(s).

The UAV Dashboard was used to create a set of text files containing all necessary information to specify these inputs. The aircraft path can be visualized through feeding the output of these algorithms back to UAV Dashboard using the User Datagram Protocol (UDP). The tracking algorithm receives the commanded position and velocity vectors from the trajectory generator in Cartesian coordinates. The aircraft model receives the control commands (elevator, aileron, rudder, and throttle) from the trajectory tracking algorithms. Figure (5) explains the data transfer signals used to pass data among the various algorithms.


Figure 5. Path Planning, Trajectory Generation, and Tracking Data Transfer.
The path planners and the trajectory generators are disabled when the manual flight is activated. In this situation, the control commands are received directly through the joystick. The joystick signal is calibrated so the control authority (the range of the control surface deflection) for the manual flight corresponds to the control authority for the trajectory trackers. Figure (6) represents the flowchart of the UAV simulation scenario setup.


Figure 6. Flowchart of the UAV Simulation Scenario Setup [39].

### 3.2. Graphical User Interface (GUI)

The WVU UAV simulation environment operates with simple user friendly interfaces using Windows 7 as an operating system for the simulation lab’s computer in the Department of Mechanical and Aerospace Engineering at West Virginia University. Figure (7) shows the interface with the UAV simulation environment. Notice that the joystick is typically used only for the manual flight operation and was not needed for the tests in this thesis.


Figure 7. User Interface with the UAV Simulation Environment.

### 3.2.1. Number of Vehicles GUI

The WVU simulation environment setup is a simple set of successive steps that start with running a MATLAB m-file followed by a GUI popup window requiring the user to choose between a single vehicle and multiple vehicles as shown in Figure (8). For the scope of this thesis, only the single vehicle was chosen to implement the results. By clicking the "LAUNCH" button this window will close and a new GUI window appears and will represent the general GUI.


Figure 8. Menu of Number of Vehicles Selection.

### 3.2.2. General GUI

The simulation environment's main selections can be accessed by users through the general GUI. The variety of the selections allows the user to test the flight of different models under different conditions and various circumstances. The general GUI consists of three sets of selections: Select Vehicle, Select Map, and Navigation and Control Option. Select Vehicle option represents five aircrafts that were modeled differently and each with its own MATLAB dynamic model, as well as a 3-D visualization implemented in FlightGear.

Only WVU YF22 was selected to implement the results in this thesis and all other models were neglected. Only one map was created for the simulation with a future ability to add new maps. This map is for the San Francisco Bay Area and the visual environment within FlightGear and UAV Dashboard map interface. The third selection of Navigation and Control Option can be neglected if the flight required is manually operated with no need for trajectory planning. Figure (9) shows these sets of selections.

Trajectory planning has to be chosen to show the 14 trajectory planning algorithms. Here we are interested in clothoid and Dubins algorithms to generate and plan a path. All other selections were neglected. Any trajectory planner can be selected in a combination with any conventional controller or adaptive controller. Otherwise only manual flights can be operated, which is not required for our test. If a conventional controller is selected, then a list of five different conventional controllers will appear as shown in Figure (10).

If the adaptive controller is selected, then six controller selections will appear as shown in Figure (11). After all the desired options are selected, clicking the "LOAD" button will save the selections in a file that will be used to start up the simulation. More details about the controllers can be found in sections 4.6 to 4.9.

### 3.2.3. Visualization

After clicking the "LOAD" button, click the "VISUALS" button to run the script which initializes both FlightGear and Dashboard interfaces for the selected aircraft. Figure (12) represents the Dashboard interface window with the selected map for the San Francisco Bay Area with a grid
consisting of a set of squares. Each square represents 200 square meters to simplify the evaluation of the flight distance and the aircraft heading angle. Figure (13) represents the FlightGear window for WVU YF22 model.


Figure 9. GUI for the Main Selections without the Navigation and Control Options.


Figure 10. Conventional Controller Selection GUI


Figure 11. Adaptive Controller Selection GUI.


Figure 12. UAV Dashboard.


Figure 13. FlightGear Visualization Software with WVU YF-22 Model.

### 3.2.4. Failure Options

The general GUI window will close when the "LAUNCH" button is clicked and the failure options GUI window will appear. The failure GUI has two selections: control surface failure and sensor failure. The control surface failure has six left and right failure situations for stabilator, aileron, and rudder. Each failure situation has two types of failures; "Locked Surface" which requires the user to specify the deflection of the locked surface (see Figure (14) and "Missing Surface" that requires the percentage of missing surface (see Figure (15)). For both failures, the time of occurrence must be set.


Figure 14. Locked Control Surface Failure GUI.


Figure 15. Missing Surface Failure.
If the sensor failure was selected, three options of sensors will be available: roll rate, pitch rate, and yaw rate. Each sensor has six options of different bias types as shown in Figure (16). The failure GUI can be ignored by unselecting any failure. If failure is either selected or unselected the "LOAD" button has to be pressed. This saves the desired parameters into a file and enables the "LAUNCH" button. Upon pressing this button, the Simulink model of the selected UAV is initialized.


Figure 16. Sensor Failure GUI.

### 3.3. Simulation Setup Using the Main Simulink Model

The main Simulink model in Figure (17) allows the user to modify the previous GUI selections instead of rerunning the simulation script and repeating all the previous steps. The main Simulink block has GPS, turbulence, and wind effects blocks, which include parameters that could be introduced to modify flight simulation scenario. The main Simulink model features a switch that allows the user to change between real time and accelerated time. Other interactive features include visualization of the results using Matlab plots and scopes and saving of results and other simulation outcomes.


Figure 17. WVU YF-22 Simulink Model.

### 3.3.1. Switch between Path Planning Algorithms

Switching between path planning algorithms can be performed by clicking on appropriate blocks within the Simulink model, without running repeatedly the GUI for setup. The masked "Follow Trajectory" block allows the user to switch between path generations algorithms. The blocks inside the blue frame in Figure (18) represent the available path generation algorithms. By clicking the desired path planning block the program will switch the algorithms. In this thesis only clothoid and Dubins algorithms are used and all other path planning algorithms will be ignored.


Figure 18. Changing the Path Generation Algorithm within the Simulink Model.

### 3.3.2. Switch between Trajectory Tracking Algorithms (Controllers)

The trajectory tracking algorithms are expected to follow a commanded trajectory while minimizing the tracking errors. To select one of the conventional controllers, the user must double click on the conventional controller block. A popup window showing all five available conventional controller blocks will appear, as seen in Figure (19). After the selection of any controller by double clicking the block, its color will change to green. Similar steps must be performed to select an adaptive controller. Six different adaptive controllers are currently implemented, as shown in Figure (20).


Figure 19. Conventional Controllers.


Figure 20. Adaptive Controllers.

### 3.3.3. Setting up a Failure Scenario

The Adjust Failure Scenario block on the main Simulink model allows the user to switch between all failures options in order to test any desired types of actuator or sensor failures for a specific scenario. By double clicking on the Adjust Failure Scenario block the failure options GUI window will appear and all the previous steps in section 3.2.4 for failure selection will be the same to select the desired type of failure.

### 3.3.4. Other Simulink Blocks and Parameters

A "manual switch" block has been created that connects and disconnects the GPS block to the trajectory trackers. The wind direction effects can be applied by double clicking on the "constant" block that is connected to the "WindDirection" block in Figure (17), and insert the desired angle in degrees. This process is repeated with the "WindSpeed" block to apply the wind speed effects in units of kts.

If for any reason the visualization software (FlightGear and UAV Dashboard) needs to be restarted, double clicking on the needed visualization software icon inside the Simulink block control will start the software. These icons are represented in Figure (17) for the FlightGear and the UAV Dashboard.

To switch between real and accelerated time, simply double click on the icon titled real time or accelerated time inside the Simulink block control. All important outcomes, plots, scopes and results can be saved by clicking on the "Save" block, which is an automated data acquisition tool described in section 5.5.

## CHAPTER IV

## PATH GENERATION ALGORITHMS

### 4.1. Dubins Algorithm

### 4.1.1. Dubins Path Planning

Dubins curve is a simple geometrical solution to solve the shortest path problem, which makes it more attractive than other approaches that require more complicated mathematical tools such as, covariance dynamics for path planning of UAV [40]. An example that demonstrates the use of a Dubins path is finding the quickest way to park a car, when the car does not face directly towards the parking space.

Dubins car's shortest path consists of a combination of three motion primitives, with constant action over a time interval applied by each individual primitive. Dubins car is considered as a nonholonomic system because it is subject to nonholonomic constraints [3]. For example, if the velocity vector for Dubins car " $u$ " is forward, has a constant magnitude, and is able to change its direction, then the only actions needed to follow the shortest paths are $u \in(L, S, R)$, where the L primitive turns the car as sharply as possible to the left, the S primitive drives the car straight ahead, and the R primitive turns the car as sharply as possible to the right. The six possible optimal combinations of these three primitives that represent the shortest path are :\{ LRL, RLR, LSL, RSR, RSL, and LSR \}. These combination can be compressed to more general terminology: "CSC" and "CCC," where "C" represents a circular arc and "S" represents a straight segment. The shortest path of any of these combinations is called Dubins curve. Figure (21) shows two combinations of Dubins curves.

UAVs can be considered similar to a Dubins car when the UAV's shortest path in two dimensional space is determined by using the Dubins approach. Like any other robots, UAVs have a minimum turning radius that depend on the geometrical and the physical properties of the UAV. Any object moving with velocity " $v$ " about a circle with radius " rturn " has an angular velocity:

$$
\begin{equation*}
\omega=\frac{v}{r_{t u r n}} \tag{1}
\end{equation*}
$$



Figure 21. The Trajectories of Two Combinations CLC and CCC.

The UAV configuration can be expressed by a point with 2-D coordinates $(x, y)$ and the heading angle $(\psi)$ as a triplet $(x, y, \psi)$. If the UAV moves in a straight line from point $\mathrm{A}\left(x_{1}, y_{1}, \psi_{1}\right)$ to point $\mathrm{B}\left(x_{1}+v \cos \left(\psi_{1}\right), y_{1}+v \sin \left(\psi_{1}\right), \psi_{1}\right)$ then the $x$ coordinate changes over time as a function of $\cos (\psi)$ therefore $\dot{x}=\cos (\psi)$.The UAV is a nonlinear system, but an approximation of linear equations describe the full system as follows:

$$
\begin{gather*}
\dot{x}=\cos (\psi)  \tag{2}\\
\dot{y}=\sin (\psi)  \tag{3}\\
\dot{\psi}=\omega=\frac{v}{r_{t u r n}} \tag{4}
\end{gather*}
$$

### 4.1.2. Dubins Trajectory Generation

The following discussion is restated from [41], in 2-D Dubins algorithm all vector components are with respect to Earth Reference Frame (E). To find the shortest path between two positions using Dubins, one should define the path curvature and its tangent, as well as the start and finish poses. The pose can be defined as follows:

$$
\begin{equation*}
[\mathrm{P}]_{\mathrm{E}}=\left[\mathrm{X}_{\mathrm{E}} \mathrm{Y}_{\mathrm{E}} \psi \mathrm{k}\right] \text {, where } \mathrm{k}=\frac{1}{R} \tag{5}
\end{equation*}
$$

where XE and YE are the x and y components with respect to the Earth Coordinate System, respectively, $\psi$ is the pose heading, k is the path maximum curvature and R is the turn radius.

The WVU simulation environment is able to generate a Dubins path passing from the start pose smoothly to the finish pose, using only four primitives' combinations: RR, RL, LR, and LL. In these combinations, R is the right turn and L is the left turn; this can be expressed in more general terminology as curve-straight-curve (CLC). Notice that the primitive S for straight line wasn't mentioned in the primitives combinations for the sake of simplicity.

The path between start and goal points is a result of a computation process that requires one to specify the start and the initial poses and their associated curvatures, which were chosen to be corresponding to the minimum turning radius of the UAV [42]. The turning path is a circular arc, and because there are left and right turns, each pose will be tangent to two circles with centers located at:

$$
\left[\vec{r}_{O C}\right]_{E}=\left[\begin{array}{l}
x  \tag{6}\\
y
\end{array}\right]_{E} \pm \frac{1}{k}\left[\begin{array}{l}
\cos \left(\psi+\frac{\pi}{2}\right) \\
\sin \left(\psi+\frac{\pi}{2}\right)
\end{array}\right]
$$

where $\vec{r}_{o c}$ is the vector from start circle to the end circle and the positive and negative signs refers to right and left turns, respectively.

The path followed by the UAV depends on the order in which each pose was created by the UAV Dashboard user, and consists of CLC segments; the straight line segment (L) is a tangent between the two circular arcs (C). Since the path is between two poses, there are four circles and four tangents: two tangents and two cross tangents. These are shown in Figure (22) [42]. Only one path is desirable and this is the shortest path. All other paths will be neglected. The shortest path is chosen based on the heading angle direction of start and end poses. The start pose heading angle will eliminate one straight tangent and one cross tangent, and the finish pose heading direction will eliminate one of the two remaining tangents; the full sketch is provided in Figure (23). If the aircraft turns to the right (clockwise), the circle it follows is called the "right circle," while the circle it follows when making left turns (counterclockwise) is called the "left circle." The CLC combinations can be explained as following:

Right-Right (RR): The straight tangent of the initial pose's right circle is connected to the straight tangent of the finish pose's right circle.

Left-Left (LL): The straight tangent of the initial pose's left circle is connected to the straight tangent of the finish pose's left circle.

Right-Left (RL): The straight tangent of the initial pose's right circle is connected to the straight tangent of the finish pose’s left circle.

Left-Right (LR): The straight tangent of the initial pose's left circle is connected to the straight tangent of the finish pose's right circle.

[^0]

Figure 22. Tangent Lines between Two Circles.


Figure 23. Relevant Path Tangents [42].
A. Computing the Straight Tangent Solutions

Figure (24) shows the straight tangent construction geometry [42].


Figure 24. Straight Tangent Construction Geometry.
The centerline distance between the centers of start and finish circles is $D$ :

$$
\begin{equation*}
D=\sqrt{\left(X_{f c}-X_{S C}\right)^{2}+\left(y_{f c}-y_{S C}\right)^{2}} \tag{7}
\end{equation*}
$$

where $\left(x_{s c}, y_{s c}\right),\left(x_{f c}, y_{f c}\right)$ are the coordinates of the centers of the start and finish circles, respectively.

The angle between the centerline and the slope of the tangent line is $\alpha$ :

$$
\begin{equation*}
\alpha=\sin ^{-1}\left(\frac{R_{f}-R_{s}}{D}\right) \tag{8}
\end{equation*}
$$

where $R_{S}, R_{f}$ are the start and finish circles' radius respectively.
The angle $\beta$ is the slope of the centerline:

$$
\begin{equation*}
\beta=\tan ^{-1}\left(\frac{y_{f c}-y_{s c}}{x_{f c}-x_{s c}}\right) \tag{9}
\end{equation*}
$$

For the right-right path combination:

$$
\begin{array}{ll}
\tau_{s}=\beta-\alpha+\frac{3 \pi}{2} & \tau_{s} \in[0,2 \pi] \\
\tau_{f}=\beta-\alpha+\frac{3 \pi}{2} & \tau_{f} \in[0,2 \pi] \tag{11}
\end{array}
$$

where $\tau_{s}$ and $\tau_{f}$ are the tangent location angles from center of start and finish circles, respectively.

For the left-left path combination,

$$
\begin{array}{ll}
\tau_{s}=\beta+\alpha+\frac{\pi}{2} & \tau_{s} \in[0,2 \pi] \\
\tau_{f}=\beta+\alpha+\frac{\pi}{2} & \tau_{f} \in[0,2 \pi] \tag{13}
\end{array}
$$

The path starts at a pose that represents the start point of the start curve segment (C), which is a part of a circle tangent to this pose. This point has a coordinate $\left(x_{s}, y_{S}\right)_{E}$, and this position is specified by the user on the UAV Dashboard. The end point of the curve segment (C) that represents the start point of the straight tangent $(\mathrm{S})$ has coordinates $\left(x_{t x}, y_{t x}\right)_{E}$ that are calculated in equation (14). The end point of the straight tangent is the start point of the finish curve segment (C) and has coordinates $\left(x_{t n}, y_{t n}\right)_{E}$ that are calculated in equation (15). The finish curve segment (C) ends at the finish pose which is also specified by the user on the UAV Dashboard. Its coordinates are $\left(x_{f}, y_{f}\right)_{E}$.

$$
\begin{align*}
& {\left[\begin{array}{l}
x_{t x} \\
y_{t x}
\end{array}\right]_{E}=\left[\begin{array}{l}
x_{s c} \\
y_{s c}
\end{array}\right]_{E}+R_{S}\left[\begin{array}{l}
\cos \left(\tau_{s}\right) \\
\sin \left(\tau_{s}\right)
\end{array}\right]}  \tag{14}\\
& {\left[\begin{array}{l}
x_{t n} \\
y_{t n}
\end{array}\right]_{E}=\left[\begin{array}{l}
x_{s c} \\
y_{s c}
\end{array}\right]_{E}+R_{S}\left[\begin{array}{l}
\cos \left(\tau_{s}\right) \\
\sin \left(\tau_{s}\right)
\end{array}\right]} \tag{15}
\end{align*}
$$

The next steps are to find the sweep angle $(\mu)$ for the start and finish curves. The direction depends on whether the solution is right-right or left-left.

$$
\left.\left.\begin{array}{l}
{\left[\begin{array}{l}
x_{c_{S}} \\
y_{c_{S}} \\
z_{c_{S}}
\end{array}\right]_{E}=\left[\left[\begin{array}{cc}
x_{s}- & x_{c s} \\
y_{s}- & y_{c s} \\
0
\end{array}\right]_{E} \times\left[\begin{array}{c}
x_{t x}- \\
x_{c s} \\
y_{t x}- \\
0
\end{array} y_{c s}\right.\right.}
\end{array}\right]_{E}\right]\left[\begin{array}{c}
x_{c_{F}} \\
y_{c_{F}}=\left[\left[\begin{array}{cc}
x_{t n}- & x_{c f} \\
z_{c_{F}}
\end{array}\right]_{E}=\left[\begin{array}{cc}
x_{f}- & x_{c f} \\
y_{f f} \\
0 & y_{c f}
\end{array}\right]_{E}\right. \tag{17}
\end{array}\right] .
$$

The following conditional statement switches directions depending on whether the solution is right-right or left-left.

If sign $z_{c_{s}}==$ start turn direction

$$
\left.\mu_{S}=\cos ^{-1}\left(\begin{array}{ll}
x_{s}- & x_{c s}  \tag{18}\\
y_{S}- & y_{c s}
\end{array}\right]_{E}^{T} \cdot\left[\left.\begin{array}{ll}
x_{t x}- & x_{c s} \\
y_{t x}- & y_{c s} \\
{\left[\begin{array}{ll}
x_{s}- & x_{c s} \\
y_{s}- & y_{c s}
\end{array}\right]_{E}}
\end{array} \right\rvert\,\left[\begin{array}{ll}
x_{t x}- & x_{c s} \\
y_{t x}- & y_{c s}
\end{array}\right]_{E}\right]\right), \mu_{s} \in[0, \pi]
$$

else if sign $z_{c_{S}}==$ opposite start turn direction

If $\operatorname{sign} \boldsymbol{z}_{\boldsymbol{c}_{\boldsymbol{F}}}==$ finish turn direction

$$
\mu_{f}=\cos ^{-1}\left(\left.\frac{\left[\begin{array}{ll}
x_{t n} & -x_{c f}  \tag{20}\\
y_{t n} & -y_{c f}
\end{array}\right]_{E}^{T} \cdot\left[\begin{array}{cc}
x_{f}- & x_{c f} \\
y_{f}- & y_{c f}
\end{array}\right]_{E}}{\|\left[\begin{array}{ll}
x_{t n}- & x_{c f} \\
y_{t n}- & y_{c f}
\end{array}\right]_{E}\left|\left[\begin{array}{ll}
x_{f}- & x_{c f} \\
y_{f}- & y_{c f}
\end{array}\right]_{E}\right|} \right\rvert\,\right), \mu_{f} \in[0, \pi]
$$

else if sign $z_{C_{F}}==$ opposite finish turn direction

$$
\mu_{f}=2 \pi-\cos ^{-1}\left(\left.\frac{\left[\begin{array}{ll}
x_{t n}- & x_{c f}  \tag{21}\\
y_{t n}- & y_{c f}
\end{array}\right]_{E}^{T} \cdot\left[\begin{array}{ll}
x_{f}- & x_{c f} \\
y_{f}- & y_{c f}
\end{array}\right]_{E}}{\left.\left.\|\left[\begin{array}{ll}
x_{n t-} & x_{c f} \\
y_{t n}- & y_{c f}
\end{array}\right]_{E}| | \right\rvert\, \begin{array}{ll}
x_{f}- & x_{c f} \\
y_{f}- & y_{c f}
\end{array}\right]_{E} \mid} \right\rvert\,\right), \mu_{f} \in[\pi, 2 \pi]
$$

Once the start and endpoints and sweep angles are determined, the aircraft trajectory is completely defined and way points can be generated.

## B. Computing the Cross Tangent Solutions

The steps are similar to the procedure above, starting with calculating the centerline distance between start and finish circles $D$ according to equation (7). Then calculating the intermediate angle $\beta$ according to equation (9), the angle $\alpha$ is calculated as shown in the following equation:

$$
\begin{equation*}
\alpha=\sin ^{-1}\left(\frac{R_{f}+R_{S}}{c}\right) \tag{22}
\end{equation*}
$$

For the right-left path combination

$$
\begin{align*}
\tau_{s} & =\beta+\alpha-\frac{\pi}{2} \tau_{s} \in[0,2 \pi]  \tag{23}\\
\tau_{f} & =\beta+\alpha+\frac{\pi}{2} \tau_{f} \in[0,2 \pi] \tag{24}
\end{align*}
$$

and for the left-right path combination,

$$
\begin{gather*}
\tau_{s}=\beta-\alpha+\frac{\pi}{2} \tau_{s} \in[0,2 \pi]  \tag{25}\\
\tau_{f}=\beta-\alpha+\frac{3 \pi}{2} \tau_{f} \in[0,2 \pi] \tag{26}
\end{gather*}
$$

Equations (14) through (21) can be used to calculate the curves end points and the sweep angles, such that the aircraft trajectory can then be generated. Figure (25) shows the cross tangent construction geometry [42].


Figure 25. Cross Tangent Construction Geometry.

### 4.2. Clothoid Algorithm

### 4.2.1. Clothoid Path Planning

Dubins trajectory generation methodology is frequently applied to produce the shortest path between two poses, due to the simplicity of the geometry that guarantees the production of a smooth path. The position and the velocity for Dubins path are continuous, while the acceleration is discontinuous because of the instantaneous changes in commanded lateral acceleration at the transition points between circular arcs and straight segments. Such a path with this instantaneous changes can be achieved with an adequate level of accuracy if the aircraft can perform these abrupt changes fast enough. Therefore, the non-continuous lateral acceleration command may make Dubins path generation non-desirable for aircrafts with slower response. Following Dubins path may lead such aircrafts to experience higher tracking errors and sometimes lose the trajectory entirely. In general, many types of UAVs are designed to maximize the flight time rather than the performance, and thus do not perform quick responses. Therefore, it may be very desirable to overcome the lack of continuous acceleration. The commanded lateral acceleration is proportional to the path curvature; in other words, in order to obtain a second order continuous path, the curvature must be a continuous function as shown in (27):

$$
\begin{equation*}
a=v^{2} K \tag{27}
\end{equation*}
$$

where $a$ is the lateral acceleration of the aircraft , $v$ is the forward velocity of the aircraft and $K$ is the path curvature.

Clothoid or Euler curves are substituted for the circular arcs used in Dubins trajectory generation to produce a trajectory that directs the aircraft through a series of commanded poses using a piecewise continuous path. Fresnel integrals can be used to generate the clothoid arcs that exhibit linearly varying curvatures as functions of path length. The discontinuous curvature of the Dubins path versus the continuous curvature of the clothoid path are illustrated in Figures (26) and (27).


Figure 26. Dubins Path with Curvature Profile.


Figure 27. Clothoid Path with Curvature Profile.

### 4.2.2. Clothoid Trajectory Generation

More steps and calculations were needed to generate a flyable clothoid path, compared to Dubins path. Clothoid is more complicated and involves iteration algorithms. To initially generate the curve within the clothoid, two parameters need to be defined. These are the total sweep angle of the clothoid arc, $\phi$, and the maximum curvature of the arc, $\kappa$. The total path length of the curve, $\mathrm{h}_{\text {max }}$ is defined in equation (28).

$$
\begin{equation*}
h_{\max }=\frac{2 \phi}{k} \tag{28}
\end{equation*}
$$

The relationship between arc length and arc sweep for the clothoid curve is illustrated in Figure (28). The entire curve profile will be altered if the total arc sweep angle is altered with a constant maximum curvature. This is in opposition to Dubins curve where the curve profile depends on the curvature only.


Figure 28. Clothoid Arc Profile with Maximum Curvature Held Constant and Sweep Angle Increasing.

There are several steps that have to be considered to have a flyable clothoid path. These steps are as follows:

## A. Define Poses

Poses are defined in the same manner as for the Dubins trajectory generation,

$$
\begin{align*}
& \mathrm{p}_{\mathrm{S}}=\left[\mathrm{X}_{S} \mathrm{Y}_{S} \psi_{S} K_{S}\right]  \tag{29}\\
& \mathrm{p}_{\mathrm{F}}=\left[\mathrm{X}_{F} \mathrm{Y}_{F} \psi_{F} K_{F}\right] \tag{30}
\end{align*}
$$

where $\mathrm{P}_{\mathrm{S}}$ and $\mathrm{P}_{\mathrm{F}}$ are the start and finish poses, respectively, for a set of maneuvers; X and Y are the position with respect to the Earth X-axis and Y-axis, respectively; $\psi$ is the heading angle with respect to the Earth X -axis, and $K$ is the maximum curvature.

## B. Coordinate Axes and Notation

The Dubins trajectory generation only used the Earth's coordinate system, while to produce a clothoid path, four important coordinate systems will be needed. These are the Earth coordinate axes, denoted by subscript E , the start coordinate axes based upon the start pose and denoted by subscript S, the finish coordinate axes based upon the finish pose and denoted by subscript F, and the connection coordinate axes based upon the straight line connecting the two clothoid arcs and denoted by subscript A. These coordinate systems are illustrated in Figure (29).


Figure 29. Coordinate Systems.

## C. Numerical Solution of the Fresnel Integrals

As mentioned before the Fresnel integral is used to generate a clothoid curve; the following equation defines this integral:

$$
\begin{equation*}
x(h)=\int_{0}^{h} \cos (\varnothing) d q, \quad y(h)=\int_{0}^{h} \sin (\varnothing) d q \tag{31}
\end{equation*}
$$

where x and y are the coordinates for the clothoid system. The length of the clothoid arc is h and its total sweep angle $\emptyset$, which is given in the following equation:

$$
\begin{equation*}
\phi(q)=\frac{\mathrm{k}}{2 \mathrm{~h}} \mathrm{q}^{2} \tag{32}
\end{equation*}
$$

The following equations represent the scaled Fresnel integrals, which are solved in order to generate the clothoid curve in the clothoid axes.

$$
\begin{equation*}
\mathrm{c}(\mathrm{~h})=\sqrt{\frac{2 \mathrm{~h}}{\mathrm{k}}} \cdot \int_{0}^{\overline{\mathrm{h}}} \cos \left(\overline{\mathrm{q}}^{2}\right) \mathrm{d} \overline{\mathrm{q}}, \quad \mathrm{~S}(\mathrm{~h})=\sqrt{\frac{2 \mathrm{~h}}{\mathrm{k}}} \cdot \int_{0}^{\overline{\mathrm{h}}} \sin \left(\overline{\mathrm{q}}^{2}\right) \mathrm{d} \overline{\mathrm{q}} \tag{33}
\end{equation*}
$$

where:

$$
\begin{align*}
& \overline{\mathrm{h}}=\sqrt{\frac{\mathrm{kh}}{2}}  \tag{34}\\
& \overline{\mathrm{q}}=\sqrt{\frac{\mathrm{k}}{2 \mathrm{~h}} \mathrm{q}} \tag{35}
\end{align*}
$$

There is no explicit solution for equation (32), therefore the scaled Fresnel equation must be approximated numerically as follows:

$$
\begin{equation*}
c(x)=\sum_{n=0}^{\infty} \frac{(-1)^{n}}{(2 n)!(4 n+1)} x^{4 n+1}, s(x)=\sum_{n=0}^{\infty} \frac{(-1)^{n}}{(2 n+1)!(4 n+3)} x^{4 n+3} \tag{36}
\end{equation*}
$$

The number of iterations depends on the desired accuracy, then the results will be scaled to the appropriate values as follows:

$$
\begin{equation*}
C(h)=\sqrt{\frac{2 h}{k}} C(x), S(h)=\sqrt{\frac{2 h}{k}} S(x) \tag{37}
\end{equation*}
$$

## D. Generating the Clothoid

The path components X and Y are defined at distinct points using the numerical approximations yielded in equations (36). Each of these X and Y path components must be scaled using the relationships from equation (37). The curves generated using the Fresnel integral are initially generated within the clothoid coordinate system independent of the Earth axes.

## E. Conversion of Clothoid to Earth Coordinate System

The clothoid was produced as a clockwise turn arc with a curvature and length varying from 0 to maximum. To generate a curve with a counterclockwise turn, the sign of the Y coordinate must be switched to negative. To convert the clothoid from the clothoid axes, denoted by subscript C, to the start and finish axes, respectively, a translation and rotation are needed, as follows:

$$
\begin{align*}
& {\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CS}} \\
\mathrm{Y}_{\mathrm{CS}}
\end{array}\right]_{\mathrm{E}}=\left(\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CS}} \\
\mathrm{Y}_{\mathrm{CS}}
\end{array}\right]_{\mathrm{E}}-\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CSfirst}} \\
\mathrm{Y}_{\mathrm{CSfirst}}
\end{array}\right]_{\mathrm{C}}\right)+\mathrm{R}_{\mathrm{EA}}\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CS}} \\
\mathrm{Y}_{\mathrm{CS}}
\end{array}\right]_{\mathrm{C}}}  \tag{38}\\
& {\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CF}} \\
\mathrm{Y}_{\mathrm{CF}}
\end{array}\right]_{\mathrm{E}}=\left(\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CF}} \\
\mathrm{Y}_{\mathrm{CF}}
\end{array}\right]_{\mathrm{E}}-\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CF}} \text { last } \\
\mathrm{Y}_{\mathrm{CF} \text { last }}
\end{array}\right]_{\mathrm{C}}\right)+\mathrm{R}_{\mathrm{EA}}\left[\begin{array}{c}
\mathrm{X}_{\mathrm{CF}} \\
\mathrm{Y}_{\mathrm{CF}}
\end{array}\right]_{\mathrm{C}}} \tag{39}
\end{align*}
$$

### 4.3. Definition of Solution Space Quadrants

The solution space was represented as four quadrants based upon the position of the finish pose relative to the start pose. The start pose when compared with the sign of the total sweep angle $\phi_{\text {total }}$ will yield the natural choice for turn directions. The natural choices for turn directions, based upon quadrant and sign of $\phi_{\text {total, }}$ are listed in the table (1) below. More details about the solution space can be found in [36].

Table 1. Direction Choices Based Upon Quadrant and Sign of Total Sweep Angle.

| Quadrant | $\phi_{\text {total }} \geq 0$ | $\phi_{\text {total }} \leq 0$ |
| :---: | :---: | :---: |
| I | RR | RL |
| II | RL | RR |
| III | LL | LR |
| IV | LR | LL |

The following Figures (30) through (37) illustrate the definition of solution space quadrants:


Figure 30. End Point in Quadrant I with $\phi_{\text {total }} \geq 0$.


Figure 32. End Point in Quadrant III with $\phi_{\text {total }} \geq 0$.


Figure 31. End Point in Quadrant II with $\phi_{\text {total }} \geq 0$.


Figure 33. End Point in Quadrant IV with $\phi_{\text {total }} \geq 0$.


Figure 34. End Point in Quadrant I with $\phi_{\text {total }} \leq 0$.


Figure 36. End Point in Quadrant III with $\phi_{\text {total }} \leq 0$.


Figure 35. End Point in Quadrant II with $\phi_{\text {total }} \leq 0$.


Figure 37. End Point in Quadrant IV with $\phi_{\text {total }} \leq 0$.

## CHAPTER V

## EXPERIMENTAL DESIGN

### 5.1. Performance Metrics

The examined algorithms did not produce identical results; this can be recognized sufficiently by a set of performance metrics produced by two initial performance objectives, developed in [43]. One is based on the minimum error of the commanded trajectory followed by the aircraft. The following were considered as performance metrics of the first objective: the maximum absolute error, average absolute error, and the standard deviation of the tracking error. The other objective is based on the avoidance of control surfaces' saturation by supplying the commands gradually. The integral of the absolute value of the rate of change of deflection of the aileron, stabilator, rudder, and throttle were considered as performance metrics with respect to the desirable characteristics, as well as the percentage of samples at saturation of each of these control surfaces. The sum of the normalized components of each of the various performance parameters was calculated to determine the performance index for each tracking algorithm. The expected performance frontiers for each of these parameters were represented by an experimentally determined threshold, which was used to normalize each parameter and assign a relative grade from 0 to 1 [43].

For the tracking error, the three primary parameters (maximum error, average error, and standard deviation) must be evaluated for the horizontal plane error, vertical error, and total error.

$$
\begin{equation*}
P V_{T T}=\left[t t_{i} \mid i=1,2, \ldots, 9\right]=\left[\bar{e}_{X Y} \bar{e}_{Z} \bar{e}_{X Y Z} \bar{e}_{\max X Y} \bar{e}_{\max Z} \bar{e}_{\max X Y Z} \hat{e}_{X Y} \hat{e}_{Z} \hat{e}_{X Y Z}\right]^{T} \tag{40}
\end{equation*}
$$

where;
$P V_{T T}$ : Trajectory tracking specific performance vector.
$\bar{e}_{X Y}$ : Average of the combined XY trajectory tracking error.
$\bar{e}_{Z}$ : Average of the vertical trajectory tracking error.
$\bar{e}_{X Y Z}$ : Average of the combined XYZ trajectory tracking error.
$\bar{e}_{\text {maxXY }}$ : Maximum of the XY plane trajectory tracking error.
$\bar{e}_{\operatorname{maxz}}$ : Maximum of the vertical trajectory tracking error.
$\bar{e}_{\operatorname{maxXYZ}}$ : Maximum of the combined XYZ trajectory tracking error.
$\hat{e}_{X Y}$ : Standard deviation of the XY plane trajectory tracking error.
$\hat{e}_{Z}$ : Standard deviation of the vertical trajectory tracking error.
$\hat{e}_{X Y Z}$ : Standard deviation of the combined XYZ trajectory tracking error.

The tracking performance index was calculated as a weighted average of the individual tracking error components, with weights designated by subjective comparative significance. A trajectory tracking specific performance index $P I_{T T}$ is defined as:

$$
\begin{equation*}
P I_{T T}=w_{T T} \cdot P V_{T T} \tag{41}
\end{equation*}
$$

For control surface performance, there are two primary parameters (integral of the absolute value of the rate of change, and saturation percentage) to be evaluated for each control surface

$$
\begin{equation*}
P V_{C A}=\left[c a_{i} \mid i=1,2, \ldots, 8\right]=\left[I \dot{\delta}_{e} I \dot{\delta}_{a} I \dot{\delta}_{r} I \dot{\delta}_{t} S_{\delta e} S_{\delta a} S_{\delta r} S_{\delta t}\right]^{T} \tag{42}
\end{equation*}
$$

where,
$P V_{C A}$ : control activity specific performance vector. $I \dot{\delta}_{e}, I \dot{\delta}_{a}, I \dot{\delta}_{r}$ and $I \dot{\delta}_{t}$ : The integral of stabilator deflection, aileron deflection, rudder deflection and throttle command rate of change, respectively. $S_{\delta e}, S_{\delta a}, S_{\delta r}$ and $S_{\delta t}$ : The stabilator, aileron, rudder, and throttle saturation index respectively.

A surface activation performance index was calculated as a weighted average of the individual surface parameters, with weights designated by subjective comparative significance. A control activity specific performance index $P I_{C A}$ is defined as:

$$
\begin{equation*}
P I_{C A}=w_{C A} \cdot P V_{C A} \tag{43}
\end{equation*}
$$

The combination of the tracking error and surface activation performance indices will result in the total performance index for the tracking algorithm test using a specific weighted average. The most efficient algorithm can be evaluated by averaging all tests for the tracking algorithms. A trajectory tracking global performance index $P_{\text {UAV }}$ is defined as:

$$
\begin{equation*}
P I_{U A V}=\bar{w}_{T T} \cdot P V_{T T}+\bar{w}_{C A} \cdot P V_{C A} \tag{44}
\end{equation*}
$$

Note that $w_{T T}, w_{C A}, \bar{w}_{T T}$ and $\bar{w}_{C A}$ are heuristically selected normalization and desirability weights.

### 5.2. General Experimental Design

The comparison between the clothoid and Dubins based path generation algorithms is expected to reveal their impact on UAV performance in correlation with several vehicle, environment, and mission conditions. Therefore, the selection of the experimental factors is dictated by these conditions, while the selection of experimental outcomes is dictated by the performance metrics. For each factor, several levels were considered based on relevance, simulation capabilities, and the need to keep the experimental grid at a manageable level.

All flight conditions were considered for five different controllers within the WVU UAV simulation environment. These flight conditions are the nominal conditions and abnormal conditions including subsystem failure conditions, such as sensors and actuators failure, and upset environment conditions, such as wind directions, wind magnitudes, and turbulence. The potential effects of path geometric characteristics has also been investigated by considering the relative position and heading of start and end poses (quadrant and turn angle effects).

The experimental grid has been reduced such that each factor level is tested once for clothoid and once for Dubins with each of the five controllers, which means that each level was tested ten times. The total number of tests is 340 tests for 34 levels. This experimental design is presented in the flowchart of Figure (38). This chart follows a single path for each test, which means each test is for a single algorithm, controller, factor and level such that a comparison between clothoid and Dubins algorithms can be performed and effects of factors isolated and assessed.


Figure 38. Experimental Design Summary.

### 5.3. Graphical Distributions of Poses

A sin wave shaped distribution of way points/poses was used to examine the first seven factors for all controllers with clothoid and Dubins. Three poses were defined to form the sine wave shape. This path is shown in Figure (39).


Figure 39. Trajectory Shape for the First Seven Factors.

The turn angle effect and the quadrant effect factors are path orientation dependent, therefore each level has a different path shape. Two poses were created for the turning angle factor, and for quadrant factor, only one pose was created.

All Dubins' factors have the same commanded distance and all clothoid's factors have the same commanded distance except for the quadrant test analysis. The commanded distance for the first seven factors (path shown in Figure (40) were calculated for Dubins ( 4592 m ) and clothoid ( 4537 m ) by using integrals between points. A grid of squares, where each square represents 200 $\mathrm{m}^{2}$, was used to calculate the commanded distance for all orientation dependent levels in order for it to approximate the distance value of the sine wave shaped path. This was done using trigonometry relationships. The turn angle factor's levels are illustrated in Figures (41) through (42). The zero degree turning angle is just a straight line which means there is no turning as shown in Figure (41), but there is still a difference in commanded distance between clothoid and Dubins, because the trajectory generator uses different algorithms to integrate this distance. The 360 degrees turning path in Figure (44) is just a combination of two turns of 180 degrees; both are in clockwise direction. The Figures (45) through (48) show the different levels for the quadrant effect factor.


Figure 40. Zero Angle Trajectory.
Figure 41. $45^{0}$ Trajectory.


Figure $42.90^{\circ}$ Trajectory.

Figure 43. $180^{\circ}$ Trajectory.



Figure 44. $360^{\circ}$ Trajectory.


Figure 45. First Quadrant Trajectory and Positive Total Sweep Angle. Figure 46. Second Quadrant Trajectory and Positive Total Sweep Angle.


Figure 47. Third Quadrant Trajectory and Positive Total Sweep Angle. Figure 48. Fourth Quadrant Trajectory and Positive Total Sweep Angle.

### 5.4. Trajectory Tracking Control Laws

### 5.4.1. Fixed parameter control laws

## A. Position PID Control Laws

The forward, lateral, and vertical distance from the commanded trajectory in the inertial Earth reference frame ( $f, l$ and $h$ ), can be minimized by the Position PID controller, as shown in Figure (49) where PID refers to proportional, integral, and derivative control. Figure (50) shows the three unique modules form the Position PID controller [44], [45].


Figure 49. Geometry of Trajectory Tracking Error [44].


Figure 50. Position PID Controller.
The trajectory variable calculation module computes forward, lateral, and vertical distance errors and relative velocities from the reference trajectory. The outer loop module is able to compensate the lateral errors by creating a bank angle command, the forward errors by creating a throttle command, and the vertical errors by creating a pitch angle command:

$$
\begin{align*}
\phi_{d} & =k_{i} \dot{l}+k_{l} l  \tag{45}\\
\delta_{T} & =k_{\dot{f}} \dot{f}+k_{f} f  \tag{46}\\
\theta_{d} & =k_{h} \dot{h}+k_{h} h \tag{47}
\end{align*}
$$

In the inner loop, the attitude angle commands produced by the outer loop are converted into required aileron, rudder, and elevator deflections using the following equations:

$$
\begin{align*}
\delta_{a} & =k_{p} p+k_{\phi}\left(\phi-\phi_{d}\right)  \tag{48}\\
\delta_{r} & =k_{r} r+k_{\emptyset r}\left(\phi-\phi_{d}\right)  \tag{49}\\
\delta_{e} & =k_{q} q+k_{\theta}\left(\theta-\theta_{d}\right) \phi \tag{50}
\end{align*}
$$

## B. Outer Loop NLDI Control Laws

The performance and the robustness of the control laws can be improved if nonlinear dynamic inversion (NLDI) is used in the outer loop [44]. The control surface deflection commands are obtained using equations (48) to (50) for the inner loop. Equation (46) is used to obtain the throttle command. The following inversion equations were used to compute the required bank and pitch angles:
$\phi_{d}=\arctan \left\{\frac{1}{\operatorname{gcos} \gamma}\left[\ddot{\ddot{l}}_{d} \cos \left(x-x_{V}\right)+\ddot{f}_{d} \sin \left(x-x_{V}\right)\right]+\frac{v}{\mathrm{~g}} \Omega_{V}+\left[\dot{i} \sin \left(x-x_{V}\right)-\dot{f} \cos \left(x-x_{v}\right)-\frac{\Omega_{V}}{\mathrm{~g} \cos \gamma}\right]\right\}$
$\theta_{d}=\frac{m}{K_{T} \cos \gamma}\left[\ddot{l}_{d} \sin \left(x-x_{V}\right)-\ddot{f}_{d} \cos \left(x-x_{V}\right)\right]+\frac{1}{K_{T}}\left(\frac{1}{2} \rho_{0} V^{2} s\left(C_{D 0}+C_{D \alpha} \alpha_{0}\right)+M g \sin \gamma-T_{0}\right)-\frac{m}{k_{T} \cos \gamma} \Omega_{V}\left[i \cos \left(x-x_{V}\right)+\right.$ $\left.\dot{f} \cos \left(x-x_{V}\right)\right]$

Approximate lateral and forward acceleration expressions were obtained using the following equations:

$$
\begin{gather*}
\ddot{l}_{d}=-k_{l s} \dot{l}-k_{l} l  \tag{53}\\
\ddot{f}_{d}=-k_{f s} \dot{f}-k_{f} f \tag{54}
\end{gather*}
$$

## C. Extended NLDI Control Laws

A method described in [46] was used to extend the inversion mechanism to the inner loop. Two phases carry out the inner loop NLDI control. The first one, called "slow mode", receives the pitch, roll, and yaw attitude angle errors and produces commands for pitch, roll and yaw rates:

$$
\begin{gather*}
{\left[\begin{array}{l}
p_{d} \\
q_{d} \\
r_{d}
\end{array}\right]=\left[\begin{array}{ccc}
1 & \sin \phi \cos \theta & \cos \phi \tan \theta \\
0 & \cos \phi & -\sin \phi \\
0 & \sin \phi \sec \theta & \cos \phi \sec \theta
\end{array}\right]^{-1}\left[\begin{array}{l}
U_{\phi} \\
U_{\theta} \\
U_{\psi}
\end{array}\right]}  \tag{55}\\
\left(\begin{array}{c}
U_{\phi} \\
U_{\theta} \\
U_{\psi}
\end{array}\right)=\left(\begin{array}{c}
K_{\phi}\left(\phi_{d}-\phi\right) \\
K_{\theta}\left(\theta_{d}-\theta\right) \\
K_{\psi}\left(\psi_{d}-\psi\right)
\end{array}\right) \tag{56}
\end{gather*}
$$

The second phase is called "fast mode". This phase receives the angular rate commands output from the slow mode controller as inputs and produces the actuators (aileron, elevator, and rudder) commands. Figure (51) shows these two stages. The angular acceleration compensation used by the fast mode inversion is defined as following:

$$
\left(\begin{array}{l}
U_{p}  \tag{57}\\
U_{q} \\
U_{r}
\end{array}\right)=\left(\begin{array}{l}
K_{p}\left(p_{d}-p\right) \\
K_{q}\left(q_{d}-q\right) \\
K_{r}\left(r_{d}-r\right)
\end{array}\right)
$$

The necessary control surface deflection commands can be obtained as following:

$$
\begin{gather*}
\delta_{e}=\frac{C_{m}(x, \delta)-C_{m o}-C_{m \alpha} \alpha-\frac{\bar{c}}{2 V} C_{m q} q}{C_{m} \delta_{e}}=\frac{\frac{M_{A d}}{\bar{a} S_{\bar{c}}}-C_{m o}-C_{m \alpha} \alpha-\frac{\bar{c}}{2 V} C_{m q} q}{C_{m} \delta_{e}}  \tag{58}\\
\delta_{a}=\frac{C_{l \delta r} b_{2}-C_{n \delta r} b_{1}}{C_{l \delta r} C_{n \delta a}-C_{n \delta r} C_{l \delta a}}  \tag{59}\\
\delta_{r}=\frac{C_{n \delta a} b_{1}-C_{l \delta a} b_{2}}{C_{l \delta r} C_{n \delta a}-C_{n \delta r} C_{l \delta a}} \tag{60}
\end{gather*}
$$

where:

$$
\begin{align*}
& b_{1}=C_{l}(x, \delta)-C_{l 0}-C_{l \beta} \beta-\frac{b}{2 V}\left(C_{l p} p+C_{l r} r\right)=\frac{L_{A d}}{\bar{q} S b}-C_{l 0}-C_{l \beta} \beta-\frac{b}{2 V}\left(C_{l p} p+C_{l r} r\right)  \tag{61}\\
& b_{2}=C_{n}(x, \delta)-C_{n 0}-C_{n \beta} \beta-\frac{b}{2 V}\left(C_{n p} p+C_{n r} r\right)=\frac{N_{A d}}{\bar{q} S b}-C_{n 0}-C_{n \beta} \beta-\frac{b}{2 V}\left(C_{n p} p+C_{n r} r\right) \tag{62}
\end{align*}
$$



Figure 51. Two Phase Dynamic Inversion Inner Loop NLDI Controller.

### 5.4.2. Adaptive Control Laws

## A. Adaptive \#1

Increased level of fault tolerance can be provided by modifying the previous control algorithms [43] as presented in Figure (52). An adaptive factor equation (63) can affect the six gains of the inner loop from equations (48) through (50).

$$
\begin{equation*}
u(k)=k_{p}\left(1-\eta f(\Delta u(k))\left(1+\frac{k_{i}}{z-1}+k_{d} \frac{z-1}{z}\right) e(k)\right) \tag{63}
\end{equation*}
$$

where $k_{p}, k_{i}$, and $k_{d}$ are PID gains, $\Delta u$ is the control gradient, and:

$$
\begin{equation*}
f(\Delta u(k))=1-\exp \left(\frac{-(\Delta u(k))^{2}}{a}\right) \tag{64}
\end{equation*}
$$

For equation (52), the pitch attitude channel is affected by the gain $k_{\theta}$ and can be modified as shown in the following equation:

$$
\begin{equation*}
k_{\theta}=\bar{k}_{1}\left\{1-\eta\left[1-\frac{2}{\exp \left[\left(\bar{k}_{2} \Delta k_{\theta}\right)^{2}\right]+\exp \left[-\left(\bar{k}_{2} \Delta k_{\theta}\right)^{2}\right]}\right]\right\} \tag{65}
\end{equation*}
$$

where $k_{1}$ and $k_{2}$ are constant gains and $\Delta k_{\theta}$ is the gradient of the adaptive gain.

All other controller parameters stay as they are without any change. The gains in equation (56) are adaptive, while the others in equation (57) remain fixed.


Figure 52. Biological Immune System Feedback Response Diagram.

## B. Adaptive \#2

To provide faster adaptation along with high robustness, an alternative adaptive control architecture based on the L1 approach [47] was attempted. L1 is referring to use L1-norm also known as least absolute deviations (LAD), least absolute errors (LAE). It is basically minimizing the sum of the absolute differences between the target value and the estimated values.

The $L 1$ control laws adapts an extension of the model reference adaptive control architecture consist a low pass filter. This filter separates performance and robustness and guarantees a bandwidth limited control signal with high adaptation rate. The benefit of using this approach is to simplify theoretical assurance for stability and performance as well as systematic design procedures, considerable minimization for the tuning attempt especially when fault tolerance is aimed. Linear type controller and a nonlinear dynamic inversion based outer/inner loop controller were combined with the L1 adaptive control within the WVU UAV simulation environment. The block diagram of the latter is presented in Figure (53). More details about L1 adaptive controller can be found in [47].


Figure 53. NLDI Inner Loop Based Control Laws with L1 Adaptive Augmentation.

### 5.5. Automated Data Acquisition Tool

The different simulation scenarios were run and the results collected using an ad-hoc developed Matlab function as an automated data acquisition tool, named "Save function". This function was implemented as a Simulink block. The 340 tests 170 tests for clothoid and 170 tests for Dubins were run by the Save function, which also facilitates saving plots and data. It creates folders and subfolders and labels them for classification, updates folders or subfolders without deleting any content, and allows test double checking and repetition without data duplication if results appear to be problematic.

Before creating the Save function, a variable "Determinpath" was defined to specify if the trajectory was using Dubins or clothoid algorithm (for Dubins, Determinepath $=0$, and for clothoid, Determinepath =1). The quadrant factor is a geometrical property of the path, therefore it was saved for each test. The quadrant factor was saved by creating one row matrix called "quadrantmatrix", each cell representing a quadrant number for the targeted pose. A folder called "cases" with two subfolders called " $2 D$ Dubins" and " $2 D$ clothoid" is first generated and then the Save function will create subfolders inside to store the required plots and data.

After each test, the user has to save the UAV dashboard scenario to the workspace folder and label it "Scenario", then click on the "Save" block on the main Simulink to run the Save function. If the user runs the Save function without saving the UAV dashboard scenario, an error message will appear asking the user to save the dashboard scenario first as shown in Figure (54).


Figure 54. Error Message for Unsaved Scenario.
When clicking on the save block, the Save function will run and a data input widow will pop up (see Figure (55)) asking the user to enter the controller number, factor number and the level number. These numbers must be the same corresponding to the numbers in Figure (38) for controllers, factors, and levels.


Figure 55. Numbering Code Inputs to Save Each Test Result.

The save function will create a folder and label it with a name associated with the specific scenario. Then all the test outputs are saved to this folder, which resides inside " $2 D$ Dubins" folder or " $2 D$ clothoid" folder. The save function will also create a (.mat) file called "datafile", which contains a matrix called "output". This matrix has 26 columns (see Appendix C) and 240 rows representing the number of tests. Figure (56) shows the process of saving the data and plots. For simplicity, the output matrix is converted into a Microsoft Excel spread sheet.


Figure 56. Flow Chart of the Automated Data Acquisition Tool.

## CHAPTER VI

# ANALYSIS AND COMPARISON OF PATH GENERATION ALGORITHMS 

### 6.1. Variation of Bank Angle and Lateral Acceleration

In terms of UAV body axes, the curvature is proportional to the yaw rate and the torsion is proportional to the roll rate. However, the UAV will conduct a bank to turn maneuver, which means rolling to a specific bank angle to turn accompanied by compensation normal to the wings using the elevator.

The bank angle and the lateral acceleration are proportional to the curvature of the path according to equation (27), which is confirmed by the simulation results presented in Figures (57) and (58). A "sine wave" path was used in this test consisting of a straight segment, turn to right, straight segment, turn to left, and straight segment. It can be seen that there are significant differences, as expected, between the variations of the bank angle when using the two path generation algorithms at the transition points between curved and straight segments. The bank angle will change continuously and smoothly for clothoid and rather abruptly for Dubins path. In both cases a maximum bank angle of 55 degrees is reached and maintained at this value for about 3 seconds for clothoid and 6 second for Dubins. The lateral acceleration also varies more smoothly and reaches lower extreme values for the clothoid path as compared to the Dubins path.

The bank angle is adjusted by the controller according to the changes in the commanded path curvature. For example, for the clothoid path, the bank angle is reduced from 55 degrees to 22 degrees which means the UAV rolls to left, then increased to 35 degrees, which means the UAV rolls to right. The UAV rolls in opposite direction without changing the sign of bank angle to compensate for the trajectory errors. These changes in bank angle are relatively smooth and do not cause the UAV to roll excessively, which in turn results in better trajectory tracking. Then the bank angle will be continuously reduced to zero degree, which represents the straight line path.

For Dubins path, the bank angle will reduce from (55 degrees) to (-36 degrees), increase to ( 4 degrees) reduce to (- 7 degrees) and then increase to ( 0 degrees). This oscillation is the result of the abrupt change in the commanded curvature. The UAV will experience undesired changes in the sign of the roll attitude angle, which leads to larger tracking errors.


Figure 57. Variation of Bank Angle with Curvature Changes.


Figure 58. Variation of Lateral Acceleration with Curvature Changes.

### 6.2. Path Length Analysis

The commanded path length for clothoid path was approximately 4592m and for Dubins it was approximately 4537 m for all tests except for the quadrant analysis tests. As expected, the clothoid algorithm produces longer commanded paths than Dubins algorithm by about 55 m . It should be noted that this difference is relatively small for the paths considered in this study and may be much more significant if the commanded path necessitates large number of turns. Figure (59) shows the curvature differences between these two algorithms. The actual distances achieved by the vehicle are obviously different depending on the trajectory tracking errors. Figures (60) and (61) shows the differences between actual and commanded trajectories for clothoid and Dubins respectively, at Nominal Conditions for $L 1+$ PPID Controller. Figure (62) shows combination of the differences between actual and commanded trajectories for both clothoid and Dubins at nominal conditions for L1+PPID controller.

From these plots, one can notice that Dubins trajectory errors are larger than clothoid trajectory errors due to the discontinuous commanded curvature. Although the Dubins commanded path length is shorter, the actual length traveled by the vehicle typically is larger. For the Dubins path, the abrupt changes required for the bank angle are producing more control surface activity and more trajectory tracking errors resulting in a longer actual path than the one achieved with the clothoid path generation algorithm.


Figure 59. Difference between Commanded Curvature for Clothoid and Dubins Algorithms.


Figure 60. Clothoid at Nominal Conditions L1+PPID Controller.


Figure 61. Dubins at Nominal Conditions L1+PPID Controller.


Figure 62. Clothoid and Dubins Distances at Nominal Conditions L1+ PPID Controller.

Figure (63) presents the actual and commanded distances for the two path generation algorithm using the five different sets of control laws at nominal conditions. In all cases, as expected, the actual distance is larger than the commanded distance for both algorithms. However, while the commanded distance for clothoid is larger than the commanded distance for Dubins, the actual distance for Dubins is larger than the actual distance for clothoid, due to larger tracking errors.


Figure 63. Distance with Nominal Conditions in (m).

The charts in appendix A present the differences between Dubins actual and commanded path lengths, and clothoid actual and commanded path lengths, for all factors at all levels with all five controllers. The tendency from all these charts follows the trend presented in Figure (63). An additional example under sensor failure is presented in Figure (64).


Figure 64. Distance with Sensor Failure in (m).
The distance for any factor with the (L1+PPID) controller is always larger than other distances for the same factor with other controllers. This tendency of the (L1 + PPID) controller does not necessarily happen because of the high trajectory tracking errors, but it may be because of how the actual path follows the commanded path. Figure (65) is an inaccurate sketch showing
a commanded path with two trajectory paths that have the same trajectory errors, but the green path has a longer length than the blue path.


Figure 65. Different Actual Distances With The Same Trajectory Errors.
The results of the distance differences between the actual distance and the commanded distance for all factors are summarized in Table (2). This table shows the controller, factor and level for maximum and minimum actual path length and the averages. The wind direction factor produces the maximum clothoid and Dubins distances at level 270 degrees with (L1+ PPID) controller, and minimum path lengths at 0 degrees wind direction with Position PID controller. Figure (66) shows the direction effect on the UAV trajectory with clothoid, (L1 + PPID) controller at level 0 degree and 270 degrees. The 0 degrees wind will push the aircraft body closer to the commanded trajectory while the 270 degrees wind will push the aircraft body farther from the commanded path. In general, wind direction will not change the trend that clothoid path is always shorter than Dubins path.


Figure 66. Wind Direction Effect on the UAV Trajectory.

Table 2. Summary of Path Length Analysis.

|  | Dis. average <br> Clothid | Dis average <br> Dubins | Max Dis. Clothoid | Max Dis. Dubins | Factor at Max <br> Dis. Clothoid | $\begin{gathered} \left\lvert\, \begin{array}{c} \text { Factor at Max } \\ \text { Dis. Dubins } \end{array}\right. \\ \hline \end{gathered}$ | Min Dis Clothoid | Min Dis <br> Dubins | Factor at Min Dis. <br> Clothoid | Factor at Min <br> Dis. Dubins |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Nominal | 4582 | 4734 | 4862 | 4989 | $\begin{array}{\|l\|l\|} \hline 20 \text { L1.PPPID } \\ \text { Nomininal } \end{array}$ | $\begin{aligned} & 20 \text { L1+PPID } \\ & \text { Nominal } \end{aligned}$ | 4468 | 4649 | 20 Position PID Nominal | 2D Position PID Nominal |
| Sensors | 4583 | 4733 | 4866 | 5031 | 2 L1TPPID Sensor Yaw | $2011+$ PPID <br> Sensor Yaw | 4468 | 4581 | 2DPosition PID Sensor Roll | 2DPosition PID Sensor Roll |
| L. Act. Locked | 4619 | 4794 | 4889 | 4986 | 2011 PPID <br> Actuatorstab Left lock | 20 L1.PPID Actuatoralier Leftlock | 4470 | 4637 | 2 POosition PID Actuatorkudd Left lock | 20Position PID <br> Actuatorkudd Left lock |
| R. Act. Locked | 4619 | 4769 | 4861 | 4985 | 2011 PPID <br> Actuatoralier Right lock | 201.1 PPID <br> Actuatorailer <br> Right lock | 4470 | 4637 | 2D Position PID <br> ActuatorRudd <br> Rightlock | 2D Position PID <br> Actuatorkudd <br> Rightlock |
| L. Act. missed | 4586 | 4755 | 4872 | 5038 | 2011 PPID <br> Actuatoralier Left miss |  | 4468 | 4634 | 2D Position PID ActuatorRudd Left miss | 2DPostion PID <br> Actuatorkudd <br> Left miss |
| R. Act. missed | 4586 | 4751 | 4872 | 5038 | 2011 PPID <br> Actuatorailer <br> Rightmiss | 2011 PPID <br> Actuatorkudd <br> Right miss | 4468 | 4634 | 2 PPosition PID Actuatorkudd Right miss | 2DPosition PID <br> Actuatorkudd Rightmiss |
| Wind Dir. | 4509 | 4642 | 4912 | 5086 | $\begin{array}{\|l\|l\|} \hline 20 \mathrm{~L} 1+\mathrm{PPID} \text { wind } \\ \text { dir 27odeg } \end{array}$ | $\begin{array}{\|l} 20 \text { L1+PPID } \\ \text { wind dir } \\ \text { 27deg } \end{array}$ | 4220 | 4280 | 2 D Position PID wind dir Odeg | 2D Position PID wind dir Odeg |
| Wind mag. | 4595 | 4771 | 4900 | 5074 | $\begin{array}{\|l\|l\|} \hline 20 \mathrm{~L} 1+\text { PPID wind } \\ \text { mag } 8 \mathrm{knots} \end{array}$ | 20 LLIPPID <br> wind mag 8 <br> knots | 4487 | 4663 | 2 D Position PID wind mag 5 knots | 20N.DD outer wind mag 8 knots |
| Turbulence | 4557 | 4701 | 4830 | 4978 | $2011+$ PPID <br> Turbulance light | $2011+$ PPID <br> Turbulance <br> Moderate | 4339 | 4603 | 2D Position PID <br> Turbulance Sever | 2DPosition PID <br> Turbulance <br> Sever |
| Total | 4582 | 4739 | 4912 | 5086 | 2D L1+PPID <br> wind dir <br> 270deg | $\begin{array}{\|c} 2 D \operatorname{L1}+\text { PPID } \\ \text { wind dir } \\ \text { 270deg } \end{array}$ | 4220 | 4280 | 2D Position PID wind dir Odeg | 2D Position PID wind dir Odeg |
| Angle | 4274 | 4330 | 5205 | 5356 | 20 LITPPID Angle 360 | $\begin{aligned} & 20 \text { L1.+PPID } \\ & \text { Angle } 360 \end{aligned}$ | 3994 | 3996 | 20 Position PID <br> Angle zero | 2DPosition PID <br> Angle zero |
| Quadrant | 4327 | 4384 | 4640 | 4747 | 2011 PPID quadrant Third quadrant | 20 L1-PPID <br> quadrant <br> Second <br> quadrant | 4201 | 4214 | 2 PPosition PID quadrant Fouth quadrant | 2DPosition PID quadrant Fouth quadrant |

The effects on the path length produced by the pose heading angle change and the relative pose position of subsequent way points follows the same trends, that is the clothoid actual path is shorter than Dubins. Obviously, the distance for the path with heading change of 360 degrees is the highest. As shown in Table (2), the 360 degrees turn reaches the highest length value with $L 1+$ PPID controller. The shortest distance occurs for 0 degrees heading change (straight line) with Position PID controller. The clothoid actual path is shorter than Dubins in all cases. However, note that for the zero curvature case, for both algorithms, theoretically, both paths have the same length. The simulation shows small differences even for this case, due to computational dissimilarity.

For the quadrant factor, the maximum distance was at the third quadrant for clothoid and Dubins with (L1 + PPID) controller, while the shortest distance for clothoid and Dubins was at first quadrant with (Position PID) controller.

### 6.3. Performance Indices Analysis

The trajectory tracking (TT), control activity (CA), and total (TOT) performance indices (PI) were investigated and represented in Appendix B for all factors at all levels with all five controllers, for both the clothoid and Dubins algorithms. In Figure (67), these PIs are presented at nominal conditions and in Figure (68) under left actuator locked conditions.


Figure 67. PI with Nominal Conditions.


Figure 68. Left Actuator Locked.
In general, the tendency for all simulation cases is that the clothoid algorithm facilitates better performance than the Dubins algorithm. There are only a few situations in which the TT for Dubins is better than for clothoid by very small amounts, which are rather negligible. Note that the TT performance represents the actual path tracking of the commanded path in three dimensions. In addition to x and y axes errors, vertical errors are also considered. Figure (69) shows a three dimensional (3D) plot for commanded and actual trajectories for clothoid algorithm with PID Position controller at nominal conditions.


Figure 69. 3D Plot for Commanded and Actual Trajectories.

Table (3) summarizes the comparative results based on TT performance indices. TT performance indices for the sine wave path do not exceed 0.448 which is the maximum TT performance index for Dubins path with NLDI outer controller that occurs when the left rudder is damaged. Clothoid and Dubins have zero TT performance indices for some factors' levels as a result of the high trajectory errors in three dimensions.

The PID Position controller is the poorest controller under both nominal and failure conditions. There are only a few exceptions, which make it the less desirable controller among the ones considered, if the mission goal is high TT performance. Conventional and adaptive controllers with NLDI have better TT for both clothoid and Dubins paths. The 360 degrees turn case produces low TT performance indices for both clothoid and Dubins with Position PID controller because it consist of two 180 degrees turns and has the longest actual path, which also maximizes the error. For the quadrant factor, the first quadrant with $L 1+E N L D I$ controller has the best TT performance indices for both clothoid and Dubins, while the poorest TT is recorded for the third quadrant for clothoid and Dubins with L1+PPID controller.

Table (4) summarizes the simulation results in terms of the CA performance indices for all factors and levels considered. The CA performance indices for clothoid are in general better than Dubins for all levels because the discontinuous commanded curvature for the Dubins path requires the UAV to perform more intense control surface activity for tracking the path. The Position PID controller produces the best CA performance indices for all levels, while the conventional and adaptive controller based on NLDI has the poorest CA performance indices. This is expected as the controller with NLDI produces high TT performance indices, which is achieved through more intense actuation of the controls. The turn angle producing the lowest CA performance index was 360 degrees and the one producing the best was zero degrees, which is consistent with the fact that tracking errors are larger in turns. The best CA performance indices when varying the quadrant was obtained for the first quadrant with $L 1+$ PPID controller. The worst CA performance indices was obtained for final pose in the second quadrant with $L 1+E N L D I$ controller for clothoid and Dubins. The first quadrant was the best and the second quadrant was the worst, independent of control laws. This result is due to the fact that the first quadrant involves typically the least amount of turn, while the second quadrant involves the most amount of turn therefore involves more
control activity. The third quadrant has less amount of turn and longer length than the second quadrant, which produce more trajectory errors and less control activity than the second quadrant.

Table (5) presents the TOT performance indices for all levels which are the weighted values for both TT and CA performance indices. Note that the evaluation in terms of TOT PI may yield different outcomes depending on the relative priority assigned to trajectory tracking versus control activity, which are generally functions of mission objective and nature. However, considering that the separate analysis shows that the clothoid path generally facilitates better performance in terms of both TT and CA, it can be concluded that the ranking of the two approaches by the TOT PI will be invariant with respect to the priority weights assigned to the two major metrics within the TOT PI. Indeed, Table (5) shows that the clothoid path has the highest TOT performance indices for all factors and levels.

Table 3. Summary of Trajectory Tracking PI Results.

|  | TT. average Clothid | TT average Dubins | Max TT Clothoid | $\begin{aligned} & \text { Max TT } \\ & \text { Dubins } \end{aligned}$ | Factor at Max <br> TT Clothoid | Factor at Max <br> TT Dubins | $\operatorname{Min} T$ Clothoid | $\operatorname{Min} T$ <br> Dubins | Factor at Min TT Clothoid | Factor at Min TT Dubins |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Nominal | 0.400 | 0.336 | 0.444 | 0.406 | NLDI outer <br> Nominal | L1+ENLDI <br> Nominal | 0.353 | 0.181 | Position PID Nominal | Position PID Nominal |
| Sensors | 0.389 | 0.330 | 0.445 | 0.436 | NLDI outer Sensor Roll | L1+ENLDI Sensor Pitch | 0.212 | 0.080 | Position PID Sensor Pitch | Position PID Sensor Pitch |
| L. Act. Locked | 0.249 | 0.237 | 0.439 | 0.401 | NLDI outer <br> ActuatorRudd <br> Left lock | L1+ENLDI <br> ActuatorRudd <br> Left lock | 0.000 | 0.000 | Position PID <br> ActuatorStab Left lock <br> , \& NLDI Extended ActuatorAiler Left lock | Position PID, NLDI outer, \& NLDI Extended, (ActuatorStab left lock) |
| R. Act. Locked | 0.290 | 0.229 | 0.439 | 0.401 | NLDI outer ActuatorRudd Rightlock | L1+ENLDI <br> ActuatorRudd <br> Rightlock | 0.000 | 0.000 | Position PID <br> ActuatorStab Right <br> lock | Position PID, NLDlouter, NLDI Extended, \& L1+ENLDI Extended (ActuatorStab Right lock ) |
| L. Act. missed | 0.375 | 0.321 | 0.445 | 0.406 | NLDI outer <br> ActuatorRudd <br> Left miss | NLDI <br> Extended <br> ActuatorStab <br> Left miss | 0.302 | 0.131 | 2D Position PID <br> ActuatorStab Left miss | Position PID <br> ActuatorRudd Left miss |
| R. Act. missed | 0.380 | 0.326 | 0.445 | 0.448 | NLDI outer ActuatorRudd Right miss | L1+ENLDI <br> ActuatorStab <br> Right miss | 0.301 | 0.131 | Position PID <br> ActuatorStab Right miss | Position PID <br> ActuatorRudd Right miss |
| Wind Dir. | 0.276 | 0.273 | 0.399 | 0.388 | Position PID wind dir 90deg | L1+ENLDI <br> wind dir <br> 90deg | 0.000 | 0.000 | L1+ENLDI wind dir 180deg | PositionPID wind dir Odeg |
| Wind mag. | 0.370 | 0.314 | 0.433 | 0.392 | NLDI outer wind mag 5 knots | NLDI <br> Extended <br> wind mag 5 <br> knots | 0.215 | 0.104 | Position PID wind mag 8 knots | Position PID wind mag 8 knots |
| Turbulence | 0.305 | 0.236 | 0.437 | 0.405 | NLDI outer <br> Turbulance <br> light | NLDI <br> Extended <br> Turbulance <br> light | 0.000 | 0.000 | L1+PPID Turbulance Moderate \& Sever | L1+PPID Turbulance <br> Moderate, Sever \& light |
| Total | 0.337 | 0.289 | 0.445 | 0.448 | NLDI outer <br> ActuatorRudd <br> Left miss | L1+ENLDI <br> ActuatorStab <br> Right miss | 0.000 | 0.000 | All zeros above | All zeros above |
| Angle | 0.330 | 0.528 | 0.965 | 0.958 | NLDI outer ActuatorRudd Left miss | L1+ENLDI <br> Angle zero | 0.625 | 0.092 | Position PID Angle 360 | Position PID Angle 360 |
| Quadrant | 0.942 | 0.920 | 0.942 | 0.916 | 2D L1+ENLDI <br> first quadrant | 2D L1+ENLDI first quadrant | 0.366 | 0.360 | 2D L1+PPID Third quadrant | 2D L1+PPID Third quadrant |

Table 4. Summary of Control Activity PI Results.

| Results <br> Factors | CA average Clothid | CA <br> average Dubins | Max CA <br> Clothoid | Max CA Dubins | Factor at <br> Max CA <br> Clothoid | Factor at Max CA Dubins | Min CA Clothoid | Min CA <br> Dubins | Factor at Min CA Clothoid | Factor at Min CA Dubins |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Nominal | 0.830 | 0.788 | 0.960 | 0.936 | 2D Position PID Nominal | 2D Position <br> PID Nominal | 0.751 | 0.700 | $\begin{array}{\|l\|} \hline 2 \mathrm{D} \mathrm{L1+ENLDI} \\ \text { Nominal } \\ \hline \end{array}$ | 2D NLDI Extended Nominal |
| Sensors | 0.829 | 0.782 | 0.960 | 0.936 | 2D Position <br> PID Sensor <br> Roll | 2D Position PID Sensor Roll | 0.748 | 0.698 | $\begin{array}{\|l\|l\|} \hline \text { 2D L1+ENLDI } \\ \text { Sensor Yaw } \\ \hline \end{array}$ | $\begin{array}{\|l} \text { 2D L1+ENLDI } \\ \text { Sensor Yaw } \\ \hline \end{array}$ |
| L. Act. Locked | 0.768 | 0.728 | 0.961 | 0.930 | 2D Position PID <br> ActuatorRud d Left lock | 2D Position PID <br> ActuatorRud <br> d Left lock | 0.512 | 0.511 | 2D NLDI Extended ActuatorAiler Left lock | 2D NLDI Extended ActuatorStab Left lock |
| R. Act. Locked | 0.794 | 0.725 | 0.961 | 0.930 | 2D Position <br> PID <br> ActuatorRud <br> d Rightlock | 2D Position PID <br> ActuatorRud <br> d Rightlock | 0.600 | 0.401 | 2D NLDI Extended <br> ActuatorStab <br> Right lock | 2D NLDI Extended <br> ActuatorStab <br> Right lock |
| L. Act. missed | 0.830 | 0.778 | 0.960 | 0.940 | 2D Position PID <br> ActuatorRud d Left miss | 2D Position PID <br> ActuatorRud <br> d Left miss | 0.750 | 0.683 | 2D L1+ENLDI <br> ActuatorStab Left miss | 2D L1+ENLDI ActuatorAiler Left miss |
| R. Act. missed | 0.829 | 0.780 | 0.960 | 0.940 | 2D Position PID <br> ActuatorRud d Right miss | 2D Position <br> PID <br> ActuatorRud <br> d Right miss | 0.744 | 0.683 | 2D NLDI Extended <br> ActuatorStab <br> Right miss | 2D L1+ENLDI <br> ActuatorAiler <br> Right miss |
| Wind Dir. | 0.731 | 0.710 | 0.818 | 0.872 | 2D Position PID wind dir 90deg | 2D Position PID wind dir 270deg | 0.571 | 0.458 | 2D NLDI Extended <br> wind dir 180deg | 2D L1+ENLDI wind dir 180deg |
| Wind mag. | 0.825 | 0.370 | 0.955 | 0.930 | 2D Position PID wind mag 5 knots | 2D Position PID wind mag 5 knots | 0.734 | 0.697 | 2D NLDI Extended wind mag 8 knots | 2D L1+ENLDI wind mag 5 knots |
| Turbulence | 0.640 | 0.602 | 0.944 | 0.918 | 2D Position <br> PID <br> Turbulance <br> light | 2D Position <br> PID <br> Turbulance <br> light | 0.355 | 0.370 | 2D NLDI Extended <br> Turbulance Sever | 2D L1+ENLDI <br> Turbulance Sever |
| Total | 0.786 | 0.696 | 0.961 | 0.940 | 2D Position PID ActuatorRud d Rightlock |  | 0.355 | 0.370 | 2D NLDI Extended <br> Turbulance Sever | 2D L1+ENLDI <br> Turbulance Sever |
| Angle | 0.892 | 0.838 | 0.987 | 0.974 | $\begin{array}{r} \text { 2D L1+PPID } \\ \text { Angle zero } \\ \hline \end{array}$ | $\begin{aligned} & \text { 2D L1+PPID } \\ & \text { Angle zero } \\ & \hline \end{aligned}$ | 0.789 | 0.679 | $\begin{array}{\|l\|l\|} \text { 2D L1+ENLDI } \\ \text { Angle } 360 \\ \hline \end{array}$ | 2D L1+ENLDI Angle 360 |
| Quadrant | 0.997 | 0.996 | 0.907 | 0.874 | $\begin{aligned} & \text { 2D L1+PPID } \\ & \text { first } \\ & \text { quadrant } \end{aligned}$ | 2D L1+PPID <br> first <br> quadrant | 0.931 | 0.890 | 2D L1+ENLDI <br> Second quadrant | 2D L1+ENLDI <br> Second quadrant |

Table 5. Summary of Total PI Results.

| Results <br> Factors | TOT. average Clothid | TOT average Dubins | Max TOT Clothoid | Max TOT <br> Dubins | Factor at Max TOT Clothoid | Factor at Max TOT Dubins | Min TOT Clothoid | Min TOT Dubins | Factor at Min TOT Clothoid | Factor at Min TOT Dubins |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| Nominal | 0.529 | 0.472 | 0.549 | 0.507 | 2D NLDI outer Nominal | 2D L1+PPID <br> Nominal | 0.517 | 0.407 | 2D L1+PPID <br> Nominal | 2D Position PID Nominal |
| Sensors | 0.521 | 0.466 | 0.549 | 0.517 | 2D NLDI outer Sensor Yaw | 2D L1+ENLDI <br> Sensor Pitch | 0.435 | 0.334 | 2D Position PID Sensor Pitch | 2D Position PID Sensor Pitch |
| L. Act. Locked | 0.405 | 0.384 | 0.550 | 0.508 | 2D NLDI outer ActuatorRudd Left lock | 2D L1+PPID <br> ActuatorRudd <br> Left lock | 0.153 | 0.153 | 2D NLDI Extended ActuatorAiler Left lock | 2D NLDI Extended ActuatorStab Left lock |
| R. Act. Locked | 0.441 | 0.378 | 0.550 | 0.508 | 2D NLDI outer <br> ActuatorRudd <br> Rightlock | 2D L1+PPID <br> ActuatorRudd <br> Rightlock | 0.217 | 0.120 | 2D NLDI Extended <br> ActuatorStab <br> Right lock | 2D NLDI Extended ActuatorStab Right lock |
| L. Act. missed | 0.511 | 0.458 | 0.555 | 0.505 | 2D NLDI outer ActuatorRudd Left miss | 2D L1+PPID <br> ActuatorAiler <br> Left miss | 0.469 | 0.367 | 2D L1+ENLDI <br> ActuatorStab Left miss | 2D Position PID ActuatorAiler Left miss |
| R. Act. missed | 0.515 | 0.462 | 0.555 | 0.526 | 2D Position <br> PID <br> ActuatorRudd <br> Right miss | 2D L1+ENLDI <br> ActuatorStab <br> Right miss | 0.472 | 0.367 | 2D NLDI Extended <br> ActuatorAiler <br> Right miss | 2D Position PID ActuatorAiler Right miss |
| Wind Dir. | 0.412 | 0.404 | 0.474 | 0.353 | 2D Position PID wind dir 90deg | 2D L1+PPID <br> wind dir 270deg | 0.340 | 0.367 | 2D L1+ENLDI wind dir 180deg | 2D Position PID wind dir Odeg |
| Wind mag. | 0.507 | 0.456 | 0.544 | 0.501 | 2D NLDI outer <br> wind mag 5 <br> knots | $\begin{aligned} & \text { 2D L1+PPID } \\ & \text { wind mag } 5 \\ & \text { knots } \end{aligned}$ | 0.436 | 0.351 | 2D Position PID wind mag 8 knots | 2D Position PID wind mag 8 knots |
| Turbulence | 0.406 | 0.346 | 0.540 | 0.480 | 2D Position PID Turbulance light | 2d L1+ENLDI <br> Turbulance <br> light | 0.121 | 0.355 | 2D L1+PPID <br> Turbulance Sever | 2D L1+PPID <br> Turbulance Sever |
| Total | 0.472 | 0.425 | 0.555 | 0.526 | 2D Position PID <br> ActuatorRudd Right miss | 2D L1+PPID <br> ActuatorRudd <br> Left lock | 0.121 | 0.120 | 2D L1+PPID Turbulance Sever | 2D NLDI Extended ActuatorStab Right lock |
| Angle | 0.754 | 0.621 | 0.963 | 0.944 | $\begin{aligned} & \text { 2D L1+PPID } \\ & \text { Angle zero } \\ & \hline \end{aligned}$ | $\begin{aligned} & \text { 2D L1+PPID } \\ & \text { Angle zero } \end{aligned}$ | 0.429 | 0.335 | $\begin{array}{\|l\|} \hline \text { 2D L1+ENLDI } \\ \text { Angle } 360 \\ \hline \end{array}$ | 2D Position PID Angle 360 |
| Quadrant | 0.938 | 0.916 | 0.938 | 0.916 | 2D L1+ENLDI <br> first quadrant | 2D L1+ENLDI <br> first quadrant | 0.548 | 0.536 | 2D L1+PPID <br> Second quadrant | 2D L1+PPID Second quadrant |

## CHAPTER VII

## CONCLUSION AND RECOMMENDATIONS

The comparison in two dimensional space, between the clothoid path a continuous curvature path and the Dubins path a discontinuous curvature path was performed in this study for nominal and abnormal flight conditions using conventional and adaptive controllers. The experimental design included the development of an automated data acquisition tool in conjunction with the WVU UAV simulation environment. It can be concluded that the geometrical structure of the UAVs' path has significant effects on the trajectory errors and the control surface activity. The simulation results confirm that the UAV actual bank angle and lateral acceleration are proportional to commanded curvature of the path. The maximum lateral acceleration for Dubins is about four times the maximum lateral acceleration for clothoid for the same turn radius due to the discontinuity in commanded curvature.

Also, one should note that the control activity of actuators surfaces increases with the increase of the trajectory errors, to compensate for these error and not loose the path. On the other hand, the control activity increases with the amount of the path turn, and the trajectory errors increase with the path length.

The path planning for UAVs depends on the UAV dynamic characteristics and the nature of the mission. The results of this research can effectively be used to select the proper algorithm for path generation depending on the nature of the mission. For example, if the mission requires significant obstacle avoidance, which typically can be achieved with tight turns and minimum overshot, then the clothoid approach should be used. Therefore, missions in urban environments, close to rugged terrain, and some combat mission are likely to benefit from the use of the clothoid. However, if the mission involves long straight segment without numerous turns, such as long distance delivery or some surveillance and reconnaissance missions, then Dubins path could be adequate.

Future work should include a comparison and analysis of the clothoid and Dubins path generation algorithms in 3-dimensional space, an investigation of the effects of additional abnormal conditions such as Global Positioning System malfunctions and finally development of path planning with moving and fixed threat regions in 3-dimensional space, which is more complicated and challenging but on the other hand is very useful for military and civilian missions.

## References

[1] N. Goyer, "AircraftMarketPlace," 5 May 2010. [Online]. Available: http://www.acmp.com/blog/history of pilotless aircraft.html. [Accessed 05 Aug. 2014].
[2] J. R. Clapper , J. J. Young, J. E. Cartwright, J. G. Grimes, S. C. Payton, S. J. Stackley and D. Popps, "Unmanned Systems Integrated Roadmap 2009-2034," Office of the Secretary of Defense, Washington,DC, 2010.
[3] D. Wenjie, "Nonholonomic Mechanical System Control," [Online]. Available: http://faculty.utpa.edu/dongw/nonholonomic_system_control.htm. [Accessed 2910 2014].
[4] M. Shanmugavel, "Path planning of multiple autonomous vehicles," Cranfield University, Shrivenham, 2007.
[5] S. M. La Valle, Planning Algorithm, Cambridge University Press, 2006.
[6] N. J. Nilsson, "A mobile Automaton: An Application of Artificial Techniques," in SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1969.
[7] T. Lozano Pérez and W. A. Michael, "An algorithm for planning collision free paths among polyhedral obstacles," Communications of the ACM, vol. 22, no. 10, pp. 560-570, 1979.
[8] F. Schøler, C. H. Anders la and M. Bisgaard, "Generating Configuration Spaces and Visibility Graphs from a Geometric Workspace for UAV Path Planning," Autonomous Robots, 2014.
[9] E. W. Weisstein, "Voronoi Diagram," MathWorld A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/VoronoiDiagram.html. [Accessed Aug. 2014].
[10] F. Aurenhammer and K. Rolf, "Voronoi diagrams," in Handbook of computational geometry, 2000, pp. 201-290.
[11] X. Chen and C. Xiangmin, "The UAV dynamic path planning algorithm research based on Voronoi diagram," in Control and Decision Conference (2014 CCDC), The 26th Chinese, 2014.
[12] J. Wilburn and M. Perhinschi, "Enhanced Modified Voronoi Algorithm for UAV Path Planning and Obstacle Avoidance," International Review of Aerospace Engineering, vol. 6, no. 1, pp. 54-63, 2013.
[13] J. Wilburn, "Development of an Integrated Intelligent Multi-Objective Framework for UAV Trajectory Generation," West Virginia Unuiversity, Morgantown, 2013.
[14] A. Stentz, "The Focussed D* Algorthim for Real Time Replanning," in Proceedings of the International Joint Conference on Artifiical Intelligence, 1995.
[15] M. Kothari, I. Pstlethwaite and D. W. Gu, "Multi UAV Path Planning in Obstacle Rich Environments Usin Rapidly Exploring Random Trees," in Proceedings of the 48th IEEE Joint Conference on Decision and Control., China : s.n., 2009.
[16] E. Bonabeau, "Editor's Introduction: Stigmergy," Artificial Life, vol. 5, no. 2, pp. 95-96, 1999.
[17] O. Holland and C. Melhuish, "Stigmergy, Self Organization, and Sorting in Collective Robotics," Artificial Life, vol. 5, no. 2, pp. 173-202, 1999.
[18] C. Zang, Z. Zhen and W. Daobo, "UAV path planning method based on ant colony optimization," in Control and Decision Conference (CCDC), 2010 Chinese, 2010.
[19] H. V. Parunak, M. Purcell and R. O’Connell, "Digital Pheromones for Autonomous Coordination of Swarming UAV's," in 1st UAV Conference, 2002.
[20] J. H. Holland, Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control and Crtificial Intelligence, U Michigan Press, 1975.
[21] T. Arora, Y. Gigras and V. Arora, "Robotic Path Planning using Genetic Algorithm in Dynamic Environment," International Journal of Computer Applications, vol. 89, no. 11, pp. 8-12, 2014.
[22] O. K. Sahingoz, "Generation of Bezier Curve Based Flyable Trajectories for Multi UAV Systems with Parallel Genetic Algorithm," Journal of Intelligent \& Robotic Systems, Vols. 1-2, no. 74, pp. 499-511, 2014.
[23] E. W. Weisstein, "Bézier Curve," MathWorld A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/BezierCurve.html. [Accessed 201411 05].
[24] . L. A. Zadeh, "Fuzzy sets," Information and Control, vol. 8, no. 3, p. 338-353, 1965. [25] J. Yaochu, "Soft Computing," Department of Computing University of Surrey, [Online]. Available: http://www.soft computing.de/def.html. [Accessed Aug. 2014].
[26] C. Sabo and C. Kelly , "Fuzzy Logic Unmanned Air Vehicle Motion Planning," Advances in Fuzzy Systems, vol. 2012, 2012.
[27] Z. Sun, T. Dong, X. Liao, R. Zhang and D. Y. Song, "Fuzzy logic for flight control II: fuzzy logic approach to path tracking and obstacles avoidance of UAVs," in Advanced Fuzzy Logic Technologies in Industrial Applications, Springer London, 2006, pp. 223-235.
[28] J. Wilburn, J. Cole, M. Perhinschi and B. Wilburn, "Comparison of a Fuzzy Logic Controller to a Potential Field Controller for Real Time UAV Navigation," in AIAA GUIDANCE, NAVIGATION, AND CONTROL CONFERENCE, Minneapolis, Minnesota, 2012.
[29] W. . S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," The bulletin of mathematical biophysics , vol. 5, no. 4, pp. 115-133 , 1943.
[30] L. Sun , . L. Yonglong and L. W. Xintao Dingand , "Path Planning and Obstacle Avoidance for Mobile Robots in a Dynamic Environment," Open Automation and Control Systems Journal, vol. 6, pp. 77-83, 2014.
[31] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," American Journal of Mathematics, vol. 79, no. 3, pp. 497-516, 1957.
[32] A. Tsourdos, B. White and M. Shanmugavel, Cooperative path planning of unmanned aerial vehicles, Wiley, 2010.
[33] Y. Lin and S. Srikanth , "Path planning using 3D Dubins Curve for Unmanned Aerial Vehicles," in Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, Orlando, FL, 2014.
[34] E. W. Weisstein, "Cornu Spiral," MathWorld A Wolfram Web Resource, [Online]. Available: http://mathworld.wolfram.com/CornuSpiral.html. [Accessed Aug 2014].
[35] H. Vorobieva, G. Sebastien, M. E. Nicoleta and M. Said, "Automatic parallel parking with geometric continuous curvature path planning," in Intelligent Vehicles Symposium Proceedings, 2014 IEEE, Dearborn, MI, 2014.
[36] J. Wilburn, M. G. Perhinschi and B. Wilburn, "Implementation of Composite Clothoid Paths for Continuous Curvature Trajectory Generation for UAVs," in AIAA Guidance, Navigation, and Control (GNC) Conference, Boston, MA, 2013.
[37] R. T. Farouki and S. Takis , "Pythagorean hodographs," IBM Journal of Research and Development, vol. 34, no. 5, pp. 736-752, 1990.
[38] O. Karas, "UAV simulation environment for autonomous flight control algorithms," West Virginia University, 2012.
[39] M. G. Perhinschi, H. Moncayo, J. Davis, B. Wilburn, O. Karas and M. Wathen, "Development of a Simulation Environment for Autonomous Flight Control Algorithms," in AIAA MODELING AND SIMULATION TECHNOLOGIES CONFERENCE, Portland Oregon, 2011.
[40] S. Park, C. Han-Lim, R. Nicholas and J. P. How, "Learning the covariance dynamics of a large-scale environment for informative path planning of unmanned aerial vehicle sensors," in AIAA Guidance, Navigation, and Control Conference. American Institute of Aeronautics and Astronautics, 2009.
[41] J. N. Wilburn, "Development of an Integrated Intelligent Multi-Objective Framework for UAV Trajectory Generation," West Virginia University, Morgantown, 2013.
[42] J. Wilburn, B. Wilburn and M. Perhinschi, "Implementation of a 3-Dimensional Dubins Based UAV," in AIAA Guidance, Navigation, and Control (GNC) Conference, Boston, MA, 2013.
[43] B. K. Wilburn, M. G. Perhinschi, H. Moncayo, O. Karas and J. N. Wilburn, "Unmanned aerial vehicle trajectory tracking algorithm comparison," International Journal of Intelligent Unmanned Systems, vol. 1, no. 3, pp. 276-302, 2013.
[44] G. Campa , G. Yu, B. Seanor, M. R. Napolitano, L. Pollini and M. L. Fravolini, "Design and flight-testing of non-linear formation control laws," Control Engineering Practice, vol. 15, no. 9, pp. 1077-1092, 2007.
[45] M. R. Napolitano, "Development of formation flight control algorithms using 3 YF-22," April 2005. [Online]. Available: www.stormingmedia.us/99/9944/A994434.html. [Accessed 2012].
[46] H. Moncayo, M. G. Perhinschi, B. Wilburn, J. Wilburn and O. Karas, "Extended nonlinear dynamic inversion control laws for unmanned air vehicles," in the 2012 AIAA Guidance, Navigation, and Control Conference, Minneapolis,MN, 2012.
[47] N. Hovakimyan and C. Chengyu, L1 adaptive control theory: Guaranteed robustness with fast adaptation, vol. 21, Siam, 2010.

## Appendices

## Appendix A

Comparison of the Generated Path Length


Figure 70. Distance with Nominal Conditions in (m).


Figure 71. Distance with Sensor Failure in (m).


Figure 72. Distance with Left Actuator Locked in (m).


Figure 73. Distance with Right Actuator Locked in (m).


Figure 74. Distance with Left Actuator Missed in (m).


Figure 75. Distance Right Actuator Missed in (m).


Figure 76. Distance with Different Wind Directions in (m).


Figure 77. Distance with Different Wind Magnitudes in (m).


Figure 78. Distance with Turbulence in (m).


Figure 79. Distance with Different Turn Angles in (m).


Figure 80. Distance with Different Quadrants in (m).

Appendix B

Comparison of Trajectory Tracking Performance Indices


Figure 81. PI with Nominal Conditions.


Figure 82. PI with Sensor Failure.


Figure 83. PI with Left Actuator Locked.


Figure 84. PI with Right Actuator Locked.


Figure 85. PI with Left Actuator Missed.


Figure 86. PI with Right Actuator Missed.


Figure 87. PI with Different Wind Angles.


Figure 88. PI with Different Wind Magnitudes.


Figure 89. PI with Turbulence.


Figure 90. PI with Different Turn Angles.


Figure 91. PI with Different Quadrants.

## Appendix C

Save Function Output Data

| 1 | Case No. |
| :---: | :---: |
| 2 | Case name |
| 3 | Actual Distance |
| 4 | Commanded Distance |
| 5 | Maximum (Actual or Commanded) |
| 7 | Differences between actual and Commanded |
| 8 | PI Trajectory Tracking |
| 9 | PI Control Activity |
| 10 | PI Total |
| 11 | Max XY Error[m] |
| 12 | Max Z Error [m] |
| 13 | Max XYZ Error [m] |
| 14 | Mean XY Error [m] |
| 15 | Mean Z Error [m] |
| 16 | Mean XYZ Error [m] |
| 17 | Standard Deviation XY Error [m] |
| 18 | Standard Deviation Z Error [m] |
| 19 | Standard Deviation XYZ Error [m] |
| 20 | Elevator Activation Index[rad/s] |
| 21 | Aileron Activation Index [rad/s] |
| 22 | Rudder Activation Index [rad/s] |
| 23 | Throttle Activation Index [\%/s] |
| 24 | Elevator Saturation Index [\%] |
| 25 | Aileron Saturation Index [\%] |
| 26 | Rudder Saturation Index [\%] |
| 27 | Throttle Saturation Index |


[^0]:    1. Tangent for path with RR turns
    2. Tangent for path with RL turns
    3. Tangent for path with LR turns
    4. Tangent for path with LL turns
